Wordified Ontologies: Evaluating a Novel Paradigm for Ontology Editing

Aisha Blfgeh[1,2] and Phillip Lord[1]

[1] *School of Computing, Newcastle University, UK*
[2] *College of Computer Sciences and Engineering, University of Jeddah, Saudi Arabia*

Abstract

Ontologies can be edited with tools such as Protégé, or with various other forms of source code. The editing process involves insertion, deletion, or fixing errors. In this paper we propose an editing process using Microsoft Word, where users can manipulate any text, adding comments and use all the facilities provided by a familiar word-processing environment. This new technique for visualising and editing ontologies has been tested and evaluated; the results are promising as modifying an ontology in Word is preferred by users to Protégé for some ontology editing tasks. We suggest, therefore, that alternative text based representations and office tools may be useful in the ontology engineering lifecycle.

Keywords:
Ontology Editing, Wordified ontology, User Evaluation

Introduction

Ontology development is a collaborative process which involves a sustained interaction between domain specialists and ontology developers. As shown in Figure 1, we designed a document-centric workflow to enable the co-ordinated use of Microsoft Office tools (Excel and Word) for ontology development. An Excel spreadsheet is used as a source of values that instantiate patterns, defined in Tawny-OWL [1] source code [1], to construct the ontology. We have also generated a Word document of an ontology; we denote this representation as a *Wordified Ontology*. It allows domain specialists to cooperate and interact with the developers in editing the ontology during the development process [2]. The use of Word documents enables us to include the documentation of the ontology with the computational components.

In some ways, this is similar to an "Intermediate representation" as defined by Rector et al˙ [3], where the knowledge from experts is transformed into a semi-formal syntax that the ontology developers use to deal with ontological information. Also, using different syntaxes to instantiate patterns is not new; for example in OPPL (Ontology Pre-Processing Language) [4], and DOSDP [5] where an abstract syntax is used for effectively editing the ontology. Office tooling has been integrated into the ontology development process before such as with Populous [6] and our own, Excel-based approach [1]), however only with the more structured forms of spreadsheets. To our knowledge, the use of arbitrary syntax tightly integrated and presented in rich Word documents is novel.

[1] `https://github.com/phillord/tawny-owl`

User evaluation is a standard part of the software engineering cycle; it has previously been applied to various aspects of ontology engineering, including the use of foundational ontology in ontology development [7], and finding frequent user activities in Protégé using Eye-tracking analysis [8].

Others have concluded that [9] the tools used for reading and understanding an ontology play a critical role in determining the usability of that ontology. Furthermore, by using a verbalised version of the ontology in their evaluation practice, they found that this supports the identification of mistakes in the ontology.

In our previous work, we have shown that it is possible to *wordify* an ontology; however, to demonstrate that it is also useful to do this, we need some form of evaluation. In this study, we assess the comprehension/manipulation of an ontology presented in this way. We conducted several experiments where users read and manipulated the Wordified ontology as well as performing the same tasks using Protégé; then we assessed their performance and measured their level of satisfaction using feedback forms.

This paper is organised as follows: first we describe existing alternative tools for ontological documentation, then we show our new visualisation version of the ontology. After that, we describe the experiments with users and show the results. Finally, we discuss the results and draw conclusions.

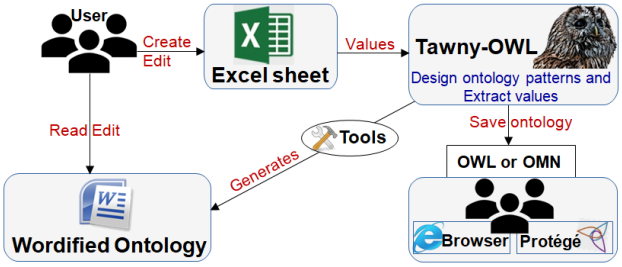


Figure 1: Document-Centric Ontology Development Workflow

Alternative representations of Ontological Knowledge

There have been many other attempts to present ontological knowledge in predominately textual formats. For example, OWLDoc is a Protégé plugin that generates HTML documentation [10]. It was inspired by JavaDoc, which does something similar with Java source code. The aim of OWLDoc is to provide a browsable, but not editable, experience of the ontology inside Protégé or in

a standalone web browser. While this works well, if errors are discovered the user has to move into a different environment to fix them.

Another mechanism for visualisation [2] was the "Intermediate Representation" [3]. This mechanism produces semi-structured text representations (Figure 2a), which were designed for domain specialists to read and edit, before being checked and cleaned by knowledge engineers who would correct their syntax and semantics; this representation was used to enable authoring of the final ontology. Many years after, a similar practice has been incorporated into software engineering with *Behaviour-Driven development* tools such as Cucumber; this allows semi-structured statements (Figure 2b) to define requirements which can be tested computationally as part of the software testing lifecycle.

```
MAIN plastic construction
    ACTS_ON ulna
    BY_TECHNIQUE transplanting
        ACTS_ON bone
WITH immobilising
BY_MEANS_OF fixation device
```

(a) Intermediate representation

```
Feature: Return Microwave
Scenario: Fred gets his money back
   Given Fred has bought a microwave
   And the microwave cost 100
   When we refund the microwave
   Then Fred should be refunded 100
```

(b) Cucumber

Figure 2: Intermediate representation and Cucumber text

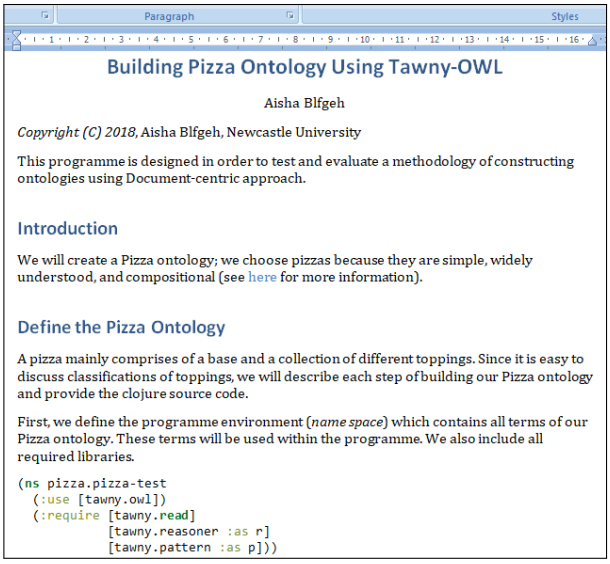
An alternative to an intermediate representation is to use an ontology syntax, which is designed to be readable to knowledge engineers such as *Manchester Syntax*; this was aimed at editing and representing all the aspects of an ontology, including the extra-logical parts which are critical for understanding the ontology in the context of its domain [11]. Unlike other syntax it is frame-based, grouping statements about a single domain concept, rather than axiom-based where statements are essentially unordered.

We use the mechanism of adding comments in the source code; we have previously described these as literate ontologies [12, 13] where we produce LaTeX, ASCII and OWL versions from a single source code. In our case, we produce a Microsoft Word version, and the final Wordified ontology is formatted as we desired. Figure 3 shows an extract of an Wordified ontology. In the following section we explain the mechanism for generating an Wordified ontology.

Microsoft Word and Ontology Representation

Although there is no clear structure of what ontology documentation should be, we started by creating a guidance document on how to build a "Pizza Ontology". We have included an explanation of logical classifications of the ontology in a narrative style as well as the Tawny-OWL source code which fulfils the chronological story of Pizza Ontology construction. Unlike Protégé, this narrative structure of the ontology offers the ability to explore ontologies as a linear, narrative document.

A *Wordified ontology* is the narrative version of the ontology and includes the whole source code of an ontology using Tawny-OWL syntax. We intentionally include the Tawny-OWL source code because of its textual representation. It was designed after Manchester syntax to be straightforward allowing a developer (or non developer) to read it without deep knowledge of the syntax. Therefore, we chose to have the complete source code of the ontology in the Wordified ontology, also we have the other purpose of testing the ability to comprehend the Tawny-OWL source code by users.


Paragraph Styles

Building Pizza Ontology Using Tawny-OWL

Aisha Blfgeh

Copyright (C) 2018, Aisha Blfgeh, Newcastle University

This programme is designed in order to test and evaluate a methodology of constructing ontologies using Document-centric approach.

Introduction

We will create a Pizza ontology; we choose pizzas because they are simple, widely understood, and compositional (see here for more information).

Define the Pizza Ontology

A pizza mainly comprises of a base and a collection of different toppings. Since it is easy to discuss classifications of toppings, we will describe each step of building our Pizza ontology and provide the clojure source code.

First, we define the programme environment (*name space*) which contains all terms of our Pizza ontology. These terms will be used within the programme. We also include all required libraries.

```
(ns pizza.pizza-test
  (:use [tawny.owl])
  (:require [tawny.read]
            [tawny.reasoner :as r]
            [tawny.pattern :as p]))
```

Figure 3: Wordified ontology: A structured and readable text with headings and source code syntax highlighted

Therefore, our mechanism for adding ontology documentation is based on adding a rich commentary text within the Tawny-OWL source code through out the ontological and software statements: a form of literate programming. Additionally, we use markup annotations to produce a structured documentation as well as the Tawny-OWL source code text. This form of source code can be transformed into a Word document; in Figure 3, we show how the text is structured and formatted with headings and subheadings; the source code of Tawny-OWL is also shown in a syntax highlighted text blocks.

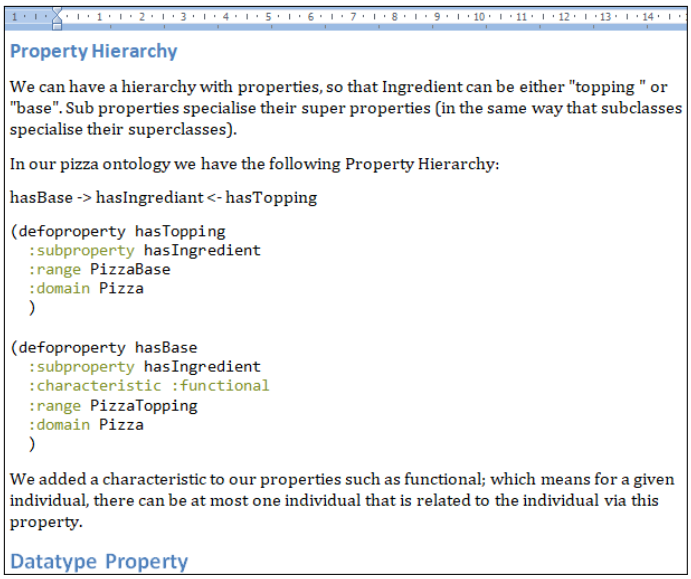
Next, we describe the evaluation process of Wordified ontologies.

Evaluation Experiments

The aim of our evaluation is to discover how easy for users to comprehend and interact with such a new form of the ontology. We achieve this by having the users read an ontology, understand their structure and search for errors introduced into our sample ontologies.

We arranged controlled testing sessions to test the comprehension of the Wordified ontology and compare the performance by

[2] We use the term *visualisation* in a broader fashion than is common, to include text on screen



Property Hierarchy

We can have a hierarchy with properties, so that Ingredient can be either "topping " or "base". Sub properties specialise their super properties (in the same way that subclasses specialise their superclasses).

In our pizza ontology we have the following Property Hierarchy:

hasBase -> hasIngrediant <- hasTopping

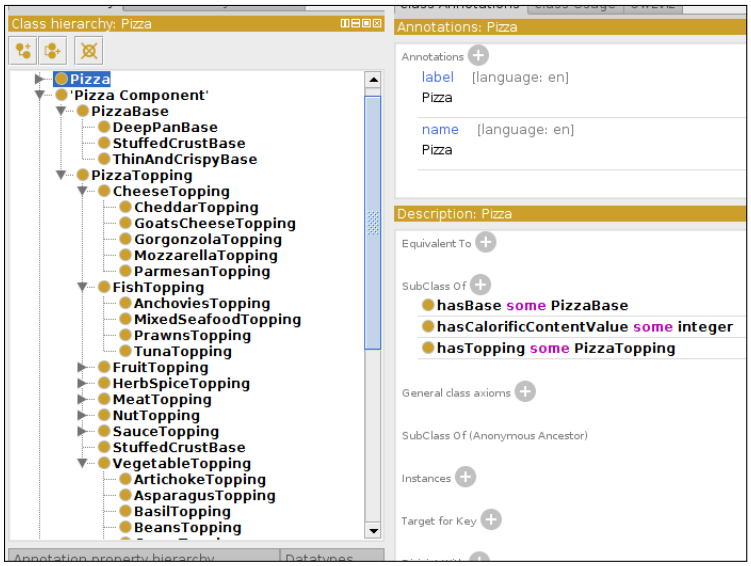
```
(defoproperty hasTopping
  :subproperty hasIngredient
  :range PizzaBase
  :domain Pizza
  )

(defoproperty hasBase
  :subproperty hasIngredient
  :characteristic :functional
  :range PizzaTopping
  :domain Pizza
  )
```

We added a characteristic to our properties such as functional; which means for a given individual, there can be at most one individual that is related to the individual via this property.

Datatype Property

(a) Wordified Ontology



(b) Ontology in Protégé

Figure 4: Testing ontologies

visualising the same ontology in Protégé. In other words, the participants examine the same ontology in the two different visualisations. These visualisations can be seen in Figure 4 below[3]. Our participants were mostly postgraduate students and researchers with a variety of backgrounds and experience in ontologies. Some have a decent knowledge in building ontologies and some are totally unfamiliar.

We wished to test the ease of reading and understanding the Wordified ontology as well as how easy it is to discover errors. Therefore we injected a few errors, both logical and extra-logical, into the ontology for users to find during the test. We prepared multiple versions of ontologies, so that the participants could explore different visualisations of the same ontology, but with different sets of errors.

In each version of the ontology, we have engineered four or five errors to be detected. These errors are either in classification or in some logical specifications of properties/classes, such as the range/domain of a property, and the disjointedness of sub-classes. Figure 4 shows examples of one error in Wordified ontology and in Protégé. Additionally, We have not included any spelling errors because they are instantly can be detected by the Microsoft Word software spelling checker, and in Protégé test; the participants are not allowed to run any reasoner checks.

After introducing the experiment objectives and the tasks to perform in the test, we provided instruction sheets in details for more assistance during the session. A participant starts with either a Wordified ontology or in Protégé, the selection process being randomly performed by the experimentor to ensure the variety between the participants in one session [5]. Hence, there are two different paths through testing experiment, depending on which version of the ontology they see first (Figure 5). This was done to ensure that preference between visualisation would not be affected by either fatigue or use of knowledge from the first test affecting the second. Additionally, we set an equal time limit to spend in each part to ensure fairness of the two tests. Finally, the participants write their feedback electronically about the following aspects:

- Clarity of ontology structure.
- Understanding the construction flow of the ontology.
- The ease of reading the ontology.
- Editing the ontology.
- Finding errors.

The feedback questions aim to measure how easy it is to read, comprehend, and edit Wordified ontology. All answers were rated using a five point Likert scale.

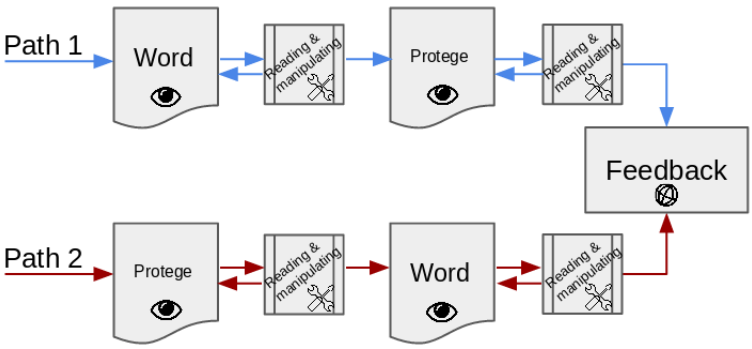


Figure 5: Evaluation Experiment Workflows

[3]Each ontology in this Figure has an error, can you find them?[4]

[5]There was no particular procedure to randomise the selection; we tried to maintain a reasonable distribution amongst our sample.

Expertise level of the participants

To maintain the fairness of our test and ensure the variety of experience levels, we asked the subjects about their level of experience and education. We asked basic demographic data as some people may under/over estimate their abilities depending on many factors, such as gender, age, etc (data not shown). The subjects experience level in ontology construction and usage is shown in Figure 6. About 32% and 40% of the subjects are "Somewhat Familiar" with ontology usage and construction respectively. We had about 16% and 20% of the participants consider themselves to have a "Mastery" level in ontology usage and construction. In our experiment, therefore, the experience was reasonably spread across range of different expertise; this allowed us to get a reasonable sample size, which would not have been possible if we restricted, for example, only to experts; it is probably reflective of the ontology development community, which also has many different levels of expertise. None of the participants were members of our lab, and had not seen Wordified ontologies previously.

Evaluation results

In this section, we explore the results of the evaluation and describe the feedback answers from users about the five aspects mentioned in the previous section showing their preferences in these aspects.

Reading and understanding the construction flow of the ontology

In this context, reading the ontology refers to the action of exploring and browsing the ontology. Because of our test on the Wordified ontology, we use the term "reading" as it also has more text and documentation of the ontology. Also, understanding the construction flow refers to the ability to follow the development process of the ontology regardless of the representation format.

Generally, over 60% of the participants find that the ontology is easy to read in both representations as shown in Figure 7. Although we designed the Wordified ontology with comprehensive text that explains the construction of the ontology, 40% feel "Neutral" about understanding the construction flow of the ontology where the same percentage can understand the construction flow in the Protégé (see Figure 8).

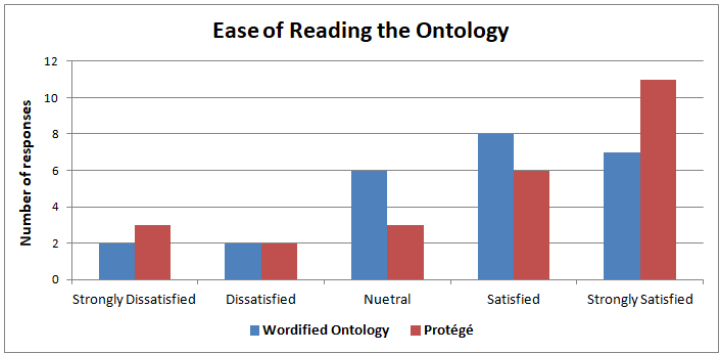


Figure 7: The ease of reading the ontology in: Wordified ontology and Protégé

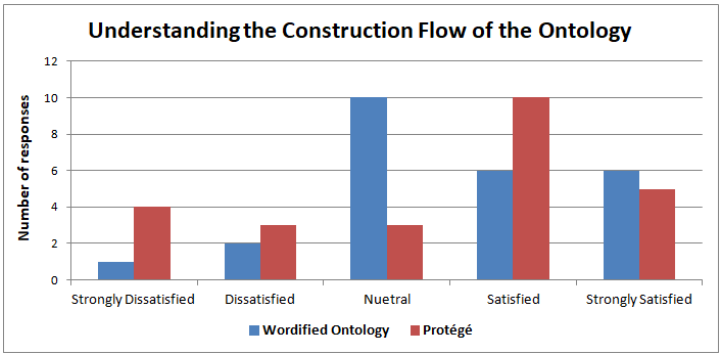


Figure 8: Understanding the construction flow of the ontology in: Wordified ontology and Protégé

Editing the ontology

Editing the ontology in this context refers to any form of the modification: deletion, insertion or update of the ontology. In a Wordified ontology, users can also add comments and annotate the text using the *Track changes* facility in Word. We asked the participants to turn on *Track changes* before they perform any changes in the ontology. This helps in saving time and effort for the ontology developers when updating the source code of the ontology accordingly.

As shown in the Figure 9 below, most participant are either "Satisfied" or "Strongly Satisfied" with their performance in the editing tasks during the test. This indicates modification of a Wordified ontology is preferred to modification in Protégé.

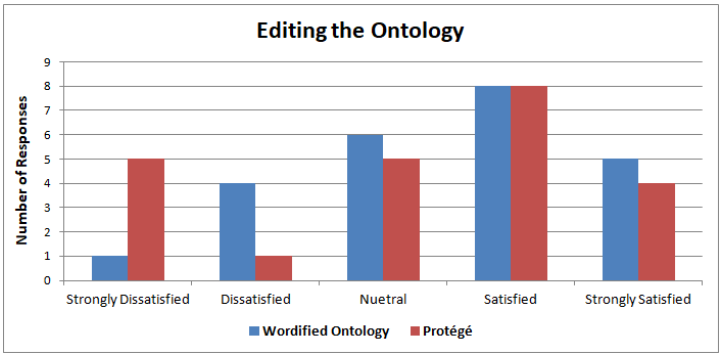


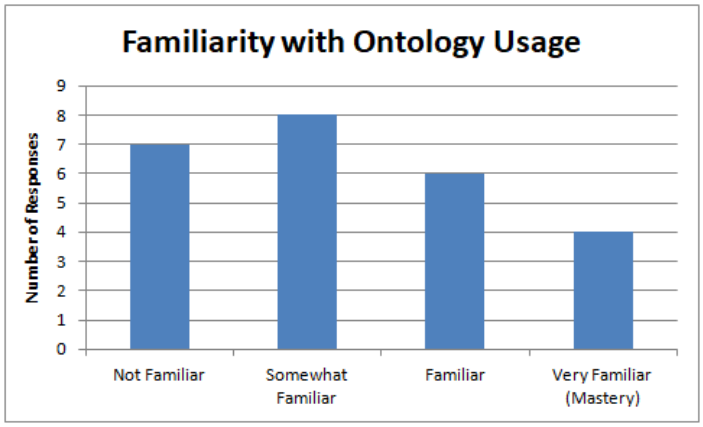
Figure 9: Results of editing the ontology in: Wordified ontology and Protégé

Finding Errors

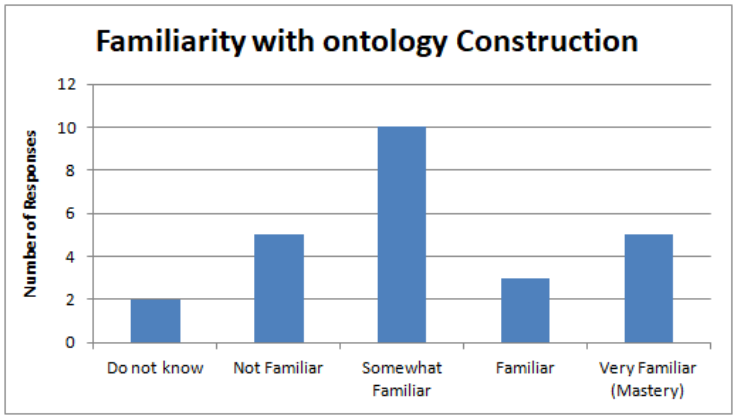
One of the participants task was to search for the errors we included in the ontology and correct them. We intended to discover how easy and quickly to spot errors in the ontology; hence we limited the time available for this task. This was a hard task, most participants [6] managed to detect at most a single error in the Wordified ontology and two errors in Protégé.

The participants feedback results are quite similar in both parts, looking at Figure 10, there are nearly the same number of participants (nine and ten) either "Satisfied" or "Strongly Satisfied"

[6] 6 participants using Wordified ontology and 4 using Protégé.



(a) Experience Level in Ontology Usage



(b) Experience Level in Ontology Construction

Figure 6: Experience level of our subjects: a) Ontology Usage and b) Ontology Construction

with this task in Wordified ontology and Protégé. A quarter find it difficult to perform the task of finding errors in Protégé and nearly a third in Wordified ontology. This could be due to the time limit we set in our experiments.

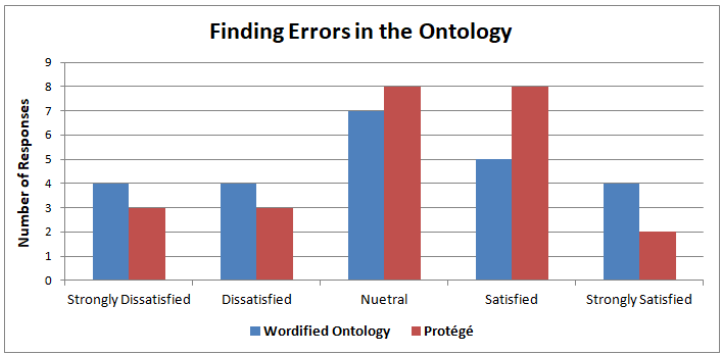


Figure 10: Results of Finding Errors in the ontology using: Wordified ontology and Protégé

Overall preferences

At the end of the session, we asked the subjects about their final preferences in using the Wordified ontology over Protégé in the main three aspects: Reading (Exploring), Editing and Learning about the ontology. The overall preferences of the users between the two forms are shown in Table 1. Wordified ontology seems to be preferable in editing, where Protégé is more convenient in reading and learning the ontology due to the hierarchical representation.

We also split the results of "Experts" preferences and "beginners" (or non-experts), in order to ensure that the level of experience does not affect users perspectives. We found no divergence in the results; the Wordified ontology remain the preferable as can be seen in the Figure 11 below.

Table 1: The preferences of user in using the Wordified ontology over Protégé.

	Wordified Ontology	Protégé	Both
Reading	20%	68%	12%
Editing	56%	24%	20%
Learning	20%	56%	24%

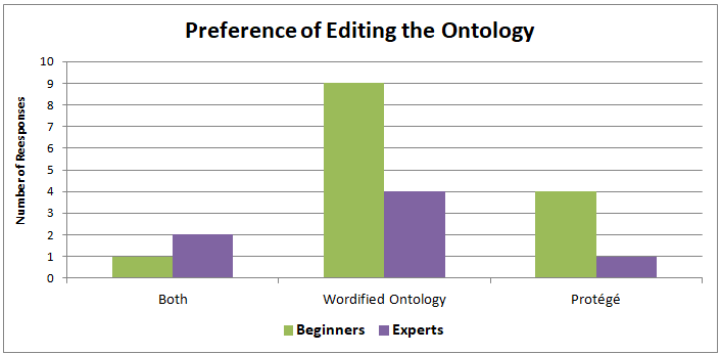


Figure 11: Users preferences in "Editing" according to the their level of expertise

Discussion

In this paper, we have evaluated whether an alternative form of representation, namely the Wordified ontology, is useful and usable by ontology users, both experts and non.

Our analysis of other work in this area shows that, in the field of ontology engineering, user experience testing is relatively limited with notable exceptions being the evaluation of an application ontology [9, 7], and the analysis of Protégé activities [8]. This paper shows the value of this form of user evaluation because the results were substantially different from our initial expectations: we thought people would prefer Protégé, especially for editing, as it is more familiar and has been longer in development.

Despite that there is no significant difference between the two formats of the ontology in our test (the p-values are less than 0.05), which means that the results are relatively close between both formats; the Wordified ontology seems to compete the Protégé

software especially in the area of documenting and editing the ontology, this is due to the familiarity of the Microsoft Word for different kinds of users, which requires no prior skills to deal with these Wordified ontologies.

Alternative representations have been tried before such as the previously mentioned "Intermediate Representation" [3], also the auto-generation of textual class definition [14]; in these cases, the representations have been textual and did not focus on the application that the users would use to interact with the text. Likewise, for more formal representations, Manchester Syntax [11] and DOSDP [5] efforts have focused on the representation alone.

In this paper, we have tested the utility of Wordified ontologies, and this shows that the Wordified ontology has a promising place in the future of ontology development. Although our word-based presentation of ontologies is relatively immature, users still found it useful for understanding and debugging ontologies. Counter to this, most users preferred Protégé for understanding overall flow, probably because of the hierarchical browser, as supported by the previous analysis of Vigo et al˙ [8], which showed that users spent 45% of their time looking at it.

There is still significant work to be done in improving the presentation of Wordified ontologies. While it is well understood how comments can be written in a light-weight markup and transformed into Word structure, we lack a good understanding of what text should go into documentary comments and what should be represented, for example, as `rdfs:comments` into the annotation of the ontology. Similarly, presenting the formal parts of the ontology as source code in Word is clearly not ideal; a more textual representation (such as [3] or [14]) might be preferred by users. For a tool such as Tawny-OWL, the non-ontological parts of the source code are also challenging. Finally, we need to explore different ways of integrating Wordified ontologies into the development process; either for new or existing ontologies.

Tools like this have, however, proven to be very popular in software development, and form the basis for behaviour-driven development (BDD) [15]; from here, we have taken some of our motivation. In addition, the use of Office tools remain popular for data handling, even though they might appear to be poorly suited for it, because users are very familiar with them.

In our tests, we have shown that, even though immature, Wordified ontologies can stand alongside or as a partial replacement for tools such as Protégé. With further development and close attention to user requirements, we believe that they could provide substantial benefits when properly integrated into the Ontology Engineering lifecycle.

Acknowledgement

We would like to express our thanks to Newcastle university for supporting this research. Also, many thanks to the University of Jeddah, Saudi Arabia for funding the scholarship.

[6]The property range in Wordified ontology and the "StuffedCrustBase" class in Protégé. If you found them, WELL DONE!

Address for correspondence

Aisha Blfgeh
a.blfgeh1@newcastle.ac.uk
abelfaqeeh@kau.edu.sa

Phillip Lord
phillip.lord@newcastle.ac.uk

References

[1] Aisha Blfgeh, Jennifer D. Warrender, Catharien M. U. Hilkens, and Phillip Lord. A document-centric approach for developing the tolapc ontology. In Frank Loebe, Martin Boeker, Heinrich Herre, Ludger Jansen, and Daniel Schober, editors, *Proceedings of the 7th Workshop on Ontologies and Data in Life Sciences, ODLS 2016, organized by the GI Workgroup Ontologies in Biomedicine and Life Sciences (OBML), Halle (Saale), Germany, September 29-30, 2016.*, volume 1692 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS.org, 2016. `http://ceur-ws.org/Vol-1692/paperB.pdf`.

[2] Aisha Blfgeh and Phillip Lord. User and developer interaction with editable and readable ontologies. In *Proceedings of the 8th International Conference on Biomedical Ontology (ICBO 2017), Newcastle-upon-Tyne, United Kingdom, September 13th - 15th, 2017.*, 2017. URL `http://ceur-ws.org/Vol-2137/paper_28.pdf`.

[3] A.L. Rector, P.E. Zanstra, W.D. Solomon, J.E. Rogers, R. Baud, W. Ceusters, W. Claassen, J. Kirby, J.-M. Rodrigues, A. Rossi Mori, E.J. Van der Haring, and J. Wagner. Reconciling users' needs and formal requirements: issues in developing a reusable ontology for medicine. *IEEE Transactions on Information Technology in Biomedicine*, 2(4):229–242, 1998. ISSN 10897771. doi: 10.1109/4233.737578. URL `http://ieeexplore.ieee.org/document/737578/`.

[4] Mikel Egaña, Robert Stevens, and Erick Antezana. Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language. *Proceedigns of OWLED*, 2009. doi: 10.1038/npre.2009.4006.1.

[5] David Osumi-Sutherland, Mlanie Courtot, James Balhoff, and Christopher Mungall. Dead simple owl design patterns. *Journal of Biomedical Semantics*, 8, 06 2017. doi: 10.1186/s13326-017-0126-0.

[6] Simon Jupp, Matthew Horridge, Luigi Iannone, Julie Klein, Stuart Owen, Joost Schanstra, Katy Wolstencroft, and Robert Stevens. Populous: a tool for building owl ontologies from templates. *BMC Bioinformatics*, 13(Suppl 1): S5, 2011. doi: 10.1186/1471-2105-13-S1-S5. URL `http://dx.doi.org/10.1186/1471-2105-13-S1-S5`.

[7] C Maria Keet. The use of foundational ontologies in ontology development: an empirical assessment. In *Extended Semantic Web Conference*, pages 321–335. Springer, 2011.

[8] Markel Vigo, Caroline Jay, and Robert Stevens. Constructing conceptual knowledge artefacts: activity patterns in the ontology authoring process. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3385–3394. ACM, 2015.

[9] He Tan, Anders Adlemo, Vladimir Tarasov, and Mats E Johansson. Evaluation of an application ontology. In *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology Bozen-Bolzano, Italy, September 21–23, 2017*, volume 2050. CEUR-WS, 2017.

[10] CO-ODE OWL Plugins. Owldoc. `https://github.com/co-ode-owl-plugins/owldoc`, 2016.

[11] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang. The manchester owl syntax. 01 2006.

[12] Phillip Lord and Jennifer D. Warrender. A highly literate approach to ontology building. abs/1512.04250, 2015. `http://arxiv.org/abs/1512.04250`.

[13] J. D. Warrender and P. Lord. The karyotype ontology: a computational representation for human cytogenetic patterns. *Bio-Ontologies*, 2013.

[14] Robert Stevens, James Malone, Sandra Williams, Richard Power, and Allan Third. Automating generation of textual class definitions from owl to english. *Journal of Biomedical Semantics*, 2(Suppl 2):S5, 2011.

[15] Bdd: Learn about behavior driven development, Dec 2018. URL `https://www.agilealliance.org/glossary/bdd/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'bdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)`.