Lifecycle of a Casual Web Ontology Development Process

Aditya Kalyanpur[1], Nada Hashmi[1], Jennifer Golbeck[1], Bijan Parsia[1]

[1]*University of Maryland, College Park*
MIND Lab, 8400 Baltimore Ave
College Park, Maryland 20740
{aditya, nada, golbeck}@cs.umd.edu, bparsia@isr.umd.edu

Abstract

Most of the existing ontology development toolkits are not catered towards 'casual web ontology development', a notion analogous to standard web page development. Key features of this process include easy and rapid creation of ontological skeletons, searching and linking to relevant existing ontologies and a natural language-based technique to improve presentation of ontologies. In this paper we elaborate the stages in a casual ontology development process, propose novel solutions to realize them and discuss implementations in an in-house ontology development toolkit – SWOOPed.

1. Introduction

In recent years, the notion of the 'Semantic Web' [1, 2] has been gaining prominence, in which users can create precise, unambiguous encodings of information in a machine readable format. Central to this notion is the idea of an *Ontology*, which is a formal specification of a conceptualization. The success of the Semantic Web is based on the existence of numerous distributed ontologies, using which users can annotate their data, thereby enabling shared machine readable content.

Ontologies, however, vary greatly in size, scope and semantics. They can range from generic upper-level ontologies (SUMO [3], Cyc [4]) to domain-specific schemas (NCI Cancer [5]). They can be created by knowledge representation (KR) experts or novice web users, differing widely in authoring style and formal semantics. They can be small ontologies containing a handful of concepts (FOAF [6]) or large ontologies containing thousands of terms and relationships (Galen [7]). In such a diverse and heterogeneous information space, *ontology engineering* assumes tremendous practical significance. Tools supporting it need to provide a seamless environment for browsing, searching, sharing and authoring ontological data.

A number of ontology development tools currently exist; notable among these are Protégé [8], Oiled [9], OntoEdit [10], OntoLingua [11] and WebODE [12]. Most of the tools provide an integrated environment to build and edit ontologies, check for errors and inconsistencies (using a reasoner), browse multiple ontologies, and share and reuse existing data by establishing mappings among different ontological entities. However, these tools are influenced by traditional KR-based ontology engineering methodologies, with steep-learning curves, making it cumbersome to use for casual web ontology development.

In this paper, we outline the lifecycle of a casual web ontology development process. Key emphasis is given to the following aspects:

- Aiding authors (esp. domain experts) build ontologies from scratch rapidly, using a short hand notation instead of a direct manipulation (DM) interface
- Facilitating reuse of existing data by providing advanced search capabilities to help locate specific concepts that can be borrowed or linked to while creating a new ontology.
- Helping novice web users explicitly understand ontological terminologies, concepts and relationships through natural language explanations

The rest of this paper is organized as follows: section 2 discusses the various stages in the casual ontology development process; section 3 explains our shorthand notation technique to create ontologies quickly; section 4 elaborates upon our concept search algorithm and section 5 discusses the use of NL-based explanations to guide ontology development and maintenance.

2. Casual Web Ontology Development

The casual web ontology development process is analogous to the standard web page development process, in which users have certain information they wish to deploy on the web, and through the use of standard HTML editors such as MS FrontPage etc. they can easily and quickly arrange and layout the information as desired, while linking to existing relevant information.

On similar lines, we envisage the casual ontology development process to be aimed at average "semantic" web users who would rather have a fast and guided approach to ontology building instead of a rigorous manual approach. Such a process would comprise of the following stages:

1. Users start with certain domain information they wish to model, and from this derive a loose terminology of concepts and relations that are needed to describe the domain.
2. The concepts are then arranged into a hierarchy and associated with specific properties (relations). A shorthand notation can be used to quickly layout the ontology this way (described in section 3).
3. A parser generates a rough (first-cut) ontology from the skeleton defined in stage 2, and a more robust ontology editing tool is used to refine the terms present in the ontology.
4. In the refinement stage, a search is provided to locate (existing) relevant ontological concepts that can be linked to or borrowed (using a copy/paste mechanism) in the current ontology (described in section 4).
5. Moreover, while browsing related ontological terms, natural language explanations of these terms are provided (described in section 5) to help users easily understand their semantics. Finally, explanations are also used to guide ontology building as noted in section 5.1.

Most of the techniques described in the subsequent sections have been implemented in an ontology engineering toolkit called SWOOPed. In addition to supporting the various stages of the casual ontology development process, SWOOPed is a highly scalable OWL [17] ontology browser and editor. It employs a plug-in architecture and is platform independent. Additional features include a hyperlinked interface to facilitate navigation across a single ontology (or between multiple ontologies), a resource holder panel to compare terms from different ontologies, and an ontology change tracker to maintain versioning information. These and other related features of SWOOPed are also noted in the subsequent sections where necessary.

3. Shorthand Notation to quickly draft an Ontology Skeleton

Most existing ontology editors provide a graphical user interface (tree-hierarchy etc) to create ontologies since its essential for users with no XML language background. Expert authors, however, may be quickly frustrated by the overhead of a direct manipulation (DM) interface. For experts, well designed DM systems can be sufficiently fast, though often using keyboard inputs will allow an expert to work faster [13]. One factor that limits the speed of experts in directly authoring semantic web documents is the overhead of the RDF/XML [18] syntax.

We support a quick shorthand notation that allows modeling experts to quickly enter information to build simple ontologies and instance data. The major features of this shorthand are:

1. Each line is prefaced with its type: Namespace (NS), Class (C), Property (P), Restriction (R), List (L) or an instance prefaced by the class of which it is an instance.
2. At least one namespace element – `default` must be defined at the start of the document
3. Indention is used to create subclasses and properties of each class. However, the same class/property can be defined at multiple places in the document (for example, as a subclass of another using indentation, and as a separate class on a new line). In this case, separately defined semantics for each resource are aggregated in the final OWL file.
4. To differentiate between terms that have identical names, NS prefixes can be used (e.g. `NS:ClassName`) to state explicit context, in the absence of which the `default` NS is assumed
5. Literals are differentiated from ontology resources when used as values of properties by enclosing them within quotation marks ("").

The following example illustrates our shorthand:

```
NS default = “http://student.owl”

C Person
 P name (String)
 P age (Integer)

 C Student
  P hasAdvisor (Person)
  P hasDegree (Degree)
   R MIN=1 hasDegree

   C PhDStudent
    R SOME hasDegree = PhD

C Degree
 L ONEOF (PhD, MS, BE)

Student JohnDoe
 hasDegree PhD
 age 25
```

This quick short hand is then automatically converted to the corresponding OWL format.

```
<owl:Class rdf:ID="Person"/>

<owl:DatatypeProperty rdf:ID="name">
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="age">
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Student">
   <rdfs:subClassOf rdf:resource="#Person"/>
   <owl:Restriction>
       <owl:onProperty
rdf:resource="#hasDegree"/>
      <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:
minCardinality>
   </owl:Restriction>
</owl:Class>

<owl:Class rdf:ID="PhDStudent">
   <rdfs:subClassOf rdf:resource="#Student"/>
   <owl:Restriction>
       <owl:onProperty
rdf:resource="#hasDegree"/>
       <owl:someValuesFrom
rdf:resource="#PhD"/>
   </owl:Restriction>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasAdvisor">
   <rdfs:domain rdf:resource="#Student"/>
   <rdfs:range rdf:resoruce="#Person"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasDegree">
   <rdfs:domain rdf:resource="#Student"/>
   <rdfs:range rdf:resoruce="#Degree"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Degree">
   <owl:oneOf rdf:parseType="Collection">
       <owl:Thing rdf:about="#PhD"/>
       <owl:Thing rdf:about="#MS"/>
       <owl:Thing rdf:about="#BE"/>
   </owl:oneOf>
</owl:Class>

<Student rdf:ID="JohnDoe">
   <hasDegree
   rdf:resource="#PhD"/>
   <age rdf:datatype="&xsd;integer">25</age>
</Student>
```

Preliminary studies (by testing a few example scenarios) suggest that this simple interface for creating ontologies and instances is an excellent interface addition for experts. This is clearly an issue that should be studied further, but initial impressions suggest that a quick, simple text based format for expert users will offer significant benefits to this package.

Note: At this point, we compare our shorthand notation with related work in this area, notably the following formats: N3 [14], Turtle [15] and OWL Abstract Syntax [16]. We find that the main purpose of these formats is to enhance readability for end-users, in place of the standard RDF/XML syntax. Our notation, however, is optimized for quick text entry and hence has less overhead than any of the formats listed above. We limit expressivity (i.e. no support for nested restrictions etc) to maintain readability, yet provide sufficient building blocks to construct a basic ontology model quickly, that can be refined later using a more powerful ontology editing tool. A more formal approach to our shorthand notation is being developed at the side.

4. Concept Search

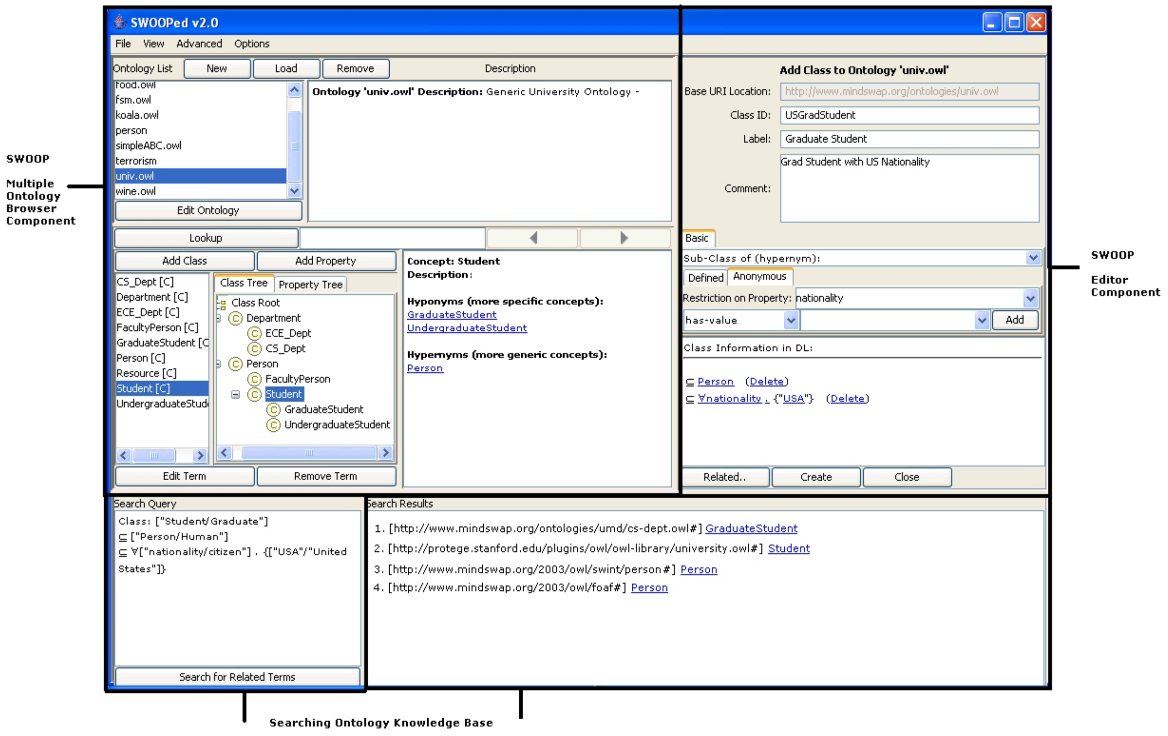
Searching for related resources in existing ontologies is a critical step in reusing ontological (T-Box) data. While most ontology engineering toolkits (listed in the introduction) have provisions to browse multiple ontologies, not much attention has been paid to searching for specific concepts defined in them. In order to accomplish this, users need to specify concept descriptions (queries) as generically or as specifically as possible. We propose a simple and intuitive technique in order to achieve this.

We regard keywords as the main part of the query (just as in standard search interfaces) and use them to represent generic descriptions. However, keywords can be combined in a constrained manner in order to create more specific descriptions. The constraints are imposed by ontology language constructs, in this case

OWL primitives. Finally, the query builder interface can guide the user in expanding/refining the keyword set (using synonyms, meronyms etc) by integrating a dictionary/thesaurus such as Wordnet at the backend.

The search is conducted over all ontologies present in the knowledge base (KB) of SWOOPed. Currently, ontologies can only be added to the KB manually, i.e. by specifying its URL or loading it from a local file. In the future, we hope to add a plug-in to SWOOPed that will crawl over OWL ontologies present on the Web and add it to the SWOOPed KB automatically.

We consider a simple example scenario to help illustrate our concept-search technique.



Ontology List: food.owl, fsm.owl, koala.owl, person, simpleABC.owl, terrorism, univ.owl, wine.owl

Ontology 'univ.owl' Description: Generic University Ontology -

CS_Dept [C]
Department [C]
ECE_Dept [C]
FacultyPerson [C]
GraduateStudent [C
Person [C]
Resource [C]
Student [C]
UndergraduateStud

Class Root
- Department
  - ECE_Dept
  - CS_Dept
- Person
  - FacultyPerson
  - Student
    - GraduateStudent
    - UndergraduateStudent

Concept: Student
Description:

Hyponyms (more specific concepts):
GraduateStudent
UndergraduateStudent

Hypernyms (more generic concepts):
Person

Add Class to Ontology 'univ.owl'

Base URI Location: http://www.mindswap.org/ontologies/univ.owl
Class ID: USGradStudent
Label: Graduate Student
Comment: Grad Student with US Nationality

Sub-Class of (hypernym):
Restriction on Property: nationality
has-value

Class Information in DL:

⊆ Person (Delete)
⊆ ∀nationality . {"USA"} (Delete)

Search Query

Class: ["Student/Graduate"]
⊆ ["Person/Human"]
⊆ ∀["nationality/citizen"] . {["USA"/"United States"]}

Search Results

1. [http://www.mindswap.org/ontologies/umd/cs-dept.owl#] GraduateStudent
2. [http://protege.stanford.edu/plugins/owl/owl-library/university.owl#] Student
3. [http://www.mindswap.org/2003/owl/swint/person#] Person
4. [http://www.mindswap.org/2003/owl/foaf#] Person

Figure 1: The SWOOPed Interface

As shown in **Figure 1**, the user can search for concepts related to those currently being defined in his ontology. In this case, the new concept being created represents a "*Graduate Student with US Nationality*". It is defined as a subclass of concept *Person* with a value restriction on property *nationality*, whose value is restricted to the string literal "*USA*".

The structure of the corresponding search query resembles the concept being defined, containing keywords joined by the appropriate DL-based constructs. As shown, the user can provide additional keywords (synonyms) to expand the search criteria. The search algorithm works as follows:

1. **For each keyword-set, it finds matching terms in the ontology KB**. Depending on the relative position of the keyword-set in the concept definition, appropriate ontological terms, i.e. classes, properties, instances and data-values are matched. While a basic string matching algorithm is used to accomplish this, the user can modify the matching criteria i.e. search against the resource *ID*, *label*, or *comment*; obtain an exact, part-of or substring match etc.

2. **It breaks the original concept description into its component axioms, creates a query for each axiom using the matched term-sets (taking terms pair-wise) and feeds each query to a Reasoner**. So in this case, supposing the keyword 'Student' matches two concepts in the

entire ontology KB - *Student, GradStudent,* and the keyword ‘Human’ matches two concepts - *Human* and *HumanBeing,* four axiomatic queries are created from the original concept description, namely:

Student rdfs:subClassOf Human?

Student rdfs:subClassOf HumanBeing?

GradStudent rdfs:subClassOf Human?

GradStudent rdfs:subClassOf HumanBeing?

These queries are posed to a DL-reasoner that reasons over the entire ontology KB, and returns a *Yes/No* answer for each of them.

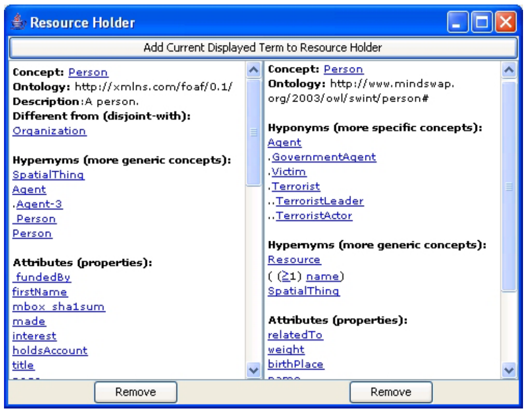
Note that the user can choose to expand the result set by allowing the tool to check satisfaction of additional axiomatic queries similar to the original (for e.g. some-value restriction instead of a has-value).

3. **The terms are ranked according to the number of axioms (based on the original concept description) satisfied in step 2 and returned as results to the user in that order.** Partial matches, based on satisfaction of similar but inexact queries, are weighted accordingly while prioritizing results.

4.1 Browsing and Comparing Related Terms

Simply finding related ontological terms in a multiple ontology environment isn’t as useful, if a seamless navigation interface to browse and compare these terms is not present. SWOOPed supports this functionality by providing navigation across ontologies via hyperlinks (between semantically linked terms) similar to standard web browsers (see the SWOOP browser component in Figure 1). Additional features such as bookmarking and a history-log are present to access specific terms quickly.

Furthermore, SWOOPed has provision to compare the description (semantic definition) of similar terms present in different ontologies. For instance, the concept *Person* defined in the *FOAF* ontology differs significantly from the concept *Person* defined in the *Mindswap.Person* [19] ontology. These two concepts can be compared against their DL-based definitions, associated properties and sample-instances (see **Figure 2**). The idea is to give the user a clear picture of the concept's intended meaning and usage, thereby aiding in the process of selection of the appropriate concept to link to or borrow.



Concept: Person
Ontology: http://xmlns.com/foaf/0.1/
Description:A person.
Different from (disjoint-with):
Organization

Hypernyms (more generic concepts):
SpatialThing
Agent
.Agent-3
 Person
Person

Attributes (properties):
 fundedBy
firstName
mbox_sha1sum
made
interest
holdsAccount
title

Concept: Person
Ontology: http://www.mindswap.org/2003/owl/swint/person#

Hyponyms (more specific concepts):
Agent
.GovernmentAgent
.Victim
.Terrorist
..TerroristLeader
..TerroristActor

Hypernyms (more generic concepts):
Resource
((≥1) name)
SpatialThing

Attributes (properties):
relatedTo
weight
birthPlace

Figure 2: Comparing Different *Person* Concepts

4.2 Linking or Borrowing Ontological Data

Having found related concepts/properties that the user could potentially use in the ontology being created, the editor interface must provide the user with the ability to either link to the data directly (with or without importing the entire external ontology) or borrow a specific subset of it (using a copy-paste mechanism). In the former case, the user can choose from among the set of ontology mapping constructs provided by the language (such as equivalence, subclass etc) to link the local ontological term to the external resource. In the latter case, we describe an iterative copy-paste mechanism to selectively borrow only a specific fragment of the external ontology.

Consider the following example scenario: A user is interested in defining a concept similar to the concept *SportsPerson* in a large external ontology. Hence, the system facilitates the copy/paste of all axioms involving *SportsPerson* from this ontology. But doing so presents some new classes and/or properties that are dependent on class *SportsPerson* in the external ontology; say it’s a subclass of concept *Person*, has property *age* and so on. Thus, the user is presented with a list of resource dependencies based on the borrowed term that he can again choose to copy/paste individually into his own ontology (see **Figure 3**). This iterative process goes on till the user is satisfied with the set of elements (and hence semantics) borrowed from the external ontology, leaving the remaining resources to be atomic if desired. Also note that during this process, the user can specify alternate names for the borrowed resources or choose to keep the same namespaces from the external ontology. In this manner, the user can quickly setup a new ontology based on a related external ontology by focusing on its relevant parts alone.

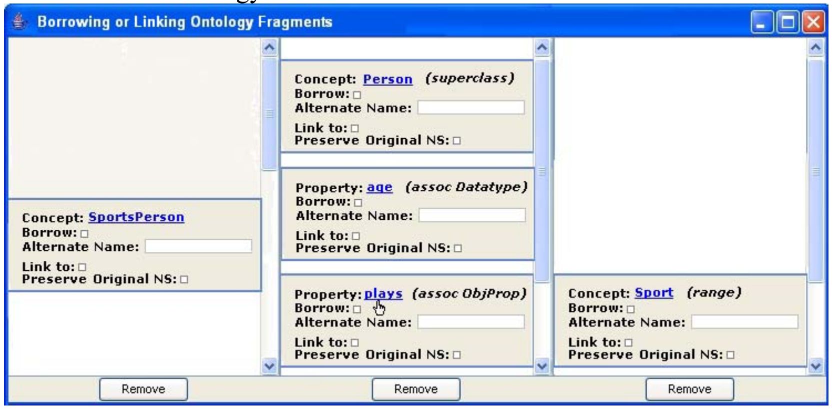


Figure 3: Iterative Copy/Paste of an Ontology Fragment

5. Natural Language Explanations

The need to provide a natural language explanation of terms in an ontology arises from the fact that the intended purpose of ontologies is for information sharing which could involve external parties that have little or no background knowledge of the ontology domain. In such cases, it becomes the responsibility of the domain experts creating the ontology to provide textual documentation for the terms within. However, when this is not available, understanding the explicit meaning behind the terms can be difficult.

Moreover, complex concepts and relationships in an ontology are constructed using a combination of DL based constructs that are difficult to read and follow. Readability can be greatly enhanced if a natural language representation of the DL-based terms is provided. Also, the NL generation system should be flexible in that the parameters used to specify textual descriptions can be customized based on user preferences.

We use shallow text generation [20] to display OWL ontologies in NL by employing predefined text plan templates, which get populated from the term's ontological description. In addition, we use various heuristics primarily focusing on Anonymous Classes (native to OWL) to improve the final quality of the results (i.e. ensure more readable sentences). **Table 1** considers a concrete well-defined example to demonstrate the utility of our NL generation system.

Table 1: Natural Language (NL) Explanation of Concepts

NL Generation Algorithm	Term Description
Original Concept 'Blandfishcourse'	intersectionOf(Mealcourse, allValuesFrom(hasFood, Blandfish), cardinality(hasDrink, 1), someValuesFrom(hasDrink, (hasValue(hasFlavor, "Delicate"))))
First iteration: (a) text-plan template - using predefined phrases to form sentence clauses, (b) handle anonymous classes - using property's domain class while describing restrictions	(is-a Mealcourse) (and) (is-a Mealcourse that-always-has value of hasFood equal to Blandfish) (and) (is-a Mealcourse that-only-has 1 value of hasDrink) (and) (is-a Mealcourse that-atleast-has one value of hasDrink equal to (is-a Drink that-has value of hasFlavor equal to "Delicate"))
Second iteration: (a) combining information about the same subject across all clauses (using conjunctions - that, with etc)	is a Mealcourse that always has value of hasFood equal to Blandfish and that has only 1 value of hasDrink equal to a Drink that has value of hasFlavor equal to "Delicate"
Third iteration: (a) using property naming conventions - parsing common prefixes - has, is, etc to improve sentence construction	is a Mealcourse that always has Food Blandfish and that has only one Drink (Drink) that has Flavor "Delicate"
Fourth iteration: (a) removing the extra words and correcting capitolization	is a Mealcourse that always has food Blandfish and only one drink that has flavor "Delicate"

5.1 Explanations to Guide Ontology Development

Currently, most ontology editors assume that the user is well aware of the semantics of the ontology language, whereas in many cases, this is not true. While the user may be familiar with the basics of the language, such as its underlying model, types of semantic constructs and its basic syntax/purpose, there are far too many hidden (interdependent) semantic nuances in a DL-based language that even an experienced user might not know.

Moreover, while the user always has a certain choice in specifying the description of an entity, not knowing the alternatives or their implications can cause the user to incorrectly model it i.e. specify its meaning different from what was intended. We feel that it's the responsibility of the ontology editor to guide the user in making the right (intended) choice while modeling the domain based on the specific semantic nuances of the ontology language.

A rule-based system seems an ideal choice to implement this functionality. A set of manually hard-coded rules can be written beforehand, each of which gets fired on specific user-actions such as adding a class, specifying an intersection class definition etc, and accordingly displays the alternatives or implications of the action based on current state of the ontology and/or semantics of the language. A back-end reasoner would be needed in order to make inferences based on the user action. This work is still in its infancy and has not been implemented in SWOOPed yet.

6. Future work

This paper represents work in progress. Some of the solutions proposed in the previous sections need to be elaborated upon, implemented and optimized in SWOOPed. Moreover, a formal evaluation of the features needs to be done by performing usability studies and comparing it against existing ontology authoring techniques.

7. Conclusion

In this paper we have outlined the lifecycle of a *casual web ontology development* process. Various key stages in this process include the use of shorthand notation to draft ontology skeletons quickly, a powerful ontology search algorithm that combines keywords with DL-based constructs to find related concepts, an iterative copy-paste mechanism to borrow relevant fragments of a related ontology, and natural language based presentation of terms in order to build and maintain ontologies effectively.

While we have identified these stages and proposed novel solutions to each of them (some of which are implemented in an ontology development toolkit – SWOOPed), more work needs to be done to fully evaluate our methodology.

8. References

[1] Berners-Lee, T. and M. Fischetti, "Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor", Harper, San Francisco, 1999.

[2] Berners-Lee, T., Hendler, J. and Lassila, O. "The Semantic Web" *Scientific American*, May, 2001

[3] Pease, A., Niles, I., and Li, J. 2002. "The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications". *In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*, Edmonton, Canada, July 28-August 1, 2002.

[4] Lenat, D. B. "Cyc: A Large-Scale Investment in Knowledge Infrastructure." *Communications of the ACM 38*, no. 11 (November 1995).

[5] Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, James Hendler, Bijan Parsia, and Jim Oberthaler. "The National Cancer Institute's Thesaurus and Ontology". *Journal of Web Semantics*, 1(1), Dec 2003.

[6] Brickley D, Miller L. The Friend of a Friend (FOAF) Vocabulary Specification: http://xmlns.com/foaf/0.1/

[7] Rector AL, Gangemi A, Galeazzi E, Glowinski A J, Rossi-Mori A. "The GALEN Model Schemata forAnatomy: Towards a re-usable Application-Independent model of Medical concepts". Published in P Barahona, M Veloso, J Bryant (eds) , *Proceedings of Medical Informatics in Europe MIE 94*, pp 229-233.

[8] Using Protégé-2000 to Edit RDF. *Technical Report. Stanford University*.

http://www.smi.Stanford.edu/projects/protege/protege-rdf/protege-rdf.html

[9] Bechhofer, S.; Horrocks, I; Goble, C.; Stevens, R. "OilEd: a Reason-able Ontology Editor for the Semantic Web". *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence*, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001.

[10] Sure, Y.; Erdmann, M.; Angele, J.; Staab, S.; Studer, R.; Wenke, D. "OntoEdit: Collaborative Ontology Development for the Semantic Web". *International Semantic Web Conference (ISWC02)*. Sardinia. Italy. June, 2002. LNCS 2342. pp. 221-235.

[11] Farquhar A., Fikes R., Rice J., "The Ontolingua Server: A Tool for Collaborative Ontology Construction". *10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada. 1996.

[12] Arpírez, J.C.; Corcho, O.; Fernández-López, M.; Gómez-Pérez, A. "WebODE: a scalable ontological engineering workbench". *First International Conference on Knowledge Capture* (K-CAP 2001). Victoria, Canada. October, 2001.

[13] Morgan, K, R.L. Morris, and S. Gibbs, "When does a mouse become a rat? or Comparing performance and preferences in direct manipulation and command line environments," *The Computer Journal,* 1991: 34 (3). 265-271.

[14] Berners-Lee T, Connolly D. "Primer: Getting into RDF & Semantic Web using N3" http://www.w3.org/2000/10/swap/Primer.html

[15] Beckett D. "New Syntaxes for RDF" Submitted to WWW2004

[16] Schneider-Patel P, Hayes P, Horrocks I. "OWL Web Ontology Language Semantics and Abstract Syntax" *W3C Recommendation 10 February 2004*. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/

[17] Word Wide Web Consortium (W3C). OWL "Web Ontology Language". http://www.w3.org/TR/owl-ref

[18] Brickley, D and R.V. Guha, "Resource Description Framework (RDF) Model and Syntax Specification", *W3C Recommendation* submitted 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[19] The MINDSWAP Person Ontology: http://www.mindswap.org/2003/owl/swint/person

[20] Busemann S, Horacek H. "A Flexible Shallow Approach to Text Generation". *Proc. 9th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Canada, August 1998, 238-247