protégé

9th International Protégé Conference

July 24-26, 2006
Stanford University, USA

Presentation Abstracts

http://protege.stanford.edu/conference/2006/schedule.html

TABLE OF CONTENTS

Table of Contents

TABLE OF CONTENTS

Embracing Change:

Financial Informatics and Risk Analytics

Mark D. Flood
Senior Financial Economist
Federal Housing Finance Board
floodm@fhfb.gov

April 17, 2006

EXTENDED ABSTRACT

Embracing Change: Financial Informatics and Risk Analytics

Overview

We present an enterprise design pattern for managing metadata in support of financial analytics packages. The complexity of financial modeling typically requires deployment of multiple financial analytics packages, drawing data from multiple source systems. Business domain experts are typically needed to understand the data requirements of these packages. Financial product innovation and research advances imply that data requirements are chronically unstable. These forces of complexity and instability motivate a software architecture that exposes financial metadata declaratively for editing by financial analysts.

The key contributions of the paper are twofold. First, we present a simple model that captures the most important software development costs involved in maintaining risk-management databases. This model allows us to identify and measure the scalability of alternative data-integration architectures. Second, we present a high-level data-integration architecture that is flexible, scalable, practical, and portable. This solution uses an ontology editor (e.g., Protégé) to expose the financial metadata for user editing in a controlled context. As a result, domain experts can make on-the-fly metadata modifications without provoking a costly design-develop-test-deploy iteration. The solution would not be practical without some facility for ontology editing.

The problem

Three basic forces – financial innovation, model risk, and strategic policy evolution – conspire to create a very unstable data integration environment for risk-managment analytics. The nature of the data coming into the models may be changing, due to financial innovation. The set of models in use may be changing, due to modeling innovations or shifting conditions and priorities. The nature of outputs requested from the models may be changing, due to changes in strategic goals. Most risk-management applications– including trading decisions, capital allocations, and hedging limits – require highly accurate data, implying a premium on the quality of the metadata. Wall Street spends billions of dollars every year addressing the operational (data integration) issues implied by these complications.

Specification and mapping costs

There is a potentially large and unstable list of source systems from which data will ultimately be drawn, and target analytical packages which the data will feed. There are two costly operations involved with managing the metadata for such a system: (a) *specification* of the data schemas for each system; and (b) *mapping* between schemas in the source systems and input schemas in the target analytics packages. To minimize the costs of mapping and create a data-integeration architecture that scales linearly in the number of software packages, we recommend the introduction of a central normalizing schema, which we call the "numeraire schema". The scalability properties of alternative architectures are established in a very simple model.

A design pattern

We specify a data-integration architecture that meets four high-level requirements:

R1. The solution must be ***flexible*** – capable of adapting to financial and research innovation in a short time frame. To achieve this, the solution exposes the metadata for editing by business analysts, as the needed responsiveness (for re-specification and re-mapping) is too quick to allow imposition of full-blown design-develop-test-deploy lifecycles.

R2. The solution must be ***scalable***. Modeling and mapping costs must not explode as the number of analytics packages, source data systems, and financial contract types grows.

R3. The solution must be ***practical***. That is, it must be implementable with production-quality tools and technologies available today.

R4. The solution should be ***portable***, to facilitate adoption. In other words, the design must be self-contained, and therefore deployable in a range of enterprise architectures (e.g., XML messaging, shared database, etc.), and software platforms (J2EE, .Net, etc.).

The design divides into four separate subsystems: data integration, position data, market data, and analytics. This division enhances portability, since the subsystems are very loosely coupled. The data integration subsystem sits at the center of the design. It includes the database maintaining the full official state of the system. Importantly, it also includes an ontology editor (e.g., Protégé). This addresses the flexibility requirement by allowing financial analysts to modify (perhaps frequently) the specifications of the various data sources and targets. The system has a number of disparate pieces – user documentation, data schemas, mappings, etc. – that must remain consistent with one another, even in the face of such changes. The design employs a formal ontology to record these facts as declarative "meta-metadata," since an ontology can include the rules to enforce consistency among the disparate pieces at all times. The ontology editor must enable users to modify both the general *structure* of a data-specification ontology as well as the individual specification *instances* corresponding to each data source and target.

The resulting four-layer architecture (consiting of: data, data schema, ontology instance, and ontological structure) is closely analogous to the Object Management Group's four-layer meta-object facility, with the innovation that an ontology editor (Protégé) is used to manage the top two layers of the metadata abstraction hierarchy. In a four-layer design, traditional metadata, typically consisting of SQL or XML schemas, are governed by a higher-level "meta-metadata" that describes the traditional metadata. In turn, these are governed by the ontological structure (the "meta-meta-metadata"), an abstract language for defining metadata. Each layer is an abstraction of the layers below it, and each layer must obey the constraints defined by the layer above it. The constraint that ontology instances must share a common ontological structure greatly facilitates programmatic access to the ontology for editing and generating documentation, data schemas, test data, etc.

Ontology Based Application Server to Execute Semantic Rich Requests

Dilvan de Abreu Moreira and Flávia Linhalis
Institute of Mathematics and Science Computing (ICMC)
University of São Paulo, São Carlos, Brazil
{dilvan, flavia}@icmc.usp.br

1. Introduction

Since computing earlier days, people have wanted computers that could interface and be programmed using easy to use tools like speech or gesture. However, until now, such a computer system remains in the realm of scientific fiction. Now, with recent advances in human computer interfaces (such as speech understanding, ubiquitous computing, tablet computers), and in the processing of semantic information by computers (through initiatives such as the Semantic Web), new technologies and tools (such as Protégé and Jena) are maturing to a level where such a system becomes a possibility.

Our work is about the application of semantic information to make computer systems smarter (using ontologies to add knowledge about application domains and common sense) and, with the help of new kinds of human-computer interfaces, to produce systems that a computer layperson can understand and program in useful ways.

In particular, this work concentrates on the processing of semantic rich requests through the activation of software components stored in an application server. Semantic information from each request is combined with semantic information from ontologies, describing the application domain and the available components, to generate arguments and activate the most appropriate component (or components) to attend the request. Ontology Based Application Servers (OBAS, section 2) can be used to improve component activation. We are testing this idea with the development of a prototype of an OBAS to execute imperative natural language requests expressed in several natural languages (NL-OBAS). In section 3 we focus on NL-OBAS ontologies.

2. Ontology Based Application Servers

The idea of improving component activation using ontologies is outlined in [1], where the authors propose an Ontology Based Application Server (OBAS): an application server, like the ones described in the Sun’s J2EE architecture, but with the capability of using semantic information about the software components it holds. In an OBAS, an ontology captures properties of, relationships between, and behaviors of components. These component descriptions may be queried, may foresight required actions, e.g. preloading of indirectly required components, may be checked to avoid inconsistent system configurations, or may improve the dynamic composition of services [1, 2].

Expanding this idea for an OBAS, a request does not need to be addressed to a particular component implementation. For instance, if one needs to send an email, an email request can be made to a generic software component described in the OBAS as an email sender component. This request will contain only the information needed to send an email (recipient, sender, subject, body, etc) the actual software component and its parameters are going to be determined by the OBAS using domain and component ontologies. Once a system capable of attending such requests is available, the next step is to connect it to human-computer interfaces that could generate the requests, from user interaction, and show the results of these requests in the context of the interaction.

To test these ideas, a prototype of an Ontology Based Application Server for the execution of imperative natural language requests (NL-OBAS) is being developed [3], as described in the next section.

3. The NL-OBAS

The NL-OBAS architecture (Figure 1) provides the UNL-Enconverter service to convert natural language requests into an interlingua named UNL (Universal Networking Language) [4]. The interlingua allows the use of different human languages to express the requests (other systems are restricted to English). Currently, the input requests must be imperative sentences; however the system can be extended to other sentence types.

The architecture also provides a Semantic Mapping Service, that uses an ontology to extract relevant information about the domain components and semantic information from the UNL representation of a request. The Component Loader service uses this information to dynamically load specific software components and execute methods to fulfill a request.

The NL-OBAS software components are related to a specific domain. These components are described in the Component Ontology and the application domain is described in the Domain Ontology. The components can query and modify the Domain Ontology.

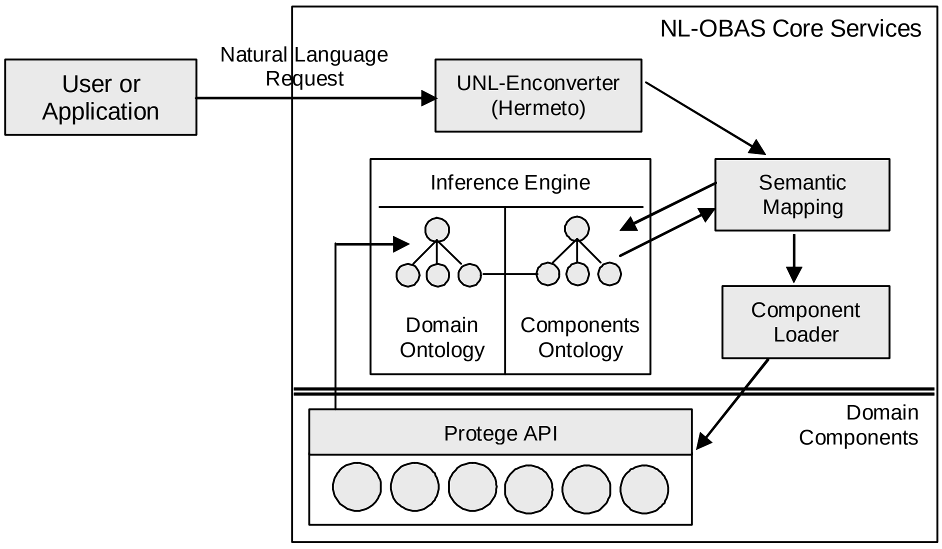


Figure 1 – NL-OBAS Architecture.

The Domain and Component ontologies were both developed using the Protégé tool [5] and are represented in OWL (Ontology Web Language) [6]. The remaining of this section is about these ontologies. Details about the services of the NL-OBAS (UNL-Enconverter, Semantic Mapping, Component Loader) are described at Linhalis & Moreira [3].

Figure 2 presents the Component Ontology classes, attributes and relationships. This ontology has to be instantiated in accordance with the syntactic and semantic characteristics of the components in the Domain Components Layer of the NL-OBAS.

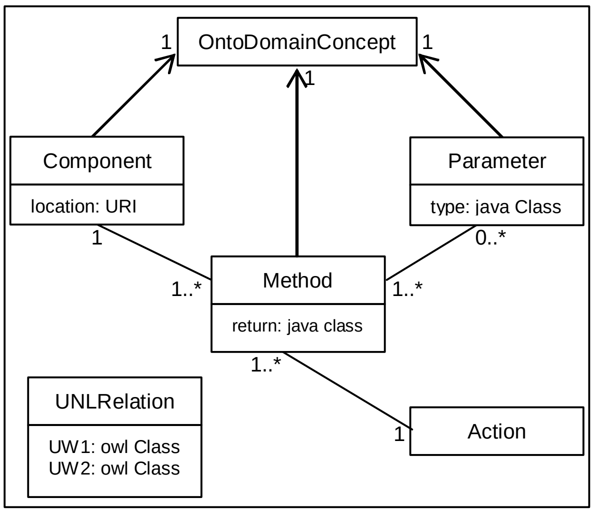


Figure 2 – Component Ontology.

The instances of the OntoDomainConcept class correspond to concepts of the application domain that are also concepts represented as classes in the Domain Ontology (Figure 3). Each Component class instance corresponds to the representation of an application domain software component that can be related to one or more concepts (represented as instances) of the OntoDomainConcept class. For example, considering the course management domain, "Student", "Teacher" and "Course" are concepts and belong to the OntoDomainConcept class. An instance of the Component class, like TeacherComponent, represents a component that is responsible for the execution of actions related to the "Teacher" concept.

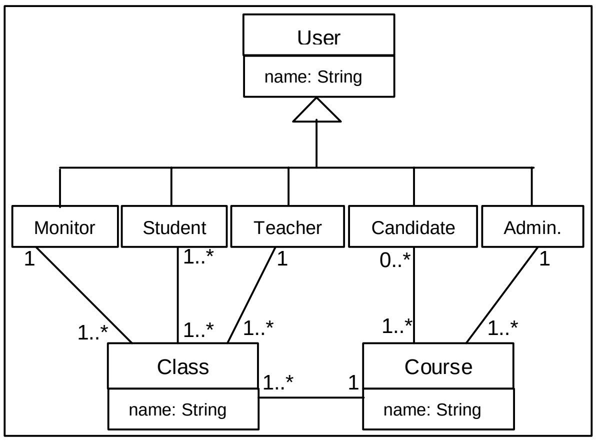


Figure 3 – Domain Ontology.

The Method class instances correspond to the methods of each component. The Parameter class instances correspond to the arguments of each method. And finally, the Action class instances correspond to imperative verbs. Each verb (action) is related to one or more methods, and each method is related to one verb.

The UNLRelation class indicates the mapping between the UNL interlingua and the software components. This class has instances representing all UNL relations being used in the imperative sentences of the application domain. Each instance of this class is related with Component, Parameter or Action classes of the Component Ontology. More information about the UNLRelation class can be found at [3].

The natural language requests are related to a specific domain that is represented in the Domain Ontology. We defined and instantiated the Domain Ontology with relationships between the concepts of the course management domain.

4. Conclusion and Future Work

The NL-OBAS architecture can be used in different application domains; it is just necessary to write the appropriate software components, define the dictionary and grammar rules (that will be used by the UNL-Enconverter service), create instances of the Component Ontology and define the Domain Ontology.

Much work still needs to be done. We plan to improve and expand the NL-OBAS prototype and transform it in a system that can be used, in restricted domains, by experts in other fields, such as biologists, physicians, geneticists, stock market traders and others, to write useful programs, without the help of computer experts.

Using natural language or other kinds of human-computer interfaces, the OBAS technology has the potential to be a hot research topic in the coming years and to offer interesting new opportunities for research as this technology can be central to the development of the future Semantic Web applications.

References

[1] Oberle, D., Staab, S. and Eberhart, A. Towards Semantic Middleware for Web Application Development. *IEEE Distributed Systems Online*, February, 2005.

[2] Sugumaran, V. and Storey, V. C. A Semantic-Based Approach to Component Retrieval. *ACM SIGMIS Database*, v. 34, n. 3, 2003, pp. 8-24.

[3] Linhalis, F. and Moreira, D. A. "Ontology-Based Application Server to the Execution of Imperative Natural Language Requests". H.L. Larsen et al. (Eds.): 7th International Conference on Flexible Query Answering Systems (FQAS), June 2006, Milano, Italy. *Lecture Notes in Artificial Intelligence (LNAI) 4027*, pp. 589–600, 2006. Springer-Verlag, Berlin Heidelberg, 2006.

[4] Ushida, H. and Zhu, M. The Universal Networking Language beyond Machine Translation. International Symposium on Language and Cyberspace, Seoul (South Korea), 2001

[5] Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W. and Musen, M. A. Creating Semantic Web Contents with Protégé-2000. IEEE Intelligent Systems, v.16 n.2, 2001, pp. 60-71.

[6] McGuinness, D. L. and van Harmelen, F. Web Ontology Language Overview. W3C Recommendation, February 2004. http://www.w3.org/TR/owl-features/.

Content Object Modelling with Protégé

Jon Carey – Producer
Helen Lippell – Information Architect
David Wood – Senior Software Engineer (david.wood.01@bbc.co.uk)
British Broadcasting Company (BBC),
BBC New Media, London

Introduction

This paper describes the process by which a project team within BBC New Media used Protégé and its OWL plug-in in order to provide a semantic description of a complex existing ontology and manage its further development.

Context

Content Management Culture is one of the biggest ongoing projects within the BBC's New Media division. Our remit is to provide predominantly web content management solutions to the new media sections of various BBC divisions. In order to do this we have built a custom interface on top of a commercial content management system (CMS).
The semantic 'content object' model, managed by Information Architects, which underlies the CMS was previously stored in Microsoft Word documents. The static nature of this format impeded both the management of the model and its implementation in a technical environment. Migrating the model into Protégé has standardised the modelling practices and increased the visibility and reusability of modelled components, thereby facilitating faster prototyping.

Content object modelling

We set out to find a software solution that would allow us to manage a number of content object models and their constituent objects and the attributes within these objects. For the purposes of the content management system we have divided all editorial content into its constituent parts.

A typical web page will be split into: article text, main image, other images, web links etc. all of which are content objects and exist as separate entities in their own right. This allows reuse of even the smallest content objects across a number of web pages without duplicating the object itself.

All content objects are defined by their attributes, which describe each property of the content object, and business rules, which control how an object can be created. Thus, the attributes of an image, for example, could be:

- Image type
- Image size (width and height)
- Image description
- Image 'Alt' tag
- Image caption
- Descriptive metadata

And its business rules could be:

- The image type must be one of a specific format (e.g. jpeg/gif)
- The image associated with an article must be of a specific size (e.g. 203 x 152 pixels)
- The image description must have a maximum number of characters in a text string
- The image 'Alt' tag and caption must have more than zero characters
- The image must have at least one piece of descriptive metadata

Some attributes may be mandatory in a given object and some may be optional. An attribute can be shared across more than one object; e.g. Headline. There is always a minimum set of attributes for any given object, e.g. the objects' file name and title.

As such:

- The model is the full set of all possible objects and all the attributes that those objects may have.
- A partner instance of the model is a collection of objects as defined for a specific partner (i.e. a department within the BBC). It will, in most cases, not be all possible objects
- An object is a collection of attributes which make up a content entity such as an image or a web link
- An attribute is a property of the object (e.g. headline, author, size)
- The attributes are of a specified type (e.g. text, date)
- The attributes have business rules, including cardinality (one or many), type, validation rules (e.g. number only or a complex regular expression) etc

Partner instances of the model are made up of a particular grouping of objects tailored to suit a particular partner. Some objects will be used across more than one partner instance of the model. There may be differences between different instances of the same object according to a partner's specific need.

There are at present two partner instances of the model and there will be more in future as we deploy our system to new partners within the BBC.

How we used Protégé

After a requirements gathering exercise, and a process of analysing various modelling tools, Protégé was selected on the basis of its support for OWL semantics. Two core requirements were that:

- The model couldn't be broken – certain rules would need to be enforced
- XML formats could be generated from whatever output was provided, for use within – but not exclusive to – the CMS

Our first step was to define a meta model of our existing content objects which we could use as a basis for 'how content objects could be created'. This defined three core building blocks of *Container*, *Property* and *ContentObject* – the subtypes of which would form the base of the model, and a starting point for our Protégé classes. Additional classes were I – to represent partner uses of an object, and *ReferenceData* – used to indicate controlled vocabularies from an external resource. Individuals within the model represent specific instances of content objects defined as subtypes of the *ContentObject* class. OWL restrictions are used to enforce content modelling constraints, such as the types of Property objects used for a certain *ContentObject*, or fixing a value of a meta model property.

We had to customise Protégé in the following ways:

- Interaction with CVS – using the NetBeans Java CVS library to version control the OWL files
- OWLTest classes were needed to check the sanity of the meta model
- XML output plug-ins were created to generate output, both for the purposes of documentation, and for use within our CMS
- A TabWidget plug-in was created for management of non-semantic presentation information

Conclusion

Protégé has provided us with an extensible solution to our content object modelling issues that would have been difficult - if not impossible - to achieve with other software. With the content object model now stored in a semantically defined format (OWL) rather than a Word document, Protégé has empowered our information architecture team to create, prototype and document the modelling process for a technical solution, with minimal future technical involvement.

Data Driven Ontology Alignment

Nigam Shah, Stanford University

In our recent work with annotations of tissue microarrays, we have automatically mapped approximately 80% of annotations for the samples in the Stanford Tissue Microarray Database to ontology terms. We have shown that a significant proportion of the diagnosis-related annotations map to terms from both the NCI thesaurus and SNOMED-CT. This mapping of a single record to terms from different ontologies presents a concrete data-driven mechanism for aligning related ontologies by using them for annotation. Such data-driven alignments have the potential to be complementary to existing alignment approaches, such as PROMPT, and may be used in tandem to better align related ontologies.

Towards Semantic Interoperability in a Clinical Trials Management System
Ravi D. Shankar, Stanford University

Abstract

Clinical trials are studies in human patients to evaluate the safety and effectiveness of new therapies. Managing a clinical trial from its inception to completion typically involves multiple disparate applications facilitating activities such as trial design specification, clinical sites management, participants tracking, and trial data analysis. There remains however a strong impetus to integrate these diverse subsystems – each supporting different but related functions of clinical trial management – at syntactic and semantic levels so as to improve clarity, consistency and correctness in specifying clinical trials, and in acquiring and analyzing clinical data. The situation becomes especially critical with the need to manage multiple clinical trials at various sites, and to facilitate meta-analyses on trials. This paper introduces a knowledge-based framework that we are building to support a suite of clinical trial management subsystems. Our initiative uses semantic technologies to provide a consistent basis for the subsystems to interoperate. We are adapting this approach to the Immune Tolerance Network (ITN), an international research consortium developing new therapeutics in immune-mediated disorders.

Towards Subject-Centric Merging of Ontologies

Jack Park
SRI International
jack.park@sri.com
Patrick Durusau
patrick@durusau.net

The challenge of Cyberinfrastructure is to integrate relevant and often disparate resources to provide a useful, usable, and enabling framework for research and discovery characterized by broad access and "end-to-end" coordination
–NSF Workshop on Cyberinfrastructure for the Social Sciences, 2005[1]

Extended Abstract

We are developing a *subject-centric* approach to federation of heterogeneous representations of subjects. Subjects are represented in many ways, including through the formal structures of ontologies. Here, we will sketch our approach to semantic interoperability among heterogeneous world views through implementations of subject maps, and briefly mention that such a subject map is being considered for installation as a plugin to the Protégé platform.

Our work exists in a field rich in experience and motivation for semantic interoperability among heterogeneous ontologies. As a sketch of that field, John Madden [1] listed "three topologies for semantic interoperability:"

1. Central semantic authority
2. Hierarchical semantics
3. Federated semantics

We consider *central semantic authority*, and *federated semantics* to be of greatest contrast in our work. Since the approach we present lies in the *federated semantics* domain and involves mapping, the key points that Madden made about semantic federation are most relevant to our work. He points out that responsibility for mapping is finely divided, which, to us, implies that many individuals and groups will contribute to the final mappings, that quality depends on peer-to-peer collaboration, that there are no global guarantees, and that there is a need to support a "market" for ontology fragments

Central semantic authority reminds of "Hobson's Choice," where a central authority grants the right to rent any horse so long as it is the horse closest to the door. In [2], we spoke to the opportunity to determine, among different world views, which subjects are being represented, and to perform merging when subject sameness is determined. We argue that a subject map provides appropriate facilities for performing subject-centric merging and marshalling the remaining, unmerged ontological entities for reference, navigation, and completeness. In this process, we argue that no particular world view

[1] NSF: http://vis.sdsc.edu/sbe/

gains privilege over any other; all world views are presented uniformly and each representation is captured in any merge process without loss of information.

Consider that the XML topic maps paradigm (XTM) [3] and [4] has served the topic mapping community well, and continues to do so. Topic maps do not replace ontologies. Rather, they augment ontologies and other world views. As a brief introduction to subject maps, consider that, as we begin to apply topic mapping to the complex use cases of, for instance, bioinformatics, where subject identity is under-specified by XTM, we need a framework that facilitates a finer-grained approach to subject identity. A framework for such a specification is known as the Topic Maps Reference Model (TMRM) [5], and we have begun to label implementations of the TMRM as *subject maps* to distinguish them from their siblings. The TMRM makes no specification of the means by which subjects are identified or ways in which subject proxies, as they are called, are merged. Rather, the TMRM leaves subject map authors free to make their own design decisions, but it specifies that subject map authors are required to *disclose* the design decisions, the ontological commitments they have made such that other implementations can create means by which merging among different subject maps can be afforded. Disclosures, in the TMRM, form a *legend* for the map, much as street maps have legends to explain the artifacts represented by the map. Steve Newcomb had this to say[2] about disclosures:

> *The Topic Maps Reference Model is our attempt to set forth a checklist of things that must be disclosed about a given body of knowledge, regardless of how that body is represented, in order to enable a specific benefit to be realized. The benefit is facilitation of the task of integrating that body of knowledge with other such bodies, on a subject-by-subject basis. I like to say that the disclosures amount to descriptions of subject address spaces.*

Implementations of the TMRM create ways in which determination of subject sameness can be evaluated among classes and properties found in different ontologies. When subject sameness is found, those classes or properties are merged and assertions can be made on related classes to support further merging opportunities. When ontological entities merge in a subject map, the statements made by those entities, that is their properties and relations are carried with them.

In some sense, there are ontological commitments made in subject mapping. But, they are made with the specific intent to facilitate the semantic integration of world views which are always the product of ontological commitments. The nature of this facilitation lies not in the ontological commitments made by any author; rather, merging is facilitated on the basis of determination of subject identity, which, in many cases, is underspecified in the constituent world views. This leaves open much room for the peer-to-peer collaboration mentioned in Madden’s presentation. But, in the case where sufficient evidence for subject identity sameness is found, merging can be handled by the subject map engine. Steve Newcomb had this to say about ontological commitments in the TMRM:

> *It's in the nature of what we've been trying to do that the semantics be left undefined. Whenever we've recognized that we've been making semantic*

[2] Steve Newcomb: Personal communication, January 24, 2006

assumptions, we've ruthlessly expunged them. We're endeavoring to create conditions favorable for discovery of relevant information expressed in terms of diverse universes of discourse, and across those universes of discourse. We need to be both inviting to knowledge resources created in the light of diverse semantic systems, cultures and communities of practice, and, at the same time, even-handed with respect to all of them. Among other things, we're looking to create a better marketplace for ideas, with more opportunities to add value for anyone with value to add.

Benefits of federation through subject-centric merging derive from viewing same-subject ontological entities together in a single subject proxy. A subject proxy is the name given to a kind of *container* for all of the properties marshaled as representations of a single subject. We have characterized [2] a prime benefit as the emergence of "worm holes" between different world views. For instance, when a particular ontology provides certain properties to a merged subject proxy not provided by another entity merged into that proxy, the fact that each property is identified with the ontology from which it comes provides cognitive links between different ontologies; Exploration of different ontologies provides opportunities for chance discovery.

Light weight implementations of the TMRM are beginning to appear. For instance, the first author is creating an implementation called TopicSpaces that is initially being applied to social bookmarking applications. It is developed using Java and is being considered for use as a plugin agent for Protégé in service of ontology federation projects.

References

[1] Madden, John, “Sharing healthcare meaning is hard to do,” Slides presented at the Face-to-Face W3C HCLSIG Meeting, January 25-26, 2006, Cambridge, MA. On the web at http://www.w3.org/2001/sw/hcls/f2f-2006/F2F-Agenda.html
[2] Park, Jack, and Patrick Durusau, "Avoiding Hobson's Choice In Choosing An Ontology", Teleconference presentation to the Ontolog Community. On the web at http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall_2006_04_27
[3] XTM Authoring Group, “XML Topic Maps (XTM) 1.0,” on the web at http://www.topicmaps.org/xtm/index.html
[4] Park, Jack, and Sam Hunting (Editors), *XML Topic Maps: Creating and Using Topic Maps for the Web*,” 2002, Addison-Wesley.
[5] Durusau, Patrick, Steve Newcomb and Robert Barta, “Topic Maps Reference Model ISO 13250-5,” available on the web at http://www.jtc1sc34.org/repository/0710.pdf

Interoperability of Protégé using RDF(S) as interchange language

Raúl García-Castro, Asunción Gómez-Pérez

Ontology Engineering Group, Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{rgarcia,asun}@fi.upm.es

Abstract. The Knowledge Web[1] European Network of Excellence, in order to assess and improve the interoperability of ontology development tools, has organised a benchmarking of the interoperability of ontology development tools using RDF(S) as interchange language. This paper presents the participation of Protégé in this benchmarking.

1 Introduction

The technology that supports the Semantic Web appears in different forms (ontology editors, repositories, reasoners, etc.) and, while all these tools use ontologies, not all of them share a common knowledge representation model. Moreover, a single tool can support different knowledge models, such as Protégé that can model ontologies using frames or OWL. This diversity in representation formalisms causes a problem when tools try to interoperate, affecting users who want either to interchange their ontologies from one tool to another, or to use their preferred tool when developing ontologies collaboratively.

Nowadays, users of ontology development tools do not know whether ontologies can be properly interchanged between two ontology development tools and, if so, which are the consequences of this interchange, such as addition or loss of knowledge. This leads to a slower uptake of ontology development tools by end users, both in the academia and the industrial world.

Knowledge Web, in order to assess and improve the interoperability of ontology development tools, has organised a benchmarking of the interoperability of ontology development tools using RDF(S) as interchange language.

This paper presents the RDF(S) interoperability benchmarking that is currently in progress, the benchmark suites that are being used in it and the participation of Protégé in it, as it is one of the leading ontology development tools.

2 Interoperability benchmarking

The benchmarking of the interoperability of ontology development tools using RDF(S) as interchange language follows the benchmarking methodology for ontology tools developed in Knowledge Web [1]. Participation in the benchmarking

[1] http://knowledgeweb.semanticweb.org/

is open to any organization, and all the relevant information about it is available in a public web page[2].

The interoperability of ontology development tools using RDF(S) for ontology interchange requires that the importers and exporters from/to RDF(S) of the tools work accurately in order to interchange ontologies correctly. Therefore, the experimentation included three consecutive stages:

– **Agreement stage**. The first step is to agree on the definition of the benchmark suites, which will be common for all the tools, as the quality of these benchmark suites is essential for the benchmarking results.
– **Evaluation stage 1**. The RDF(S) importers and exporters of the ontology development tools are evaluated with the agreed versions of the benchmark suites.
– **Evaluation stage 2**. The second evaluation stage covers the evaluation of the ontology interchange between ontology development tools.

Seven tools are currently participating in the benchmarking, of these four are ontology development tools: KAON, OntoStudio, Protégé using its RDF backend, and WebODE; and three are RDF repositories: Corese, Jena and Sesame.

When writing this paper, the benchmarking participants are in the *Evaluation stage 2*. By the beginning of June 2006, the experimentation will be finished, and the results obtained will be available in the benchmarking web page.

3 Benchmark Suites

The benchmark suites used in the benchmarking are composed of benchmarks that import, export or interchange an ontology that models a simple combination of knowledge model components (classes, properties, instances, etc.). The process followed for defining these benchmark suites can be found in [2].

The **RDF(S) Import Benchmark Suite** is used to perform an exhaustive evaluation of the RDF(S) import capabilities of ontology development tools. It contains 82 benchmarks and has been built regarding the components of the RDF(S) knowledge model (classes and class hierarchies, properties, instances, and literals) and the combinations that can be obtained with these components.

The **RDF(S) Export Benchmark Suite** is used to perform an evaluation of the RDF(S) export capabilities of ontology development tools. It contains 66 benchmarks and has been built regarding the common components of the knowledge model of ontology development tools (classes and class hierarchies, datatype and object properties, instances, and literals) and the combinations that can be obtained with these components.

The **RDF(S) Interoperability Benchmark Suite** is used to evaluate the interoperability of ontology development tools by testing the interchange of ontologies from one origin tool to a destination one, and vice versa. It contains 66 benchmarks and has been built regarding the common components of the

[2] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/

knowledge model of ontology development tools and the combinations that can be obtained with these components. As these components are the same as those in the RDF(S) Export Benchmark Suite, the ontologies defined in these two benchmark suites are the same.

4 Protégé results

The interoperability benchmarking described in this paper is now taking place and the authors of this paper are carrying out the experimentation with Protégé.

The results for the Protégé ontology editor using its RDF backend that are available at present are the raw results of the evaluation of its RDF(S) importers and exporters[3]. By June 2006, once the benchmarking has finished, we will get public results with detailed information about the current interoperability of Protégé, including:

- Analysis of the evaluation of Protégé's RDF(S) importer and exporter.
- Analysis of the evaluation of Protégé's interoperability with the other participant tools using RDF(S) as interchange language.
- Recommendations for Protégé users on practices and best practices for achieving interoperability.
- Recommendations for Protégé developers for improving the interoperability of Protégé.
- Recommendations for tool developers on developing interoperable systems.

We have also started to organise a benchmarking activity[4], similar to this, for benchmarking the interoperability of ontology development tools using OWL as the interchange language.

Acknowledgments

This work is partially supported by a FPI grant from the Spanish Ministry of Education (BES-2005-8024), by the IST project Knowledge Web (IST-2004-507482) and by the CICYT project Infraestructura tecnológica de servicios semánticos para la web semántica (TIN2004-02660).

References

1. García-Castro, R., Maynard, D., Wache, H., Foxvog, D., González-Cabero, R.: D2.1.4 specification of a methodology, general criteria and benchmark suites for benchmarking ontology tools. Technical report, Knowledge Web (2004)
2. García-Castro, R., Gómez-Pérez, A.: A method for performing an exhaustive evaluation of RDF(S) importers. In: Proceedings of the Workshop on Scalable Semantic Web Knowledge Based Systems (SSWS2005). Number 3807 in LNCS, New York, USA, Springer-Verlag (2005) 199–206

[3] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/evaluation_stage_1_results.html
[4] http://knowledgeweb.semanticweb.org/benchmarking_interoperability/owl/

Semantic Versioning Manager: Integrating SemVersion in Protégé

Tudor Groza
DERI, National University of Ireland,
Galway, Ireland
www.deri.org
tudor.groza@deri.org

Max Völkel
FZI, Forschungszentrum Informatik,
Karlsruhe, Germany
www.fzi.de
voelkel@fzi.de

Siegfried Handschuh
DERI, National University of Ireland,
Galway, Ireland
www.deri.org
siegfried.handschuh@deri.org

ABSTRACT

Knowledge domains and their semantic representations via ontologies are typically subject to change in practical applications. Additionally, engineering of ontologies often takes place in distributed settings where multiple independent users interact. Therefore, change management for ontologies becomes a crucial aspect for any kind of ontology management environment. We introduce a new RDF-centric versioning approach and an implementation called SemVersion integrated as the Semantic Versioning Manager plug-in in Protégé. SemVersion provides structural and semantic versioning for RDF models and RDF-based ontology languages like RDFS.

INTRODUCTION

For many practical applications, ontologies (cf. Staab and Studer 2004) can not be seen as static entities, they rather change over time. Support for change management is crucial to support uncontrolled, decentralized and distributed engineering of ontologies. First approaches have been described in (Klein 2004 and Stojanovic 2004). But, there is no tool that functions as a standard versioning system for ontologies like CVS does in the field of software development.

We introduce an RDF-based approach that provides versioning for RDF models and RDF-based ontology languages like RDFS, OWL flavors or TRIPLE (Sintek and Decker 2002). We present a working methodology accompanied by its implementation in the system SemVersion. We then integrate it in Protégé as a tab plug-in, the Semantic Versioning Manager Tab, in order to provide to the end-users a natural way of editing and versioning their ontologies.

Our approach is inspired by the classical CVS system for version management of textual documents (e.g. Java code). The core element of our approach is the separation of language-specific features (the semantic diff) from general features, such as structural diff, branch and merge, management of projects and metadata.

A first survey on causes and consequences of changes in an ontology is presented in (Klein and Fensel 2001), followed by OntoView, an implementation for ontology versioning (Klein and Fensel 2002) that is based on the comparison of two ontology versions in order to detect changes. Basically, the system compares ontological classes, displays them side-by-side in RDF/XML and leaves it to the user to state “identical” or “conceptual change”.

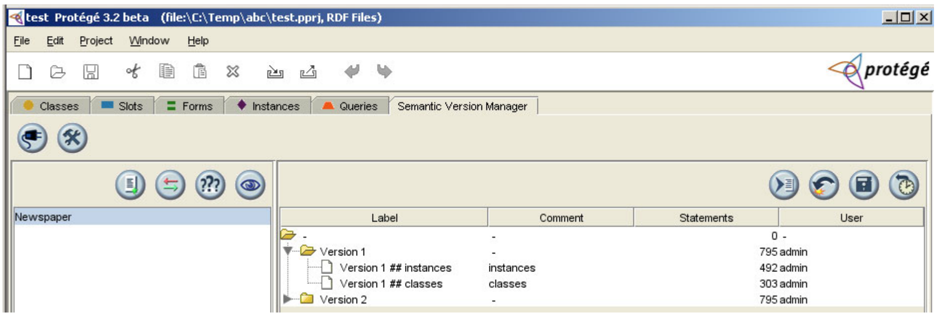
ISOCO KPOntology[1] is another example of a library, somehow similar to our system. It provides a high level API for managing ontologies, with support for multiple different triple stores. While KPOntology focuses on the ontology management, SemVersion’s primary focus is on versioning, the management aspects being transparent to the end user.

SEMANTIC VERSIONING MANAGER TAB

The Semantic Versioning Manager Tab (SVM Tab) is our solution for integrating SemVersion in the most natural environment for creating and editing ontologies, i.e. Protégé. The Ontology Lifecycle has 4 phases:

[1] http://kpontology.isoco.com/

Creation/generation, versioning, evaluation/visualization and negotiation. The goal of this integration is to add a plus of functionality in order to provide to the end-user within Protégé, the package of functions mapping to three of the four phases of the ontology lifecycle.



Label	Comment	Statements	User
-	-	0	-
Version 1	-	795	admin
Version 1 ## instances	instances	492	admin
Version 1 ## classes	classes	303	admin
Version 2	-	795	admin

Fig. 1. The Semantic Versioning Manager Tab in Protégé

As core functionalities, the versioning manager tab offers the possibility to:

- create new versioned models (an ontology under version control)
- add versions to a particular versioned model from:
    - the current edited ontology
    - external sources – now it accepts only local files, but in the future, it will accept also URIs
- export a particular version as a file (RDF syntax)
- load a particular version as the current editable knowledge base

Because SemVersion is an RDF-based versioning system, the provided diff operations have to be considered from the RDF semantics point of view. The structural diff represents the set-theoretic difference of two RDF triple sets. Following, we will describe how the semantic diff is achieved: consider two models A and B that are versions of the same RDF Schema model. In order to compute the semantic diff, we use RDF Schema entailment on model A and infer all triples we can (*Inf*(A)). We apply the same operation for model B (*Inf*(B)). Then we calculate a structural diff on *Inf*(A) and *Inf*(B). To summarize, a way to compute a semantic diff is thus to materialize the complete entailment (transitive closure) of two RDFS models and then perform a structural diff on the transitive closure.

The current visualizations provided for the structural and semantic diffs are in terms of statements. Our goal is to offer a more intuitive visualization, for example, by displaying the two ontologies in parallel and create graphic connections to indicate the added and removed statements. Note that the Protégé Prompt Tab has a very clear presentation of the structural diff and represents a good example for us. We also intend to build a graphical visualization for the semantic diff, since the information provided by it, is more expressive and intuitive.

The Semantic Versioning Manager Tab has a graphical visualization for the structural diff, but only in terms of classes and subclasses between several versions. This functionality was created using the Aduna Cluster Map Library[2] and it is depicted in Fig. 2. The idea behind the visualization is to emphasize the differences between several versions by creating clusters of common subclasses of the same class in different versions of the same ontology. By checking more entities, the graph will grow, displaying all the selected classes together with their relationships and clusters of subclasses (an arrow in the graph represents a parent-child relationship).

In terms of implementation, we wanted as a first step to reuse the functionalities provided by Protégé for dealing with RDF ontologies. That is why, two of the most important functions of the SVM Tab (create version from current ontology and load version as current knowledge base) are realized by using the Protégé RDF Backend. Although in terms of reutilization, this was a good design decision, it also introduced a constraint which has to be taken into account by the end-user: the Protégé project on which the user is working has to have an RDF knowledge base. This limitation will be corrected as soon as we will implement our own stream-based RDF importing mechanism, part of the second step of development.

[2] http://aduna.biz/products/technology/clustermap/index.html

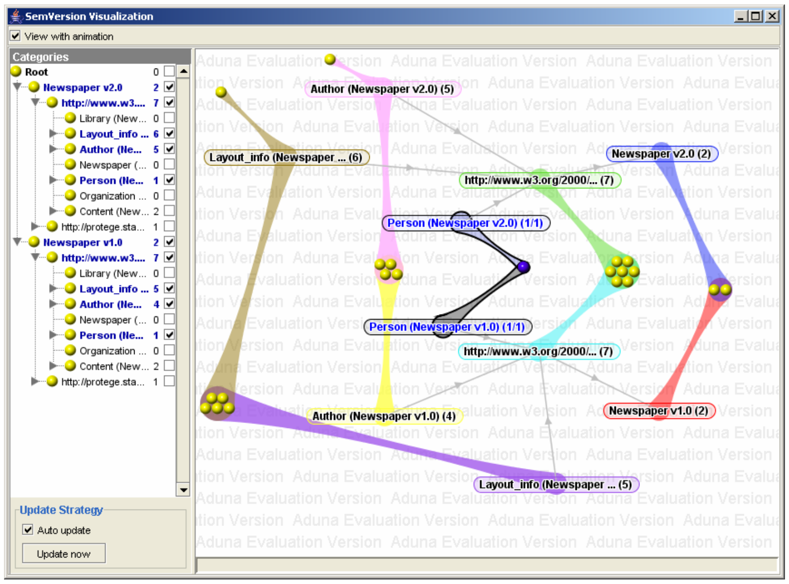


Fig. 2. Structural diff visualization in terms of classes and subclasses

SUMMARY

Versioning support for ontologies is crucial especially in dynamic environments. We presented here a methodology for RDF-based versioning that separates the management aspects from the versioning core functionality. By integrating it into Protégé, we wanted to provide the end user with the extra functionality needed to follow the normal ontology life-cycle. We intend in the future to correct the current limitations and to advance a step forward with the versioning system, first by providing collaborative versioning functionalities, like sharing versions between several users and afterwards by developing a joint negotiation mechanism for committing new versions.

ACKNOWLEDGEMENT

Part of this work has been funded by the European Commission 6th Framework Programme in the context of the Knowledge Web - Network of Excellence project, FP6-507482 and the EU IST NEPOMUK IP – The Social Semantic Desktop, FP6-027705.

REFERENCES

Klein, M., Fensel, D. 2001. Ontology versioning for the semantic web. In: *1st Int Semantic Web Working Symp. (SWWS)*, Stanford, CA, USA. pp. 75–91

Klein, M., Fensel, D. 2002. *OntoView: web-based ontology versioning*. Technical report, Vrije Universiteit Amsterdam. Submitted, draft at http://www.cs.vu.nl/˜mcaklein/papers/ontoview.pdf

Sintek, M., Decker, S. 2002. Triple - a query, inference, and transformation language for the semantic web. In: *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, London, UK, Springer-Verlag. pp. 364–378

Staab, S., Studer, R., eds. 2004. Handbook on Ontologies in Information Systems. *Int Handbooks on Information Systems*. Springer

Stojanovic, L. 2004. *Methods and Tools for Ontology Evolution*. PhD Thesis, University of Karlsruhe.

Franz Inc. is the world's leading vendor of dynamic object-oriented development tools featuring Allegro Common Lisp. Since 2004, Franz has been heavily engaged in the Semantic Web through RacerPro and AllegroGraph. Racer is a very familiar tool for Protege users. AllegroGraph is a new product that we will demo at the International Protégé Conference this year. AllegroGraph is a highly scalable, persistent triple store for Semantic Web applications that need to work with billions of triples. The triple store loads, stores and queries uniquely fast. Load time surpasses 10,000 triples per second all the way to a billion triples. Retrieval times for (sequences of) triples on disk are measured in single digit microseconds. What might be very interesting to Protege users is that we are working on deep integration with RacerPro.

```
Steve Sears                        Internet: ssears@franz.com
Product Marketing Manager        http: // _www.franz.com/ < http://www.franz.com/>_ <
http://www.franz.com/ >
555 12th St #1450,                Phone: (510) 452-2000, ext. 154
Oakland, CA 94607                 FAX: (510) 452 0182
```

ENRICHMENT OF OBO ONTOLOGIES

Michael Bada, Ph.D. and Lawrence Hunter, Ph.D.
University of Colorado at Denver and Health Sciences Center, Aurora, CO, USA

An enormous effort has gone into the creation and maintenance of the Open Biomedical Ontologies (OBO) project, and this has been met with the ever-increasing use of the constituent ontologies by biological researchers. There are currently over 50 OBO ontologies, ranging over such domains as anatomy, behavior, phenotype, experiment, and sequence. The flagship OBO ontology, the Gene Ontology (GO), with its three subontologies detailing molecular functions, biological processes, and cellular components, in particular has experienced phenomenal growth in terms of numbers of terms and also of its extensive use by annotators to describe gene and gene-product entries in a number of prominent model-organism databases.

Although many of these ontologies are large, they are structurally quite simple, typically consisting of only a few relationship types apart from the fundamental is-a relationship that forms the backbone of an ontology. Furthermore, there are no links between terms from separate ontologies, even between terms from different GO subontologies. However, most concepts do have relationships with other concepts, as evidenced by their natural-language names and definitions, and computational agents cannot take advantage of these relationships if they are not formally represented. The work presented in this demo expands on previous efforts that have taken advantage of the compositionality of GO terms to produce more formal definitions of GO terms in the form of added relationships with other ontological terms.

In this demo will be shown a frame-based integration of the three GO subontologies, the Chemicals of Biological Interest ontology (ChEBI), and the Cell Type Ontology in which relationships between elements of the ontologies are modeled in a way that better captures the relational semantics between biological concepts represented by the terms, rather than between the terms themselves, than previous frame-based efforts. A methodology for creating suggested enriching assertions of the form [subject, relationship, object] by identifying patterns in GO terms, correlating these patterns and subpatterns with relationships, matching concepts to these patterns and subpatterns, and integrating these assertions into the ontologies will be described. Using this methodology, a large number of new, reliable assertions linking ontological terms using a variety of specific, hierarchically arranged relationship were created: A predicted assertion was made for 62% of GO terms that matched one of 31 patterns, and 97% of these predicted assertions were assessed to be valid; a further 429 assertions (corresponding to 6% of the matching terms) were manually created, resulting in an initial set of 4,497 assertions. Furthermore, this methodology programmatically integrates assertions into the base ontology such that each assertion is fully consistent with respect to higher (*i.e.*, more general) relevant class and slot levels.

OntoSphere3D: a multidimensional visualization tool for ontologies

Alessio Bosca, Dario Bonino
Politecnico di Torino, Torino, Italy
{alessio.bosca, dario.bonino}@polito.it

Introduction

Visualizing knowledge in two dimensions is a challenging task, since many dimensions are involved: instances, concepts, hierarchical relations just for naming a few. And it is also a well studied paradigm that already produced good solutions such as: Jambalaya [1, 2], GraphViz [3], OntoViz [4], etc. Nevertheless, mapping the many dimensions involved by an ontology, on only two dimensions can sometimes be too restrictive, especially in the case of very large and complex knowledge domains.

In this presentation, we propose a novel approach for inspecting and editing ontologies using a visually enriched 3-dimensional space. Ontology information is represented on a 3D view-port merging structural information (i.e. concepts and relations) with context information (the amplitude of *Is-A* hierarchies rooted in a given ontology concept,…) by means of visual cues. Being the 3-dimensional view quite natural for humans, especially for what concerns navigation, the proposed approach aims at being more effective in browsing ontological data than the currently available 2-dimensional solutions. Improvements are obtained by involving direct manipulation operations such as zooming, rotating, and translating objects, and by introducing more dimensions to convey information on the visualized knowledge model as the color or the size of visualized entities.

The proposed work aims also at tackling space allocation issues for ontology visual models, in fact, in the traditional solutions, big ontologies can easily lead to overcrowded representations that are difficult to browse and that can be more confusing than aiding. Some attempts exist to overcome these problems, as in OntoRama [5,6], where the nodes being inspected are magnified with respect to the other nodes in the ontology. However, even these approaches tend to collapse when visualizing big ontologies such as SUMO [7], counting over than 5'000 concepts. The proposed application, instead adopts a dynamic collapsing mechanism and different views, at different granularities, for granting a constant navigability of the rendered model.

Proposed Approach

A three-dimensional environment is the starting point of the proposed ontology visualization tool, as a 3D space offers one more dimension than traditional 2D approaches to represent ontology data, so simplifying its interpretation. In addition many more dimensions are added to improve completeness and readability of the representation. Two main principles guide the visualization effort: increasing the number of “dimensions” (colors, shapes, transparency, etc.) which represent concepts features and convey additional information without adding the burden of further graphical elements (such as labels) on the scene, and automatically identifying the part of knowledge to be displayed and the detail level to be used in the process, on the base of user interaction with the scene. The latter principle is particularly important for improving the overall system performances since scale factor indeed constitutes a strong issue in visualizing complex graph structures like ontologies. As the cardinality of elements increases, the number of items to be concurrently displayed on the screen worsens the graphical perception of the scene and complicates spotting details. When the amount of visualization space needed to represent all the information within the KB outnumbers the space available on the screen, a few options remain available: to scale down the whole image to the detriment of readability, to present on the screen just a portion of it and allow its navigation or to summarize the information in a condensed graph and provide means for exploration and expansion. As the effectiveness of these options depends on the use case involved (consistency checking, domain comprehension, KB updates) a combined usage of them offers a suitable approach.

To combine seamlessly the options above, taking advantage of their strength points whenever possible, the proposed solution exploits different scenes that present and organize the information on the screen according to differently detailed perspectives. Such scenes interchange in managing the graphical space as user attention shifts from one concept to another, by implicitly inferring the focus from user’s interaction with the scene (e.g., a concept selection with a mouse click). In this way, the idea of “focusing” as the application capability of highlighting the elements of interest while leaving out the others, is applied.

The OntoSphere3D [8] user interface is rather minimalist and allows direct manipulation of scenes through rotation, panning and zoom; it allows to browse the ontology as well as to update it and to add new concepts and relations (taking advantage of functionalities provided by the Protégé framework in which is deployed). Every concept within a given scene is clickable with two different results: a single left-click maintains the current perspective and simply navigates through elements, while a double-click leads to a focusing operation, shifting the scene to a more detailed level. Right-clicking on a concept that has direct instances visualizes them, while the same action applied to an instance bring the focus back on the related concept (see section 3.3) .

Root Focus Scene

This perspective presents a big “earth-like” sphere bearing on its surface a collection of concepts represented as small spheres (Figure 1). The scene does not visualize any taxonomic information and only shows direct “semantic” relationships between elements of the scene, usually a graph not fully connected. Atomic nodes, the ones without any subclass, are smaller and depicted in blue while the others are white and their size is proportional to the number of elements contained in their own sub-tree.

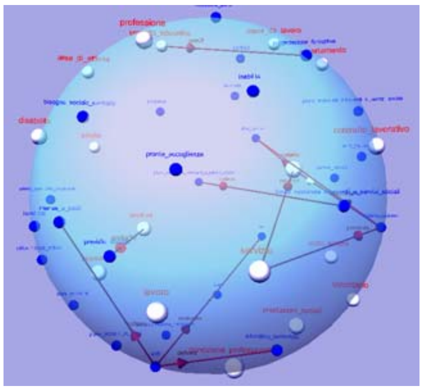


Figure 1. The Root Focus scene.

This view is particularly intended for representing the primitive concepts (i.e., the roots), but can also be used, during the ontology navigation, to visualize direct children of a given node; a pretty useful option in case of heavily subclassed concepts. Representing primitive concepts within the ontology, and the relations between them, allows to easily identify the conceptual boundaries of the represented domain and provides a very good hint to the question: “what’s the ontology about?”

Tree Focus Scene

This scene shows the sub-tree originating from a concept; it displays the Is-A hierarchy as well as other semantic relations between represented classes. Since experimental evidence proves that too many elements on the screen, at the same time, hinder user attention, the scene completely presents only three fully-expanded levels at a time. As the user browses the tree, the system automatically performs expansion and collapse operations in order to maintain a reasonable scene complexity (Figure 2). Collapsed elements are coloured in white and their size is proportional to the number of elements present in their sub-tree; instead concepts located at the same depth level within the tree have the same colour in order to easily spot groups of siblings. *Is-A* relations are displayed with a neutral colour (grey), without labels, whereas other semantic relations involving concepts already in the scene are displayed in red, and are accompanied by the name of the relation. When an element of the scene is related to a node that is not present on the view-port, a small sphere is added for the hidden node in the proximity of the given element, so terminating the end of the arrow; in such cases, incoming relations are represented with a green arrow, while outgoing links with a red one.

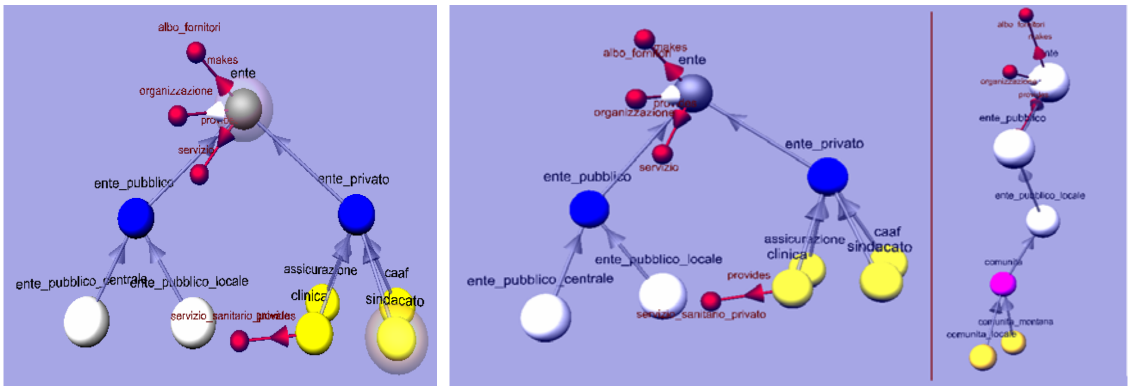


Figure 2. The Tree Focus scene.

Concept Focus Scene

In the concept focus scene, all the available information about a single concept is provided, at the highest possible level of detail. The concept’s children and parent(s) are therefore shown as well as its ancestor root(s) and its semantic relations, including the inherited ones. Semantic relations are drawn as arrows terminating in a small sphere: red if the relation is outgoing and green otherwise. Direct relations are drawn close to the concept, with an opaque color, while inherited ones are located a bit farther from the center and depicted with a fairly transparent color.

This scene can be extremely useful during consistency checking operations because it eases the spotting of inconsistent concepts or relations, e.g. whenever a concept inherits from an ancestor a property that "logically" contrasts with other features of its own

Instance Focus Scene

Whenever a concept has direct instances (in the tree focus scene or in the concept focus scene) its sphere is depicted surrounded by a transparent sphere resembling a sort of a shell (Figure 3). By right-clicking the concept, its direct instances are shown, using the same representation paradigm adopted by the concept focus scene. The resulting view represents instances and their properties as inter-connected cubes.

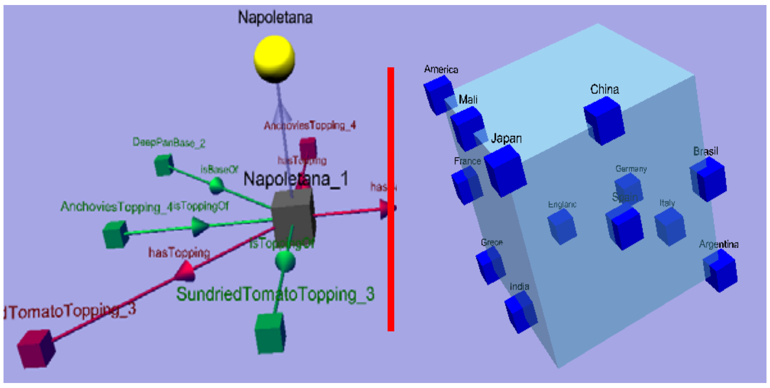


Figure 3 Instance Focus Scene

Results

In order to confirm the initial claim of the proposed work (i.e. improved navigability and inspection capabilities with respect to 2-dimensional approaches) the authors set up an efficiency comparison between 4 different visualization tools: the proposed OntoSphere3D plug-in, the Jambalaya plug-in, the OWLViz plug-in and the TgViz plug-in. These plug-ins have been tested against a predefined set of ontology related operations, namely: visualization of the top concepts, visualization of the relations between the top concepts, visualization of concepts located at level *n* in the *Is-A* hierarchy of the ontology, visualization of the concepts related to a given one, visualization of relations between concepts at the same hierarchy level, navigation of the ontology from one concept to another, search for a given concept.

The required Protégé-related skills have also been taken into account in the evaluation. Each operation has been assigned a predefined difficulty score, as reported in Table 1. Evaluation results are, instead, reported in
Table 2.

Table 1 Difficulty scores for several user interactions.

User interaction	Difficulty score
Mouse click	2
Mouse double-click	2
Look at the screen	0
Mouse over	1
Mouse scroll	3
Search (filling a form)	4

Table 2. Results of efficiency evaluation on 4 different visualization plug-ins.

Operation	Jamb.ya	On.3D	OWLViz	TViz
User Experience	1	1	2	2
Top concepts	0	0	3	6
Relations between top concepts	1	0	isA = 0 not-isA = ∞	1
Level–n concepts	n	2n	n.d.	n.d.
Related concepts	4	0	isA = 0 not-isA= 7	0
Relations between concepts at the same level	1	0	isA=0 not-isA=7	1
Concept navigation	2	2	0-3	0-3
Search	10	6	4	6
Jamb.ya = Jambalaya, On.3D = OntoSphere3D, TViz=TgViz				

It is easy to notice that, in most cases the proposed approach outperforms the other applications, except for visualizing concepts at a given level *n* in the ontology hierarchy, for which *n* mouse clicks are required, and for searching concepts, where the offered functionality is the one of the Protégé framework as for TgViz. In concept navigation however, data is quite difficult to compare since both Jambalaya, OWLViz and TgViz require scrolling for navigating between ontology concepts. According to the evaluation grid in Table 1, this is not a too heavy task but, when the ontology size grows up from few tens of concepts to several thousands, the required scrolling may become much more cumbersome and thus shall probably be re-weighted. On the contrary, the OntoSphere3D behavior is size-independent, becoming more suitable on really big ontologies such as SUMO.

References

[1] M.A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In Workshop on Interactive Tools for Knowledge Capture, Victoria, B.C. Canada, October 2001.

[2] Jambalaya. http://www.thechiselgroup.org/chisel/projects/jambalaya/jambalaya.html.

[3] Gansner, E. R. & North, S. C. (1999), An open graph visualization system and its applications to software engineering, Software Practice and Experience 30(11),.

[4] OntoViz Tab: Visualizing Protégé Ontologies. http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

[5] P.W.Eklund, N.Roberts, S.P.Green, OntoRama: Browsing an RDF Ontology using a Hyperbolic-like Browser, *The First International Symposium on CyberWorlds (CW2002)*, pp.405-411, Theory and Practices, IEEE press, 2002

[6] OntoRama. http://www.ontorama.com/

[7] Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.

[8] http://sourceforge.net/projects/ontosphere3D

Semantic Data Preparation: The Instance-Selection plug-in

Paulo Gottgtroy
Knowledge Engineering and Discovery Research Institute
School of Computer and Information Sciences
Auckland University of Technology
Private Bag 92006 Auckland 1020, New Zealand
E-mail: paulo.gottgtroy@aut.ac.nz

Abstract— Computer scientists have been working on data preparation techniques in order to improve the quality of the KDD results. In largely parallel research, ontologies have been widely used by the Artificial Intelligence community to represent domain knowledge and to integrate different database models. This work investigates the application of ontologies in the data preparation step of the KDD process. We present an ontology instance selection tool able to export an ontological representation into specific formats used by different data mining workbenches.

Introduction

Knowledge Discovery in Databases (KDD) is an iterative process based on the analysis of current facts or data, pre-processing to clean and transform that data, application of mining algorithms, and deployment using the mining results on new data. Computer scientists have been working on cleaning data, data transformations, selection of samples and - in case of large data sets - performing feature selection operations to find relevant variables to the problem in order to improve the quality of data processes. Their objectives include: building simpler and more comprehensible models, improving data mining performance, and helping to prepare, clean, and understand data. In largely parallel research, ontologies have been widely used by the Artificial Intelligence community to represent domain knowledge and to integrate different database models.

A common background underlying database and ontology research is well known. Although ontologists and data modelers have been working together to bridge both areas, for example, in topics like conceptual modeling, database integration and metadata representation, less work has been undertaken in relation to feature selection in large and multi-dimension spaces. Our approach explores this gap and presents an ontology feature selection tool able to translate its representation into a tabular format. This tabular format can further be exported to different data mining workbenches or data formats.

The Instance Selection plug-in is part of a set of tools which supports the Ontology Driven Knowledge Discovery (ODKD) [1]. ODKD investigates a hybrid approach bringing together the state of art of artificial intelligence methods for knowledge discovery in large databases (KDD) and ontology engineering. ODKD analyses how data mining can assist in efficient and effective large-volume data analysis in order to build a sharable and evolving knowledge repository, while at the same time leveraging the semantic content of ontologies to give intelligent support and to improve knowledge discovery in complex and dynamic domains.

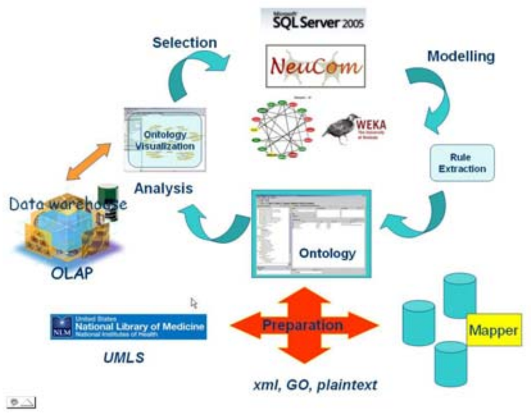


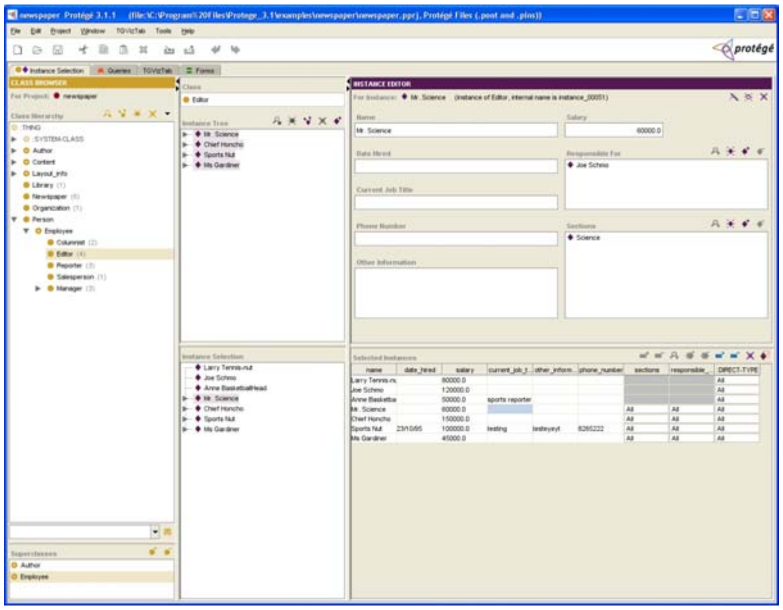
Figure 1.OKDD process.

The semantic data preparation phase of ODKD encompasses: ontology preparation, ontology analysis and instance selection - figure 1. It covers the first three steps of CRoss-Industry Standard Process for Data Mining process (CRISP-DM) [2] - business understanding, data understanding and data preparation - creating an alternative ontology driven pipeline for the data mining process.

The Instance Selection Tool supports the third step of CRISP-DM (data preparation) by converting ontology concepts into a simple tabular representation which allow further concept analysis by data mining workbenches, business intelligence tools, data visualization techniques and so on.

Instance Selection Tab

The Instance Selection Tab (figure 2) extends the instance tree tab [4] by adding a panel with a table able to store instances selected from the knowledge base through the instance tree, from results of a query and from the selection of instances in a visualization tool. We have also developed additional features to the query tab [5] and instance browser panel which allow us, for example, to select instances from the TGViz Tab [6].



Selected Instances

name	date_hired	salary	current_job_t	other_inform	phone_number	sections	responsible_	DIRECT-TYPE
Larry Tennis-n		80000.0						All
Joe Schmo		120000.0						All
Anne Basketba		50000.0	sports reporter					All
Mr. Science		60000.0				All	All	All
Chief Honcho		150000.0				All	All	All
Sports Nut	23/10/95	100000.0	testing	testeyeyt	6265222	All	All	All
Ms Gardiner		45000.0				All	All	All

Figure 2 – Instance tree tab.

After the selection of instances, the Tab allows a full manipulation of instances and its slots defining among others: which slots should be exported, select the values of multiple value slots, select a template including the internal name (:name) and/or display slot, etc – figure 3.

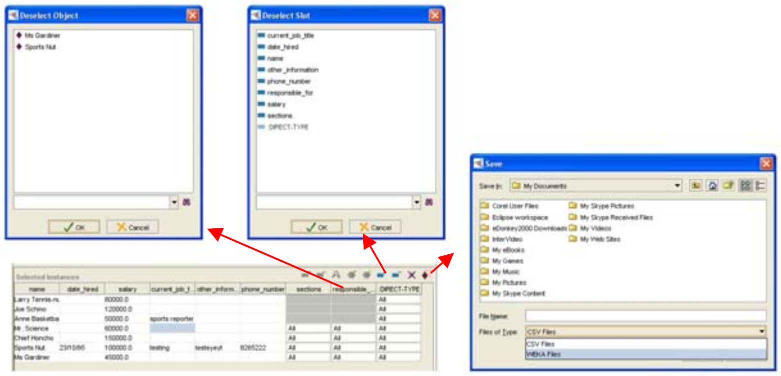


Figure 3 – Examples of instance manipulation options.

It also allows a recursive manipulation of slot type instance defining which slots of the instances should be exported for further analysis. This feature enables a full investigation of the relationships among concepts, for example, in a bioinformatics problem a user can select the genes responsible for a disease and further cluster them by the molecular function, gene expression in order to test new/different hypothesis without any extra data transformations

We believe that the Instance Selection Tab can help in any scenario where there is a need for extract/manipulate/export instances from a knowledge base without programmatically access the Protégé API.

Using a Degree-of-Interest Model for Adaptive Visualizations in Protégé

Tricia d'Entremont Margaret-Anne Storey

Department of Computer Science
University of Victoria
Victoria, BC Canada
http://www.thechiselgroup.org
email: tdent@uvic.ca; mstorey@uvic.ca

Abstract

Visualizations are commonly used as a cognitive aid for presenting large ontologies and instance data. One challenge with these visual techniques is that the generated views are often very dense and complex. It is difficult to know which concepts to include in the visualization to meet a user's information needs. In this talk, we present recent work that proposes using an attention-reactive interface to provide adaptive visualizations in Protégé. This furthers our recent work in providing visualization "on demand" for maintenance, editing, and understanding tasks by drawing users' attention to concepts of interest within the context of the current task.

1. Introduction

Understanding and maintaining the structure of large ontologies is a cognitively demanding task. Over the last several years the CHISEL group at the University of Victoria has been working on developing advanced visual interfaces to help users browse and understand large, complex ontologies.

Our main ontology visualization tool, Jambalaya, is an integration of the SHriMP[1] (Simple Hierarchical Multi-Perspective) visualization toolkit with Protégé [1]. In Jambalaya, the ontology is represented as a graph where classes and instances are depicted as nodes, and relationships between the classes are represented as directed arcs. Jambalaya provides multiple, inter-changeable views of the graph structure allowing users to explore multiple perspectives of information at different levels of abstraction.

As the size and complexity of the ontology grows, however, the usefulness of Jambalaya's advanced visualizations and Protégé's standard views decreases. Users report that they often work for extended periods of time on a small subset of the ontology. The concepts relevant to their task become difficult to locate because non-interesting concepts consume valuable screen real estate, obscuring the interesting concepts as illustrated in Figure 1.

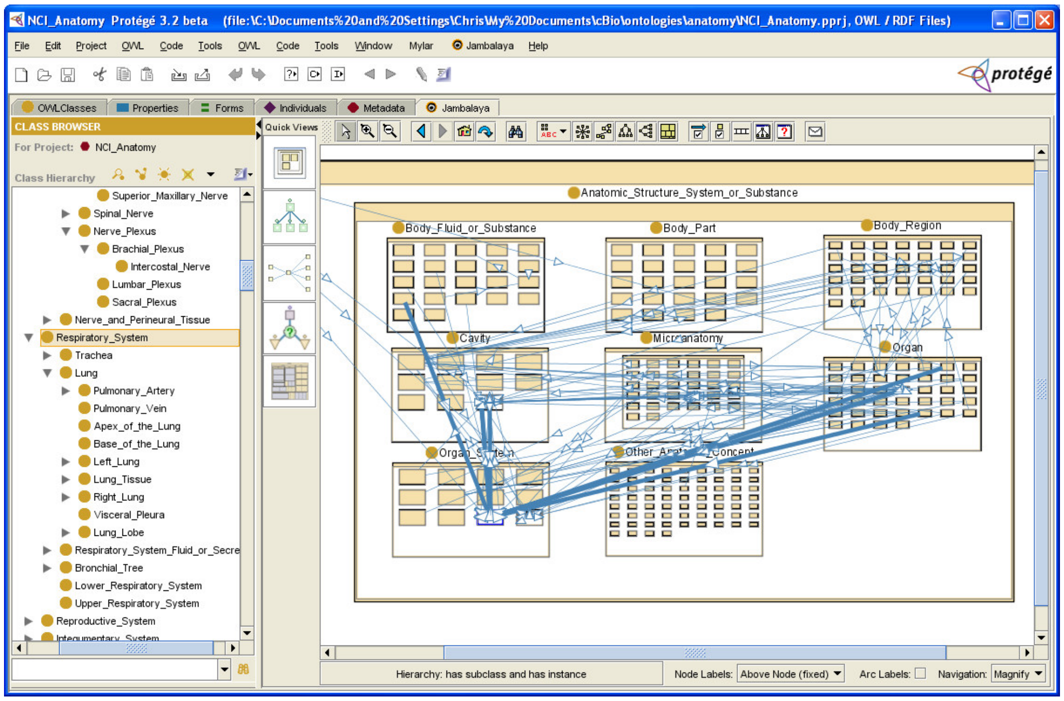
Within Protégé's class browser, for example, the concepts of interest given a particular task may not be visible. Users are therefore forced to spend considerable effort navigating the ontology by scrolling, expanding and collapsing nodes in order to find concepts of interest.

2. Approach

To address this problem and to help users find concepts of interest within the ontology, we are developing a plug-in for Protégé which applies principles of attention-reactive user interfaces to provide adaptive visualizations within Protégé

Attention-reactive interfaces consist of two components: a mechanism to continuously calculate the user's degree of interest (DOI) and a dynamic display of the information that uses the DOI calculation to draw users' attention to interesting elements in order to reduce navigation overhead [2].

1. http://www.thechiselgroup.org/shrimp



Hierarchy: has subclass and has instance
Node Labels: Above Node (fixed)
Arc Labels:
Navigation: Magnify

Figure 1: Protégé with the Jambalaya Tab selected displaying a portion of the NCI Thesaurus

The approach will be applied to both the standard tree browser views in Protégé as well as to the graphical based views in Jambalaya.

To calculate the user's DOI within the ontology, we have adapted Mylar[2], a plug-in for the Eclipse[3] integrated software development environment to work with Protégé [3]. Within Eclipse, Mylar monitors the programmer's activities, computes the users' DOI of the various code entities, and adapts the respective Eclipse views using the DOI calculation. To integrate Mylar with Protégé, we have written our own monitor to track user activity within the ontology and to pass that information to Mylar.

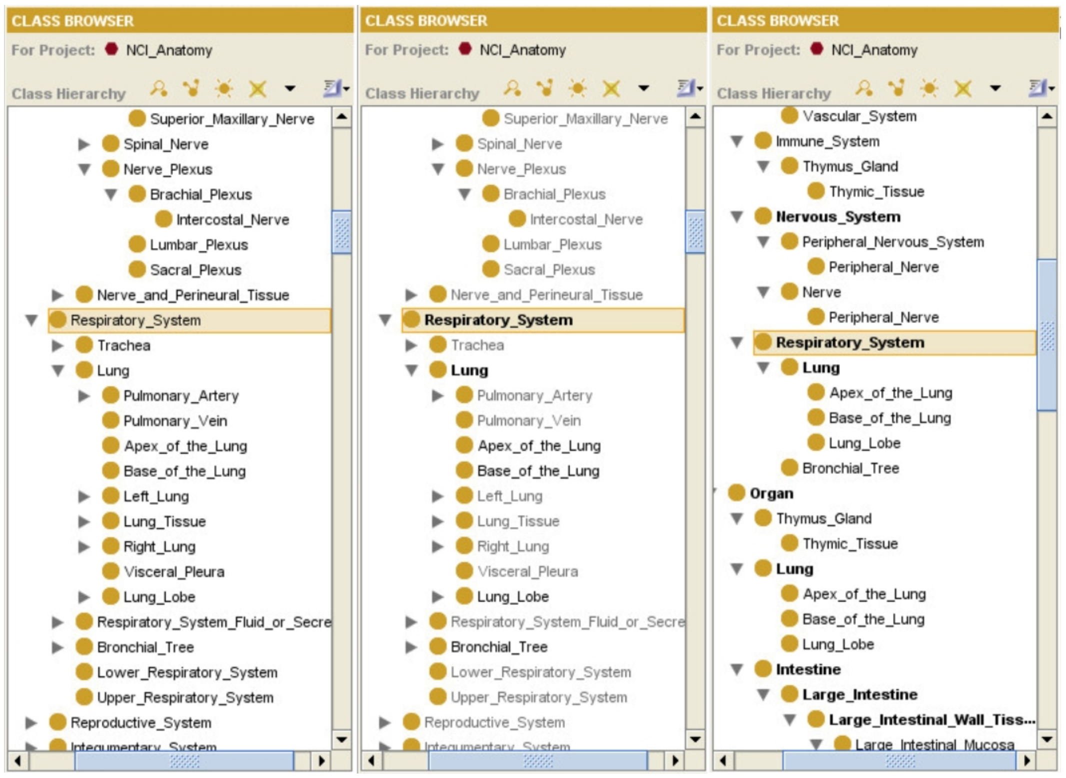
Mylar's DOI model associates an interest value with each concept in the ontology. When a concept is selected or edited, its DOI value increases. The DOI calculation also contains a *decay* function which decreases an element's interest value if it has not been selected.

We use the DOI calculation to adapt the appropriate views in Protégé to reduce navigational overhead and to draw user's attention to concepts of interest within the context of the current task. A primary consideration in the design of these adapted views has been to provide lightweight, easily reversible mechanisms to focus user's attention without deviating significantly from the existing, familiar Protégé views.

3. Adaptive Visualizations

Within Protégé's class, owl, and instance browsers, a user's DOI over the concepts is visualized using font weight and font color as shown in Figure 2b. Non-interesting concepts, ones that the user has not selected or whose DOI value has decayed to zero, are displayed in gray. *Landmark* concepts, those with a high DOI calculation or which the user has manually specified to be a landmark, are highlighted in black, bold text. Interesting concepts, those that have been selected but whose calculated DOI value falls below a threshold value are shown in black. To provide for finer granularity, we are also exploring the use of font size as a mechanism for highlighting a user's DOI.

2. http://www.eclipse.org/mylar/
3. http:// www.eclipse.org



```
CLASS BROWSER
For Project: ● NCI_Anatomy
Class Hierarchy
        Superior_Maxillary_Nerve
      ▶ Spinal_Nerve
      ▼ Nerve_Plexus
        ▼ Brachial_Plexus
            Intercostal_Nerve
          Lumbar_Plexus
          Sacral_Plexus
    ▶ Nerve_and_Perineural_Tissue
  ▼ Respiratory_System
    ▶ Trachea
    ▼ Lung
      ▶ Pulmonary_Artery
        Pulmonary_Vein
        Apex_of_the_Lung
        Base_of_the_Lung
      ▶ Left_Lung
      ▶ Lung_Tissue
      ▶ Right_Lung
        Visceral_Pleura
      ▶ Lung_Lobe
    ▶ Respiratory_System_Fluid_or_Secre
    ▶ Bronchial_Tree
      Lower_Respiratory_System
      Upper_Respiratory_System
  ▶ Reproductive_System
  ▶ Integumentary_System
```

```
CLASS BROWSER
For Project: ● NCI_Anatomy
Class Hierarchy
        Superior_Maxillary_Nerve
      ▶ Spinal_Nerve
      ▼ Nerve_Plexus
        ▼ Brachial_Plexus
            Intercostal_Nerve
          Lumbar_Plexus
          Sacral_Plexus
    ▶ Nerve_and_Perineural_Tissue
  ▼ Respiratory_System
    ▶ Trachea
    ▼ Lung
      ▶ Pulmonary_Artery
        Pulmonary_Vein
        Apex_of_the_Lung
        Base_of_the_Lung
      ▶ Left_Lung
      ▶ Lung_Tissue
      ▶ Right_Lung
        Visceral_Pleura
      ▶ Lung_Lobe
    ▶ Respiratory_System_Fluid_or_Secre
    ▶ Bronchial_Tree
      Lower_Respiratory_System
      Upper_Respiratory_System
  ▶ Reproductive_System
  ▶ Integumentary_System
```

```
CLASS BROWSER
For Project: ● NCI_Anatomy
Class Hierarchy
      Vascular_System
  ▼ Immune_System
    ▼ Thymus_Gland
        Thymic_Tissue
  ▼ Nervous_System
    ▼ Peripheral_Nervous_System
        Peripheral_Nerve
    ▼ Nerve
        Peripheral_Nerve
  ▼ Respiratory_System
    ▼ Lung
        Apex_of_the_Lung
        Base_of_the_Lung
        Lung_Lobe
      Bronchial_Tree
Organ
  ▼ Thymus_Gland
      Thymic_Tissue
  ▼ Lung
      Apex_of_the_Lung
      Base_of_the_Lung
      Lung_Lobe
  ▼ Intestine
    ▼ Large_Intestine
      ▼ Large_Intestinal_Wall_Tiss...
        ▼ Large_Intestinal_Mucosa
```

(a) (b) (c)

Figure 2: a) Standard Protégé class browser b) Highlighting concepts in the class browser. c) Highlighting and filtering concepts in the class browser.

In addition to displaying the user's DOI by highlighting the concepts in the different views, we provide a mechanism for filtering non-interesting concepts from the view as shown in Figure 2c. Although it is still possible for the view to display a vertical scrollbar, the number of items through which the user must search can be drastically reduced thereby decreasing the time spent finding relevant concepts. Users can also specify that a concept is no longer of interest, the associated DOI value is updated correspondingly, and, if filtering is enabled, the concept is removed from the view.

4. Ongoing and Future Work

As a first step toward investigating this approach, we are designing studies of users' interaction patterns within Protégé and Jambalaya. Throughout our work in this domain, we have been concerned with which tasks could benefit from visualization support and when visualization support should be provided [4]. These studies will provide important insight into these issues as well as a basis for preliminary evaluation of the impact the adapted views have on users' navigation.

Our proposed user studies will consist of two phases. During the first phase, users' interactions within the ontology will be monitored using the standard views provided by Protégé and Jambalaya. The second phase will involve monitoring user activity with the adapted views. The goal of these studies will not only be to evaluate and refine the approaches we have presented here but also to discover additional situations in which an attention-reactive interface may provide cognitive support.

This work is in its earliest stages, and it is very important to us to get feedback from the Protégé user community. During our presentation, we will lead a discussion on how these adapted visualizations may, or may not, be helpful.

Our future work will explore ways to adapt the advanced visualizations in Jambalaya given a user's DOI model. Beyond highlighting concepts using color or size, we will explore using motion to make elements above a certain degree of interest "pop out" from the graph using motion techniques [5].

In addition, we are interested in investigating the possibilities of sharing DOIs among users, for example, sharing an expert's DOI with a novice to provide guidance.

5. Acknowledgements

This work was supported by the National Center for Biomedical Ontology, under roadmap-initiative grant U54 HG004028 from the National Institutes of Health.

We would like to acknowledge the work of Mik Kersten and the rest of the Mylar project team.

Chris Callendar and other members of the CHISEL group have been instrumental in this work.

References

1. M. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R.Fergesen, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. *In Workshop on Interactive Tools for Knowledge Capture*, Victoria, B.C. Canada, October 2001.
2. S. Card and D. Nation. Degree-of-interest trees: A component of an attention-reactive user interface. In *International Conference on Advanced Visual Interfaces (AVI02)*, 2002.
3. M. Kersten and G. Murphy. Mylar: a degree-of-interest model for IDEs. In *Proceedings of the 4th international conference on Aspect oriented software development*, pages 159–168, New York, NY, USA, 2005. ACM Press.
4. N. Ernst, M. Storey, and P. Allen. Cognitive support for ontology modeling. *Int. J. Hum.-Comput. Stud.*, 62(5):553-577, May 2005.
5. C. and R. Bobrow. Motion to support rapid interactive queries on node-link diagrams. *ACM Transactions on Applied Perception*. 1: 1-15, 2004.

Design of an ontology / conceptual graph-based gross anatomy dissection knowledgebase for real-time table-side implementation in a first-year medical dissection laboratory

HISLEY, Kenneth C., Larry D. ANDERSON*, Stacy E SMITH*, Stephen KAVIC*, Paul G NAGY*. Departments of Radiology, Anatomy and Neurobiology, and Surgery, University of Maryland School of Medicine, Baltimore, MD.

Abstract
Our stated goal here is to provide dissection students with 24/7 access to expert knowledge, maximize the information's relevance and completeness, while simplifying its presentation. Expert assistance to students "at the dissection table" is required for them to learn the synthesis and application of mental models during problem solving, and usually is present in distilled form as a "dissector". Types of information in such dissectors include: 1) didactic (text descriptions), 2) spatial (diagrams, cadaver and radiological image sets) and 3) procedural (sequential technique). Unfortunately, timely access to sophisticated face-to-face faculty guidance above and beyond the dissector is limited. To achieve our goal, we are developing an online web-based gross anatomy dissection guide using PROTEGÉ, OWL, JAMBALAYA and XML software methods to visualize, capture and replay expert-level dissection knowledge and guidance. In our design, the ontology/conceptual graph of a given exploratory procedure in the human cadaver becomes a faculty member's "conceptual backbone" integrating multiple classes of teaching resources into the expert's sophisticated event sequence. This form of knowledge representation, an ontology and conceptual graphs comprise networks of conceptual nodes and associations linking successive steps of dissection viewpoints, suggested observations, potential discoveries, mechanical technique and anatomical illustrations supported by supporting hyperlinks embedded within the images at relevant points. Thus, dissection students will have distance-independent, on-demand access to the sophisticated knowledge of experts focusing on a well-constrained anatomical region and task.

Design of Ontological Data for Data Navigation and Media-Rich Document Display Application

Tatiana Malyuta, Insook Choi, and Robin Bargar
New York City College of Technology of the City University of New York
tmalyuta@citytech.cuny.edu

The initiative of the Downtown Brooklyn project is to prototype a media-rich interactive navigation system in a media-rich document environment. To assume an urban environment such as Downtown Brooklyn as a content base, the system must support documents of different types that can be stored and reproduced in digital formats, such as photographs, moving images, drawings, architectural plans, text documents and sounds. The main objectives of this project include:

- Implementation of a navigation system guided by ontological data.
- Interactive media-assisted query and documents display in a multimodal environment.
- Extensible data model that affords efficient incorporation of rapidly-developing urban data.

System Description

The prototype system differs in several respects from mainstream map-based information systems. The primary mode of accessing documents is through semantic coordinates rather than map coordinates, and the user interface is anchored by a visualization tool for navigating semantic space. Documents and visualizations are displayed dynamically on multiple screens by a digital media signal processing system that schedules the timing, sampling and sequencing of images and sounds based upon semantic relevance. Ontological data design increases the capacity for prioritization of signal processing resources, facilitating the implementation of a concept that requires time-critical parallel access to multiple documents of multiple types.

In terms of data representation for users of the system, two approaches benefit directly from the data design methodology:

- Visualization of semantic space and its correspondence with map-based coordinates of individual documents; and
- Representation of the semantic interests of multiple users traversing a shared virtual model of a complex urban environment.

These representations are assisted by the application of ontological data design for modeling the profiles of the users, the semantic interests in the document resources, and the metadata of the documents about the urban environment.

Data Design

Design of meta and ontological data (see Figure 1) is provided based on the following approach. The system supports documents of different types. We consider documents *data*. Documents are modeled with the help of the class <`Document`> and subclasses of this class that represent documents of specific types, e.g. <`PhotoDocument`> and <`ArchitecturalDrawing`>.

Except for several basic features, different features of interest (attributes) of documents are modeled not as properties of the document classes but as separate classes. The general class <`Attribute`> has subclasses: subclass <`Common`> of attributes relevant for documents of each type and the subclasses of attributes relevant for documents of different types, e.g. subclasses <`PhotoSpecificAttribute`> and <`ArchitectureSpecificAttribute`>. Within each subclass of attributes we may define additional superclass-subclass hierarchies of attributes, e.g. common attributes contain as a subclass <`Location`>, which in turn contains subclasses <`Street`>, <`CrossStreet`>, and <`District`>.

All documents share a few basic properties e.g. <`DateOfDocument`>, and they can be related to the leaf <`Common`> attributes. Documents of each type are related to their leaf <`Specific`> attributes. We consider documents' attributes *metadata.*

Documents of different types also are made related to each other with the help of various relationships between attributes. We call such various relationships of interest between attributes *ontologies* because these relationships define meaningful associations between documents. These relationships are asserted as separate knowledge. Each ontology is based on one or more conditions involving attributes; such conditions constitute the *ontology contents*. Ontologies define additional classifications of documents. For example, a user may want to see documents of different types related to the 'Business district' of Downtown, where the concept of 'business' district is defined based on the values 'Downtown' and "Metrotech' of the metadata attribute <`District`>.

Profiles and Invariants
To enforce documents search, the system supports several user *profiles* that reflect different users' interests. The system defines several *invariants* that participate in building the profiles; each profile is defined by invariant ontologies. For example, for the 'District' invariant, the profile 'Architecture Preservationist' is related to the ontology 'Historical district', while the profile 'Real Estate Developer' is related to the ontology 'Business district'. Document navigation and search are performed from profiles to invariant ontologies, then to other ontologies and metadata, and from there to the documents.

We consider hierarchies of attributes, ontologies, profiles, and invariants *ontological data*. Such design of meta and ontological data has several advantages: integrity of metadata, flexibility of modifications and expansion of meta and ontological data, and better utilization of OWL features by the reasoning mechanism.

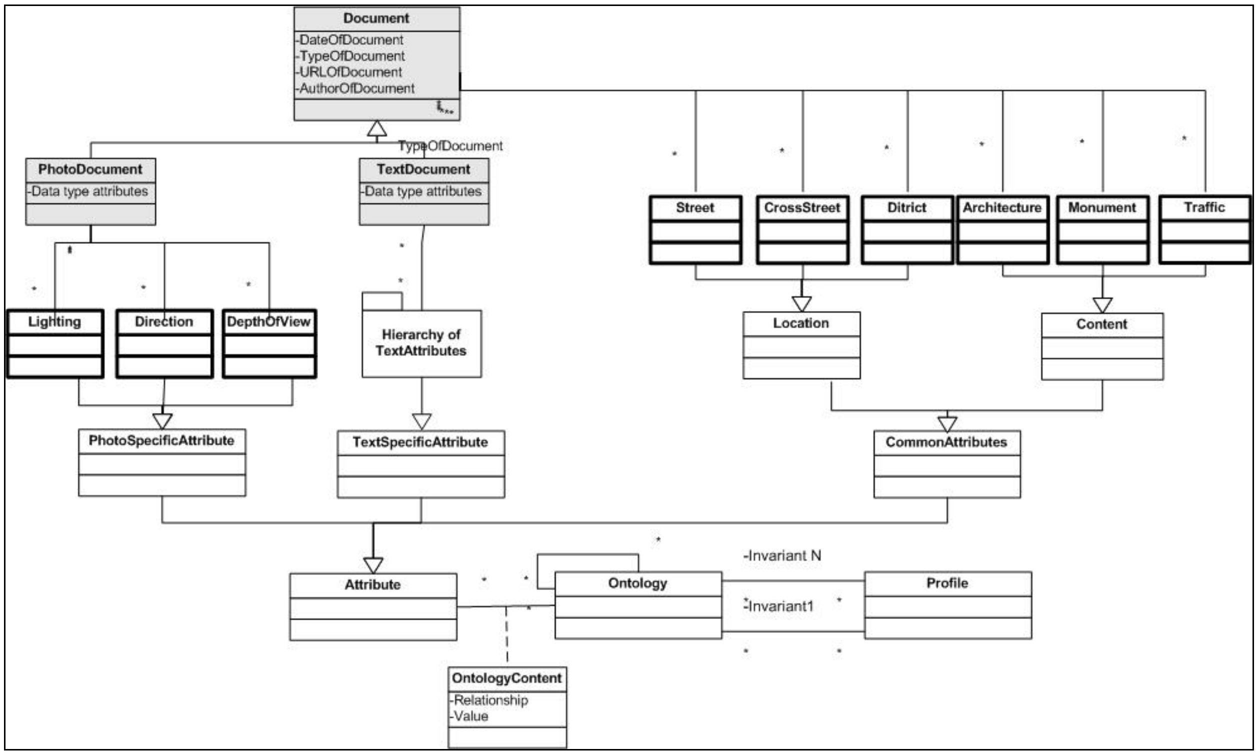


Figure 1. Design of meta and ontological data.

9th International Protégé Conference 2006
Poster Submission

Development of the Generation Challenge Program (GCP) Scientific Domain Model-Associated Ontology

Genevieve Mae Aquino[1], Victor Jun Ulat[1#], Kevin Manansala[2], Sergio Gregorio[3], Ramil Mauleon[1], Kouji Satoh[4] , Guy Davenport[5], Tom Hazekamp[6], Thomas Metz[1], Manuel Ruiz[7], Reinhard Simon[8], Masaru Takeya[4], Jennifer Lee[9,10], Martin Senger[1], Graham McLaren[1], Theo Van Hintum[11] and Richard Bruskiewich[1*]

[1]International Rice Research Institute, DAPO Box 7777, Metro Manila, Philippines
[2]University of the Philippines, Diliman, Quezon City, Philippines
[3]University of the Philippines, Los Baños, Laguna, Philippines
[4]National Institute of Agrobiological Sciences, Ibaraki, Japan
[5]Centro Internacional de Mejoramiento de Maíz y Trigo, Texcoco, Mexico
[6]International Plant Genetic Resources Institute, Rome, Italy
[7]Centre International de Recherche Agronomique pour le Développement, Montpellier, France
[8]Centro Internacional de la Papa, Lima, Peru
[9]University of Dundee, Nethergate, Dundee, Scotland
[10]Scottish Crop Research Institute, Invergowrie, Dundee, Scotland
[11]Centrum voor Genetische Bronnen Nederland, Wageningen Universiteit & Researchcentrum, Wageningen, The Netherlands
* Project Principal Investigator: r.bruskiewich@cgiar.org
# Poster Presenter: v.ulat@cgiar.org

Abstract

We present the construction of RDF/OWL Ontology for coding the semantic structure of the Generation Challenge Program (GCP) domain models and associated ontology for crop information systems. The ontology is an output of the GCP Subprogramme 4 commissioned research: “Task 22 - Development of GCP Domain (Data) Models,” and was concurrently developed with the common scientific domain model to ensure semantic compatibility across the GCP (see http://www.generationcp.org/model). Protégé-2000 was used to develop a formal "controlled vocabulary," or network of discretely enumerated named crop informatics concepts. Our ontology is focused on the representation of domain model feature types (attributes) and certain feature values as ontology and is explicitly modeled in the ontology metadata model of the GCP domain model (see http://pantheon.generationcp.org/demeter/Ontologies.html). Ongoing efforts are focused on the cataloguing of pertinent sub-domain entity ontology (using established international standards where available, e.g. Gene and Plant Ontology), the software implementation of the domain model and ontology in the GCP platform middleware (see http://pantheon.generationcp.org), and the translation of domain models into data type and service type ontology specifications for web services and semantic web implementation (see http://pantheon.generationcp.org/moby).

Keywords: crop, plant, agriculture, ontology, domain model

Topics: ontology development

Development of NeuronBank: An online knowledge base of identified neurons and synaptic connections

Calin-Jageman, RJ[1]; Sunderraman, R[2]; Zhu, Y[2]; Katz PS[1]

1 Department of Biology, Georgia State University, Atlanta, GA
2 Department of Computer Science, Georgia State University, Atlanta, GA

Neuroscientists seek to understand the brain by mapping its network topology. This involves finding ways to reliably classify neurons and then delineating their interconnections. Although knowledge of circuit topology is invaluable, no standard has evolved for representing this knowledge. The primary difficulty is that neuroscientists do not yet understand the determinants of a “natural class” of neurons. Instead, neuroscientists have built neuron typologies from whatever attributes they can currently measure. Extant typologies are thus *species specific* (reflecting the techniques currently available for that species) and *provisional* (changing as new techniques become available).

The NeuronBank project seeks to develop tools to allow neuroscientists to represent, explore, and share their knowledge of identified neuron types and neural circuits. Our approach is to provide a common framework that can accommodate species differences and changes in the way neuron types are described. The goal is to create a federation of extensible knowledge bases that can be seamlessly queried.

To represent neurons, we are using Protégé to develop an extensible ontology. Our ontology includes a “core” branch defining invariant concepts (neuron class, connection, etc.) and an extensible branch of attributes that can be used for describing and defining neuron types.

To store the knowledge of neuron types, we are developing the *NeuronBank Species Server*, a web-enabled knowledge base that uses our extensible ontology. Different communities of neuroscientists will be able to setup their own server and customize the ontology to describe the neuron typology in use for their species of interest. We are developing clients to enable users to input, browse, search, and compare neural circuits within a species. We are currently using Protégé as a back-end for the Species Server.

To share knowledge across neuroscience communities, we are developing the *NeuronBank Meta Server*, a portal for collecting information across groups of species-servers. Together, these components will form a federation of knowledge-bases that can serve species-specific needs while allowing wide-spread data sharing in the neuroscience community.

This project is funded by NIH grant # R21 MH76753 and the Georgia State University Brains and Behavior Program.

Intelligent Integration of Railway Systems
Poster Description

Richard Lewis[1], Clive Roberts[1], Gerhard Langer[2], Michael Pirker[2], Roger Shingler[3]

[1] University of Birmingham, Edgbaston, Birmingham, United Kingdom
[2] Siemens Transportation, Werner-von-Siemens Strasse, Erlangen, Germany
[3] Bombardier Transportation, Litchurch Lane, Derby, United Kingdom

This poster will contain a description of the InteGRail project, Intelligent Integration of Railway Systems. InteGRail is a Europe wide, European Commission funded project that has 39 stakeholders representing all of the main industry players. The list includes operators, suppliers, maintainers and technical advisors such as Universities.

The poster will contain an introduction to the project highlighting the main objective of the work. The objective of the project is to develop a trans-national, Europe wide, integrated railway system. The aim to meet this objective is to create a **holistic, coherent information system** to integrate the major railway subsystems and deliver a higher level of coordination and cooperation between the key railway processes by intelligent information sharing. It is believed that the result will be an integrated system that will facilitate the improved information sharing and provide decision support. It is considered that the key aspects of this work are interoperability based on a common railway ontology as a basis for standardisation.

The experience gained in previous integrative projects is being used to identify key challenges for higher level semantic and ontological information sharing areas and define specific areas where improvements can be made. These improvements are based on integrating existing information systems to gain the maximum value from information. The problem areas are believed to lie in the integration of heterogeneous technical systems that rely on different terminology without appropriate self-descriptiveness. On an ontological level the capture and integration of the domain experts view is relevant.

In consideration of these challenges the solution is believed to lie in the creation of models to represent domain concepts and their related attributes, in the form of on-line ontological representation of measurements within this virtualisation of the perception of real-world sensor information. The capture of the expert's knowledge (based on scenarios) will then be used to create sequences of description logic queries that implement the system services based on distributed reasoning. . The sensor data will be mapped on-line to the models where it can be classified to identify a situation e.g. condition based monitoring, and provide appropriate system actions and activities.

Initial work with onotology and reasoners has resulted in the development of example railway domain ontology. Ongoing work and discussions with experts will yield more comprehensive models displaying the strengths of this approach over traditional database/data warehouse implementation.

Contact - Clive Roberts (C.Roberts.20@bham.ac.uk)

OWLFCAView Tab: A Visualization and Modeling Tool for Composite Expressions of SNOMED CT using Formal Concept Analysis (FCA)

Guoqian Jiang, Harold R. Solbrig, James D. Buntrock, Christopher G. Chute
Division of Biomedical Informatics, Mayo Clinic, Rochester, MN, 55905

Introduction

Modern terminologies have advanced well beyond simple one-dimensional subsumption relationships through the introduction of composite expressions. The ability to form composite expressions opens a whole new realm of expressive possibility. Meanwhile, it also brings the great challenges to the terminology service software. In clinical domain, SNOMED-CT [1] now provides a platform where composite expressions have the potential to be used in clinical situations. The complexity, however, of using SNOMED-CT in its tabular form can still be quite daunting and the need for an intermediate service layer is becoming an absolute necessity. Our general hypothesis is that reformulating the rules of composition and compositional transformations in the language of lattice theory will possibly provide a solution for the challenges described above. Formal concept analysis (FCA) provides a fertile ground for exploitation with its generic structure of lattice building algorithms to visualize the consequences of partial order that the underlying mathematical lattice theory builds on [2-3]. In this study, we developed a visualization and modeling tool for the composite expressions of SNOMED-CT using FCA technique and discussed the issues related to the future potentials of the FCA-based approaches.

Materials

A subset of 2006 US edition of SNOMED-CT with OWL format was used. The concepts of SNOMED-CT fall into two types: primitive concepts and fully-defined concepts. For the former, the asserted conditions of the concepts are necessary but not sufficient and for the latter, the asserted conditions are both necessary and sufficient.

System Construction

The visualization and modeling tool was developed as a Protégé Tab Plug-in called “OWLFCAView Tab” in a Protégé OWL platform. The Protégé platform is an ontology edit environment which was developed by Stanford Medical Informatics [4]. The JAVA API of an open source software Concept Explorer version 1.2 was integrated to generate the cross-table context and the concept lattice [5].

Based on analysis of the asserted conditions of composite expressions represented in OWL file of SNOMED-CT, three basic specifications were implemented in current version of our tool. The first perspective focused on one of properties in a selected set of the SNOMED-CT composite expressions. Here, the selected set of the expressions was used as the formal objects and the fillers of the restriction of the selected property were used as the formal attributes. The second one focused on all asserted restrictions in a selected set of the composite expressions. Here, the selected set of the expressions was used as the formal concepts and the fillers of all asserted restrictions defined for the expressions were used as the formal attributes. The third one focused on all super-classes of a selected set of the composite expressions. Here, the selected set of the expressions was used as the formal objects and the super-classes of each expression were used as the formal attributes.

Results & Discussions

One of the main features in this study is that we developed a property-oriented way for visualization of Protégé OWL ontologies using the FCA technique, especially focusing on the composite expressions of SNOMED-CT. Although there exists several other kinds of visualization tools (e.g. built-in OWLViz Tab plug-in [4]), however, most of them are class-oriented. Through the property-oriented way, the visualization could become more flexible and scalable. For example, there is a specific property named “RoleGroup” assigned to most of the composite expressions. Guiding by the “RoleGroup”, we could easily identify graphically in a concept lattice whether the definitions of the “RoleGroup” among the composite expressions at question are the same or not. Therefore, through capturing every granularity of the composite expression, our FCA-based visualization tool could provide foundations to achieve our goal of seeking solutions to the challenges of the current terminology services facing on. In addition, the current version of our visualization tool has been generalized for the Protégé OWL ontologies and available in the plug-in library of Protégé [4].

References

[1] URL: http://www.snomed.org/; last visited: April 14, 2006.
[2] Granter B and Wille R. Formal concept analysis: mathematical foundations. Springer, 1999. ISBN: 3-540-62771-5.
[3] Kalfoglou Y, Dasmahapatra S and Chen-Burger Y. FCA in knowledge technologies: experiences and opportunities. In Proceedings of the 2nd International Conference on Formal Concept Analysis (ICFCA’04), Sydney, Australia, Feb, 2004.
[4] URL: http://protege.stanford.edu/; last visited: April 14, 2006.
[5] URL: http://sourceforge.net/projects/conexp; last visited: April, 2006.

A STUDY OF ONTOLOGY-BASED DESIGN INFORMATION REPRESENTATION, INDEXING AND PRODUCT LIFECYCLE SEARCH

Zhanjun Li and Karthik Ramani
Purdue Research and Education Center for Information Systems in Engineering (PRECISE)
Purdue University, West Lafayette IN 47907-2024, USA

Abstract

Due to the increasing complexity of the product and the design process, as well as the popularity of computer-aided documentation tools, the number of electronic and textual design documents generated has exploded. The availability of such extensive design document resources has created new challenges and opportunities for research. These include improving the design information retrieval in order to achieve a more coherent environment for design exploration, learning, and reuse. One critical issue is related to the construction of design ontology for indexing and retrieval design documents. The ontology model can explicitly and accurately capture the important design concepts as well as the engineering context which differentiate these concepts so that engineers can locate their documents of interest with less effort.

We propose to develop and use a handcrafted, domain-specific knowledge source for design information indexing and retrieval. Figure 1 demonstrates the system architecture and functional modules. The knowledge source includes a design ontology and design lexicon, representing domain knowledge as well as domain-related linguistic knowledge. The acquisition and maintenance process are supported by Protégé 3.2. The knowledge source is used to assist the extraction of engineering semantics from unstructured design documents, i.e. convert from free text documents into a structured representation with all the terms which carry engineering semantics uniquely identified As ontology concepts and in XML representation, i.e. partXMLs. Using design ontology and design lexicon makes the extraction process less dependent on natural language processing techniques in order to understand the texts while acquiring improved accuracy of concept recognition. More effective design information retrieval is achieved by using the knowledge source to disambiguate query keywords at concept level and to detect query intents under engineering contexts as well. An ontology-based query processing algorithm is developed to fulfill this task. To support the product lifecycle search, we also integrate ontology and shape based approaches for design information retrieval.

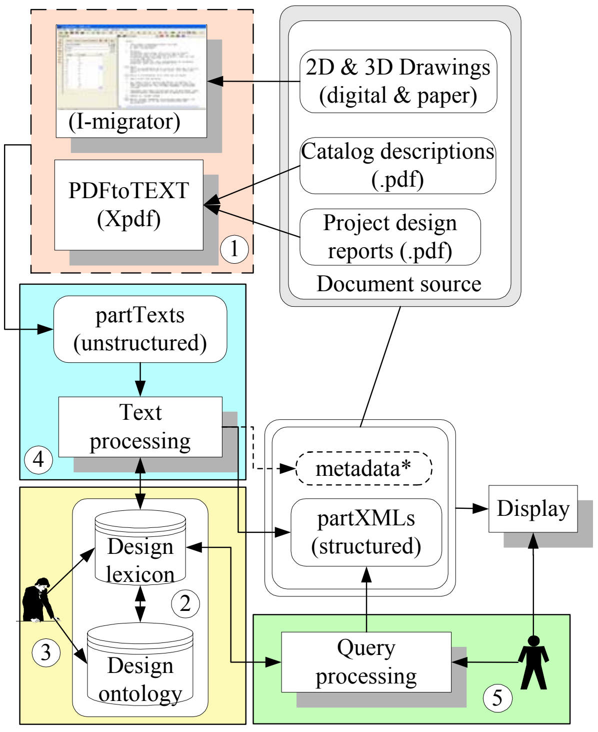


Figure 1. System architecture of ontology-based design information indexing and retrieval

1. Pre-processing
2. Domain-specific knowledge source
3. Knowledge acquisition and maintenance using protégé 3.2
4. Engineering semantics extraction
5. Document retrieval

The Termontography Workbench: a Protégé-based tool for the compilation of multilingual terminological resources

Peter De Baer, Koen Kerremans and Rita Temmerman

Erasmushogeschool Brussel - Centre for Terminology and Communication (http://cvc.ehb.be/)

In this poster we describe the Termontography Workbench, a tool for the compilation of ontologically structured multilingual terminological resources starting from a multilingual domain-specific corpus.
This tool supports the termontography methodology, a multidisciplinary approach in which theories and methods for a multilingual terminological analysis in accordance with sociocognitive theory [1] are combined with methods and guidelines for ontology engineering. A clear distinction is made between conceptual modeling at a culture-independent level and a culture-specific analysis of units of understanding. Hence, the prototypical structuring of understanding is taken into consideration.
We currently use the workbench for the PoCeHRMOM-project to build a trilingual (Dutch, English and French) terminological resource and application ontology in the field of e-HRM. By providing an online e-HRM platform that includes terminological information on functions, educational background and competencies we wish to improve the matching between job offers and candidate profiles. Furthermore we want to make it easier for SME's to participate in emerging e-HRM processes.
The workbench uses Protégé as an object database. Our model (supporting the TBX-standard) is used to structure the multilingual terminological information and contains the following information:

1. Source: a corpus text document.
2. Term: a language and text string combination.
3. (Term) description: a description of a specific term or a meaningful text phrase (i.e. law).
4. Term relation: a relation between two terms, for example the full form of an abbreviation.
5. Concept: a relevant unit of understanding within the domain of interest.
6. Concept relation: a relation between two concepts, for example a required competency for a certain function.
7. Category: a unit of understanding used to classify sources, terms, (term) descriptions, term relations, concepts, concept relations and categories.

Categories and concepts can be represented by near-synonymous or near-equivalent terms. The categories are structured in a partitive hierarchy while the concepts are structured in a generic hierarchy on the domain. In the workbench both hierarchies are displayed as a tree model.
Our decision to use Protégé as the underlying database was mainly based on the following considerations:

- We wanted the model to be flexible while building the prototype of our workbench without having to hardcode and maintain complex SQL-statements.

- Using the database backend of Protégé we expect the database to be expandable over time.
- Using the server version of Protégé we believe it will be possible to build a multi-user version of the workbench.
- We hope to be able to exchange ontology information with other knowledge engineers using the Protégé tool(s).
- In the future we want to research how ontology inference can support the creation of terminological resources.

[1] R. Temmerman, Towards New Ways of Terminology Description. The sociocognitive approach, John Benjamins, Amsterdam/Philadelphia, 2000.

Theory-specific sub-ontologies and theory-neutral general ontologies

Graham Wilcock

Department of General Linguistics
University of Helsinki
graham.wilcock@helsinki.fi

This poster describes work on combining a theory-specific sub-ontology with a more general theory-neutral ontology. The domain is linguistic description.

Background
Academic linguists tend to hold strong views on linguistic theories. Each major theory has its own dedicated annual conference, where its advocates present theory-specific linguistic descriptions, and show how their theory explains the data better than rival theories. Field linguists by contrast tend to be busy recording examples of endangered languages before they disappear forever, and less interested in which theory may offer a more elegant analysis. They want a general framework which helps them to capture data about an unknown language with a minimum of theoretical preconceptions.

HPSG: A specific linguistic theory
Head-driven Phrase Structure Grammar (HPSG) (Pollard & Sag 1994) is one specific linguistic theory. Unlike most other theories, it is based on a rigorous foundation of typed feature structures with a clear type hierarchy. HPSG is usually implemented by unification-based typed feature structure systems such as LKB (Copestake 2002).

HPSG OWL: A theory-specific ontology
Using Protégé we have developed a theory-specific HPSG OWL ontology. Linguistic theories, like ontologies, evolve through different versions: we adopted the version of HPSG described by Sag et al. (2003), with an emphasis on multiple inheritance and cross-classification. This work is ongoing. The current focus is on using reasoners to support cross-classification and SWRL to support lexical rules and grammar rules.

GOLD: A theory-neutral general ontology
General Ontology for Linguistic Description (GOLD) (Farrar & Langendoen 2003) (http://www.linguistics-ontology.org/) is an OWL ontology that offers a more general framework intended for field linguists. GOLD uses theory-neutral class names, such as GrammaticalUnit, SemanticUnit, FeatureSpecification. Although GOLD initially attempted to be entirely theory-neutral (as regards competing linguistic theories), it has moved towards supporting different "communities of practice" within the overall framework (Farrar & Lewis 2005).

COPE: Community of practice extensions

A community of practice in GOLD may focus on developing a consensus within a specific area, for example in phonology or in describing Bantu languages. In such a case, the aim is to encourage the development of "best-practice" resources. On the other hand, communities of practice may focus on competing theories, where each sub-community has its own distinctive terminology and divergent conceptualization. In this case, the aim is different: to capture explicitly the relationship between the sub-community view and the overall framework, in the form of a Community of Practice Extension (COPE) (Farrar & Lewis 2005). A COPE is a sub-ontology that inherits from, and extends, appropriate parts of the overall GOLD ontology.

HPSG COPE: A theory-specific sub-ontology inside GOLD

HPSG OWL has an HPSG-specific class hierarchy using HPSG terminology. In order to combine this with GOLD, we define the HPSG-specific classes as subclasses of the theory-neutral GOLD classes. For example, in the HPSG type hierarchy there is the type "sign" (Sag et al. 2003, p. 475) with these attributes:

PHON	type: list(form)	(a sequence of word forms)
SYN	type: gram-cat	(a grammatical category)
SEM	type: sem-struc	(a semantic structure)

and in GOLD there is the class LinguisticSign which includes the properties:

hasForm	Range: PhonologicalUnit
hasGrammar	Range: GrammaticalUnit
hasMeaning	Range: SemanticUnit.

Using namespaces, we define HPSG types as subclasses of GOLD classes, and define HPSG attributes as subproperties of GOLD properties. For this example:

hpsg:sign	subclass of	gold:LinguisticSign
hpsg:form	subclass of	gold:PhonologicalUnit
hpsg:gram-cat	subclass of	gold:GrammaticalUnit
hpsg:sem-struc	subclass of	gold:SemanticUnit
hpsg:PHON	subproperty of	gold:hasForm
hpsg:SYN	subproperty of	gold:hasGrammar
hpsg:SEM	subproperty of	gold:hasMeaning

The work on both HPSG OWL and HPSG COPE is ongoing.

References

Copestake, Ann (2002) *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford.

Farrar, Scott & Terence Langendoen (2003) A linguistic ontology for the Semantic Web. GLOT International 7(3), 97-100.

Farrar, Scott & William Lewis (2005) The GOLD Community of Practice: An infrastructure for linguistic data on the web. URL: http://www.u.arizona.edu/~farrar/.

Pollard, Carl & Ivan Sag (1994) *Head-driven Phrase Structure Grammar*. CSLI Publications, Stanford.

Sag, Ivan, Thomas Wasow & Emily Bender (2003) *Syntactic Theory: A Formal Introduction* (2nd edition). CSLI Publications, Stanford.

Sequences in Protégé OWL

Nick Drummond[1], Alan Rector[1], Robert Stevens[1], Georgina Moulton[2],
Matthew Horridge[1], Hai H. Wang[1], Julian Seidenberg[1]

1. Bio Health Informatics Group, School of Computer Science, The University of Manchester, UK
2. Northwest Institute for Bio Health Informatics, Manchester, UK
nick.drummond@cs.manchester.ac.uk

ABSTRACT: Sequences are a natural part of the world to be modeled in ontologies. Yet the Web Ontology Language, OWL, contains no built in support specifically for sequences or ordering. It does, however, have constructs that can be used to model many aspects of sequences, albeit imperfectly. This paper describes a design pattern for modeling order using existing OWL-DL constructs. These constructs allow us to use standard DL reasoning to perform pattern matching akin to regular expression matching. Although clearly not the most efficient mechanism, pattern matching with standard DL reasoners works surprisingly well and brings real benefits to users by allowing them to work at a higher level of abstraction than raw sequences and to deal with situations in which the details of the sequences are under specified.

Introduction

OWL has no inbuilt support for ordering. However, the world to be modeled in ontologies expressed in OWL is full of sequences:

- Time related events – *e.g.* sequences of sub-processes, plans, life-stages etc.
- Physically linked structures – *e.g.* Protein sequences and other macromolecules, carriages in a train, etc.
- Conceptually linked structures – *e.g.* documents, data structures, travel itinerary etc.

Unfortunately, the natural constructs from the underlying RDF vocabulary – `rdf:List` and `rdf:nil` – are unavailable in OWL-DL because they are used in the RDF serialization of OWL[1]. Although `rdf:Seq` is not illegal, it depends on lexical ordering and has no logical semantics accessible to a DL classifier. Despite these limitations, we have strong reasons for wanting to express and reason with sequential constructs in OWL-DL.

- *Expressivity* – OWL-DL includes constructs such as transitive properties, which allow more of the semantics of sequences to be represented explicitly than in RDF or OWL-Lite.
- *Reasoning* – DL reasoners can be used to check consistency and infer subsumption – *e.g.* to confirm that a sequence of amino acids contains only amino acids or to infer that a class of sequences is subsumed by another class of sequences – *i.e.* that one pattern is a subset of another pattern.

Our example is that of sequences of amino acids, although the patterns are applicable to many domains. An example ontology is available along with a more general demonstration on the co-ode wesite[2].

Modelling lists as data structures

To model lists as data structures, we follow the standard pattern for linked lists in which each item is held in a "cell" (`OWLList`); each cell has contents ("head") and a pointer to the next cell ("tail"); and the end of the list is indicated by a terminator (`EmptyList`) which also serves to represent the empty list. In RDF these constructs are

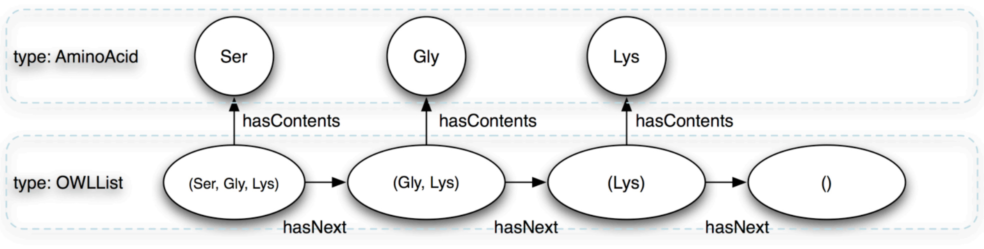


Figure 1: List data structure – simple example

[1] http://www.w3.org/TR/owl-semantics/mapping.html#rdf_List_mapping
[2] http://www.co-ode.org/ontologies/lists/

implemented using the class `rdf:List` for the cell, the individual `rdf:nil` as the terminator, and the two properties `rdf:first` and `rdf:next` for the contents and pointer to the next cell respectively.
However, because we cannot use the RDF vocabulary in OWL-DL we must define our own (Figure 2), an example of which is shown diagrammatically in Figure 1. Whereas the semantics of the properties `rdf:first` and `rdf:next` are implicit in RDF, in OWL we can express more. We want each cell to have exactly one contents item and one next cell, and we want to represent the notion of being a member of the list. This can be done by making `hasContents` and `hasNext` functional, and by defining a transitive property, `isFollowedBy`, as a super-

```
Class(OWLList partial
        restriction(isFollowedBy allValuesFrom(OWLList)))
Class(EmptyList complete
        OWLList
        restriction(hasContents maxCardinality(0)))
EquivalentClasses(EmptyList
                  intersectionOf(OWLList
                                 NOT restriction(isFollowedBy SOME owl:Thing)))
ObjectProperty(hasListProperty domain(OWLList))
ObjectProperty(hasContents Functional super(hasListProperty))
ObjectProperty(hasNext Functional super(isFollowedBy))
ObjectProperty(isFollowedBy Transitive super(hasListProperty) range(OWLList))
```

Figure 2: OWL vocabulary for lists as data structures in concrete abstract syntax

property of `hasNext` as shown. Since this means that `hasNext` implies `isFollowedBy`, any sequence of entities linked by `hasNext` will be inferred to be a chain linked by `isFollowedBy`.

In other words the members of any list are the contents of the first element plus the contents of all of the following elements. The intention is that cells should be directly linked by the functional property `hasNext`. The transitive superproperty, `isFollowedBy,` is typically used in definitions and queries. An example of a fully specified list is shown in Figure 3.

Note that we are representing classes of lists rather than individual lists. We treat classes of lists as patterns. We then use the reasoner to determine which other classes of lists and/or individual lists are subsumed by – *i.e.* match – those patterns.
For uniformity we have chosen to create a class of empty lists, which have neither content nor following members. (The negated existential restriction is used with the property `isFollowedBy` rather than the apparently simpler `cardinality(0)`, because cardinality constraints are not permitted on transitive properties[3]). Note that, from the definitions and equivalence axioms given, we can infer that any list that provably has no contents can have no following elements and *vice versa*.

```
OWLList AND
       hasContents SOME Ser AND
       hasNext SOME (
            OWLList AND
                hasContents SOME Gly AND
                hasNext SOME (
                    OWLList AND
                        hasContents SOME Lys AND
                        hasNext SOME EmptyList))
```

Figure 3: Example of a class of OWL Lists of the form (Ser, Gly, Lys) in simplified syntax

Types of lists

Possible constructs are shown in Figure 4 along with a simplified syntax and examples – there is not enough space to describe the OWL for each, but there is at least one example of each in the demo ontology. We use a sugared shorthand syntax for lists that should be intuitive given the definitions and examples. Space does not permit an exhaustive enumeration, but it is clear that constructs supported are similar in expressivity to that available in regular expressions. Below we describe some of the properties of lists modeled using this pattern:

- The elements are classes, which may be fully defined, e.g. "tiny polar amino acid" or "large charged amino acid". A defined class can be considered as an implied disjunction of its subclasses. This would equivalent to being able to name a disjunction[4] in a regular expression. Most regular expression languages do not support the use of named subexpressions. Even if named subexpressions were supported, the disjunction would have to be enumerated manually in advance. By contrast, in OWL, the classifier can infer the subclass hierarchy based on

[3] http://www.w3.org/TR/owl-ref/#OWLDL

[4] In regular expression parlance, an "alternation", usually written [P1 P2 P3] where each Pi is itself a regular expression.

	terminology	meaning	examples
1	`(A, B, C)`	Exactly ABC (terminated)	`abc`
2	`(A!)`	A list consisting only of As	`aaa, aa, etc.`
3	`(A, B, C, …)`	Starting with ABC (non-terminated)	`cbc, abcx`
4	`(…, A, B, C)`	Ending With ABC (terminated)	`abc, xabc`
5	`(…, A, B, C, …)`	Containing ABC	`abc, xabc, xabcx, abcx`
6	`(A*B )`	A string of As followed by B	`ab, aaab,`
7	`([A, B, C], B, C)`	A or B or C, followed by B then C	`abc, bbc, cbc`
8	`(hasProp some X, B, C)`	Restriction followed by B then C	`Any abc where a hasProp x`
9	`¬(A, B, C, …)`	Not starting ABC	`cbaxx`
10	`((A, B, C, …), (D, E, F, …))`	Starting ABC, followed by anything, followed by DEF, followed by anything	`abcdef, abcxxdefx`
11	`(A, B, C, …) AND (…, A, B)`	Starting ABC, and ending AB	`abcab, abcxxab`
12	`()`	Empty list (nil)	

Figure 4: Examples of the constructs that can be expressed in OWL Lists

the properties of the amino acids. Different abstractions over the same amino acids can be used for different problems.

- The notion of "0 or more `As`" or "1 or more `As`" cannot be expressed on its own without including the terminating pattern or item, even if this is simply the empty list as this requires a recursive definition in OWL.
- There is no way of stating "`n As`", *i.e.* `n` repetitions of `A,` other than by explicitly expanding the list with `n` occurrences of `A`.
- It is possible to assert a class of lists in a conjunction, or more generally a boolean combination, of classes defined using the above patterns – as in line 11. However, care is required, as this can give unexpected results. For example, the class of lists that starts with "ABC" and ends with "BCD" is different from the class of lists starting with "ABC" followed by "BCD". That is:
  `(A, B, C, ...) AND (..., B, C, D)` subsumes `(A,B,C,D)`
  whereas (A, B, D, ... ,B, C, D) does not.
- There is no way to define a class of "lists" so that it excludes cycles. (Note that the class of lists`(A, B, A)` does not imply a cycle, merely a list beginning with an `A`, followed by a `B`, followed by another (possibly the same) `A`. Both individual lists `(a1, b, a2)` and `(a1, b, a1)` satisfy this definition.
- There is no way to define the class of lists of a specific length except by exhaustively representing the member classes. A more compact form would require the use of cardinality constraints on `isFollowedBy`, which is transitive. Cardinality constraints on transitive properties are excluded from OWL-DL[5].
- There is no way to define a class of "lists" so that it excludes additional branches being defined using `isFollowedBy` instead of its functional sub-property `hasNext`.
- It is not possible to represent a class of lists in which one named sublist *directly follows* another named sublist without an intervening element (as in pattern 10). This can be done only by explicitly re-representing the concatenation of the two patterns as a single list. To make this practical, a macro like mechanism would be required in the tools.

Using OWL lists

The mechanisms described have been used with biologists to capture notions that they would otherwise find difficult to express. Biologists find the ability to work with under-specified sequences and to consider abstractions over sequences useful. Biologists form amino acid patterns (motifs) by looking at collections of similar proteins and deducing that all these proteins have either Arginine or Lysine at this position. A regular expression is then made that captures this inference. With OWL lists, we can capture this notion using pattern 7 or we can enable is a greater abstraction, "large and positive", using pattern 8:

Pattern 7 element: `Arg or Lys`

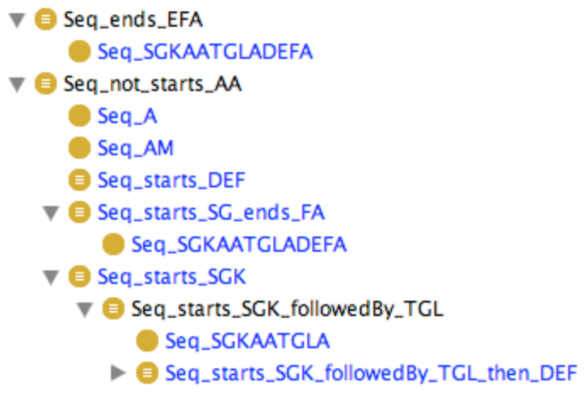
Pattern 8 element: `AminoAcid and ((hasSize some Large) and (hasCharge some Positive))`

The abstraction is more expressive than the disjunction and by underspecifying we may also find further matches that suggest the disjunction is over-constrained. In addition, finding that one under-specified pattern subsumes another has intriguing biological possibilities.

We have implemented a series of wizards and tools in Protégé-OWL to make the construction of the required subset of patterns possible and made the user interface practical and at least partly hide the raw OWL syntax. Further tools to help users formulate and display these notions and to make the pattern easily and completely generic are in progress.

[5] and OWL 1.1

For the test-cases that have been run, results are surprisingly fast, although we would like to investigate further whether some constructs have a greater effect on the reasoning speed than others. The example on the web includes reasonably complex, although intentionally short, classes of lists and classifies on a moderately fast laptop, using Protégé-OWL connected to FaCT++[6] or Pellet[7], in approximately 2 seconds. The accompanying fingerprint example, which models real biological data, uses pattern 10 to "join" six motifs together in sequence. Each motif matches a pattern of approximately 20 elements, with a number of alternative elements at each position. Running through Pellet took between 80-170s to correctly classify various test proteins up to 450 elements long. Because of the recursive nature of the definitions, the main practical difficulties witnessed were that of stack size within editing and reasoning software which can be easily resolved with careful programming.



```
▼ Seq_ends_EFA
    Seq_SGKAATGLADEFA
▼ Seq_not_starts_AA
    Seq_A
    Seq_AM
    Seq_starts_DEF
  ▼ Seq_starts_SG_ends_FA
      Seq_SGKAATGLADEFA
  ▼ Seq_starts_SGK
    ▼ Seq_starts_SGK_followedBy_TGL
        Seq_SGKAATGLA
      ▶ Seq_starts_SGK_followedBy_TGL_then_DEF
```

Figure 5: Extract from inferred hierarchy of example lists

Conclusion

In summary, we have described a design pattern for describing sequences in OWL-DL. There are alternatives to be investigated, such as the idea of directly linking elements together without any intervening structure, but the ontological differences between these approaches lie outside the scope of this paper.

Standard reasoners can be used to infer subsumption corresponding to pattern matching over classes of lists and to recognize lists of individuals as belonging to given classes, *i.e.* to treat classes of lists as patterns and to determine whether other classes are more specialized patterns and whether lists of individuals match those patterns. For illustration, a small extract of the classified online example is shown in figure 5. The use of tableaux reasoners, ensures that our matches are sound and complete. However, users must take care to provide complete definitions including both disjointness and closure axioms. Algorithms based on deterministic finite automata as in standard regular expression matchers would almost certainly be faster but the main purpose of using a tableaux reasoner has been to express the list structures along with other notions in a single representation and use reasoners already integrated with OWL-DL.

The most important result of this work is our experience that representing and reasoning over classes of ordered structures in OWL-DL is useful. It provides users with new ways to organise and browse their knowledge at a higher level of abstraction than would otherwise be possible. We have used real examples from protein sequences in biology as a motivation, but the notions are general and can be applied to other notions that are intrinsically ordered.

[6] http://owl.man.ac.uk/factplusplus/
[7] http://www.mindswap.org/2003/pellet/

Frames and OWL Side by Side

Hai H. Wang[1] Natasha Noy[2] Alan Rector[1] Mark Musen[2] Timothy Redmond[2] Daniel Rubin[2]
Samson Tu[2] Tania Tudorache[2] Nick Drummond[1] Matthew Horridge[1] Julian Seidenberg[1]

[1]Bio-Health Informatics Group
The University of Manchester,

[2]Stanford Medical Informatics
Stanford University

1 Introduction

Understanding the relation of the two most widely-used ontology modeling paradigms, OWL (and other DL paradigm) to frame paradigms is an important issue. Because the Protégé tool supports both paradigms within a single overall framework, it provides an opportunity to compare the two paradigms both in theory and practice. In this paper, we focus on the relationship between the Frames paradigm, as defined in the OKBC specification [1] and implemented in core Protégé, and OWL and its implementation in Protégé-OWL.

OWL provides three increasingly expressive sublanguages, OWL Lite, OWL DL, and OWL Full, for different communities of users. In this paper, we will primarily use OWL DL as the basis for comparison, as OWL DL provides the description-logic reasoning capabilities that are the distinctive features of OWL. Therefore, unless otherwise noted, when we use "OWL", we are actually referring to OWL DL. Contrastingly, there is not a single standard frame representation language analogous to OWL. The past standardization effort has produced the Open Knowledge-Base Connectivity (OKBC) protocol, which is a generic access and manipulation layer for knowledge representation services. For the purpose of this paper, we will primarily use the Protégé-Frames implementation of the OKBC Knowledge Model. When appropriate, we comment on features of frame representations that may not be available in Protégé. Unless otherwise noted, when we use "Frames," we are actually referring to Protégé-Frames.

2 Similarities and Differences

The two paradigms have many similar modeling constructs: both are built around the notion of classes, representing concepts in the domain of discourse; classes have instances; properties (slots) describe attributes of those classes and relationships between them; restrictions and facets express constraints on the values of properties and slots. There are however major differences in the semantics of these constructs and in the way these constructs are used to infer new facts in the ontology or to determine if the ontology is consistent. As a result, the way that the modeling constructs are used in the two paradigms and the implications of definitions are different.

2.1 Semantics and implication

The following represents some of the major differences between Frames and OWL:

Unique name assumption In Frames, if two objects have different names, they are assumed to be different, unless explicitly stated otherwise. In OWL, no such assumption is made.

Closed World Assumption vs Open World Assumption In Frames, everything is prohibited until it is permitted; in OWL, everything is permitted until it is prohibited. Nothing can be entered into a Frames KB until there is a place for it in the corresponding template. Anything can be entered into an OWL KB unless it violates one of the constraints.

Single vs Multiple models A Frames ontology only has one model which is the minimal model that satisfies each of the assertions of the Frames ontology. This means that models for a Frames ontology can only contain instances that are explicitly specified. In general an OWL ontology will have many models consisting of all possible interpretations that satisfy each of the assertions in the OWL ontology.

These differences have direct implications on how inference is performed, and therefore, implications for the use of modeling constructs in the two paradigms. The foremost of these differences are:

[1]http://www.ai.sri.com/okbc/spec.html

Assertion vs Classification In Frames, defining facets on a slot at a class, or defining a constraint on a slot at the top level, makes a statement about all instances of that class (except for possible exceptions provided by default values), describing *necessary* conditions for instances of that class. In OWL, there are effectively two kinds of statements about classes: a) those that, as in Frames are true of all individuals in a class, and b) those that are collectively *necessary and sufficient* to recognize members of a class. A OWL classifier can use the sufficient conditions to infer which classes are subclasses of the defined class. There is no equivalent feature in Frames.

Constraint checking vs Consistency checking The same reasoner that checks the classification also checks that an OWL KB is consistent. The classifier tries to build a model that satisfies all the axioms in the ontology. If no such model can be built, the ontology is inconsistent. When building a model that satisfies all the assertions, a classifier may assign new types to ontology instances, in addition to the types explicitly asserted by a modeler. By contrast, a Frames reasoner checks if the constraints are satified by the property values on instances; if they are not, the instance is non-conformant. In Frames, the inference cannot assign a new type to an instance.

Other differences include association of slots and properties to classes and individuals: In OWL, properties can be used with any class or individual unless such use violates explicitly specified constraints. In Frames, slots must be explicitly attached to classes before we can add facets to them or use them to assign slot values for the instance of the class. Treatment of multiple domains and ranges for slots and properties also works differently: In OWL, multiple domains and ranges are treated as an intersection, and in Frames—as a union.

In practice, these major differences in the sanctioned inference lead to differences in modeling style. A developer of an OWL ontology thinks in terms of necessary and sufficient conditions to define a class, building new concepts from existing ones by fitting them together in definitions like blocks of Lego and determining what conditions sufficiently define something as an instance of a class. A developer of a Frames ontology addresses the problem from another angle, deciding what are the implications of being a member of a particular class.

2.2 Expressive Power

2.2.1 Frames

Meta-modelling: Metaclasses are classes whose instances are themselves classes and constitute an integral part of the Frames formalism. In OWL DL, the sets of classes and instances are disjoint, and therefore metaclasses are not available[2].

Classes as property values: Another implication of the requirement that classes and instances are disjoint in OWL DL is the the prohibition on using classes as property values. There are a number of ways to approximate this meaning in OWL DL , using classes directly as property values could be much more intuitive in many cases.

Default Information and Exceptions: Default reasoning has always been one of the great strengths of frames systems. Frames are designed around the notion of "prototypes" in which defaults are used to fill partial knowledge in our perceptions. Many domains, such as Biology and Medicine, strongly rely on exceptions. Because OWL is a logic formalism. It leaves no room for exceptions. Currently, there are some attempts to use design patterns to represent default information in OWL . However, the proposed approaches are far from perfect and impractical in all but the simplest cases. Default reasoning remains one of the key strengths of Frames over OWL.

Concrete Domains and User Defined Datatypes: One of the omissions in the current OWL language that prevents many potential users from adopting OWL is poor representation of numeric expressions in order to be able to express quantitative relations. Issues preventing this support from being in the current version of OWL include how to refer to an XML schema user defined datatype with a URI and the denotational semantics of the XML data type etc. However, these are being resolved and OWL 1.1 will support user-defined datatypes.

2.2.2 OWL DL

Defined Classes: As discussed before, OWL allows us to 'define' classes, allowing inferred subclass and type information to be calculated. There is no equivalent to defined classes in Frames.

Embedding Class Definition: OWL allows complex expressions to be built up without having to name each intermediate concept before use. This embedding of 'anonymous concepts' allows more expressive representations to be built easily and naturally. To some extents, the embedding of concepts in expressions can be considered as a special case of definition. This can result in a dramatic reduction in the number of facts that have to be maintained explicitly. Frames does not have a notion of anonymous classes.

[2]Metaclasses are allowed in OWL Full.

Set Combination on Classes: OWL allows users to apply the standard set operators (Intersection, Union and Complement) on class descriptions. With the combination of the OWL class axioms, like equivalent class, we can model statements like covering, disjointness etc.

Characters of Properties: Frames allows uses to say that a slot is *functional* (by setting the maximum cardinality to be one) or *symmetric* (by setting the invert slot to be itself). In OWL, apart from *functional* and *symmetric*, the properties can be set as *transitive* as well. Statements like "If A is part of B and B is part of C, then A is part of C as well" which could not be modeled in Frames could be model in OWL. The transitive property is important for some application domains like anatomy and geography.

3 Tool Support

3.1 OWL DL

3.1.1 OWL reasoner

As we have discussed, one of the most important distinguishing features of logic based ontology languages like OWL from other ontology formalisms is that it has formal semantics. Staying within OWL DL, allows us to build reasoners which can make automatic inferences over our knowledge base. Reasoners can be used at development time to help users to build and manage their ontologies more easily. We believe that building large, reusable ontologies with rich multiple inheritance by manual effort unaided by formal checks is virtually impossible. This is doubly so in any collaborative environment where work from several authors must be reconciled and integrated. However, many of our users develop draft versions of ontologies without benefit of classifiers to which we propose "enrichments" with the help of classifiers and then return to the original authors for checking. At the end of the development process, we deliver most of the ontologies we build to applications in forms that do not require the applications to have classifiers or even know that classifiers have been used. In general, the use of OWL reasoners includes but is not limited to the following:

Diagnosis An OWL ontology is an engineered artifact which may inevitably contain flaws which may include logical inconsistency, unexpected subsumptions inference and unexpected type coercion for individuals. Using OWL reasoners directly the first kind of flaws can be automatically and efficiently identified. With tool support and additional interaction from users, the causes of the last two flaws can normally be discovered as well.

Composition OWL allows us to build more complex concepts out of simpler concepts – this is sometimes referred to as "Conceptual Lego". The fundamental advantage of using classifiers is that they allow composition i.e. they allow new concepts to be defined systematically from existing concepts. This can result in a dramatic reduction in the number of facts that have to be maintained explicitly.

Normalisation and Managing Polyhierarchies Managing and maintaining a complex polyhierarchy of concepts is hard. Changes often need to be made in several different places, and keeping everything in step is error prone and laborious. Using a classifier, we have developed a mechanism for 'normalisation' of ontologies. In normalised ontologies, even the most complex polyhierarchies are built out of simple non-overlapping trees. Concepts bridging the trees are created by definitions. The relationships amongst defined concepts are maintained by the classifier. Changes are always made in exactly one place.

Pre- and Post- Coordination and the Ontology Life Cycle Querying a knowledge base is one of the fundamental tasks needed in an ontology-driven application. Depending on the type of ontology and the type of queries that need to be asked, we can determine whether the reasoner is needed at run-time, or if it can be simply used at "publish" time. Many applications need an ontology primarily as a fixed vocabulary with fixed relations. They do not expect to define new concepts in terms of existing concepts "on the fly" at run time. What is required at run time is a "pre-coordinated" ontology in which all concepts needed and the relations between them are explicitly pre-defined. That the pre-coordinated ontology was, or was not, derived through normalisation and classification is irrelevant at run time. What is required is a hierarchy (or polyhierarchy) of concepts to "fill the boxes" in the information or decision support model. In this case the life cycle of the ontology is that it is authored, maintained and demonstrated correct in OWL with the help of the classifier, but then "published" as a pre-coordinated form in RDF(S), OWL Lite, CLIPS, or some proprietary form. On the other hand, some applications require more concepts than can be conveniently enumerated. Indeed the number of concepts that can be defined from the elements of an ontology may be indefinitely large or even demonstrably infinite. If it cannot be predicted in advance which concepts are needed, then the classifier is needed at run time to create and classify the concepts found.

3.2 Frames

3.2.1 Knowledge-Acquisition Forms

Class definitions in Protégé-Frames provide constraints on instances of those classes. Tools like Protégé-Frames use these constraints to guide and verify the knowledge-acquistion process. Protégé-Frames, for example, generates knowledge-acquisition forms based on Slot widgets in Protégé-Frames (the user- interface components for acquiring slot values) limit the number of choices that users get and flag the violation of constraints: for instance, Protégé-Frames will put a red border around a slot with more values than the maximum cardinality constraints allows. We can simulate similar knowledge-acquisition support for OWL, but this does not reflect the semantics of the language.

3.2.2 Constraint languages

Most frame-based systems lack the ability to express disjunction, existential quantification about frames, or relationships between properties of the same frame or different frames. For example, one cannot restrict a value of one property based on that of another property. A remedy for this limitation is the introduction of a more expressive constraint language based on First Order Logic (FOL) developed specifically to express these more complex relationships. In Protégé the Protégé Axiom Language (PAL) serves this purpose. PAL is a subset of first-order logic that allows users to express constraints on slot values while making the unique-name and closed-world assumptions.A plugin to Protégé enables users to find the instances in the knowledge base that violate the PAL constraints.

3.2.3 Query tab

Protégé provides different ways in which the user can query the content of an ontology. The **QueryTab** is part of the Protégé system and allows the user to retrieve instances in the ontology that match certain criteria. The queries are defined following the pattern *(class, slot, operator, value)* and can be composed in a conjunction or a disjunction of queries. The operator can be set according to the type of values that the slot may take. For instance, a slot of type *String* allows besides the *equals*, other lexical operators as well, like *contains* or *begins with*. The query engine is built-in Protégé-Frames and uses the closed world assumption, i.e. it will only return as true the things that have been explicitly asserted in the knowledge base. A query like, ”Give me all instances of *Pizza* that have the value of *hasCountryOfOrigin* slot *Italy* ” will only return the instances of pizzas that have this assertion explicitly made.

4 Frames or OWL: Guidelines

The many differences between frames and OWL could prove daunting to users who need to select an approach for their applications. As might be expected, in some cases, the frame representation with its closed-world semantics is a good fit for the application, while in other cases, the capabilities and expressiveness of OWL are needed to deliver the functional requirements.

In general Frames is particularly useful when the application has the following requirements:

- Creating ontologies for domains in which the closed-world assumption is appropriate.
- The application focuses on data acquisition.
- The application domain requires constraints on slot values.
- The model relates classes to other classes.

Likewise, the following requirements of applications may make OWL a preferred choice:

- Creating robust terminologies in which classes are defined.
- Need for DL reasoning to ensure logical consistency of ontologies.
- Controlled terminologies are published on the Semantic Web and accessed by other applications.
- Applications in which classification is a paradigm for reasoning.

5 Conclusion

In this paper, we compared the two most important ontology modeling paradigms – Frames and DL both in theory and practice. We hope that from this comparison users can understand the major difference between these two paradigms and be able to choose the most suitable modeling languages for their applications.

A Mechanism to Define and Execute SWRL Built-ins in Protégé-OWL

Martin J O'Connor , Amar Das
Stanford Medical Informatics, Stanford, CA 94305-5479

We have developed an extension to Protégé-OWL that provides mechanisms to define Java implementations of SWRL built-ins, to dynamically load these implementations, and to invoke them from a rule engine.

1. SWRL Built-Ins

SWRL[1] provides a very powerful extension mechanism that allows user-defined methods to be used in rules. These methods are called built-ins[2] and are predicates that accept one or more arguments. Built-ins are analogous to functions in production rule systems. A number of core built-ins are defined in the SWRL specification. This core set includes basic mathematical operators and built-ins for string and date manipulations. These built-ins can be used directly in SWRL rules. For example, the core SWRL mathematical built-in called `greaterThanOrEqual` can be used as follows to indicate that a person with an age of 18 or greater is an adult:

```
Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThanOrEqual(?age, 18) -> Adult(?p)
```

When executed, this rule would classify individuals of class `Person` with an `hasAge` property value of 18 or greater as members of the class `Adult`. The `swrlb` qualifier before the built-in name indicates the alias of the namespace containing the built-in definition. In this case, it indicates that the built-in comes from the core SWRL built-in ontology.

Users can also define their own built-in libraries. Example libraries could include built-ins for currency conversion, and statistical, temporal or spatial operations. Again, these user-defined built-ins can be used directly in SWRL rules.

An OWL definition for a SWRL built-in is provided by the class `Builtin`, which is contained in the files `swrl.owl` and `swrlb.owl`. These files have the namespace base `http://www.w3.org/2003/11/`. A new user-defined built-in is described in OWL as an instance of this class. The individual name is set to the name of the built-in. In general, a set of related built-ins are defined in a single OWL file. For example, a user-defined set of temporal built-ins could be defined in a file called `temporal.owl`. A specific built-in, such as, say, `before`, would then be defined in this file as an individual named `before`, which would be an instance of the class `Builtin`. The argument properties of each built-in can be used to specify the number arguments it is expecting.

To use these user-defined built-ins in SWRL rules, the file containing them must be imported. Sets of built-ins are usually given a user-friendly alias when they are imported. For example, the built-ins defined in `temporal.owl` could be give the alias `temporal` that can be used to qulaify their use in SWRL rules.

2. Defining Java Built-in Implementations

We have developed an extension to the Protégé-OWL SWRLTab[3],[4] called the *SWRL Built-in Bridge*[5] to provide support for defining and dynamically loading built-in implementation written in Java. Users wishing to provide implementations for a library of built-in methods must first define a Java class that contains definitions for all the built-ins in the library. The bridge is expecting this built-in implementation class to be called `SWRLBuiltInMethodsImpl`. This class must implement the

[1] http://www.w3.org/Submission/SWRL/
[2] http://www.daml.org/2004/04/swrl/builtins.html
[3] http://protege.stanford.edu/plugins/owl/swrl/
[4] M. J. O'Connor, H. Knublauch, S. W. Tu, B. Grossof, M. Dean, W. E. Grosso, M. A. Musen. Supporting Rule System Interoperability on the Semantic Web with SWRL. Fourth International Semantic Web Conference (ISWC2005), Galway, Ireland, 2005.
[5] http://www.w3.org/Submission/SWRL/BuiltInBridge.html

interface SWRLBuiltInMethods[6]. This interface acts as a typing or structuring mechanism - it does not define any methods itself.

The package name of the SWRLBuiltInMethods class should be the namespace qualifier of the built-ins appended to the Java package name edu.stanford.smi.protegex.owl.swrl.bridge.builtins. For example, the standard SWRL built-in swrlb:greaterThan should be defined as a method called greaterThan in the class SWRLBuiltInMethodsImpl, which should be located in the package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.swrlb.

Each implementation of a specific built-in in the SWRLBuiltInMethodsImpl class should have a signature of the form:

```
public static boolean <builtInName>(List arguments) throws BuiltInException
```

The single arguments parameter is a list that should contains one or more Argument objects[7]. The three possible types of argument objects expected by the bridge are LiteralInfo, IndividualInfo and VariableInfo. The LiteralInfo and IndividualInfo objects are used to pass information to built-ins: the LiteralInfo object contains OWL literals, such as integers or strings, and the IndividualInfo object specifies the name of an OWL individual. The VariableInfo class can be used by a built-in to assign a value to a variable. This value can be either an OWL individual name or a literal.

The three parameter classes have constructors to create them from their matching types. LiteralInfo objects can be constructed from RDFFSLiteral objects or from basic Java types. Accessor methods are provided to get these values. IndividualInfo and VariableInfo classes can be constructed from Java instances of OWLIndividual and SWRLVariable classes, respectively. Both of these classes also have a getName call to retrieve the variable or individual name.

For example, the SWRL rule atom swrlb:add(?x, 2, 3) could use the core SWRL add built-in to add two integer literals. When this built-in is invoked by a rule engine, the first argument should be an instance of the VariableInfo class with its name set to x; the second and third arguments should be instances of the LiteralInfo class that hold their respective values.

Each built-in class must declare the exception BuiltInException, which is defined in the exceptions subpackage of the standard bridge package. This abstract class has concrete subclasses for the four possible exceptions that can be thrown by a built-in implementation: (1) InvalidBuiltInArgumentNumberException, which is used indicate that an incorrect number of arguments have been passed to the built-in; (2) InvalidBuiltInArgumentException, which should be used used to indicate that an argument of the wrong type has been passed to the built-in; (3) LiteralConversionException, which can be used to indicated that literal argument is invalid in some way; and (4) BuiltInNotImplementedException, which can be used to indicate that a built-in (or variants of it for a particular argument type) has not been implemented.

Here is an example Java method defining a built-in called stringEquals from the core SWRL built-in library:

```
private static String STRING_EQUALS = "stringEquals";

public boolean stringEqualIgnoreCase(List arguments) throws BuiltInException {

  String argument1, argument2;

  SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(STRING_EQUALS, 2, arguments.size());

  argument1 = SWRLBuiltInUtil.getArgumentAsAString(STRING_EQUALS, 1, arguments);

  argument2 = SWRLBuiltInUtil.getArgumentAsAString(STRING_EQUALS, 2, arguments);

  return argument1.equals (argument2);

} // stringEquals
```

This method illustrates the use of a utility class called SWRLBuiltInUtil that can be used to process arguments and generate appropriate exceptions. In the above example, the checkNumberOfArgumentsEqualTo method will throw an InvalidBuiltInArgumentNumberException if two arguments are not passed to the built-in; the getArgumentAsAString method can be used to extract a string value from a supplied LiteralInfo object and will throw an InvalidBuiltInArgumentException if both supplied arguments are not strings. Most of the methods in this utility class take the name of the built-in as their first parameter so that the offending built-in name can be displayed if an error is thrown.

[6] This class is defined in the Protégé-OWL package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.
[7] This class is defined in the Protégé-OWL package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.

3. Invoking a Built-in Method from a Rule Engine

A built-in methods can be invoked by a rule engine through the `SWRLRuleEngineBridge` class. The constructor for this class[8] takes an instance of an `OWLModel` class that contains the relevant knowledge base for the rules being executed, in addition to the rules themselves. To support built-in invocation, this class has a method called `invokeSWRLBuiltIn` that takes the name of a built-in and a list of `Argument` objects for that built-in. It returns a boolean value that holds the result of the built-in invocation. The built-name passed to the invoke method must be of the form `<builtInLibraryAlias>:<builtInName>`. For example, the `add` method in the core SWRL built-in library would be referred to as `swrlb:add`.

The invoke method itself can directly throw three possible exceptions: (1) `InvalidBuiltInNameException`, which is used to indicate that a supplied built-in name is not a valid name for a SWRL built-in, i.e., no OWL individual of class `BuiltIn` with the name of the built-in exists in the OWL model supplied to the bridge; (2) `UnresolvedBuiltInException`, which indicates that no Java implementation method for the built-in could be found in any of the dynamically loaded built-in libraries; and (3) `InvalidBuiltInMethodsImplementationClass`, which indicates that the Java implementation class found for the built-in did not correctly implement the interface `SWRLBuiltInMethods`.

If an implementation is found for the supplied built-in, it is invoked and is supplied with the argument list passed to the invoke method. As discussed above, a built-in implementation can throws four possible exceptions (which are also subclasses of the `BuiltInException` class). If an exception is throws it is passed directly back to the caller. If the method executes successfully, its (boolean) return value is passed back from the invoke method.

Rule engines that wish to access this invocation mechanism are responsible for creating an instance of the `SWRLRuleEngineBridge` class[9] and calling the invoke method at run-time as rules are executed. The mechanism that connects invocations of built-ins from inside a rule engine to the bridge's invoke method is going to be rule engine specific.

It is worth noting that there is going to be considerable overhead to invoking built-in implementations from inside a rule engine. In addition to marshalling built-in arguments, the process of invoking external methods from inside a rule engine during rule evaluation can dramatically slow down its execution. This overhead may not be dramatic if the rule engine is Java-based, but for non Java-based engines the overhead could be significant. In general, if a particular built-in is expected to be used a lot, a native rule engine implementation of the method should be developed. Many engines have in libraries of in-built methods available that can be used in this implementation process.

4. Loading a Built-in Implementation Class at Runtime

Sets of related built-ins are contained in the same `SWRLBuiltInMethodsImpl` class. For example, the implementation class containing the above `stringEquals` method can then be defined as follows:

```
package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.swrlb;

import edu.stanford.smi.protegex.owl.swrl.bridge.builtins.*;

import edu.stanford.smi.protegex.owl.swrl.bridge.exceptions.*;

public class SWRLBuiltInMethodsImpl implements SWRLBuiltInMethods {

  public boolean stringEquals(List arguments) throws BuiltInException { ... }

  ....

} // SWRLBuiltInMethodsImpl
```

To allow Protege-OWL to find this built-in implementation classes at run time it must first be placed in a JAR file. This JAR file should then be placed in the Protege-OWL plugins directory. Protege will automatically add this JAR file to the applications class path so that a class loader will be able to find this class at run time. The bridge employs a lazy loading mechanism: When a built-in from a particular implementation class is invoked for the first time, the bridge loads the implementation class using Java's class loader. Any subsequent invocations of built-ins from the class will be routed directly to the loaded class. The Java package name of the built-in implementation classes

An example `SWRLBuiltInMethodsImpl` class that implements most of the core SWRL built-ins can be found in the `edu.stanford.smi.protegex.owl.swrl.bridge.builtins.swrlb` package in the standard Protege-OWL distribution.

[8] This class is defined in the Protégé-OWL package `edu.stanford.smi.protegex.owl.swrl.bridge`.
[9] Described here: http://smi-web.stanford.edu/people/moconnor/swrl/RuleEngineBridge.html.

Integrating data into an OWL Knowledge Base via the DBOM Protégé plug-in

Raphaël Squelbut, Olivier Curé
Université de Marne-la-Vallée, Laboratoire ISIS
5, boulevard Descartes
Marne-la-Vallée 77454 France
Email: (ocure,rsquelbu)@univ-mlv.fr

Abstract

This paper presents a Protégé plug-in which enables end-users to design and instantiate an OWL knowledge base from multiple existing relational databases. This approach is inspired by data integration and exchange solutions and presents some functionalities which facilitate the development and maintenance of Semantic Web applications from frequently updated and domain-concerned databases.

1 Introduction

The omnipresence of information technologies is leading to the increase of available data sources. This situation motivates many application designers to access and combine several data sources within a unique solution. The approach we have adopted to tackle this problem consists in merging data contained in the sources into one single, materialized model. The main motivation of this solution is that data sources may be frequently updated and not always accessible at application run-time. In such a context, a data integration approach [2], with a virtualized target, is not adapted because replying to queries will not be possible as data sources are not all accessible on-demand.

A peculiarity of our approach is characterized by the target model we have adopted : a Description Logics (DL) [6] compliant with the Semantic Web, namely OWL DL. It is well-understood that the success of the next generation web partly depends on the availability of efficient ontological engineering tools. The implemented framework proposes to process the following set of ontological engineering's tasks : (i) creation of an ontology schema from the schemata of multiple relational databases (DBs), (ii) instantiation of the knowledge base (KB) with tuples from the sources.

In the context of practical and expressive ontologies, these tasks are considered complex, time-consuming and financially expensive because they require a collaboration between knowledge engineers and domain experts. Thus approaches aiming to facilitate the design of ontologies may be valuable for organizations willing to implement Semantic Web applications.

The system we have developed, named DBOM (DataBase Ontology Mapping), tackles very expressive ontologies, those enabling the expression of general logical constraints [3]. This choice is motivated by our need to semantically enrich the data contained in relational DBs and fulfils a need to implement practical and efficient inference enabled services which are based on application-dependent ontologies. We believe that DBOM's features may encourage the design of ontologies based on practical databases exploited in the "Deep Web" and thus accelerate the adoption of the Semantic Web.

The first application implemented using DBOM's functionalities is a medical informatics web application named XIMSA (eXtented Interactive Multimedia System for Automedication) [1]. In this system, it is essential to integrate several data sources, mostly related to drugs (the French Health Products Safety Agency (AFSSAPS) drug database, ATC/DDD system, etc.), in order to design a KB which enables inferences on drugs and symptoms related to self-medication.

A first non graphical version of DBOM was limited to processing XML maping files in order to create a DL TBox and instantiate the DL ABox. In the context of our medical informatics application, mapping files are usually created and maintained by health care professionals. These tasks, usually performed within text editors, are considered error-prone and painful for end-users. Thus we aimed to develop a graphical user interface via a Protégé plug-in. Additionally, with this approach, many new functionalities, not available in the previous DBOM implementation, are provided and help the mapping designers during their tasks.

The remainder of this article is organized as follows. Section 2 presents the available solution of DBOM and its principle. Our plugin and its features are described in Section 3. Section 4 provides a discussion and presents some future work on this plug-in.

2 The DBOM Framework

In DBOM, the design of an ontology is created from the DB conceptual schema in a semi-automatic way. End-users familiar with the DB schema are responsible for the design of the mapping file. Still all the other operations are processed automatically : creation of the TBox, instantiation and maintenance of the ABox.

The DBOM system is supported by a migration system which is a set of formulas linking a set of source schemas of DBs and the target ontology schema formalized in OWL DL. The DBOM system is supported by a migration system $\mathcal{MI}$ which is a triple $(\mathcal{S}, \mathcal{O}, \mathcal{M})$, where :

- $\mathcal{S}$ is a set of source schemas of relational DBs.
- $\mathcal{O}$ is the (target) ontology schema formalized in OWL DL.
- $\mathcal{M}$ is a set of formulas of a language $\mathcal{L}_{\mathcal{M}}$ over $\mathcal{S}$ and $\mathcal{O}$.

The definition of $\mathcal{MI}$ emphasizes relations with data exchange [5] and data integration [2] systems. We now contrast the DBOM approach with the comparison of data exchange and integration provided in [4] :

- as in both data exchange and integration, the source schemas are given and the mapping is a set of formulas constructed by a human expert.
- as in data integration, the ontology (target) schema is a reconciliation of the sources and is constructed from the processing of the source schemas given a mapping.
- as in data exchange, the target instances are materialized, while they are virtualized in the case of data integration.

In this work, we consider that data stored at the sources are always locally consistent, meaning that they respect their set of integrity constraints. Anyhow, the integration of these autonomous and consistent sources may result in an inconsistent KB. This is due to violations of integrity constraints specified on the target schema.

In the related domain of data integration, two approaches are proposed to deal with inconsistent data : (i) a procedural approach which is based on domain-specific transformation and cleaning, (ii) a declarative approach. In this last approach, information integration semantics given in terms of *repairs* is usually proposed to solve inconsistency.

Our method to deal with inconsistent data adopts a declarative approach which exploits information on view preferences. In this solution, we consider that all sources do not have the same level of reliability and we enable the mapping designer to set confidence values over the views of the mapping. Using these information at mapping processing time, there is a motivation to prefer values coming from a view wrt data retrieved from another view.

The structure of the mapping imposes to define (i) a prolog which contains the namespaces used in the target ontology schema (declared in Protégé's *Metadata* tab) and bindings to the database sources, (ii) the body of the mapping which contains concrete members and their associated SQL queries.

We refer to "members" of the mapping as the set of concepts, denoting sets of objects, and object properties, denoting binary relations between objects. We make the distinction between concrete and abstract members. The comprehension of concrete and abstract members is relatively straightforward as it is equivalent to the assumption made in Object-Oriented Programming. Thus instances (individuals) can be created for a concrete concept and a concrete object property assertions relate two existing individuals. Abstract members can not be instanciated and they aim to design a hierarchy of members where final (leafs in a tree representation) members should be concrete. In the context of the DBOM Protégé plug-in, the abstract members are defined in the *OWL Classes* and *Properties* tabs while the concrete members are declared and defined in the *DBOM* tab. The creation of concrete classes and properties are made with SELECT data manipulation operations of the SQL language.

3 The DBOM plug-in

The DBOM plug-in proposes to load DBs in the DBOM widget tab (fig.1 point 1) and represents the related schemas using a tree shape (fig.1 point 2, this area also proposes a view of the ontology tree). Thus all informations (tables, attributes and their types, primary keys, etc..) on the DB are efficiently represented in the main Protégé window.

The declaration of concrete members is processed in the following order

- choose the type of the member to create (concept or object property in area 3 of fig.1),
- optionally define a super arbitrary (meaning that this can be either concrete or abstract) member for this member. In the case of an object property, the end-user has to declare the arbitrary concepts related to the domain and range of this role (area 4 of fig.1),
- type an SQL query to be associated to this member (area 6). Several queries (views), usually defined over different sources, can be associated to a given member with the button of area 5. Each view can be accessed and maintained with the tab of area 5.
- optinally define a confidence value associated with this member's view (area 7). Intuitively, confidences on the views of a given member define a partial order on the views. This order serves to instantiate consistently the DL ABox at mapping processing time.
- bind to each elements in the SELECT clause of this query a previously defined datatype property (area 8 of fig.1).

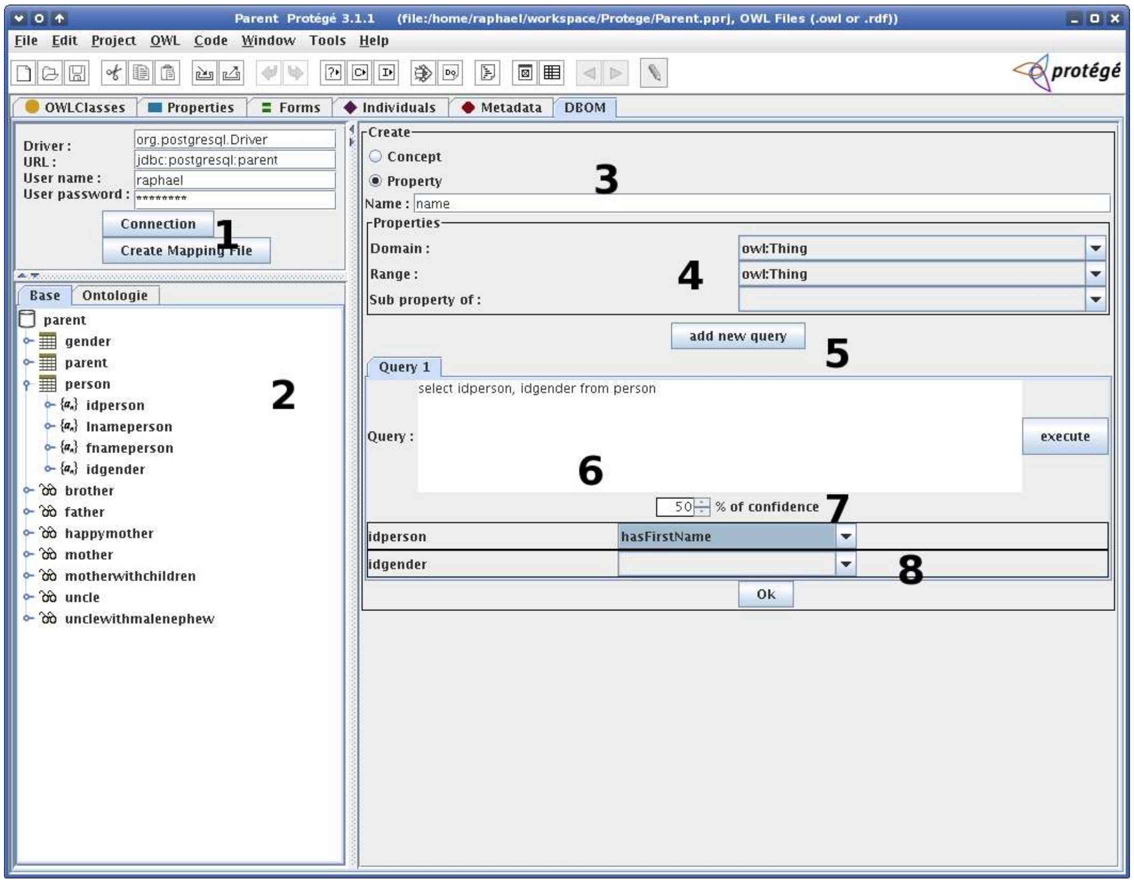
Interactions between Protégé tabs enable to envision and enrich the TBox. Once the end-user is done with the mapping, she has the opportunity to save the TBox and the mapping file. The mapping instance can then be processed to instantiate the DL ABox.

4 Discussion

In this paper, we have introduced a solution to enable the migration of data stored in multiple data sources to a DL knowledge base. Because data from the sources may not be retrieved and shared on-demand at query time, we have opted for a materialization of the migrated data. In order to deal with possible cases of inconsistencies, we support the setting of confidence values over the views of the mapping. Based on a *global-as-view* (GAV) approach, our solution uses a notion of possible answers, corresponding to credulous entailment, in order to instantiate the ABox.

In order to ease the creation of mapping files, the DBOM Protégé plug-in has been implemented. Although additional tests still need to be conducted with end-users, we are already very positive about the first results. The integration of our approach in the Protégé faciliates and accelerates the design of mappings for end-users comfortable with this ontology editor and knowledge-base framework (creation of abstract members, axiomatizations, determine the OWL sublanguage, etc.).

The creation of this plug-in opened new perspectives with the possibility to apply DBOM functionalities to existing ontologies. This approach was not possible with the previous stand alone DBOM implementation : the mapping was responsible for the creation from scratch of the TBox.



Driver : org.postgresql.Driver
URL : jdbc:postgresql:parent
User name : raphael
User password : ********

Create: ( ) Concept [✓ Property]
Name : name

Properties
Domain : owl:Thing
Range : owl:Thing
Sub property of :

Query : select idperson, idgender from person

50 % of confidence

idperson	hasFirstName
idgender	

Figure 1: DBOM widget tab

In order to enhance the user-friendliness of the plug-in, we are currently implementing a Query By Example (QBE) solution to design queries associated with concrete members. This solution will be based on interactions with the DB trees of area 2 (fig.1) during the definition of members' views.

References

[1] Curé, O. : *XIMSA : eXtended Interactive Multimedia System for Auto-medication* Proceedings of IEEE Computer-Based Medical Systems (CBMS) 2004. pp 570-575

[2] M. Lenzerini : *Data integration : a theoretical perspective.* Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp 233-246, 2002.

[3] Gómez-Pérez, A., Fernández-López, M., Corcho, O. *Ontological Engineering.* Springer-Verlag. November 2003.

[4] Fagin, R., Kolaitis, P., Miller, R., Popa, L. : *Data exchange : semantics and query answering* Proceedings of ICDT 2003. LNCS 2572. pp 207-224.

[5] P. Kolaitis : *Schema mappings, data exchange, and metadata management* Proceedings of ACM Symposium on Principles of Database Systems (PODS), pp 61-75, 2005.

[6] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. : *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press (2003).

AVICENA, ontology for the design of executable clinical practice guidelines

Alfonso Díez, Alberto Cortés, Rafael Gil (BET), Pablo Castells, Fernando Díez, José Mª Fuentes, Álvaro Valera (UAM/EPS)

BET Value
{adiez,rgil,acortes}@betvalue.com
http://www.betvalue.com
Universidad Autónoma de Madrid, Escuela Politécnica Superior (UAM/EPS)
{fernando.diez,chema.fuentes,pablo.castels}@uam.es
alvaro.valera@estudiante.uam.es
http://nets.ii.uam.es

Abstract

Clinical practice guidelines (CPG) are increasingly more demanded as a necessary tool in the supply of health care. In spite of their continuous evolution and spreading, their usage in the context of the new information technologies is still not enough developed. There is much work to be done in the area of execution of CPG, and, consequently, it is difficult to find repositories of formal and executable CPG. The project AVICENA is a new proposal of an ontology for the modelling of executable guidelines, that (1) creates new ways for the representation of clinical knowledge and (2) allows the management of the dynamics of health-related processes.

Motivation

Medical organizations are today doing a very high pressure in order to standardize clinical processes, which have grown very much in the last decades, both in complexity and quantity. Factors such as universal access to health resources, ubiquity in the provision of services, growing complexity and specialization of medical organizations, and higher costs, between others, and driving the health supply system to adopt more efficient approaches, based in information technologies and managed business processes. The development of this new entrepreneurial organization is based in a mixture of new methods and techniques: standardisation and personalization of clinical activities, cost-effectiveness analysis, fast adaptability to new practices and resources, evidence based practices, effective management of knowledge, etc.

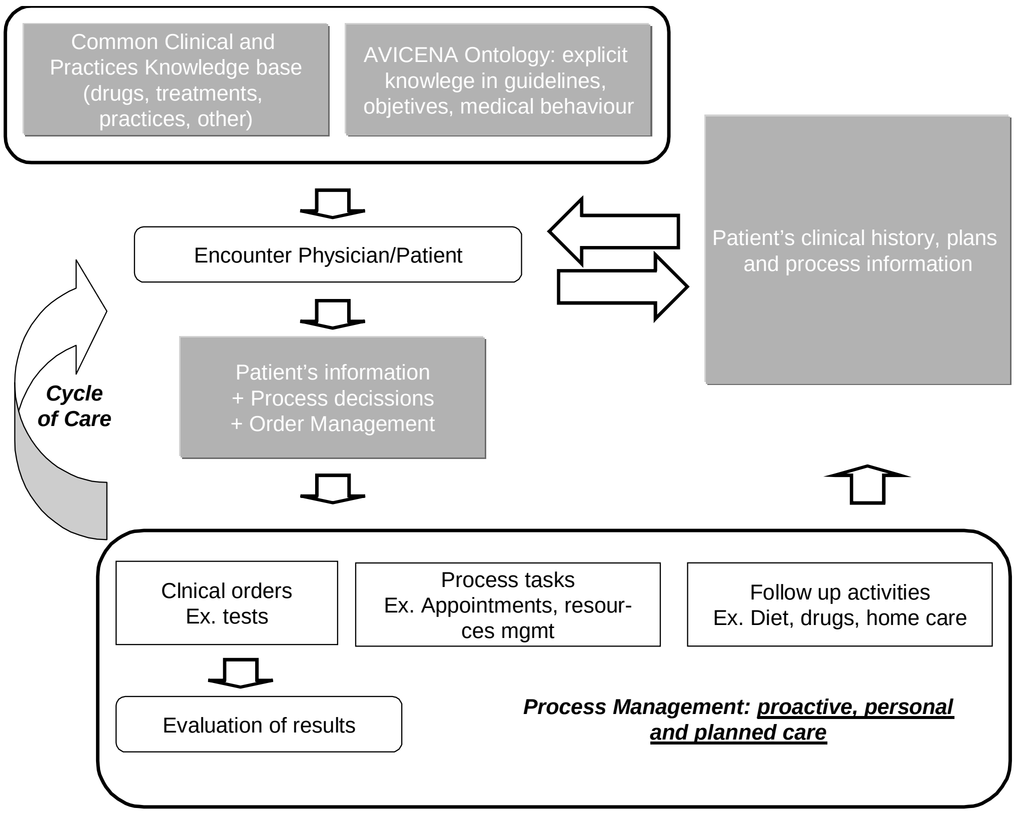
The standardization of practices is largely solved with the definition and use of clinical guidelines and pathways, that determine the expected behaviour and clinical outputs for the different classifications of diagnosis, techniques or symptoms, enhancing the capability of the medical system to provide care in an efficient way. The benefits are very clear:

- Reduce care variance: medical attention is guided by standard procedures, derived from clinical evidences, and local policies.
- Enhance quality of attention: optimise the use of resources and the cost-efficiency ratio, eliminate unnecessary tests, reduce hospitalisation time and optimise quality and time used for encounters
- Personalize patient attention. Guides define and control the concrete behaviour of the whole care system for each single patient, not for the complete universe of all possible patients. Therefore, the attention turns to be proactive and planned, instead of reactive and spontaneous.
- Using guides, the medical system distributes knowledge and maintains homogeneous and up-to-date procedures. New evidences are added to guides, and this knowledge is quickly disseminated along the organization.

Project scope

The AVICENA project was started in 2005 by BET and UAM with the support of the Spanish Ministry of Industry, Tourism and Commerce (FIT-350300-2005-33). The objective of the project is to build a family of ontologies and systems for performing health care activities by means of the management of processes and knowledge. The project joints two areas of systems development: the capability to define and represent knowledge (clinical, organizational), and the ability to manage business processes based on that knowledge.

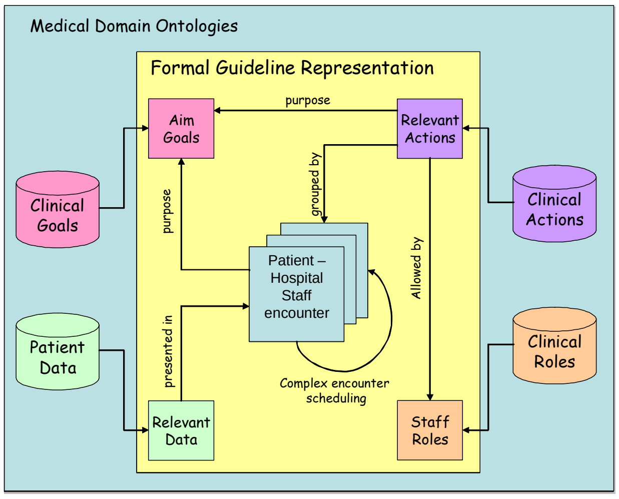
The following diagram summarises the scope of the project, with their different components and working areas:


Common Clinical and Practices Knowledge base (drugs, treatments, practices, other)
AVICENA Ontology: explicit knowlege in guidelines, objetives, medical behaviour
Encounter Physician/Patient
Patient's clinical history, plans and process information
Cycle of Care
Patient's information + Process decissions + Order Management
Clnical orders Ex. tests
Process tasks Ex. Appointments, resour-ces mgmt
Follow up activities Ex. Diet, drugs, home care
Evaluation of results
Process Management: proactive, personal and planned care

The system is designed to interact in the care process at its very centre: the relation between the doctor or nurse and the patient, making a coherent view on the various domains of information related: the process information about the patient, the clinical history, the management of orders, and the reference clinical information used.

AVICENA ontology

AVICENA is an ontology for the representation of executable clinical guidelines that surges from the application of Semantic Web concepts to the fields of health and process representation. This ontology is defined in OWL, and has been developed using Protégé. The following diagram illustrates the composition of a clinical guideline.



AVICENA introduces various areas for the representation of the guides:

– The encounters, and the encounters planning network, that allow managing a non-linear approach in the process.
– The clinical objectives, that permit to trace a different path of activities depending on the evaluation (qualitative, quantitative, maybe automated) of certain variables and on its evolution
– The role of the person that is attending the patient, to adapt to different situations with doctors, nurses, other support personnel, etc.
– The interoperability (via semantic web) of the process with other systems around: clinical history, order processing, tests results, etc.
– The plan and distribution of activities along the persons, groups and organization(s) involved in the supply of services for the patient

Other main areas of interest taken on account in AVICENA, although not still developed, are reasoning and inference (within the guide, and in relation with the process and patient data), reusability, management of clinical objectives and versioning.

Process Guidelines

The whole business in health care has the core procedure in the contact of the physician and the patient: the encounter. Each time the doctor/nurse meets the patient, the situation is totally evaluated, the history of the patient and his current situation, test results, treatments on course, along with all the related information about the process type, the state-of-the-art, the local usual practices, etc. Decisions taken in the encounter will drive the future of the patient and of the medical organization(s) responsible of the care. AVICENA defines encounters of two types: the most classical discrete (when it occurs in a definite moment of time) and continuous, where the encounter takes some time (maybe days) to reach an objective and finish, usually adapted to hospital treatment.

The encounter has three domains of information: (1) The knowledge used by the physician, which can be represented in a knowledge base repository; (2) The AVICENA ontology where all formal clinical strategies and procedures are defined and (3) the specific clinical and process data for that particular patient. These data is composed of annotations in the HCE, the information produced by the process on its own, and all reference data related with order processing and results.

Process Management

This approach of the relation between the physician and the patient allows to manage the consequences, using a Business Process Management System (BPMS), coupled with the ontology. Using the information provided by the ontology, the system is able to asses physician in the ways for the attention to the patient, to define and distribute activities along the organization, to create and track activity plans for that patient, and to track order processing and evaluate or distribute results.

Each encounter type (a subset of a guideline) becomes a node in a graph of pathways. Each path is a strategy to treat the specific problem o a given patient. Therefore, the doctor determines a planned strategy each time he/she meets the patient. Those strategies can be thought as a superset of a pathway, and are a common policy of the organization where the assistance is provided.

The care cycle is therefore managed using a process perspective, that is, each single patient is an instance of a managed process, which comprises the situations, singularities, resources and plans. Thus, the attention is personal, proactive, controlled and planned. This has very deep impact in the way care is delivered, the quality of the service, in the efficient use of resources and in the control of costs.

Current work done

The ontology and its conceptualization has been developed as an abstraction of various works done in clinical organizations: Adeslas (a health insurance company with over two million customers) and Hospital de la Ribera (a large private hospital) specifically in the areas of cardiovascular risk control, diabetes, and HIV with some thousands patient’s processes currently under control.

Conclusions and future work

AVICENA is a valuable approach to the state-of-the-art of representation of clinical processes. First, adds new concepts in this area, like the definition of encounters, the use of objectives to define treatment paths, and the incorporation of Support Processes to the definition of the care cycle. Secondly it imports techniques from other areas, like Semantic Web, to cover recently discovered needs, such as the execution of guides in distributed and heterogeneous organizations.

AVICENA core components have passed the research phase, and currently we are in development phase, profiting from the prototypes already in use.

We have identified some kinds of processes with evident and direct benefits: preventive treatments for risk populations, geriatrics, pregnancy, patients with multiple pathologies, and chronic diseases, between others, although the ontology is not limited in its usage to any specific type of process.

Grading lung tumors using OWL-DL based reasoning

Olivier Dameron[1], Élodie Roques[1], Daniel Rubin[2], Gwenaëlle Marquet[1] and Anita Burgun[1]

[1]UPRES-EA 3888, Université de Rennes1, France
[2]Stanford Medical Informatics, Stanford University School of Medicine, Stanford CA 94305, USA
`olivier.dameron@univ-rennes1.fr`

1 Introduction

The treatment and prognosis for a tumor depend to a large extent on its stage. The goal of this article is to analyze to what extent tumor grading can be performed automatically using the OWL-DL description logic language. We focused on the grading of lung tumors. Section 2 is a review of the authoritative cancer ontology, in which we conclude that the NCIT has to be extended in order to perform automatic tumor grading. Section 3 is a description of the TNM classification used for defining the grade of a tumor and the anatomical concepts that we used. Section 4 compares how Description Logic's class-based classification and instance-based classification support tumor grading.

2 The NCI thesaurus

The NCI Thesaurus[1] (NCIT) is a controlled vocabulary designed to meet the needs of the cancer research community [1, 2]. Its goal is to provide definitions for basic and clinical concepts used in cancer research. These definitions, the taxonomic structure of the thesaurus and the presence of explicit relationships between classes make the NCIT more than a controlled vocabulary.

Several works analyzed the NCIT as a terminology and as an ontology. They identified limitations concerning the term-formation principles, the missing or inappropriately assigned verbal and formal definitions [3], as well as classification, synonymy, relations and definitions [4].

The NCIT was originally designed using a proprietary description-logic-based languages and has been converted into OWL-Lite [2]. The current version is composed of 41,717 named classes, all of which are primitive classes, i.e. they can have constraints, but do not have any necessary and sufficient definition that can be used to infer that an individual is an instance of this class.

The published limitations (although some of them have been fixed, and other are on the process of being fixed) and the lack of defined classes, specially for the T, N and M criteria make the NCIT useless for our grading purpose. As we focus on the OWL-DL language capabilities, rather than on the model, it was easier to devise our own ontology, drawing inspiration from the NCIT whenever possible.

[1]`ftp://ftp1.nci.nih.gov/pub/cacore/EVS/`

3 Methods: OWL-DL Ontology for tumor grading

The TNM Classification of Malignant Tumours[5] (TNM) is the cancer staging system developed and maintained by the International Union Against Cancer (UICC) and the American Joint Committee on Cancer (AJCC) to maintain consensus on one globally recognized standard for categorising cancer[2].

The TNM criteria for staging tumors involve (among others) the anatomical location of the primary tumor and of its possible metastasis as well as the involved lymph nodes. Therefore, different types of tumors (for example lung tumors or colon tumors) have different sets of TNM criteria. Section 3.1 describes the criteria used for grading lung tumors. Section 3.2 presents the anatomical entities used for describing tumor locations.

3.1 The T, N and M axis and the grades

Staging a tumor is a two steps process.

The first step consists in giving a score along three axis describing the tumor (from Tis to T_4), the spreading into lymphatic nodes (from N_0 to N_3) and the possible metastasis (M_0, M_1).

The second step consists in determining the stage according to the previous scores. Each combination of a score on the T, N and M axis define the unique stage of the tumor, from 0 to IV. Note that several combinations of T, N and M scores can lead to the same stage.

For lung tumors, we used the lung carcinoid tumors staging guide[3].

We used the previous descriptions to provide necessary and sufficient definitions for the various T, N and M criteria, and for the various stages.

3.2 Anatomical entities

Some of the T, N and M criteria refer to anatomical entities as location landmarks. In order to address the difference of granularity between clinical descriptions and the definitions of the staging criteria, it is important that our model contains not only the anatomical entities mentionned in the staging criteria, but also the various direct and indirect parts of these entities.

In order to demonstrate this principle, we selected some anatomical entities of interest such as Heart and Lung, for which we added direct and indirect parts. For this anatomical decomposition, we drew inspiration from the Foundational Model of Anatomy (FMA) [6], which is a canonical model of healthy human anatomy. We were not trying to be exhaustive in the description of anatomy, as we expect that it would require to deal with other problems such as the automatic extraction of the relevant portion of the FMA (including pruning classes with a too small level of granularity such as cells), or such as dealing with the great number of additional classes.

The staging criteria definitions were adapted in order to support such parthood descriptions.

4 Results: Mechanisms for grading tumors

The generated OWL-DL ontology was composed of 123 classes, among which 66 were defined. There were 57 classes representing anatomical entities.

This ontology was imported by an ontology in which simulated patient conditions were represented using classes, and by another ontology in which the same conditions were represented using individuals.

In both cases, the classification time ranged from a few seconds to a few minutes.

The files for the ontology and the test cases are available online[4].

[2]`http://en.wikipedia.org/wiki/TNM`

[3]`http://www.cancer.org/docroot/CRI/content/CRI_2_4_3X_How_is_lung_carcinoid_tumor_staged_56.asp?sitearea=`

[4]`http://www.med.univ-rennes1.fr/ dameron/ontology/tnmClassification/`

4.1 Class-based reasoning

For performing class-based reasoning, we represented clinical situations by creating specific subclasses for each entity. Depending on the goal, the reasoning process consisted either in retrieving the inferred subclasses of a particular stage, or in retrieving the inferred superclass of a tumor that is also a stage.

We devised 240 tests, all of which were correctly classified.

4.2 Instance-based reasoning

For performing instance-based reasoning, we represented clinical situations by creating instances of the corresponding classes. Depending on the goal, the reasoning process consisted either in retrieving the inferred instances of the class representing a particular stage, or in retrieving the inferred type of the tumor individual that is also a stage.

We devised 150 tests, all of which were correctly classified.

5 Discussion

We demonstrated that automatic staging of lung tumors can be performed using OWL-DL classification. First, this result was obtained by providing logical definitions to a limited number of classes. Second, this result shows that applications can be expected to reuse leverage the symbolic knowledge represented in separate ontologies. The logical and computational feasibility of reusing (portions of) existing ontologies such as the NCI Thesaurus, the Foundational Model of Anatomy and possibly bioinformatics ontologies [7] remains to be studied.

We also highlighted some of the limitations of description logics-based classification for this task of tumor staging.

- Particularly, the open-world assumption makes it clumsy to perform instance classification (though theoretically possible). In practice, when the patient's record mentions that the primary tumor is located in the lower lobe of the right lung and involved the right hilar lymph nodes, it implicitly indicates that no other lymph nodes were involved and that there were no detectable distant metastasis. If we do not represent this assumption, the open world assumption will make it impossible for the classifier to infer that this is a N_1 situation, as there might be unspecified involved lymph nodes somewhere else, or some unspecified distant metastasis. It would be possible to make the individual representing the tumor an instance of the class of tumors involving ipsilateral lymph nodes and having no distant metastasis, but such classes and the corresponding closure would have to be generated on the fly. The problem is that in this context, we try to perform some closed-world based reasoning.

- The previous limitation can easily be solved by modeling the patient condition using classes, for which we can add closure constraints. Admittedly, this is tweaking the OWL-DL possibility for achieving a computational goal and is not very elegant. However, we can also consider that this method adequately represents that fact that the class of all the tumors located in the lower lobe of the right lung and involving only the right hilar lymph nodes are N_1 tumors.

- Another limitation of OWL-DL is the impossibility to use numeric range constraints for representing "the cancer is larger than 3cm" or "the tumor is closer than 2 cm to the point where the trachea branches into the left and right main bronchi". This makes impossible the automation of tumor staging using only description logics reasoning. An additional functionality has to be added to the system that makes the individual or the class representing a patient's tumor an instance or a subclass of `TumorEqualOrBiggerThan3cm` or of `TumorSmallerThan3cm` depending not the actual value of its size.

- For our model to be complete, we should have included classes for all the possible anatomical entities (for example by generating them from the FMA). This would have lead to a major decrease of the classification performance in terms of computing time and of memory footprint. Moreover, the FMA is a model of healthy anatomy, so we should even have doubled the classes for considering all the possibly abnormal entities.

We noticed that the way we modeled things mattered. For example, it was easier to define N_3 and to reuse its definition for N_2 and N_1, rather than to start with the definition of N_1 and to have to handle complex closures for N_2 and N_3.

Although we have not actually done it, we assume that our approach could be applied to other kinds of tumors and that similar conclusions would have been reached. The definitions we provided for T_0... T_4; N_0... N_3 and M_0,M_1 are specific to lung tumors. They should be renamed T_0-Lung... in order to avoid confusion between the name of these classes and there logical definitions. Other definitions could be defined for other kinds of tumors, and they should be named accordingly T_0-Colon...

Acknowledgments

This work was partially funded by the SMI, Stanford University, while Olivier Dameron was a postdoctoral fellow.

References

[1] Jennifer Golbeck, Gilberto Fragoso, Franck Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. The national cancer institute's thésaurus and ontology. *Journal of Web Semantics*, 1(1):75–80, 2003.

[2] Franck W. Hartel, Sherri de Corronado, Robert Dionne, Gilberto Fragoso, and Jennifer Golbeck. modeling a description logic vocabulary for cancer research. *Journal of Biomedical Informatics*, 38:114–129, 2005.

[3] Werner Ceusters, Barry Smith, and Louis Goldberg. A terminological and ontological analysis of the NCI thesaurus. *Methods of Information in Medicine*, 44:498–507, 2005.

[4] Anand Kumar and Barry Smith. Oncology ontology in the nci thesaurus. In *Proceedings of the Artificial Intelligence in Medicine Europe conference AIME 2005*, pages 213–220, 2005.

[5] Vincent T. DeVita, Samuel Hellman, and Steven A. Rosenberg. *Cancer: Principles and Practice of Oncology.* Lippincott Williams and Wilkins, 7th edition, 2005.

[6] C. Rosse and J.L.V Mejino. A reference ontology for bioinformatics: the foundational model of anatomy. *Journal of Biomedical Informatics*, 36:478–500, 2003.

[7] Anand Kumar, Yum Lina Yip, Barry Smith, and Pierre Grenon. Bridging the gap between medical and bioinformatics: An ontological case study in colon carcinoma. *Computers in Biology and Medicine*, page In press, 2005.

Knowtator: A plug-in for creating training and evaluation data sets for Biomedical Natural Language systems

Philip V. Ogren

Division of Biomedical Informatics

Mayo Clinic College of Medicine

Rochester, MN, USA

Ogren.Philip@mayo.edu

Abstract

A general-purpose text annotation tool called Knowtator is introduced. Knowtator facilitates the manual creation of annotated corpora that can be used for evaluating or training a variety of natural language processing systems. Building on the strengths of the Protégé knowledge representation system, Knowtator has been developed as a Protégé plug-in that leverages Protégé's knowledge representation capabilities to specify annotation schemas. Knowtator's unique advantage over other annotation tools is the ease with which complex annotation schemas (e.g. schemas which have constrained relationships between annotation types) can be defined and incorporated into use. Knowtator is available under the Mozilla Public License 1.1 at http://bionlp.sourceforge.net/Knowtator.

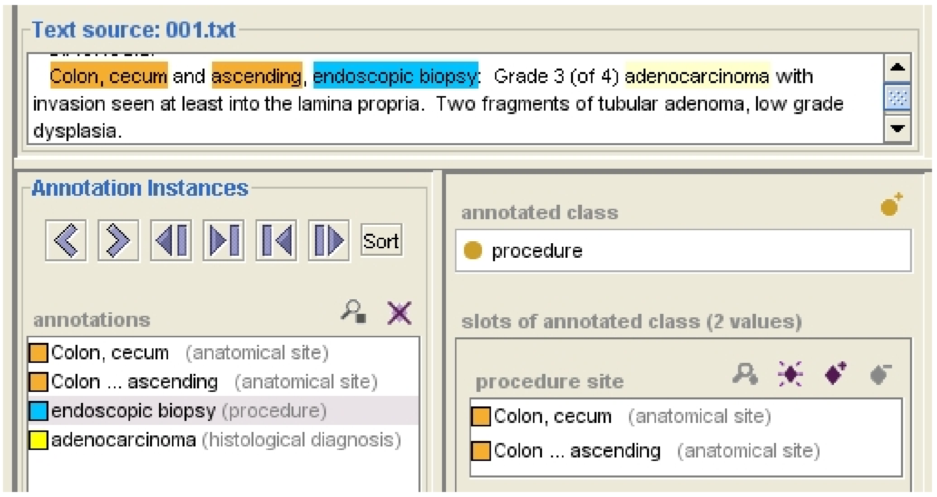
1 Introduction

Knowtator is a general-purpose text annotation tool for creating annotated corpora suitable for evaluating or training Natural Language Processing (NLP) systems. Such corpora consist of texts (e.g. documents, abstracts, or sentences) and *annotations* that associate structured information (e.g. POS tags, named entities, shallow parses) with extents of the texts. The annotations correspond to a gold standard data set that can be used to directly compare against NLP system output. Alternatively, the annotations can be used as input to an NLP system's training algorithms. An *annotation schema* is a specification of the kinds of annotations that can be created. Knowtator provides a very flexible mechanism for defining annotation schemas. This allows it to be employed for a large variety of corpus annotation tasks.

Knowtator has been implemented as a Protégé plug-in and runs in the Protégé environment. In Knowtator, an annotation schema is defined with Protégé class, instance, slot, and facet definitions using the Protégé knowledge-base editing functionality. The defined annotation schema can then be applied to a text annotation task without having to write any task specific software or edit specialized configuration files. Annotation schemas in Knowtator can model both semantic (e.g. protein-protein interactions) and linguistic phenomena (e.g. coreference resolution).

2 Related work

There exists a plethora of manual text annotation tools for creating annotated corpora. While it has been common for individual research groups to build customized annotation tools for their specific annotation tasks, several text annotation tools have emerged in the last few years that can be employed to accomplish


Text source: 001.txt

Colon, cecum and ascending, endoscopic biopsy: Grade 3 (of 4) adenocarcinoma with invasion seen at least into the lamina propria. Two fragments of tubular adenoma, low grade dysplasia.

Annotation Instances

annotations

- Colon, cecum (anatomical site)
- Colon ... ascending (anatomical site)
- endoscopic biopsy (procedure)
- adenocarcinoma (histological diagnosis)

annotated class: procedure

slots of annotated class (2 values)

procedure site

- Colon, cecum (anatomical site)
- Colon ... ascending (anatomical site)

Figure 1 Sample annotations capturing mentions of diagnostic procedure for colorectal cancer.

a wide variety of annotation tasks. Some of the better general-purpose annotation tools include Callisto[1], WordFreak[2], GATE[3], and MMAX2[4]. Each of these tools is distributed with a limited number of annotation tasks that can be used 'out of the box.' Many of the tasks that are provided can be customized to a limited extent to suit the requirements of a user's annotation task via configuration files. In Callisto, for example, a simple annotation schema can be defined with an XML DTD that allows the creation of an annotation schema that is essentially a tag set augmented with simple (e.g. string) attributes for each tag. In addition to configuration files, WordFreak provides a plug-in architecture for creating task specific code modules that can be integrated into the user interface.

A complex annotation schema might include hierarchical relationships between annotation types and constrained relationships between the types. Creating such an annotation schema can be a formidable challenge for the available tools either because configuration options are too limiting or because implementing a new plug-in is too expensive or time consuming.

3 Implementation

3.1 Annotation schema

Knowtator approaches the definition of an annotation schema as a knowledge engineering task by leveraging Protégé's strengths as a knowledge-base editor. Protégé has user interface components for defining class, instance, slot, and facet frames. A Knowtator annotation schema is created by defining frames using these user interface components as a knowledge engineer would when creating a conceptual model of some domain. For Knowtator the frame definitions model the phenomena that the annotation task seeks to capture. An annotation schema consists primarily of a target ontology (see §3.2.1 below) but also contains additional configuration information such as the annotator's names and class color assignments.

As a simple (and hypothetical) example, an annotation schema that captures mentions of diagnostic procedures for the colorectal cancer domain could be modeled with the classes *procedure* and *anatomical site*. A slot called *procedure site* is constrained to be an instance of *anatomical site* and captures the anatomical location where the procedure took place. Annotations in Knowtator using this simple schema are

[1] http://callisto.mitre.org
[2] http://wordfreak.sourceforge.net
[3] http://gate.ac.uk/. GATE is a software architecture for NLP that has, as one of its many components, manual text annotation functionality.
[4]http://mmax.eml-research.de/.

shown in Figure 1. An annotation for the text ‘endoscopic biopsy’ is associated with the class *procedure* and is the selected annotation in Figure 1.

A key strength of Knowtator is its ability to relate annotations to each other via the slot definitions of the corresponding annotated classes. In the example the *procedure site* slot relates the *procedure* annotation with the annotations corresponding to the class *anatomical site* for the texts ‘Colon, cecum’ and ‘Colon … ascending.’ The constraint on the slot ensures that the values of the *procedure site* slot are filled with annotations corresponding to *anatomical sites.*

Protégé is capable of representing much more sophisticated and complex conceptual models which can be used, in turn, by Knowtator for text annotation.

3.2 Knowledge Model

The Knowtator data model consists of three components: 1) an ontology that defines the concepts that are the target of the annotation task 2) a set of instances that capture assertions about relations between and properties of concepts mentioned in text, and 3) a mapping between the target text and members of 2.

3.2.1 Target Ontology

The set of class, instance, slot, and facet frames that define the set of named entities and relations that are the subject of the annotation task is called a *target ontology* and is roughly equivalent to an annotation schema (described in §3.1 above.) A target ontology models the semantic and/or linguistic phenomena that are found in or described by the target texts. The annotation schema has no dependencies on any Knowtator specific class definitions (e.g. classes in the target ontology do not inherit slots from Knowtator specific classes.) The target ontology provides a conceptual model of the target domain and could be used independently of Knowtator.

3.2.2 Concept Mentions

Human language is not constrained to conform to a formal knowledge model regardless of how carefully constructed it[5] might be. Rather than assume that target texts will conform to a pre-defined model of a domain, the Knowtator data model provides a mechanism to describe mentions of concepts with respect to a knowledge model. The term *concept mention* is defined as a description of a concept that has been found in the target text. A *concept mention* provides a means for describing how a concept (e.g. a class or instance frame) is being discussed in text and the assertions that can be drawn from the text about that concept (e.g. relationships between other mentioned concepts or properties of the concept). While Knowtator utilizes the annotation schema to help the human annotator make consistent annotations, it is not constrained by the data model to create annotations that strictly adhere to the target ontology with respect to the constraints that have been defined therein.

3.2.3 Annotations

Annotations provide a mapping between the target texts and concept mentions that have been created. An annotation consists of a concept mention, a span (or spans) of text, and other book keeping information such as the annotator that created the annotation, the date the annotation was created, and a pointer to the target text. The annotation definition is customizable so that additional book keeping information can be captured if desired.

For NLP tasks such as named entity recognition and relationship extraction the Knowtator data model provides a clean separation between the concepts that are being talked about and how they are being talked about. However, for modeling linguistic phenomena (e.g. a deep parse) this clean separation may

[5] Ambiguous anaphora intentional.

not always make sense (i.e. the extent of text in question may actually be an instance of noun phrase rather than a mention of one.) Knowledge engineering issues such as these can be addressed by Knowtator to some extent but such discussion is outside the scope of this extended abstract.

3.3 Features

In addition to its flexible annotation schema definition capabilities, Knowtator has many other features that are useful for executing text annotation projects. A consensus set creation mode allows one to create a gold standard using annotations from multiple annotators. First, annotations from multiple annotators are aggregated into a single Knowtator annotation project. Annotations that represent agreement between the annotators are consolidated such that the focus of further human review is on disagreements between annotators.

Inter-annotator agreement (IAA) metrics provide descriptive reports of consistency between two or more annotators. Several different match criteria (i.e. what counts as agreement between multiple annotations) have been implemented. Each gives a different perspective on how well annotators agree with each other and can be useful for uncovering systematic differences. IAA can also be calculated for selected annotation types giving very fine grained analysis data.

Knowtator provides a pluggable infrastructure for handling different kinds of text source types. By implementing a simple interface, one can annotate any kind of text (e.g. from xml or a relational database) with a modest amount of coding.

Knowtator provides stand-off annotation such that the original text that is being annotated is not modified. Annotation data can be exported/imported to/from a simple XML format.

Annotation filters can be used to view a subset of available annotations. This may be important if, for example, viewing only named entity annotations is desired in an annotation project that also contains many part-of-speech annotations. Filters are also used to focus IAA analysis and the export of annotations to XML.

Knowtator can be run as a stand-alone system (e.g. on a laptop) without a network connection. For increased scalability, Knowtator can be used with a relational database backend (via JDBC).

Knowtator and Protégé are provided under the Mozilla Public License 1.1 and are freely available with source code at http://bionlp.sourceforge.net/ Knowtator and http://protege.stanford.edu, respectively. Both applications are implemented in the Java programming language and have been successfully deployed and used in the Windows, MacOS, and Linux environments.

4 Conclusion

Knowtator has been developed to leverage the knowledge representation and editing capabilities of the Protégé system. By modeling syntactic and/or semantic phenomena using Protégé frames, a wide variety of annotation schemas can be defined and used for annotating text. New annotation tasks can be created without writing new software or creating specialized configuration files. Knowtator also provides additional features that make it useful for real-world multi-person annotation tasks.

Using a Degree-of-Interest Model for Adaptive Visualizations in Protégé

Tricia d'Entremont Margaret-Anne Storey

Department of Computer Science
University of Victoria
Victoria, BC Canada
http://www.thechiselgroup.org
email: tdent@uvic.ca; mstorey@uvic.ca

Abstract

Visualizations are commonly used as a cognitive aid for presenting large ontologies and instance data. One challenge with these visual techniques is that the generated views are often very dense and complex. It is difficult to know which concepts to include in the visualization to meet a user's information needs. In this talk, we present recent work that proposes using an attention-reactive interface to provide adaptive visualizations in Protégé. This furthers our recent work in providing visualization "on demand" for maintenance, editing, and understanding tasks by drawing users' attention to concepts of interest within the context of the current task.

1. Introduction

Understanding and maintaining the structure of large ontologies is a cognitively demanding task. Over the last several years the CHISEL group at the University of Victoria has been working on developing advanced visual interfaces to help users browse and understand large, complex ontologies.

Our main ontology visualization tool, Jambalaya, is an integration of the SHriMP[1] (Simple Hierarchical Multi-Perspective) visualization toolkit with Protégé [1]. In Jambalaya, the ontology is represented as a graph where classes and instances are depicted as nodes, and relationships between the classes are represented as directed arcs. Jambalaya provides multiple, inter-changeable views of the graph structure allowing users to explore multiple perspectives of information at different levels of abstraction.

As the size and complexity of the ontology grows, however, the usefulness of Jambalaya's advanced visualizations and Protégé's standard views decreases. Users report that they often work for extended periods of time on a small subset of the ontology. The concepts relevant to their task become difficult to locate because non-interesting concepts consume valuable screen real estate, obscuring the interesting concepts as illustrated in Figure 1.

Within Protégé's class browser, for example, the concepts of interest given a particular task may not be visible. Users are therefore forced to spend considerable effort navigating the ontology by scrolling, expanding and collapsing nodes in order to find concepts of interest.

2. Approach

To address this problem and to help users find concepts of interest within the ontology, we are developing a plug-in for Protégé which applies principles of attention-reactive user interfaces to provide adaptive visualizations within Protégé

Attention-reactive interfaces consist of two components: a mechanism to continuously calculate the user's degree of interest (DOI) and a dynamic display of the information that uses the DOI calculation to draw users' attention to interesting elements in order to reduce navigation overhead [2].

1. http://www.thechiselgroup.org/shrimp

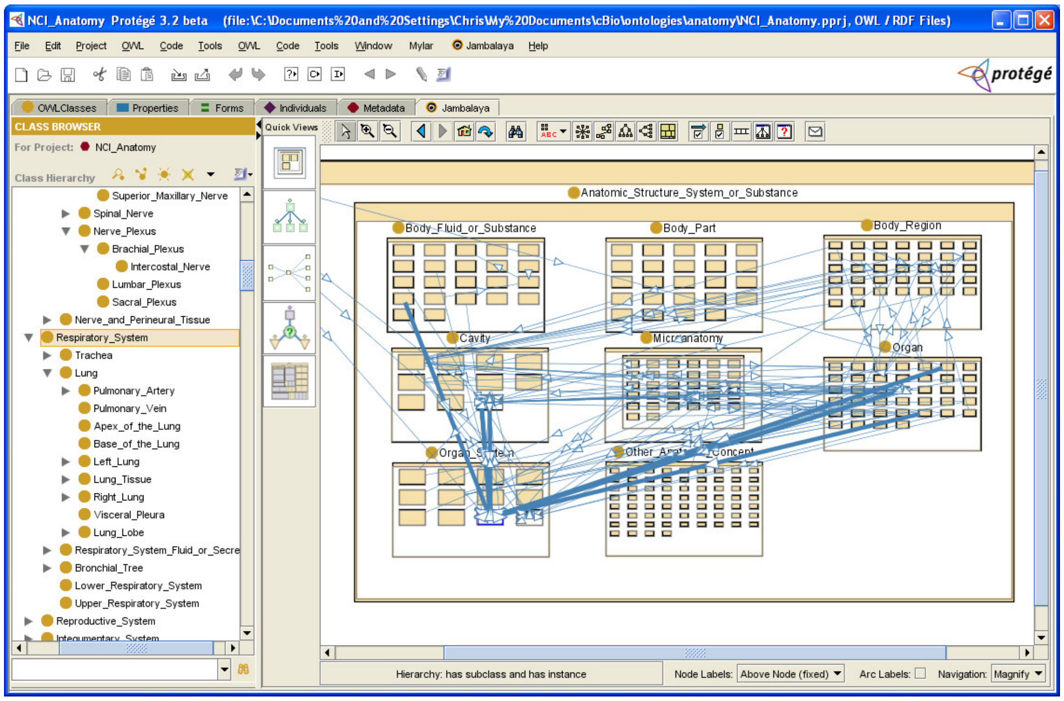


Figure 1: Protégé with the Jambalaya Tab selected displaying a portion of the NCI Thesaurus

The approach will be applied to both the standard tree browser views in Protégé as well as to the graphical based views in Jambalaya.

To calculate the user's DOI within the ontology, we have adapted Mylar[2], a plug-in for the Eclipse[3] integrated software development environment to work with Protégé [3]. Within Eclipse, Mylar monitors the programmer's activities, computes the users' DOI of the various code entities, and adapts the respective Eclipse views using the DOI calculation. To integrate Mylar with Protégé, we have written our own monitor to track user activity within the ontology and to pass that information to Mylar.

Mylar's DOI model associates an interest value with each concept in the ontology. When a concept is selected or edited, its DOI value increases. The DOI calculation also contains a *decay* function which decreases an element's interest value if it has not been selected.

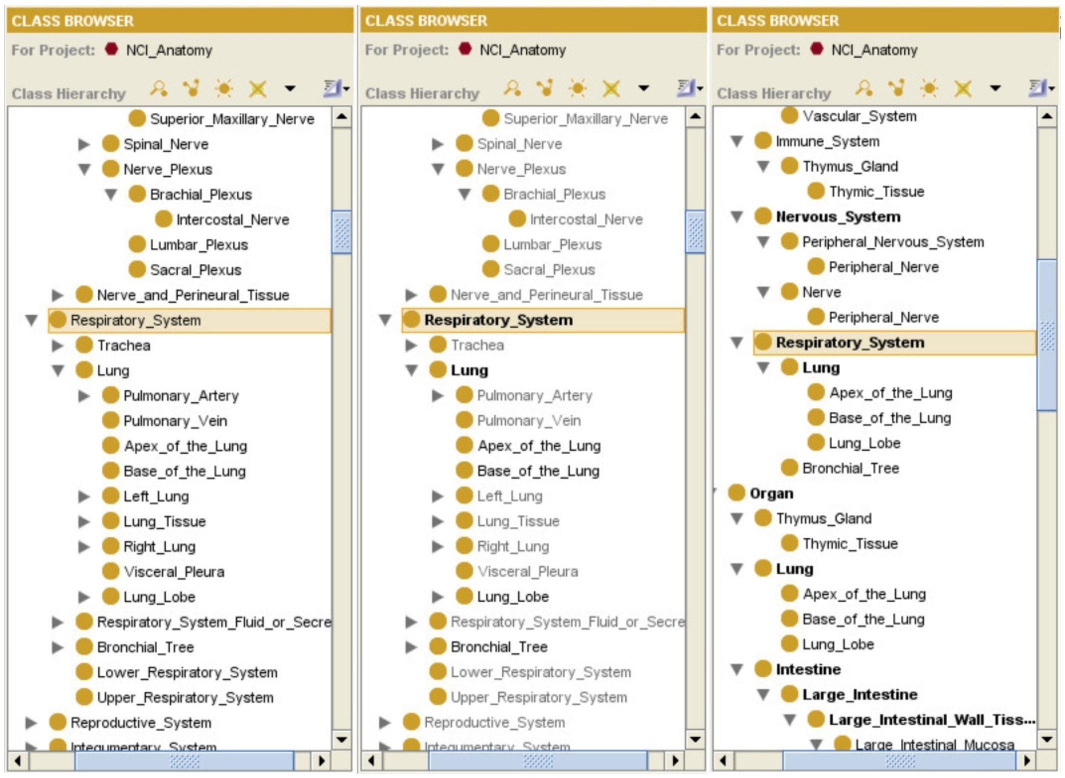
We use the DOI calculation to adapt the appropriate views in Protégé to reduce navigational overhead and to draw user's attention to concepts of interest within the context of the current task. A primary consideration in the design of these adapted views has been to provide lightweight, easily reversible mechanisms to focus user's attention without deviating significantly from the existing, familiar Protégé views.

3. Adaptive Visualizations

Within Protégé's class, owl, and instance browsers, a user's DOI over the concepts is visualized using font weight and font color as shown in Figure 2b. Non-interesting concepts, ones that the user has not selected or whose DOI value has decayed to zero, are displayed in gray. *Landmark* concepts, those with a high DOI calculation or which the user has manually specified to be a landmark, are highlighted in black, bold text. Interesting concepts, those that have been selected but whose calculated DOI value falls below a threshold value are shown in black. To provide for finer granularity, we are also exploring the use of font size as a mechanism for highlighting a user's DOI.

---

2. http://www.eclipse.org/mylar/
3. http:// www.eclipse.org



(a) (b) (c)

Figure 2: a) Standard Protégé class browser b) Highlighting concepts in the class browser. c) Highlighting and filtering concepts in the class browser.

In addition to displaying the user's DOI by highlighting the concepts in the different views, we provide a mechanism for filtering non-interesting concepts from the view as shown in Figure 2c. Although it is still possible for the view to display a vertical scrollbar, the number of items through which the user must search can be drastically reduced thereby decreasing the time spent finding relevant concepts. Users can also specify that a concept is no longer of interest, the associated DOI value is updated correspondingly, and, if filtering is enabled, the concept is removed from the view.

4. Ongoing and Future Work

As a first step toward investigating this approach, we are designing studies of users' interaction patterns within Protégé and Jambalaya. Throughout our work in this domain, we have been concerned with which tasks could benefit from visualization support and when visualization support should be provided [4]. These studies will provide important insight into these issues as well as a basis for preliminary evaluation of the impact the adapted views have on users' navigation.

Our proposed user studies will consist of two phases. During the first phase, users' interactions within the ontology will be monitored using the standard views provided by Protégé and Jambalaya. The second phase will involve monitoring user activity with the adapted views. The goal of these studies will not only be to evaluate and refine the approaches we have presented here but also to discover additional situations in which an attention-reactive interface may provide cognitive support.

This work is in its earliest stages, and it is very important to us to get feedback from the Protégé user community. During our presentation, we will lead a discussion on how these adapted visualizations may, or may not, be helpful.

Our future work will explore ways to adapt the advanced visualizations in Jambalaya given a user's DOI model. Beyond highlighting concepts using color or size, we will explore using motion to make elements above a certain degree of interest "pop out" from the graph using motion techniques [5].

In addition, we are interested in investigating the possibilities of sharing DOIs among users, for example, sharing an expert's DOI with a novice to provide guidance.

5. Acknowledgements

This work was supported by the National Center for Biomedical Ontology, under roadmap-initiative grant U54 HG004028 from the National Institutes of Health.

We would like to acknowledge the work of Mik Kersten and the rest of the Mylar project team.

Chris Callendar and other members of the CHISEL group have been instrumental in this work.

References

1. M. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R.Fergesen, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. *In Workshop on Interactive Tools for Knowledge Capture*, Victoria, B.C. Canada, October 2001.
2. S. Card and D. Nation. Degree-of-interest trees: A component of an attention-reactive user interface. In *International Conference on Advanced Visual Interfaces (AVI02)*, 2002.
3. M. Kersten and G. Murphy. Mylar: a degree-of-interest model for IDEs. In *Proceedings of the 4th international conference on Aspect oriented software development*, pages 159–168, New York, NY, USA, 2005. ACM Press.
4. N. Ernst, M. Storey, and P. Allen. Cognitive support for ontology modeling. *Int. J. Hum.-Comput. Stud.*, 62(5):553-577, May 2005.
5. C. and R. Bobrow. Motion to support rapid interactive queries on node-link diagrams. *ACM Transactions on Applied Perception*. 1: 1-15, 2004.

OntoSphere3D: a multidimensional visualization tool for ontologies

Alessio Bosca, Dario Bonino
Politecnico di Torino, Torino, Italy
{alessio.bosca, dario.bonino}@polito.it

Introduction

Visualizing knowledge in two dimensions is a challenging task, since many dimensions are involved: instances, concepts, hierarchical relations just for naming a few. And it is also a well studied paradigm that already produced good solutions such as: Jambalaya [1, 2], GraphViz [3], OntoViz [4], etc. Nevertheless, mapping the many dimensions involved by an ontology, on only two dimensions can sometimes be too restrictive, especially in the case of very large and complex knowledge domains.

In this presentation, we propose a novel approach for inspecting and editing ontologies using a visually enriched 3-dimensional space. Ontology information is represented on a 3D view-port merging structural information (i.e. concepts and relations) with context information (the amplitude of *Is-A* hierarchies rooted in a given ontology concept,…) by means of visual cues. Being the 3-dimensional view quite natural for humans, especially for what concerns navigation, the proposed approach aims at being more effective in browsing ontological data than the currently available 2-dimensional solutions. Improvements are obtained by involving direct manipulation operations such as zooming, rotating, and translating objects, and by introducing more dimensions to convey information on the visualized knowledge model as the color or the size of visualized entities.

The proposed work aims also at tackling space allocation issues for ontology visual models, in fact, in the traditional solutions, big ontologies can easily lead to overcrowded representations that are difficult to browse and that can be more confusing than aiding. Some attempts exist to overcome these problems, as in OntoRama [5,6], where the nodes being inspected are magnified with respect to the other nodes in the ontology. However, even these approaches tend to collapse when visualizing big ontologies such as SUMO [7], counting over than 5'000 concepts. The proposed application, instead adopts a dynamic collapsing mechanism and different views, at different granularities, for granting a constant navigability of the rendered model.

Proposed Approach

A three-dimensional environment is the starting point of the proposed ontology visualization tool, as a 3D space offers one more dimension than traditional 2D approaches to represent ontology data, so simplifying its interpretation. In addition many more dimensions are added to improve completeness and readability of the representation. Two main principles guide the visualization effort: increasing the number of "dimensions" (colors, shapes, transparency, etc.) which represent concepts features and convey additional information without adding the burden of further graphical elements (such as labels) on the scene, and automatically identifying the part of knowledge to be displayed and the detail level to be used in the process, on the base of user interaction with the scene. The latter principle is particularly important for improving the overall system performances since scale factor indeed constitutes a strong issue in visualizing complex graph structures like ontologies. As the cardinality of elements increases, the number of items to be concurrently displayed on the screen worsens the graphical perception of the scene and complicates spotting details. When the amount of visualization space needed to represent all the information within the KB outnumbers the space available on the screen, a few options remain available: to scale down the whole image to the detriment of readability, to present on the screen just a portion of it and allow its navigation or to summarize the information in a condensed graph and provide means for exploration and expansion. As the effectiveness of these options depends on the use case involved (consistency checking, domain comprehension, KB updates) a combined usage of them offers a suitable approach.

To combine seamlessly the options above, taking advantage of their strength points whenever possible, the proposed solution exploits different scenes that present and organize the information on the screen according to differently detailed perspectives. Such scenes interchange in managing the graphical space as user attention shifts from one concept to another, by implicitly inferring the focus from user's interaction with the scene (e.g., a concept selection with a mouse click). In this way, the idea of "focusing" as the application capability of highlighting the elements of interest while leaving out the others, is applied.

The OntoSphere3D [8] user interface is rather minimalist and allows direct manipulation of scenes through rotation, panning and zoom; it allows to browse the ontology as well as to update it and to add new concepts and relations (taking advantage of functionalities provided by the Protégé framework in which is deployed). Every concept within a given scene is clickable with two different results: a single left-click maintains the current perspective and simply navigates through elements, while a double-click leads to a focusing operation, shifting the scene to a more detailed level. Right-clicking on a concept that has direct instances visualizes them, while the same action applied to an instance bring the focus back on the related concept (see section 3.3) .

Root Focus Scene

This perspective presents a big “earth-like” sphere bearing on its surface a collection of concepts represented as small spheres (Figure 1). The scene does not visualize any taxonomic information and only shows direct “semantic” relationships between elements of the scene, usually a graph not fully connected. Atomic nodes, the ones without any subclass, are smaller and depicted in blue while the others are white and their size is proportional to the number of elements contained in their own sub-tree.

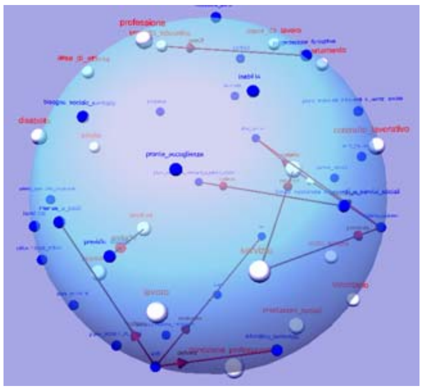


Figure 1. The Root Focus scene.

This view is particularly intended for representing the primitive concepts (i.e., the roots), but can also be used, during the ontology navigation, to visualize direct children of a given node; a pretty useful option in case of heavily sub-classed concepts. Representing primitive concepts within the ontology, and the relations between them, allows to easily identify the conceptual boundaries of the represented domain and provides a very good hint to the question: “what’s the ontology about?”

Tree Focus Scene

This scene shows the sub-tree originating from a concept; it displays the Is-A hierarchy as well as other semantic relations between represented classes. Since experimental evidence proves that too many elements on the screen, at the same time, hinder user attention, the scene completely presents only three fully-expanded levels at a time. As the user browses the tree, the system automatically performs expansion and collapse operations in order to maintain a reasonable scene complexity (Figure 2). Collapsed elements are coloured in white and their size is proportional to the number of elements present in their sub-tree; instead concepts located at the same depth level within the tree have the same colour in order to easily spot groups of siblings. *Is-A* relations are displayed with a neutral colour (grey), without labels, whereas other semantic relations involving concepts already in the scene are displayed in red, and are accompanied by the name of the relation. When an element of the scene is related to a node that is not present on the view-port, a small sphere is added for the hidden node in the proximity of the given element, so terminating the end of the arrow; in such cases, incoming relations are represented with a green arrow, while outgoing links with a red one.

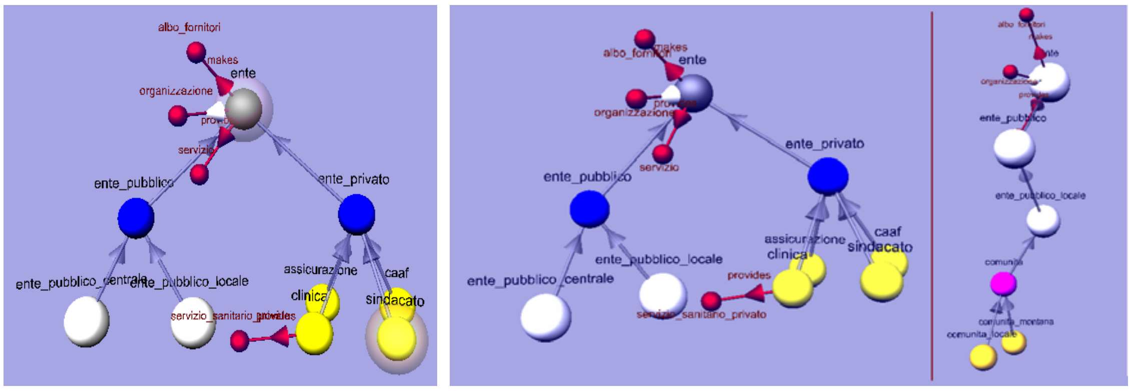


Figure 2. The Tree Focus scene.

Concept Focus Scene

In the concept focus scene, all the available information about a single concept is provided, at the highest possible level of detail. The concept’s children and parent(s) are therefore shown as well as its ancestor root(s) and its semantic relations, including the inherited ones. Semantic relations are drawn as arrows terminating in a small sphere: red if the relation is outgoing and green otherwise. Direct relations are drawn close to the concept, with an opaque color, while inherited ones are located a bit farther from the center and depicted with a fairly transparent color.

This scene can be extremely useful during consistency checking operations because it eases the spotting of inconsistent concepts or relations, e.g. whenever a concept inherits from an ancestor a property that "logically" contrasts with other features of its own

Instance Focus Scene

Whenever a concept has direct instances (in the tree focus scene or in the concept focus scene) its sphere is depicted surrounded by a transparent sphere resembling a sort of a shell (Figure 3). By right-clicking the concept, its direct instances are shown, using the same representation paradigm adopted by the concept focus scene. The resulting view represents instances and their properties as inter-connected cubes.

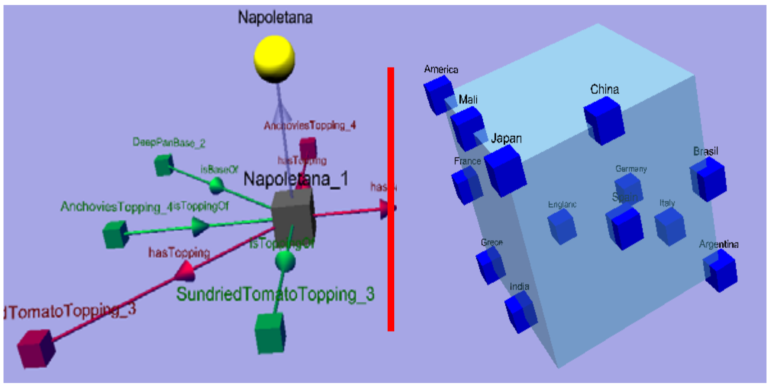


Figure 3 Instance Focus Scene

Results

In order to confirm the initial claim of the proposed work (i.e. improved navigability and inspection capabilities with respect to 2-dimensional approaches) the authors set up an efficiency comparison between 4 different visualization tools: the proposed OntoSphere3D plug-in, the Jambalaya plug-in, the OWLViz plug-in and the TgViz plug-in. These plug-ins have been tested against a predefined set of ontology related operations, namely: visualization of the top concepts, visualization of the relations between the top concepts, visualization of concepts located at level *n* in the *Is-A* hierarchy of the ontology, visualization of the concepts related to a given one, visualization of relations between concepts at the same hierarchy level, navigation of the ontology from one concept to another, search for a given concept.

The required Protégé-related skills have also been taken into account in the evaluation. Each operation has been assigned a predefined difficulty score, as reported in Table 1. Evaluation results are, instead, reported in
Table 2.

Table 1 Difficulty scores for several user interactions.

User interaction	Difficulty score
Mouse click	2
Mouse double-click	2
Look at the screen	0
Mouse over	1
Mouse scroll	3
Search (filling a form)	4

Table 2. Results of efficiency evaluation on 4 different visualization plug-ins.

Operation	Jamb.ya	On.3D	OWLViz	TViz
User Experience	1	1	2	2
Top concepts	0	0	3	6
Relations between top concepts	1	0	isA = 0 not-isA = ∞	1
Level–n concepts	n	2n	n.d.	n.d.
Related concepts	4	0	isA = 0 not-isA= 7	0
Relations between concepts at the same level	1	0	isA=0 not-isA=7	1
Concept navigation	2	2	0-3	0-3
Search	10	6	4	6
Jamb.ya = Jambalaya, On.3D = OntoSphere3D, TViz=TgViz				

It is easy to notice that, in most cases the proposed approach outperforms the other applications, except for visualizing concepts at a given level *n* in the ontology hierarchy, for which *n* mouse clicks are required, and for searching concepts, where the offered functionality is the one of the Protégé framework as for TgViz. In concept navigation however, data is quite difficult to compare since both Jambalaya, OWLViz and TgViz require scrolling for navigating between ontology concepts. According to the evaluation grid in Table 1, this is not a too heavy task but, when the ontology size grows up from few tens of concepts to several thousands, the required scrolling may become much more cumbersome and thus shall probably be re-weighted. On the contrary, the OntoSphere3D behavior is size-independent, becoming more suitable on really big ontologies such as SUMO.

References

[1] M.A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. Noy. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In Workshop on Interactive Tools for Knowledge Capture, Victoria, B.C. Canada, October 2001.

[2] Jambalaya. http://www.thechiselgroup.org/chisel/projects/jambalaya/jambalaya.html.

[3] Gansner, E. R. & North, S. C. (1999), An open graph visualization system and its applications to software engineering, Software Practice and Experience 30(11),.

[4] OntoViz Tab: Visualizing Protégé Ontologies. http://protege.stanford.edu/plugins/ontoviz/ontoviz.html.

[5] P.W.Eklund, N.Roberts, S.P.Green, OntoRama: Browsing an RDF Ontology using a Hyperbolic-like Browser, *The First International Symposium on CyberWorlds (CW2002)*, pp.405-411, Theory and Practices, IEEE press, 2002

[6] OntoRama. http://www.ontorama.com/

[7] Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.

[8] http://sourceforge.net/projects/ontosphere3D

Cognitive support for an argumentative structure during the ontology development process

Alexander Garcia Castro[1,2,3,4,§], Angela Norena[5], Andres Betancourt[5], Mark A. Ragan[2,4]
[1] *International Center for Tropical Agriculture*
[2] *Institute for Molecular Bioscience, The University of Queensland, Brisbane, Qld 4072 Australia*
[3] *Australian Center for Plant Functional Genomics*
[4] *Australian Research Council Centre in Bioinformatics*
[5] *Javeriana University, Engineering Facultity*
agcastro@cgiar.org, m.ragan@imb.uq.edu.au, {amnorena, andresbetancourt@puj.edu.co}

Abstract: Structuring and supporting the argumentative process that takes place within the knowledge elicitation process is a major problem when developing ontologies. Knowledge elicitation relies heavily on the argumentative process amongst domain experts. The involvement of geographically distributed domain experts and the need for domain experts to lead the design process, adds an interesting layer of complexity to the whole process. We consider that the argumentative structure should facilitate the elicitation process and serve as documentation for the whole process; it should also facilitate the evolution and contextualization of the ontology. We propose the use of conceptual maps as means to support and scaffold an argumentative structure during the development of ontologies within loosely centralized communities.

1. Introduction

The applications of knowledge engineering are growing larger and more systematic, now encompassing more ambitious ontologies—sizes in the hundreds of thousands of concepts will not be uncommon [1]. Furthermore, the development of those ontologies is usually a participatory exercise in which different experts interact *via* virtual means, resembling thereby a loosely centralized community. We believe the requirements of the Semantic Web (SW) bring with it an associated need for enhanced cognitive support in those tools we use.

Cognitive support is used to leverage innate human abilities, such as visual information processing, to increase human understanding and cognition of challenging problems [2]. Developing ontologies in loosely centralized environments as those described by Pinto *et al.* [3] poses challenges not previously considered by most existing methodologies. This user-centric design relies heavily on the ability of domain experts to interact with each other and with the knowledge engineer. By doing so the ontology evolves. Mailing lists, web forums, and WIKI pages usually support this interaction. Despite this combination of tools (none of them an ontology editor per se, nor a knowledge engineering tool), information is lost, documentation is poorly structured, and the process is not always easy to follow. This results in a decreased participation by the domain experts.

One of the key components in the development of ontologies in loosely centralized environments is the discussion related to each and every term and relationship/property. Pinto *et al*, as well as Tempich *et al* [3, 4] have proposed an argumentative structure to support and facilitate the discussion within the process of developing ontologies in loosely centralised environments. Both, Garcia *et al* and Hayes *et al* [5, 6], have studied the use of CMs during the elicitation process when developing ontologies in distributed environments. However, it is not clear how to support the proposed structure, nor what is the role of the argumentative process within the development of the ontology. The knowledge elicitation process, part of the whole ontology development, is a major bottleneck, particularly within those communities in which domain experts are geographically distributed. In order to assist the elicitation process and improve the interaction we propose the use of CMs as a means to scaffold the argumentative structure.

This paper is organized as follows. Firstly we provide some background information, and present our approach to the problem of supporting argumentative structures. We explain in section two what is an argumentative structure within the context of ontology development, we also present in this section the

[§] Corresponding author: agcastro@cgiar.org relationship between a CM and the argumentative structure proposed by Tempich *et al* [4]. In section three we present our CM plug-in for Protégé and elaborate further how our plug-in supports, assists and facilitates the argumentative process. We present a brief discussion and conclusions in section four.

2. Argumentative structure and CMs

Central to ontology development is the process by which domain experts and the knowledge engineer argue about terms/types and relationships. This collaborative interaction generates threads of arguments [3, 4, 7], and there is a need to support the evolution and maintenance of this argumentative process in a way that makes it easy to follow and, more importantly, links to evidence and provides room for conflicting points of view. Figure #1 {not italicised} presents the argumentative structure proposed by [4].

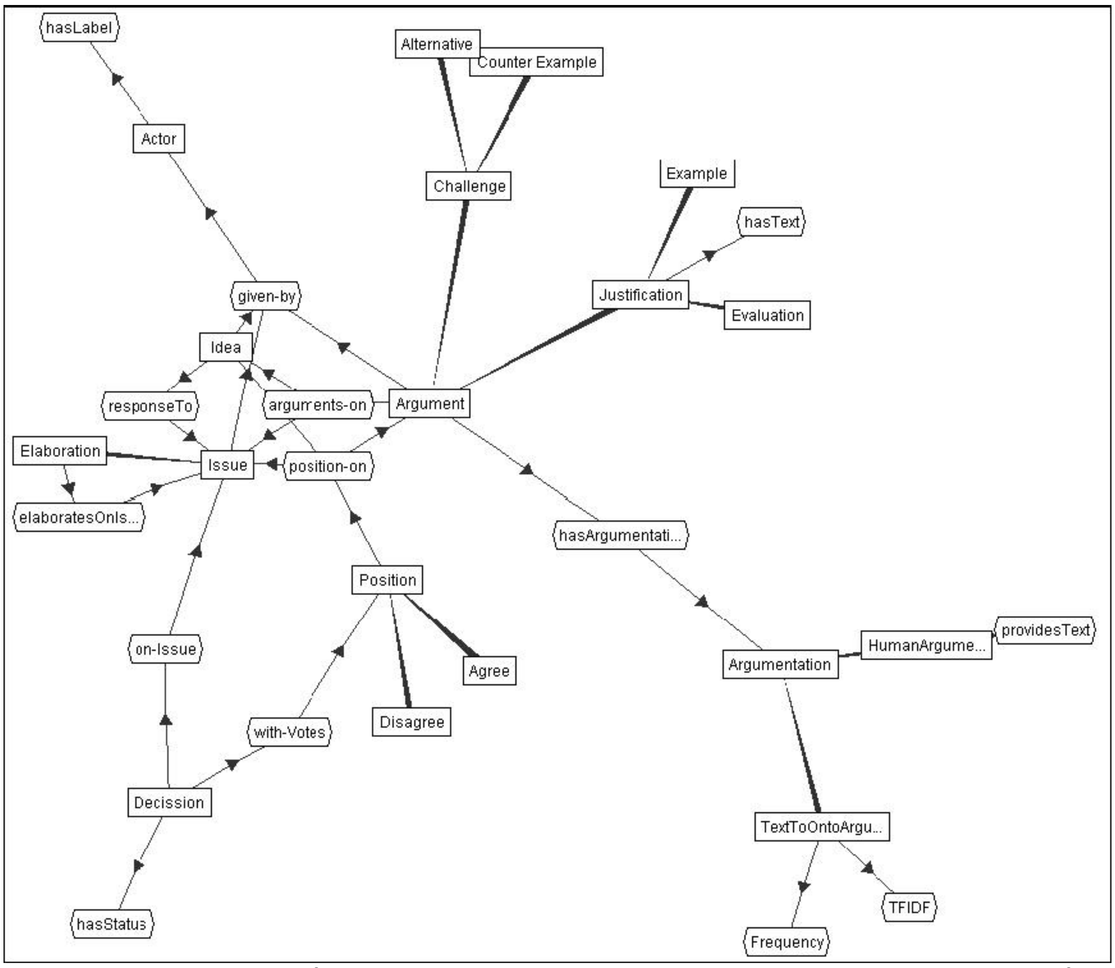


Figure #1 The major concepts of the argumentation ontology and their relations. Reproduced with permission from [4]

CMs are semantically valid artefacts without OWL constraints; concepts and relationships are the main scaffold of a CM. At any given point during the argumentative process one has a concept/class and a relationship/property. The evolution of the discussions increases the amount of information attached to the concept or relationship, the argumentative structure is enriched as domain experts provide arguments and base them upon evidence, which may be a paper, a commentary, or more generally a file of any kind (*idem* information source). The different views of the world can be represented with a CM, and the evidence may be attached to the particular concept/class or relationship/property at hand. This graphic representation facilitates the continuous exchange of information amongst domain experts –sharing knowledge. Following the threads of the discussions is not always easy for domain experts. The information exchanged is usually structured as an email-based chat. The knowledge engineer has to follow these text-based discussions in which there is mostly verbal knowledge, filter them, and at some point "formalise" that implicit knowledge. Moving from verbal knowledge into formalised-shared knowledge is difficult; some information is usually lost, the evidence supporting those different positions is not always provided by domain experts, and most importantly keeping domain experts engaged throughout the entire process is not always possible. Cognitive support is thus required so we may facilitate the useful flow/exchange of information and at the same time record the entire process.

3. Argumentation *via* CMs

Concepts and relationships resemble the two key components within an argumentative structure: arguments and positions. During the development process we argue in relation to a concept and/or a relationship. Positions are supported upon evidence, and the simple argumentative structure is by itself a particular view of the world that is being modelled. Figure *#2* illustrates the basics behind the relationship between CMs and an argumentative structure.

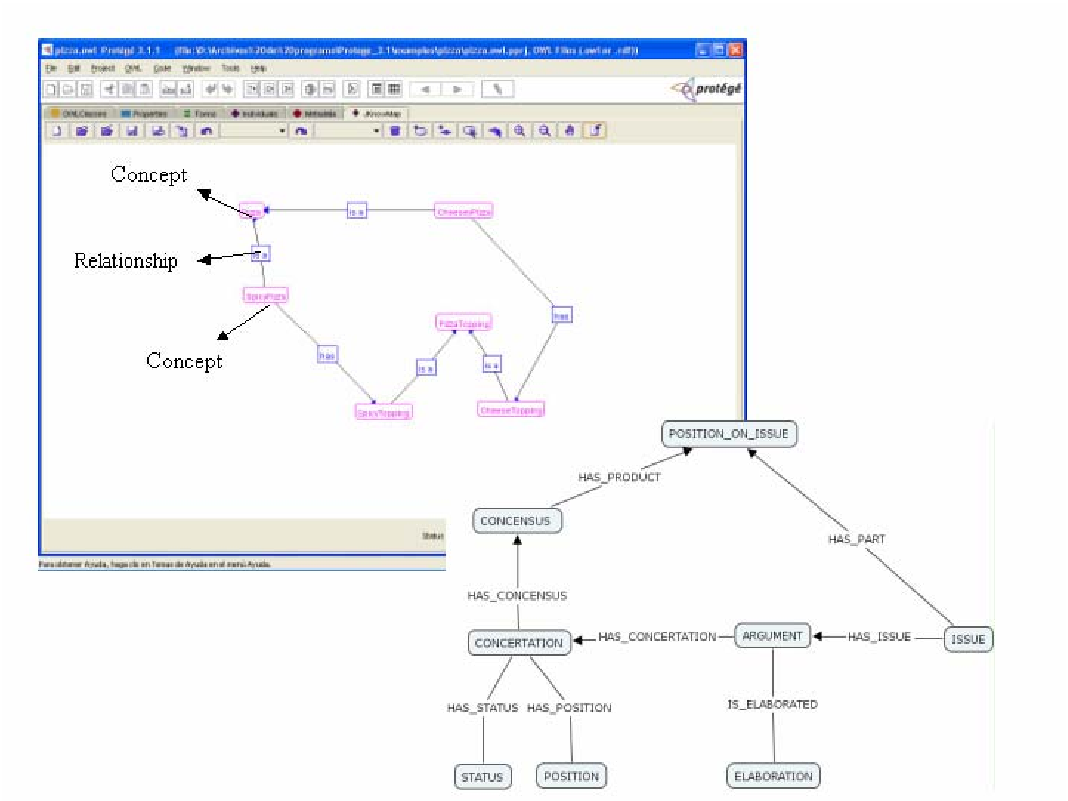


Figure #2. A simplification of the argumentative structure presented by [4]. The pizza example (http://www.co-ode.org) is used in order to illustrate our simplified argumentative structure.

For any given **issue** there is an **argument** that is elaborated by presenting the conflicting positions. The **elaboration** provides instances --concrete examples. For any issue there is a *concertation*[1] process that presents argument-elaborated conflicting positions. Once a consensus is reached there is a position on the issue initially at hand. The issue is well focused and specific, the same is true for the argument. It supports a position with simple and few words whereas the elaboration of the argument tends to be larger, and supported by different files (idem. pdf, ppt, doc, xls). Although there may be more than one argument for any given issue, there is only one elaboration for each argument. The dispute resolution process (also known as conciliatory process) produces a position on the particular issue; with in this process the knowledge engineer acts as a facilitator. Discussions over terminology, and over conceptual models, tend to address one issue at a time, this is highly dependent on the knowledge engineer.

A very important part of the whole process is the management of the history. Tracing back the argumentation process from the position_on_issue to the elaboration for a particular argument; being able to "see" the argumentative structure in order to "stand" on a particular place. The history should also allow us to go back and take an alternative route, thus we see the history not just as a simple undo" but as a more complex feature. An interesting starting point for complex history management is the one

[1] Concertation. From the French *concertation*. A conciliatory processes by which two parts reach an agreement.

provided by PhotoShop, a framework in which it is possible to see the whole process, take different routes, define blocks of work, and many other features that are always part of the manipulated image.

4. Discussion and conclusions

For any given **issue** there is an **argument** that is elaborated by presenting the conflicting positions. The **elaboration** provides instances -concrete examples. For any issue there is a process that presents argument-elaborated conflicting positions. Once a consensus is reached there is a position on the issue initially at hand. The issue is well-focused and specific, the same is true for the argument. It supports a position with simple and few words, whereas the elaboration of the argument tends to be larger and supported by different files (idem. pdf, ppt, doc, xls). Although there may be more than one argument for any given issue, there is only one elaboration for each argument. The dispute-resolution process (also known as the conciliatory process) produces a position on the particular issue; within this process the knowledge engineer acts as a facilitator. Discussions over terminology, and over conceptual models, tend to address one issue at a time and this is highly dependent on the knowledge engineer.

References

1. Ernst A, Neil. , Storey M-A, Allen P: **Cognitive support for ontology modeling**. *Int J Human-Computer Studies* 2005, **62**:553–577.
2. Walenstein A: **Cognitive support in software engineering tools: a distributed cognition framework**. Simon Fraser University; 2002.
3. Pinto HS, Staab S, Tempich C: **Diligent: towards a fine-grained methodology for Distributed, Loosely-controlled and evolving engineering of ontologies**. In: *European conference on Artificial Intelligence: 2004; Valencia, Spain*; 2004: 393-397.
4. Tempich C, Pinto HS, Sure Y, Staab S: **An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT)**. In: *Second European Semantic Web Conference: May 2005; Greece*; May 2005: 241--256.
5. Garcia C, Alexander., Rocca-Serra P, Stevens R, Taylor C, Nashar K, Ragan MA, Sansone S: **The use of concept maps during knowledge elicitation in ontology development processes - the nutrigenomics use case**. *BMC* 2005, **Submitted**.
6. Hayes P, Eskridge CT, Saavedra R, Reichherzer T, Mehrotra M, Bobrovnikoff D: **Collaborative Knowledge Capture in Ontologies**. In: *K-CAP 05: 2005; Banff, Canada*; 2005.
7. Garcia CA, Sansone AS, Rocca-Serra P, Taylor C, Ragan AM: **The use of conceptual maps for two ontology developments: nutrigenomics, and a management system for genealogies.** In: *8th Intl Protégé Conference Protégé: 2005; Madrid, Spain*; 2005: 59-62.

A PDF Storage Backend for Protégé

Henrik Eriksson

Dept. of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden

her@ida.liu.se

Introduction

Protégé is a successful environment for editing ontologies and knowledge bases. In Protégé, there are several tab-level extensions available to ontology developers, which add to the user-interface of Protégé [4]. However, there are few alternatives for storing ontologies and knowledge bases. Currently, Protégé stores ontologies and knowledge bases as files or tables in relational databases. The file storage consists of a group of separate files; that is, a project file (with the .pprj extension) and files for classes and instances (individuals). Some storage formats combine classes and instances in a single file. The advantage of using several files for storage is that the files can be used directly by other applications, such as applications processing instances only. The advantage of a single storage file is that it simplifies file management, such as renaming and copying. Furthermore, it might be easier for novice users to handle single-file storage than to manage multiple files.

Fortunately, the Protégé architecture separates its internal knowledge representation from the external serialisation of the knowledge-base content. The Protégé application-programming interface (API) supports storage-backend extensions, which allow developers to change the way Protégé saves and loads ontologies and knowledge bases. The standard Protégé distribution contains different storage-backend implementations for Clips, XML, RDFS, OWL, and databases. The standard file-based backend extensions, however, tend to use multiple files and voluminous syntax without compression. Thus, from a pure data-storage point of view, these formats are inefficient.

The goal of the *PDF backend approach* is to use the Portable Document Format (PDF) as the basis for a Protégé storage backend. This approach allows Protégé users to store ontologies and knowledge bases inside PDF files. It is possible to use pre-existing PDF documents as templates and add ontologies and knowledge bases to them. The resulting PDF files will still be documents that users can view on-screen and print. Tools for handling PDF, such as Adobe Acrobat Reader, will continue to work as before.

We have previously explored the use of PDF documents as the basis for *semantic documents* [2,3]. The goal of the semantic-document approach is to bring documents and knowledge bases closer together. The difference between semantic documents and the PDF storage backend is that semantic documents are intended for document annotations, which relate concepts in the ontology to selected text in the documents, whereas the PDF storage backend stores ontologies as attachments to PDF files.

We believe that the PDF backend approach will help Protégé users manage file-based storage of ontologies and knowledge bases. One of the advantages of using PDF as the storage format is that it supports compression of attached files. In addition to working as a normal document, the resulting file will be compact and will combine the project, ontology, and instance data in one package.

Background

PDF is an open format developed by Adobe [1]. In addition to Adobe products, such as Acrobat, there are several commercial and open-source tools that create, manage, and read PDF documents. Originally, Adobe designed PDF to support platform and device-independent printing and fast on-screen viewing. PDF documents consist of an indexed structure of internal document objects, such as text and binary streams.

The PDF specification supports the *attachment* of files to PDF documents. Just as it is possible to attach an arbitrary file to an e-mail, PDF allows for additional files to be inserted into the internal structure of PDF files. Although this feature is commonly available in Adobe products, such as Acrobat Professional (for reading and attaching) and Acrobat Reader (for reading), it is less known.

Saving to PDF

The PDF storage back-end is a prototype implementation of the Protégé back-end API. It redirects the saving of projects and knowledge bases to PDF files. If the PDF document does not exist, the back-end will create a new one automatically. In the current implementation, it creates a one-page document with the name of the knowledge base and some information about the document. In principle, it is possible to generate other types of initial documents, such as tables with knowledge-base metrics, instructions for downloading Protégé and reports generated from knowledge-base content.

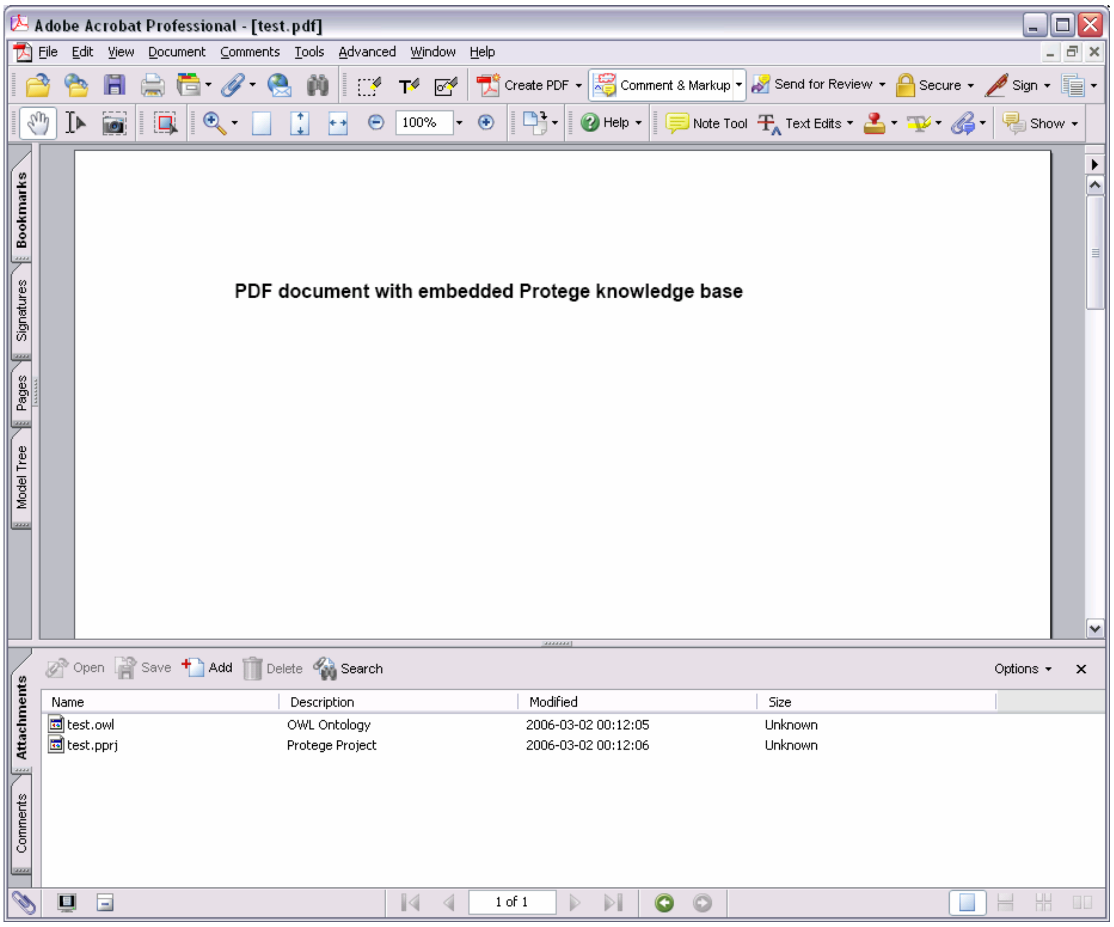
The PDF storage back-end uses PDFBox for accessing PDF documents and adding attachments to them. PDFBox is an open-source Java-based library for reading, manipulating, and writing PDF documents (see http://www.pdfbox.org/). PDFBox can parse PDF documents and represent the document content using object structures in Java. Furthermore, PDFBox supports document creation from Java and insertion of attachments into documents. We have found that PDFBox works well with Protégé and that it provides the functionality required for document generation, and reading and writing attachments to/from streams in a document. In PDFBox, adding compression to attachments is a straightforward task.

The attachments saved with the PDF document are available in tools such as Acrobat Reader (by opening the “Attachments” tab). Users can easily extract these attachments and use them as normal project and knowledge-base files for Protégé. When Protégé loads a PDF document, it first parses the file using PDFBox and then loads the attachments from input streams provided by PDFBox. This process is quite efficient. Because the objects in PDF documents are indexed, PDFBox can use fast random-file access to extract the attachments without processing the entire document.

Example

Let us consider an example of how the PDF storage backend saves Protégé projects and ontologies in documents. Figure 1 shows a sample PDF document generated by the storage backend. This document contains two file attachments, test.owl and test.pprj. These files correspond to the files saved normally by the OWL backend. Furthermore, it is possible to extract these attachments from the PDF document (e.g. using Acrobat) and use them as normal Protégé files. Likewise, ontology developers can manually construct new documents for the PDF backend by attaching normal Protégé files to a PDF document.

The document view in Acrobat shows the generated one-page document with the template text, “PDF document with embedded Protégé knowledge bases.” Note that it is possible to open this document with any PDF viewer (although some third-party viewers and old Acrobat versions may show the attachment). In principle, this generated front page can contain any information about the ontology, such as instructions for using the ontology, ontology metrics, and text generated from the ontology content. An alternative is to start with an existing PDF document and let Protégé add the project and ontology attachments to it. Another possibility is to add other project-related files, such as plug-in specific resource files, as document attachments.



PDF document with embedded Protege knowledge base

Name	Description	Modified	Size
test.owl	OWL Ontology	2006-03-02 00:12:05	Unknown
test.pprj	Protege Project	2006-03-02 00:12:06	Unknown

Figure 1. Sample PDF document opened in Acrobat. In this case, the PDF back-end generated a single document page and stored the project and ontology as attachments. The attachments tab in Acrobat shows a list of the document attachments.

Discussion

One of the major challenges for implementing the prototype of the PDF back-end was to modify the libraries and APIs that the back-end uses. The back-end required modifications (bug fixes and/or extensions) to PDFBox, the Protégé OWL extension, and the Protégé core API. In particular, the Protégé core API assumed the existence of a project (.pprj) file for storing project-specific information, such as custom adjustments to forms. It was necessary to add API functionality for storing the project information in alternative formats and locations. Currently, there are still some unresolved issues in terms of using the same storage extension for several knowledge-based formats, such as frames, RDFS, and OWL.

The implementation of PDF back-end shows that it is possible to develop file-based storage solutions that overcome some of the problems with multiple files, and that provide compact storage for XML-based formats. It is possible to use a similar method for developing storage back-ends for other archive and compression formats, such as jar, tar and zip. These formats allow other applications to read files produced by Protégé without the overhead of PDFBox. Furthermore, new storage formats help make Protégé useful for new groups of users, and expand potential uses of ontologies.

Summary and Conclusions

We believe that PDF is a viable storage format for Protégé ontologies and knowledge bases. The advantage of storing project, class and instance files as PDF attachments is that it is a straightforward method that allows document viewing and printing using standard PDF tools. Note, however, that this approach is different from semantic documents, which relate document content to concepts in the ontology. The prototype implementation illustrates that PDF storage works. However, there are still some limitations to the Protégé API that make it difficult to develop a storage plug-in that works for all knowledge-base formats. Nevertheless, we believe that PDF storage will be a highly useful feature of Protégé.

Acknowledgements

This work was supported by Vinnova (grant no. 2003-01415), by The Swedish Research Council (grant no. 621-2003-2991), and by Statistics Sweden.

References

[1] Adobe (2004a). PDF Reference Version 1.6. Adobe Press, Berkeley, CA, 5th edition.

[2] Henrik Eriksson. The semantic document approach to combining documents and ontologies. *International Journal of Human–Computer Studies*, in press.

[3] Henrik Eriksson. Support for semantic documents in Protégé. In *Proceedings of the Eight International Protégé Conference,* Madrid, Spain, July 18–21, 2005.

[4] John H. Gennari, et al. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human–Computer Studies,* 58(1):89–123, 2003.

Content-based Ontology Ranking

Matthew Jones
Dept. of Electronics & Computer Science
University of Southampton
msj103@ecs.soton.ac.uk

Harith Alani
Dept. of Electronics & Computer Science
University of Southampton
h.alani@ecs.soton.ac.uk

ABSTRACT

Techniques to rank ontologies are crucial to aid and encourage the re-use of publicly available ontologies. This paper presents a system that obtains a list of ontologies from a search engine that contain the terms provided by a knowledge engineer and ranks them. The ranking of these ontologies will be done according to how many of the concept labels in those ontologies match a set of terms extracted from a corpus of documents related to the domain of knowledge identified by the knowledge engineer's original search terms.

1 INTRODUCTION

Ontologies can be very time-consuming and expensive to construct. As the use of ontologies for the representation of domain knowledge increases, so will the need for an effective set of tools to aid the discovery and re-use of existing knowledge representations. This is because a major advantage of ontologies is their ability to be re-used as well as easily adapted to work with new knowledge-based applications.

Recently, a small number of search-engines to aid in the discovery of ontologies have been developed, but the techniques for ranking the results of these search engines are still in their early stages. The ontology search engines; Swoogle [3] ranks its results using an adaptation of Google's PageRank [4] scoring system. A major downside with this method is that many ontologies are poorly inter-referenced, which does not necessarily reflect in the quality of the ontologies. AKTiveRank[1] is an ontology ranking method that applies a number of graph analysis measures to estimate how well does each potential ontology represent the classes of interest. This ranking method is purely dependent on the terms given by users when searching for ontologies.

When developing ontology ranking techniques, it is important to also consider how users perform ontology searches in the first place. The ontology search engine mentioned above allows searching for specific terms, which has to exist in the ontology (e.g. part of a class or a property name) for that ontology to be retrieved. However, when analysing queries sent by ontology seekers to the Protégé mailing lists, we found that they all describe a domain (e.g. History, Economy, Algebra), rather than specific entity labels. In this paper we introduce a ranking method that is based on the content similarity of an ontology to a corpus that is selected for the given search terms.

2 CONTENT-BASED RANKING

In order to rank ontologies, our system will attempt to find a corpus that relates to the domain that the user requires an ontology to represent. This method is inspired by [2], but differs from it in that the corpus is selected based on the user query, rather than the ontology itself. The corpus will then be analysed to identify domain-related terms to use for evaluating the existing ontologies in terms of how well they cover the domain of interest. Using a representative corpus allows terms to be extracted using term frequency measures (tf-idf [5]). The terms which get the highest Tf-idf score from this corpus can then be considered as potential concept labels. This system uses the top 50 words of such an analysis. An ontology which has more class labels that match these words is deemed more suitable by the system and is therefore ranked higher than others. The following sections demonstrate our ranking method.

2.1 Obtaining a Corpus

To obtain a set of documents relevant to a user query, this system uses a Google search, and takes the first 100 pages as its corpus. Initially we thought that using the same search terms provided by the user would be enough to get a set of documents rich in domain-related information. However, many of the documents returned in such cases where too general (e.g. charity sites and general organisations' web pages when searching for 'Cancer'). As a remedy to this problem, WordNet was used to expand user search terms to make the search for pages more specific to the domain of knowledge required.

2.2 Adding WordNet

For more specific queries in Google, more terms need to be added to the Google query string, other than those given by users when searching for ontologies. These extra query terms can be obtained from WordNet. The use of WordNet has two benefits; while specifying a more specific query to Google; it also allows the system to disambiguate any terms provided by the user which may have more than one meaning (e.g. Cancer as a disease rather than a zodiac sign).

Table 1 shows the top 50 terms used in a corpus obtained from Google for the term ‘Cancer’, compared with those obtained using a query expanded with WordNet, by specifying the disease sense of the word cancer. The words added to the Google query are simply synonyms, hyponyms and meronyms of the original query terms. The addition of these extra words is simply to obtain a more specific query for Google. The improvement in the selection of potential concept labels in column (b) is quite apparent when compared with column (a).

(a) Using Basic Google Search						(b) Using WordNet Expanded Google Search					
1.	cancer	18.	loss	35.	cliphead	1.	cancer	18.	thymoma	35.	studies
2.	cell	19.	dine	36.	apologize	2.	cell	19.	malignant	36.	ovarian
3.	breast	20.	mine	37.	changed	3.	tumor	20.	gene	37.	information
4.	research	21.	dinner	38.	unavailable	4.	patient	21.	clinical	38.	research
5.	treatment	22.	cup	39.	typed	5.	document	22.	neoplasm	39.	drug
6.	tumor	23.	strikes	40.	bar	6.	carcinoma	23.	pancreatic	40.	related
7.	information	24.	heard	41.	spelled	7.	lymphoma	24.	Tissue	41.	associated
8.	color	25.	signposts	42.	correctly	8.	disease	25.	therapy	42.	neoplastic
9.	patient	26.	teddy	43.	typing	9.	access	26.	lesion	43.	oral
10.	health	27.	bobby	44.	narrow	10.	treatment	27.	blood	44.	bone
11.	support	28.	betrayal	45.	entered	11.	skin	28.	study	45.	chemotherapy
12.	news	29.	portfolio	46.	refine	12.	liver	29.	thyroid	46.	body
13.	care	30.	lincoln	47.	referenced	13.	leukemia	30.	smoking	47.	oncology
14.	wealth	31.	inn	48.	recreated	14.	risk	31.	polyp	48.	growth
15.	tomorrow	32.	endtop	49.	delete	15.	breast	32.	human	49.	medical
16.	entering	33.	menuitem	50.	bugfixes	16.	genetic	33.	health	50.	lung
17.	writing	34.	globalnav			17.	tobacco	34.	exposure		

Table 1: Comparison of tf-idf results from a corpus of 100 documents obtained from Google for the term ‘cancer’ (column a) and terms expanded using WordNet (column b).

2.3 Calculating Ontology Score

Each ontology is then ranked according to how many of these new terms match class labels within them; the class match score (CMS).

Definition: Let *O* be the set of ontologies to be ranked and *P* be the set of potential class labels obtained from the corpus. And *n* is the number of terms collected from the corpus.

$$CMS[o \in O] = \sum_{i=1}^{n} I(P_i, o) \times 5 \log (n + 2 - i)$$

$$I(P_i, o) = \begin{cases} 1 & : \text{ if } o \text{ contains a class with label matching } P_i \\ 0.4 & : \text{ if } o \text{ contains a class with label which contains } P_i \\ 0 & : \text{ if } P_i \text{ does not appear in any of } o\text{'s class labels} \end{cases}$$

The values 1 & 0.4 can be adjusted according to how much emphasis is put on a complete class label match compared with a partial one. The ontologies are also analysed to see if any literal text, e.g. comments, matches the potential class labels. The literal text match score (LMS) is the same as CMS, except that $I(P_i, o)$ is now 1 if the ontology (*o*) contains text that matched a given terms (P_i), and 0 otherwise.

The total score for each ontology is a combination of these scores, which are weighted, to emphasise the importance of one over the other; $Total = \alpha\ CMS + \beta\ LMS$. Where α & β are weights, which the experiment in section 4.1 is concerned with manipulating.

3 EXPERIMENTS

In this section two experiments are presented that show how manipulations of how the system ranks the ontologies affect the ranking order. The example used here is a search for ontologies for ‘Cancer’. The results from the experiments will then be compared and evaluated in section 5. The set of ontologies to be ranked in these experiments appear in table 2, and were chosen carefully from Google results, after throwing away duplications and broken ontologies.

3.1 Experiment 1

This experiment looks at how changing the significance of the class match and literal text match score affects ontology's ranking. This is done by changing the α & β values described in section 2.3. For this experiment two attempts to rank the ontologies are made. Experiment 1(a) uses 0.8 and 0.2 for α & β respectively (a class match being considered more important). For 1(b), both α & β take the value 0.5 (both being assumed to have the same importance for ranking purposes).

3.2 Experiment 2

This experiment looks at the effects of restricting the corpus to being comprised solely of Wikipedia pages. This experiment is repeated twice using two sets of weights for α & β as done in experiment 1.

ID	Ontology URL
1	http://semweb.mcdonaldbradley.com/OWL/Cyc/FreeToGov/60704/FreeToGovCyc.owl
2	http://www.inf.fu-berlin.de/inst/agnbi/research/swpatho/owldata/swpatho1/swpatho1.owl
3	http://www.mindswap.org/2003/CancerOntology/nciOncology.owl
4	http://sweet.jpl.nasa.gov/ontology/data_center.owl
5	http://compbio.uchsc.edu/Hunter_lab/McGoldrick/DataFed_OWL.owl
6	http://www.cs.umbc.edu/~aks1/ontosem.owl
7	http://homepages.cs.ncl.ac.uk/phillip.lord/download/knowledge/ontologyontology.owl
8	http://www.daml.org/2004/05/unspsc/unspsc.owl
9	http://envgen.nox.ac.uk/miame/MGEDOntology_env_final.owl
10	http://www.fruitfly.org/%7Ecjm/obo-download/obo-all/mesh/mesh.owl

Table 2: URLs of ontologies to be used in all experiments.

4 EVALUATION

In order to evaluate our results, it was necessary to compare the system's ranking with those produced by humans. Two medical students were asked to rank each of our selected ontologies according to how well the ontologies cover the concept of cancer. These comparisons are shown in Table 3.

The results obtained from the system are promising, but obviously non-conclusive due to the very small size of the experiment. Our system agreed with our experts by ranking the NCI ontology first. The comparison shows that while some of the ontologies are ranked similarly by the system, there are still a few of the ontologies that seem to be out of place. Notably, a number of the larger, more general ontologies (e.g. ID 1) are given lower scores by our experts. This is possibly due to the fact that they considered the ontologies 'too general'. This indicates that perhaps there is a need for checking the specificity of an ontology when evaluating its relevance to a search query.

The results of experiment 2(b) turned out to be the most similar to the ranks produced by our experts, scoring 0.693 in Pearson's Correlation Coefficient (PCC), where a value of 1 is a perfect match, 0 is total randomness, and -1 is an inverse match (table 4). This similarity drops to 0.492 when the user search terms are *not* expanded with WordNet. Note that the average PCC value between the ranks of our two experts is 0.92, indicating high agreement.

Ontology ID	Human Rank	Expt. 1(a)	Expt. 1(b)	Expt. 2(a)	Expt. 2(b)
3	1	1	1	1	1
10	2	8	5	8	5
6	3	2	2	2	2
2	4	6	8	5	6
5	5	5	6	6	7
1	6.5	3	3	3	3
9	6.5	7	7	7	8
8	8	4	4	4	4
7	9	10	10	10	10
4	10	9	9	9	9

Table 3: Comparison of rankings from system experiments along with average human rank

Rankings	Pearson Correlation Coefficient
Humans:Rank1a	0.565
Humans:Rank1b	0.650
Humans:Rank2a	0.577
Humans:Rank2b	**0.693**
Humans:No WordNet	0.492

Table 4: Comparison of ranks with those given by users

5 CONCLUSION

Ontologies can be evaluated and ranked is many different ways, based on variant characteristics. Here we experimented with evaluating ontologies based on their coverage of the domain of interest. As we are simply looking for a set of particular concepts there is no attempt to look at how well they are connected or how well each is defined in an ontology. Combining our method with some of the ranking metrics from [1] would possibly be beneficial. The set of potential concepts which are extracted from the corpus were very acceptable in our experiment. However, retrieving a suitable corpus can not be guaranteed for all ontology search queries, especially if the search terms are too specific.

The results from a Wikipedia-only corpus where better in our experiments, but it did not differ dramatically from the unrestricted one. This renders limiting the corpus to Wikipedia questionable, especially that the ontology topics that users might be after may not be covered well enough in Wikipedia.

REFERENCES

[1] Alani, H. and Brewster, C., Ontology Ranking based on the Analysis of Concept Structures. In Proceedings of the 3rd International Conference on Knowledge Capture (K-Cap), Banff, Alberta, Canada. 2005.

[2] Brewster, C., Alani, H., Dasmahapatra, S. and Wilks, Y.. Data driven ontology evaluation. In International Conference on Language Resources and Evaluation, Lisbon, Portugal, 2004.

[3] Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivari, P., Doshi, V. C. and Sachs, J. Swoogle: A semantic web search and metadata engine. In Proceedings of the 13th ACM Conference on Information and Knowledge Management, Nov. 2004.

[4] Page, L., Brin, S., Motwani, R. and Winograd, R. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Uni., 1999.

[5] Salton, G. and Buckley, C. 1988 Term-weighting approaches in automatic text retrieval. Information Processing & Management 24(5): 513–523

A CONSUMER ONTOLOGY ANALYSIS TOOL

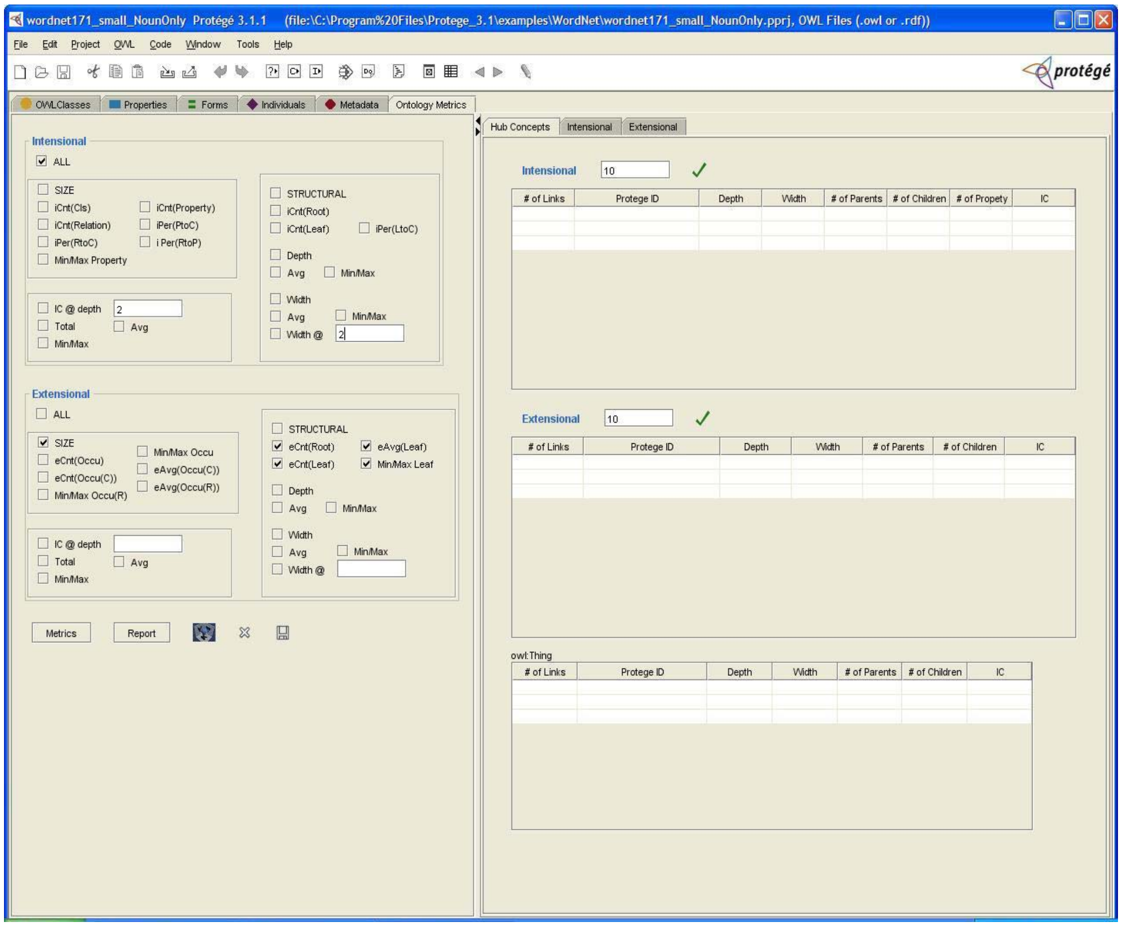
Valerie V. Cross and Anindita Pal
Computer Science and Systems Analysis
Miami University
Oxford, OH 45056
{crossv, pala}@muohio.edu

Ontologies capture domain knowledge in a generic way and provide a commonly agreed upon understanding. As such, they are becoming the backbone of the Semantic Web which has as a primary role the sharing of knowledge among Internet users. Numerous ontologies are being developed and published across varying domains such as biomedical, computational linguistics, and business. Development and deployment of extensive ontology-based software solutions represent considerable challenges in terms of the amount of time and effort required to construct the ontology. These challenges can be addressed by the reuse of ontologies. In some domain areas such as e-commerce, recent research indicates that a major concern in the development and maintenance of ontologies for products and services is "the terminological dynamics in markets" [Hepp et al 2005]. Such domains would greatly benefit from research to improve the ontology maintenance process so that the acquisition of dynamic domain vocabulary could be more timely and comprehensive. Another important issue for many Semantic Web applications is the need for interoperability between interacting software agents, each having their own but different ontologies. In this case, the objective is not reuse of an existing ontology but instead one of communication between the two software agents. Establishing a level of semantic interoperability between the two could require both agents to evaluate the quality and usefulness of the other agent's ontology. An ontology analysis tool could assist in the process of ontology evaluation for re-use, facilitate monitoring the development and maintenance of ontologies, and help each agent to determine characteristics of the other agent's ontology for assessing the degree of semantic interoperability.

This presentation describes the research, implementation, and evaluation undertaken in order to develop an ontology analysis tool that can help address issues related to ontology re-use, maintenance and interoperability. By examining the current ontology evaluation research literature and reviewing metrics research in other areas such as conceptual modeling, software development, information systems development, and information retrieval, suites of metrics have been identified and incorporated into an ontology analysis tool. First, an overview of related research discusses the various approaches to ontology evaluation. Ontology evaluation is a broad research area. Different frameworks have been proposed for evaluating how "good an ontology is." Some like One-T [Bouillon et al 2002] consider evaluating the content of the ontology for completeness, consistency and correctness in terms of lack of inconsistencies, redundancy and errors. Others suggest developing methodologies to evaluate an ontology during the development process throughout its entire lifetime [Guarino and Welty 2002]. Furthermore, many argue that the only true way of evaluating ontologies is to use them in applications and assess the application performance. Recent research in ontology evaluation such as AKTiveRank [Alani and Brewster 2005], OntoQA [Tartir 2005], and ONTOMETRIC [Lozano-Tello and Gómez-Pérez 2004] is summarized and compared to the consumer ontology analysis tool.

Another earlier and more detailed study compares different ontologies with respect to 28 characteristics grouped into eight categories [Noy and Hafner 1997]. These approaches provide ontology users valuable information, but consumers, who are the developers of applications using ontology, need additional ways to evaluate ontologies to determine how well the ontology fits their application's needs. Noy [2004] suggests that to make reusing ontologies easier as the number of ontology library increases, more research needs to address the evaluation of an ontology from the consumer point of view. The ontology analysis tool's development was based on the consumer point of view [Noy 2004]; i.e., the consumer is shopping for an ontology to re-use or adapt for a particular project or need. The ontology analysis tool is a Protégé tab plug-in, the most popular among the different categories of Protégé plug-ins. The metrics produced by the tool should provide the consumer with insights about the level of sophistication and the amount of detail provided by the ontology. It allows the user to examine the intensional ontology characteristics separately from those of the extensional ontology. An intensional ontology only includes the ontology schema or definitions. Thus an intensional ontology consists of a set of concepts (or classes), their definitions and their inter-relationships (specified in properties or slots). An extensional ontology includes the instances of an ontology, where instances are occurrences of a concept. Thus an extensional ontology not only references the concept schemas, but also includes the instances of the concepts.

The metrics are divided into size and structural categories to facilitate user interaction. The implementation is generalized in order to handle the structural difference in ontologies. Thus the design is parameterized so that users can easily switch between an intensional and extensional ontology. For an intensional ontology, the taxonomical structure is built using the sub-class relationship. But the extensional taxonomical structure is more complex and differs from ontology to ontology. The users must select the relationship that will be used to build the taxonomical structure of extensional ontology. Much research has focused on extensional ontologies, in some part, because the consideration for reuse of ontologies has often been on terminological ontologies in the biomedical fields. The complete set of metrics produced by the tool and its user interface are presented. The following figure shows the interface of the plug-in. The interface consists of two split panes; a left "Selection" panel for the various parameters to be selected and a "Result" panel to display the result of the metrics calculated.


wordnet171_small_NounOnly Protégé 3.1.1 (file:\C:\Program%20Files\Protege_3.1\examples\WordNet\wordnet171_small_NounOnly.pprj, OWL Files (.owl or .rdf))

File Edit Project OWL Code Window Tools Help
protégé
OWLClasses Properties Forms Individuals Metadata Ontology Metrics
Metrics Report

Intensional
- [✓] ALL
- SIZE: iCnt(Cls), iCnt(Property), iCnt(Relation), iPer(PtoC), iPer(RtoC), i Per(RtoP), Min/Max Property
- IC @ depth 2; Total; Avg; Min/Max
- STRUCTURAL: iCnt(Root), iCnt(Leaf), iPer(LtoC); Depth: Avg, Min/Max; Width: Avg, Min/Max, Width @ 2

Extensional
- ALL
- [✓] SIZE: eCnt(Occu), eCnt(Occu(C)), Min/Max Occu(R), Min/Max Occu, eAvg(Occu(C)), eAvg(Occu(R))
- IC @ depth; Total; Avg; Min/Max
- STRUCTURAL: [✓] eCnt(Root), [✓] eAvg(Leaf), [✓] eCnt(Leaf), [✓] Min/Max Leaf; Depth: Avg, Min/Max; Width: Avg, Min/Max, Width @

Hub Concepts | Intensional | Extensional

Intensional 10 ✓

# of Links	Protege ID	Depth	Width	# of Parents	# of Children	# of Propety	IC

Extensional 10 ✓

# of Links	Protege ID	Depth	Width	# of Parents	# of Children	IC

owl:Thing

# of Links	Protege ID	Depth	Width	# of Parents	# of Children	IC

Users can select all metrics in the group intensional or extensional by checking the "All" buttons in these groups. Users can also select either size or structural metrics within a group by checking the corresponding buttons. The next set of parameters to input after selection of metrics is the root concept to use in metric calculations and the relationship used to build the extensional taxonomical structure. User can select these parameters by clicking the "Metrics" or "Report" buttons. For the "Metric" button if no class is selected, then only the ontology level metrics are displayed since there is no space in the UI to display the metric results for all classes in the ontology. In case of the "Report" button, if users do not select any class, then the metrics are measured on all classes in the ontology and reported.

A hub summary lists the hub concepts, those with the largest number of links in and out of them. The “Hub Concepts” tab displays the intensional and extensional hub summaries of the ontology. The tab consists of three tables. The intensional table lists the hub concepts, that is, classes with the maximum number of subclasses and super-classes, for entire intensional ontology. The extensional table lists the extensional hub concepts, that is, instances with the maximum number of links in and out of it for the complete extensional ontology. These instances could be from different classes. The Selected Class table lists the extensional hub concepts from the user-specified class. The summary also consists of the following measure for each hub; depth, width, number of properties and information content measures [Seco et al 2004]. By default the plug-in displays the top 10 hub concepts. Users can specify the number or percent of hubs to display. The hub summary may also be visualized by displaying with the paths connecting the hubs. An individual hub may be selected from the hub summary report and a visualization of three levels up and down from the selected hub concept is displayed.

The consumer ontology analysis tool is applied to several different ontologies, primarily terminological ontologies: WordNet, UMLS, UNSPSC, and eCl@ss. WordNet [Miller 1995] is an online English lexical reference system whose extensional ontology organizes nouns, verbs, adjectives and adverbs into synonym sets into semantic network with sets of synonymous terms, or synsets associated with lexical concepts. The synsets differentiate word senses from each other. UMLS (Unified Medical Language System) (http://www.nlm.nih.gov/research/umls/documentation.html) is an ontology that combines many distinct terminologies and was created by the National Library of Medicine (NLM) in Bethesda MD. It helps retrieve information from different biomedicine and health sources. UMLS consists of a large vocabulary database, Metathesaurus. It contains 1.8 million biomedical and health-related concepts, various string concept names, and their relationships. In Metathesaurus, there are more than 100 source vocabularies including different terminologies, classifications, and some thesauri. UNSPSC (United Nations Standard Products and Services Code (UNSPSC) is a hierarchical convention used to classify all products and services. eCl@ss is a Product and Service classification standards developed by leading German companies and offers a standard for information exchange between suppliers and their customers.

The results from applying the ontology analysis tool to the four ontologies are presented and the usefulness of tool and the insights gained for each ontology are discussed. The significance of and recommendations for future directions of this research are also presented.

REFERENCES

H. Alani and C. Brewster, “Ontology Ranking based on the Analysis of Concept Structures”, *International Conference On Knowledge Capture*, Alberta, Canada, 2005.

Y. Bouillon, O. Corcho, M. Fernández-López, and A. Gómez-Pérez, “A survey on ontology tools”, *OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce*, May 2002.

N. Guarino and C. Welty, “Evaluating ontological decisions with OntoClean”, *Communications of the ACM*, Volume 45, Number 2, February 2002

M. Hepp, J. Leukel, and V. Schmitz. "Content Metrics for Products and Services Categorization Standards," *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pp. 740-745, 2005.

A. Lozano-Tello, and A. Gómez-Pérez, “ONTOMETRIC: A Method to Choose the Appropriate Ontology,” *Journal of Database Management*, 15(2), 1-18, April-June 2004.

G. Miller, “WordNet: a lexical database for English,” *Communications of the ACM 38 (11)*, 39 – 41, 1995

N. Noy, “Evaluation by Ontology Consumers,” in “Why Evaluate ontology technologies? Because it works!” *Intelligent Systems, IEEE*, Volume 19, Issue 4, Jul-Aug 2004

N. F. Noy, and C. Hafner, “The State of the Art in Ontology Design”, *AI Magazine*, Fall 1997, p. 53-74, 1997.

N. Seco, T. Veale, and J. Hayes, “An Intrinsic Information Content Metric for Semantic Similarity in WordNet”, *ECAI 2004*, 1089-1090, 2004.

S. Tartir, I.B. Arpinar, M. Moore, A.P. Sheth, A.P. and Aleman-Meza, B. OntoQA: Metric-Based Ontology Quality Analysis *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, Houston, TX, USA, 2005.

Marginalia Ontology:
A Comprehensive Ontology for Describing the Metadata of Ontologies

Ravi Tiruvury, Stanford University

Abstract:
More than often, ontology users face situations wherein they would like to know more about an ontology in general - such as the latest curatorial activities, experiences of previous users of the ontology, and so on. These seemingly simple facts about an ontology help the users choose an ontology best-suited to meet their needs. However, associating this "metadata" with ontologies is not an easy task for two reasons:

1) There is no structured repository of this metadata
2) Even if metadata were available, there is no formal representation/schema by which the metadata could be expressed.

The current work takes the approach of first identifying this schema - what we call Marginalia Ontology. This could then drive a more structured effort to record metadata about ontologies as per the specifications of Marginalia Ontology - which could potentially present ontology users with very useful and relevant metadata.

Lessons Learned From Ontology Design

Jean-André BENVENUTI[1] Laure BERTI–ÉQUILLE[2]
Éric JACOPIN[1]

[1]MACCLIA, Écoles de Saint-Cyr Coëtquidan, France
{jean.benvenuti,eric.jacopin}@st-cyr.terre.defense.gouv.fr

[2]IRISA, Université de Rennes 1, France
berti@irisa.fr

Introduction Since 2001, we have been involved in the development of a system named SABRE [1, 2, 3, 4] for supporting the training of military students of the French Army; we expressed the knowledge of the environment and concepts of military work as ontologies using Protégé. We consider that the ontology contains explicit descriptions of the concepts (represented by words: e.g. "to serve one's homeland") used in our military domain. The knowledge is also at stake in the military training and then must be understandable by the students. Finally, not only our ontology must produce usable data structures, but it also must be accepted by the French Army instructors. The SABRE system must consequently possess the necessary data structures so that this knowledge can be expressed, edited, completed, presented, used in a training session to check the current knowledge of the trained students, and stored for future references. However, building ontologies and designing the data structures for such a knowledge-based system are not easy tasks; we recently found that some set-theory-based characteristics extracted from our "in progress" ontology could considerably help the ontology builder. This paper describes our experience on building the ontology of the French Military training using Protégé and how this step-by-step process can be supported and guided on-the-fly by systematically checking the theoretical properties of the instance sets of the ontology.

During the design of an ontology for military training [1, 2], we partially solved the ambiguity (getting different interpretations for identical values of different slots), completeness (adding interpretations for all values) and minimality (avoiding intractability) problems with extra design and rules [4]. We here present several (polynomial-time) functions which can automatically detect where *i)* to add relevant interpretation rules or *ii)* to complete our ontology (adding classes and slots) for assisting the ontology design.

Illustrative Example This paragraph illustrates the problem we encoutered in the case of ontology design in the domain of military training with the help of a geometrical problem (chosen for the sake of simplicity). Then we present a set of routines which aims at pointing to the classes, slots and instances which are either incomplete or need rules for further interpretation.

On the left of Figure 1 is an UML class diagram (a) where Polygon and Point are two classes; at least 3 ordered Points can be the vertices of 1 Polygon (adapted from [8, page 68]). The object diagram (b) is correct with respect to the UML class diagram (a); Triangle1 and Square are instances of Polygon and Triangle2 is an instance of Polygon whose vertices are made from vertices of Triangle1 and Square. Triangle1 has vertices {P1,P2,P3}, Square has vertices {P4,P5,P6,P7} and Triangle2 has vertices {P1,P5,P7}. One interesting question is : May {P1,P5,P7}, instances of Point and vertices of Triangle2, be also vertices of both Triangle1 and Square? The correctness of Figure 1(b) with respect to the UML class diagram (a) entails a positive answer to this question[1].

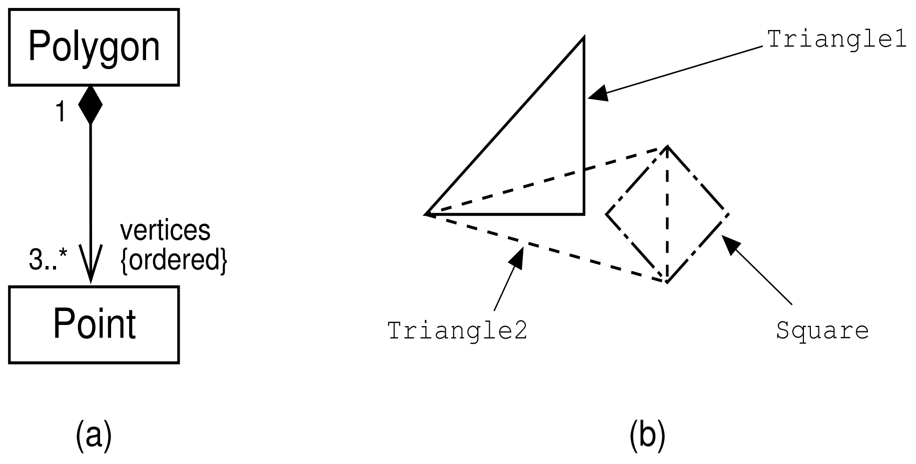


Figure 1: A Geometrical Example Illustrating the Need for Disambiguization

Now consider the following set of coordinates of instances of Point that can be associated to Figure 1(b): $\{(0,0),(0,2),(2,2),(0,3),(0,5)\}$; which of these coordinates are that of the three vertices of Triangle2? Or, in other words, which of these coordinates activate Triangle2? As Triangle2 shares all its vertices with both Triangle1 and Square, it should be obvious that the answer to this question is impossible when some information is missing (and may not be available) such as the orientation of the axes and the position of their origin: interpreted only from the coordinates, Figure 1(b) is ambiguous. Could we modify Figure 1(a) to avoid this ambiguity, and how?

As illustrated by this geometrical discussion, disambiguization and completeness are the problems we encountered when editing the knowledge of the French Military Training and designing our ontology with Protégé [1]. Our objective is to activate this knowledge so as to educate military staff to the recently designed French soldier's code [5, 6, 7]; the proposed pedagogical method [5] is to confront the learner to a concrete case, describing a real-life military situation where the learner is demanded to act as an actor of the concrete case and play his rôle throughout. In the context of the French Military Training ontology designed from several referential documents [6, 7], the word "discipline" may have different interpretations depending on the articles studied in the current pedagogical training session: for example, in the context of the 10th article of the French Soldier's Code, "discipline" should be interpreted as close as possible as "reserve"(*i.e.*, "secrecy"). But in the 2nd article, "discipline" should be interpreted as close as possible as "courage". The previous question now stands as follows: which of the learner's behaviours activate the articles and themes of the current pedagogical session? To answer this question, we first designed a set of rules to interpret the behaviours along the pedagogical session [2, 3, 4]. However, as we came to discover that some parts of our military ontology did not need rules, we came to design a set of functions to automatically detect our need for disambiguization rules (*i.e.*, for enriching our ontology). The next paragraph formally describes these functions.

Towards automatic detection of ambiguities in ontologies We begin by an analogy with linear algebra, where eigenvalues are the coordinates of the eigenvectors defining the main axes of a vector space (all vectors of a vector space can be written as a combination of these eigenvectors). We thus use the word eigen together with: *(i)* values of a slot of a class, *(ii)* a slot of a class, *(iii)* all instances of a class and finally *(iv)* a class, to denote that the values of the slot of an instance of a class can uniquely determine this instance.

[1]Note that object-oriented design answers "no" [8, page 68]: "The general rule is that, although a class may be a component of many other classes, any instance must be a component of only one owner. [...] The "no sharing" rule is the key to composition. Another assumption is that if you delete [a] polygon, it should automatically ensure that any owned Points also are deleted."

Table 1: Algorithm for Determining Eigen Values, Eigen Instances and Eigen Classes of a Given Ontology

```
let $\mathcal{V}$ be a set;
let $c$ be a class with a slot s taking as value a subset of $\mathcal{V}$ (from $\emptyset$ to $\mathcal{V}$ itself);
let $\mathcal{I}(c)$ be the set of instances of class $c$;
let $\mathcal{V}$(s,i) be the set of values of the slot s of the instance i of class $c$;

function GetEigenValues(s, i):
    let V ← $\mathcal{V}$(s,i);
    for = all j∈ $\mathcal{I}$(GetClass(i)) with j≠i do
        V ← V \ $\mathcal{V}$(s,j)
    end for all;
    return V;

function EigenSlotQ(s, i):
    return (GetEigenValues(s, i)≠ $\emptyset$);

let $\mathcal{S}(c)$ be the set of slots of class $c$;
function EigenInstanceQ(i):
    for = all $s \in \mathcal{S}$(GetClass(i)) do
        when EigenSlotQ(s,i) throw true
    end for all;
    return false;

function EigenClassQ($c$):
    return ($\bigwedge_{i\in\mathcal{I}(c)}$EigenInstanceQ(i));
```

When EigenSlotQ(s,i) returns **true**, the EigenValues of slot s uniquely determine the instance i; which means, in our case of ontology for military training, that (at least) one behaviour the learner is asked to play corresponds to only one article of the soldier's code or else only one pedagogical theme, thus avoiding the need for interpretation of the behaviour in the context of the article or the theme of the pedagogical session.

When EigenInstanceQ(i) returns **true**, the instance i has at least one EigenSlot, which, in our case of ontology for military training, means that an article has at least one behaviour as unique representative.

Finally, EigenClassQ(c) returns **true** when all the instances of class c have at least one unique representative.

Here is what these functions return in the case of Figure 1(b):

Table 2: Characterization of the Example

GetEigenValues(vertices, Triangle1)	{P2,P3}
GetEigenValues(vertices, Triangle2)	$\emptyset$
GetEigenValues(vertices, Square)	{P4,P7}
EigenSlotQ(vertices, Triangle1)	true
EigenSlotQ(vertices, Triangle2)	false
EigenSlotQ(vertices, Square)	true
EigenInstanceQ(Triangle1)	true
EigenInstanceQ(Triangle2)	false
EigenInstanceQ(Square)	true
EigenClassQ(Polygon)	false

As vertices is not an EigenSlot of Triangle2, Polygon is not an EigenClass. We either need to add rules to interpret the coordinates correctly, or add at least one slot to Polygon (e.g. barycenter) or further add classes to the ontology (e.g. subclasses of Polygon, e.g. Triangle, Square, Pentagon, ...) to distinguish between polygons.

Conclusion The set of functions presented in this paper greatly helped to automatically detect the need to design rules to interpret ambiguous values in our ontology (in particular, for setting up the use of this ontology in a pedagogical session) and ultimately, to complete the ontology design. These functions run in polynomial time in the number of values of the same slot across all the instances of the same class.

References

[1] Jean-André BENVENUTI, Laure BERTI–ÉQUILLE & Éric JACOPIN, *Ontological Parsing of XML Documents: A Use Case in the Domain of Training Military Staff*, in: Proceedings of the 21st International Conference on Distance Education (ICDE'04), Hong-Kong (February 2004), 10 pages.

[2] Jean-André BENVENUTI, Laure BERTI–ÉQUILLE & Éric JACOPIN, *From Ontology to Inference*, in: Poster Proceedings of Ingénierie des Connaissances (IC'04), Lyon (May 2004) (in french), 2 pages.

[3] Jean-André BENVENUTI, Laure BERTI–ÉQUILLE & Éric JACOPIN, *The* SABRE *Project: an Intelligent Tutoring System for Military Behaviours*, Poster at the CNRS Summer School, Autrans (Juillet 2004) (in french), 2 pages.

[4] Jean-André BENVENUTI, Laure BERTI–ÉQUILLE & Éric JACOPIN, *When Protégé and Rules become Parsing for Learning*, Workshop on Protégé with Rules, Madrid (July 2005), 3 pages.

[5] Commandement de la Formation de l'Armée de Terre, *Guide pour l'enseignement des principes de l'Exercice du Métier des Armes et du Code du Soldat*, Saint-Maixent l'École (2002), 92 pages.

[6] État-Major de l'Armée de Terre, *l'Exercice du Métier des Armes dans l'Armée de Terre : Fondements et Principes*, SIRPA Terre (1999), 41 pages.

[7] État-Major de l'Armée de Terre, *Code du Soldat et Guide du Comportement*, Directive n^{o} 004496/DEF/-EMAT/CAB du 28 juin 1999, 21 pages.

[8] Martin FOWLER, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley (3rd Edition, 2004), 175 pages.