RoundTrip Ontology Authoring

Brian Davis[1], Ahmad Ali Iqbal[1,3], Adam Funk[2], Valentin Tablan[2], Kalina Bontcheva[2], Hamish Cunningham[2], and Siegfried Handschuh[1]

[1] Digital Enterprise Research Institute, Galway, Ireland
[2] University of Sheffield, UK
[3] University of New South Wales, Australia

Abstract. Controlled Language (CL) for Ontology Editing tools offer an attractive alternative for naive users wishing to create ontologies, but they are still required to spend time learning the correct syntactic structures and vocabulary in order to use the Controlled Language properly. This paper extends previous work (CLOnE) which uses standard NLP tools to process the language and manipulate an ontology. Here we also generate text in the CL from an existing ontology using template-based (or shallow) Natural Language Generation (NLG). The text generator and the CLOnE authoring process combine to form a RoundTrip Ontology Authoring environment: one can start with an existing imported ontology or one originally produced using CLOnE, (re)produce the Controlled Language, modify or edit the text as required and then turn the text back into the ontology in the CLOnE environment. Building on previous methodology we undertook an evaluation, comparing the Round Trip Ontology Authoring process with a well-known ontology editor; where previous work required a CL reference manual with several examples in order to use the controlled language, the use of NLG reduces this learning curve for users and improves on existing results for basic ontology editing tasks.

1 Introduction

Formal data representation can be a significant deterrent for non-expert users or small organisations seeking to create ontologies and subsequently benefit from adopting semantic technologies. Existing ontology authoring tools such as Protégé[4] attempt to resolve this. but they often require specialist skills in ontology engineering. This is even more exasperating for domain specialists, such as clinicians, business analysts, legal experts, etc. Such professionals cannot be expected to train themselves to comprehend Semantic Web formalisms and the process of knowledge gathering; involving both a domain expert and an ontology engineer can be time-consuming and costly. Controlled languages for Knowledge creation and management offer an attractive alternative for naive users wishing to develop small to medium sized ontologies or a first draft ontology which can subsequently post-edited by the Ontology Engineer. In previous work[1], we presented CLOnE - *Controlled Language for Ontology Editing* which allows naive

[4] `http://protege.stanford.edu`

users to design, create, and manage information spaces without knowledge of complicated standards (such as XML, RDF and OWL) or ontology engineering tools. CLOnE's components are based on GATE's existing tools for IE (information extraction) and NLP (natural language processing).[2]

The CLOnE system was evaluated using a *repeated-measures, task-based* methodology in comparison with a standard ontology editor (Protégé). CLOnE performed favourably with test users in comparison to Protégé. Despite the benefits of applying Controlled Language Technology to Ontology Engineering, a frequent criticism against its adoption is the learning curve associated with following the correct syntactic structures and/or terminology in order to use the Controlled Language properly. Adhering to a controlled language can be, for some naive users, time consuming and annoying. Where the CLOnE system uses natural language analysis to unambiguously parse CLOnE in order to create and populate an ontology. The reverse of this process, (NLG) Natural Language Generation, involves the generation of the CLOnE language from an existing ontology.

The text generator and CLOnE authoring processes combine to form a RoundTrip Ontology Authoring(ROA) environment: a user can start with an existing imported ontology or one originally produced using CLOnE, (re)produce the Controlled Language using the text generator, modify or edit the text as required and subsequently parse the text back into the ontology using the CLOnE environment. The process can be repeated as necessary until the required result is obtained. Building on previous methodology [1] we undertook a repeated-measures, task-based evaluation, comparing the Round Trip Ontology Authoring process with (Protégé); Where previous work required a reference guide in order to use the controlled language, the substitution of NLG can reduce this learning curve for users, while simultaneously improving upon existing results for basic Ontology editing tasks. The remainder of this paper is organized as follows: Section 2 discusses related work Section 3 discusses the design and implementation of the ROA pipeline focusing on the NLG component - the ROA text generator. Section 4 presents our evaluation and discusses our quantitative findings. Finally, Section 5 and Section 6 offer conclusions and future work.

2 Related work

"Controlled Natural Languages (CL)s are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity."[3] CLs were later developed specifically for computational treatment. CLs have subsequently evolved into many variations and flavours such as Smart's Plain English Program (PEP) [4], White's International Language for Serving and Maintenance (ILSAM) [4] and Simplified English.[5] They have also found favour in large multi-national corporations, usually within the context of machine translation and machine-aided translation of user documentation. [3, 4]

[5] `http://www.simplifiedenglish-aecma.org/Simplified_English.htm`

The application of CLs for ontology authoring and instance population is an active research area. *Attempto Controlled English*[6] (ACE) [5], is a popular CL for ontology authoring. It is a subset of standard English designed for knowledge representation and technical specifications, and constrained to be unambiguously machine-readable into discourse representation structures, a form of first-order logic. (It can also be re-targeted to other formal languages.). [6] The Attempto Parsing Engine (APE) consists principally of a definite clause grammar, augmented with features and inheritance and written in Prolog. [7] ACE OWL, a sublanguage of ACE, proposes a means of writing formal, simultaneously human- and machine-readable summaries of scientific papers. [8, 9] Similar to RoundTrip Ontology Authoring, ACE OWL also aims to provide reversibility (translating OWL DL into ACE) The application NLG, for the purposes editing existing ACE text, is mentioned in [10]. The paper discusses the implementation of the shallow NLG system, or OWL Verbalizer, focusing primarily on the OWL to ACE rewrite rules, however no evaluation is provided, nor any quantitative results in attempt to quantify the impact of NLG in the authoring process. Furthermore OWL's *allValuesFrom* must be translated into a construction which can be rather difficult for humans to read. A partial implementation is available for public testing[7].

Another well-known implementation which employs the use of NLG to aid the Knowledge creation process is WYSIWYM (*What you see is what you meant*). It involves direct knowledge editing with natural language directed feedback. A domain expert can edit a knowledge based reliably by interacting with natural language menu choices and subsequently generated feedback which can then be extended or re-edited using the menu options. WYSIWYM was used initially in the context of the DRAFTER project and the multilingual NLG system included in DRAFTER was re-engineered for DRAFTER II. [11] DRAFTER II and WYSIWYM will most likely be deployed in CLEF[8] project. The work is conceptually similar to RoundTrip Ontology Engineering however the natural language generation occurs as a feedback to guide the user during the editing process as opposed to providing an initial summary in Controlled Language for editing. A usability evaluation is provided in [12] in the context of knowledge creation, some of which is based on IBM heuristic evaluations[9] whereby a substantial amount of user feedback is collected, yet no specific quantitative data that we are aware of, is presented. However a significant amount of empirical evaluation was undertaken as part of evaluation for the MILE (Maritime Information and Legal Explanation) application which used WYSIWYM in the context of query formulation for the CLIME[10] project of with the outcome was favourable. [12]

---

[6] `http://www.ifi.unizh.ch/attempto/`

[7] `http://attempto.ifi.uzh.ch/site/tools/`

[8] `http://www.clinical-escience.org/`

[9] `http://www-03.ibm.com/able/resources/uebeforeyoubegin.html`

[10] CLIME, Cooperative Legal Information Management and Explanation, Esprit Project EP25414

Similar to WYSIWYM is *GINO* (Guided Input Natural Language Ontology Editor) provides a guided, controlled NLI (natural language interface) for domain-independent ontology editing for the Semantic Web. GINO incrementally parses the input not only to warn the user as soon as possible about errors but also to offer the user (through the GUI) suggested completions of words and sentences—similarly to the"code assist" feature of Eclipse[11] and other development environments. GINO translates the completed sentence into triples (for altering the ontology) or SPARQL[12] queries and passes them to the Jena Semantic Web framework. Although the guided interface facilitates input, the sentences are quite verbose and do not allow for aggregation. Static grammar rules exist for the controlled language but in addition, dynamic grammar rules are generated from the Ontology itself, however this does not constitute surface realization in the context of natural language generation, but the amendment of additional parsing rules to GINO's grammar to guide the user. This permits the system to handle a domain shift, however this is heavily dependent on any linguistic data or RDF label data encoded the ontology. A full textual description of the Ontology is not realized as is the case of the CLOnE text generator. [13] Furthermore, similar, to our evaluation, a small usability evaluation was conducted using SUS [14], however the sample set of six was too small to infer any statistically significant results [15]. In addition, GINO was not compared to any existing Ontology editor during the evaluation. Finally, [16] present an Ontology based Controlled Natural Language Editor, similar to GINO, which uses CFG (Context-free grammar) and lexical dependencies - CFG-DL to generate RDF triples. To our knowledge the system ports only to RDF and does not cater for other Ontology languages. Furthermore no quantitative user evaluation is provided.

Other related work involves the application of Controlled Languages for Ontology/Knowledge base querying, which represent a different task than that of knowledge creation and editing but are worth mentioning for completeness sake. Most notably *AquaLog*[13] is an ontology-driven,portable question-answering (QA) system designed to provide a natural language query interface to semantic mark-up stored in knowledge base. PowerAqua [17] extends AquaLog, allowing for an open domain question-answering for the semantic web. The system dynamically locates and combines information from multiple domains.

3 Design and Implementation

In this section, we describe the overall architecture of the Round Trip Ontology Authoring (ROA) Pipeline which is implemented in GATE [2]. We discuss briefly extensions to existing CLOnE components of ROA, but focus the attention of this section towards describing the CLOnE text generator, the algorithm used

[11] `http://www.eclipse.org/`
[12] `http://www.w3.org/TR/rdf-sparql-query/`
[13] `http://kmi.open.ac.uk/technologies/aqualog/`

and the XML containing templates needed to configure the controlled language output of the generator.

3.1 RoundTrip Ontology Authoring (ROA) and CLOnE

ROA builds on and extends the existing advantages of the CLOnE software and input language.

1. ROA requires only one interpreter or runtime environment, the Java 1.6 JRE.
2. ROA like CLOnE uses a sub-language of English.
3. As far as possible, CLOnE is grammatically lax; in particular it does not matter whether the input is singular or plural (or even in grammatical agreement).
4. ROA can be compact; the user can create any number of classes or instances in one sentence.
5. ROA is more flexible and easier to learn by using simple examples of how to edit the controlled language generated by the text generator in order to modify the Ontology. It reduces the need to learn the Controlled Language by following examples, guiding rules or CLOnE syntactic rules. Instead, a user can create or modify various classes and instances in one (generated) sentence or (using simple copy and paste) create new properties between new or existing classes and instances.
6. The CLOnE grammar with ROA has been extended to handle simple verbs and phrasal verbs.
7. Like CLOnE any valid sentence of ROA can be unambiguously parsed.
8. The advantage of the GATE Ontology API allows users to import existing Ontologies for generation, subsequent editing in ROA and export to different Ontology formats.
9. SimpleNLG[14] has been added into the ROA text generator to lexicalize unseen properties.

Procedurally, CLOnE's analysis consists of the ROA pipeline of processing resources (PRs) shown in Figure 1 (left dotted box). This pipeline starts with a series of fairly standard GATE NLP tools which add linguistic annotations and annotation features to the document. These are followed by three PRs developed particularly for CLOnE: the gazetteer of keywords and phrases fixed in the controlled language and two JAPE[15] transducers which identify quoted and unquoted chunks. (Names enclosed in pairs of single or double quotation marks can include reserved words, punctuation, prepositions and determiners, which are excluded from unquoted chunks in order to keep the syntax unambiguous.) The last stage of analysis, the CLOnE JAPE transducer, refers to the existing

[14] `http://www.csd.abdn.ac.uk/~ereiter/simplenlg/`

[15] GATE provides the *JAPE* (Java Annotation Pattern Engine) language for matching regular expressions over annotations, adding additional annotations to matched spans, and manipulating the match patterns with Java code.

ontology in several ways in order to interpret the input sentences. The following table provides an excerpt the grammar rules of the CLOnE language. We refer the reader to [1, 18] for additional rules and examples.

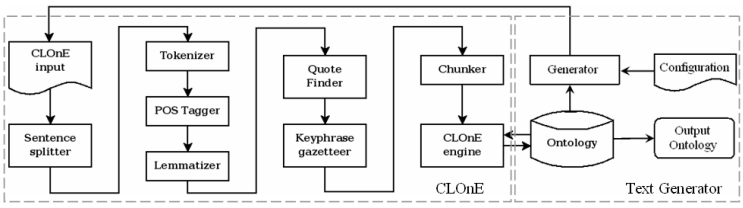

Fig. 1. The ROA RoundTrip Ontology Authoring pipeline

Sentence Pattern	Example	Usage
Forget everything.	`Forget everything.`	Clear the whole ontology corpus to start with the new ontology.
(Forget that) There is/are <CLASSES>.	`There are researchers, universities and conferences.`	Create or delete (new) classes.
(Forget that) <INSTANCES> is a/are <CLASS>.	`Ahmad Ali Iqbal and Brian Davis are 'Ph.D. Scholar'.`	Create (or delete) instances of the class.
(Forget that) <SUBCLASSES> is/are a type/types of <SUPERCLASS>.	`'Ph.D. Scholar' is a type of Student.`	Make subclass(es) of an existing super-class. 'Forget that' only unlinks the the subclass-superclass relationship.
(Forget that) <CLASSES/INSTANCES> <VERB PROPERTY> <CLASSES/INSTANCES>.	`Professor supervises student.`	Create the property of the form *Domain_verb_Range* either between two classes or instances.

3.2 Text generation of CLOnE

The text generation component in Figure 1 (right dotted box) displayed in the ROA pipeline is essentially an Ontology Verbalizer. Unlike some NLG systems, the communicative goal of the text generator is not to construct tailored reports for specific content with the Knowledge base or to respond user specific queries. Hence no specific content selection subtask or "choice" is performed since our

goal is to describe and present the Ontology in textual form as unambiguous subset of English - the CLOnE language for reading, editing and amendment. We select the following content from the Ontology: top level classes, subclasses, instances, class properties and their respective domain and ranges, and instance properties. The text generator is configured using an XML file, whereby text templates are instantiated and filled by the values from the Ontology. This file is decoupled from the text generator PR. Two templates which are used to generate top level classes and class properties are displayed in Figure 2.

Stage 1 carried out by the text generator converts the input ontology into an internal GATE ontological resource and flattens it into RDF style triples. This is done breadth-first—so lists are created where super-classes always precede their corresponding subclasses—in the following order: top-level classes, subclasses, instances, class properties, and instance properties.

Stage 2 matches generation templates from the configuration file (See Figure 2) with the triples list derived from the Ontology in Stage 1. A generation template has three components: (1) an `in` element containing a list of triple specifications, (2)an `out` element containing phrases that are generated when a successful match has occurred and an (3) optional `ignoreiIf` element for additional triple specifications that cause a match specified in the `in` element to be ignored if the conditions are satisfied. The triple specifications contained within the `in` portion of the template can have a subject, property, object XML elements. The triple specifications act as restrictions or conditions, such that an input triple generated from the Ontology must match this template. If more than one triple is included in the `in` element they are considered as a conjunction of restrictions, hence the template will only match if one or more actual triples for all triple specifications within the `in` element are found. One triple can reference another, i.e., a specification can constrain a second triple to have the same object as the subject of the first triple. Only backward referencing is permitted since the triples are matched in a top down fashion according to their textual ordering. An example of referencing can be seen in line 188 of the `out` element of the template shown in Figure 2 for generating class properties.

The `out` section of the template describes how text is generated (Stage 3) from a successful match. It contains phrase templates that have text elements and references to values matched within the `in` elements. Phrases are divided into singular and plural forms. Plural variants are executed when several triples are grouped together to generate a single sentence (Sentence Aggregation) based on a list of Ontology objects (i.e., `There are Conferences, Students and Universities`). Text elements within a template are simply copied into the output while reference values are replaced with actual values based matching triple specifications. We also added a small degree of lexicalization into the Text Generator PR, whereby an unseen property, which is treated as a verb is inflected correctly for output `study` and `studies`. This involves a small of amount of dictionary look-up using the SimpleNLG Library to obtain the third person singular inflection `studies` from `study` to produce `Brian Davis studies at NUIG`. The `out` elements of the generation template also provide several phrase

templates for the singular and plural sections. These are applied in rotation to prevent tedious and repetitious output.

Stage 2 also groups matches together into sets that can be expressed together in a plural form. For this to proceed, the required condition is that the only difference between the matches in set, occurs in only one of the references used in the phrase templates, i.e., if singular variants would only differ by one value. A specialized generation template with no `in` restrictions is also included in the configuration file. This allows the production of text where there are no specific input triple dependencies.

```
<!-- Template for all the other top classes -->
<template>
    <in>
        <triple id="t1">
            <property ns="rdf" name="type"/>
            <object ns="owl" name="Class"/>
        </triple>
        <triple id="t1.subject">
            <subject ref="t1.subject"/>
            <property ns="rdfs" name="subClassOf"/>
            <object name="Entity"/>
        </triple>
    </in>
    <out>
        <singular>
            <phrase>There are
            <ref ref="t1.subject" number="plural"/>.
            </phrase>
        </singular>
        <plural>
            <phrase>There are
            <ref ref="t1.subject" number="plural"/>.
            </phrase>
        </plural>
    </out>
    <ignoreIf>
    </ignoreIf>
</template>
```

```
<!-- Template for defining class properties -->
<template>
    <in>
        <triple id="t1">
            <property ns="rdf" name="type"/>
            <object ns="rdf" name="Property"/>
        </triple>

        <triple id="t2">
            <subject ref="t1.subject"/>
            <property ns="rdfs" name="domain"/>
        </triple>

        <triple id="t3">
            <subject ref="t1.subject"/>
            <property ns="rdfs" name="range"/>
        </triple>
    </in>
    <out>
        <singular>
            <phrase>
                <ref ref="t2.object" number="singular"/>
                <ref ref="t1.subject" number="singular"/>
                <ref ref="t3.object" number="singular"/>.
            </phrase>
        </singular>
        <plural>
            <phrase>
                <ref ref="t2.object" number="singular"/>
                <ref ref="t1.subject" number="singular"/>
                <ref ref="t3.object" number="plural"/>.
            </phrase>
        </plural>
    </out>
</template>
```

Fig. 2. Example of generation template

4 Evaluation

4.1 Methodology

Our methodology is deliberately based on the criteria previously used to evaluate CLOnE [1, 18], so that we can fairly compare the earlier results using the CLOnE software with the newer Round Trip Ontology Authoring(ROA) process. The methodology involves a *repeated-measures, task-based* evaluation: each subject carries out a similar list of tasks on both tools being compared. Unlike our previous experiment, the CLOnE reference guide list and examples are withheld from the test users, so that we can measure the benefits of substituting the

text generator for the reference guide and determine its impact on the learning process and usability of CLOnE. Furthermore, we used a larger sample size and more controls for bias. All evaluation material and data are available online for inspection, including the CLOnE evaluation results for comparison[16]. The evaluation contained the following.

- A pre-test questionnaire asking each subject to test their degree of knowledge with respect to ontologies, the Semantic Web, Protégé and controlled languages. It was scored by assigning each answer a value from 0 to 2 and scaling the total to obtain a score of 0–100.
- A short document introducing ontologies, the same 'quick start' Protégé instructions as used in [18] (partly inspired by Protégé's *Ontology 101* documentation [19]), and an example of editing CLOnE text derived from the text generator. The CLOnE reference guide and detailed grammar examples used in for the previous experiment [18] were withheld. Subjects were allowed to refer to an example of how to edit generated Controlled Language but did *not* have access to CLOnE reference guide.
- A post-test questionnaire for each tool is the *System Usability Scale* (SUS), which also produces a score of 0–100 to compare with previous results. [14]
- A comparative questionnaire similar to the one used in [18] was applied to measure each user's preference for one of the two tools. It is scored similarly to SUS so that 0 would indicate a total preference for Protégé, 100 would indicate a total preference for RoundTrip, and 50 would result from marking all the questions *neutral*. Subjects were also given the opportunity to make comments and suggestions.
- Two equivalent lists of ontology-editing tasks, each consisting of the following subtasks:
  - creating two subclasses of existing classes,
  - creating two instances of different classes, and
  - either (A) creating a property between two classes and defining a property between two instances, or (B) extending properties between two pairs of instances.

  For both task lists, an initial ontology was created using CLOnE. The same ontology was loaded into Protégé for both tasks and the text generator was executed to provide a textual representation of the ontology for editing purposes(see Figure 3), again for both tasks.

For example, Task List A is as follows.

- Create a subclass *Institute* of *University*.
- Create a subclass *Workshop* of *Conference*.
- Create an instance *International Semantic Web Conference* of class *Conference*.
- Create an instance *DERI* of class *Institute*.
- Create a property that *Senior Researchers supervise Student*.
- Define a property that *Siegfried Handschuh supervises Brian Davis*.

[16] `http://smile.deri.ie/drupal/node/98`

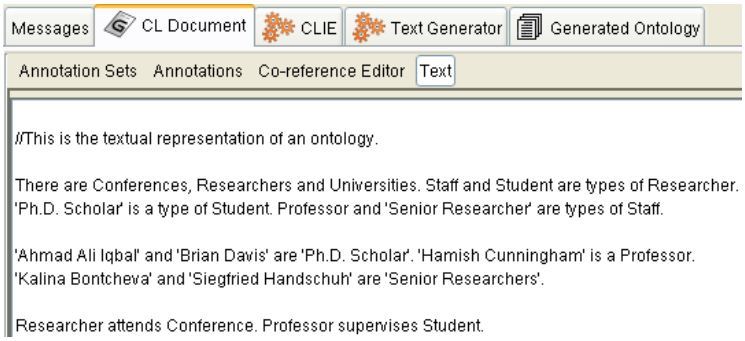

Fig. 3. Text Generated by ROA

4.2 Sample quality

We recruited 20 volunteers from the Digital Enterprise Research Institute, Galway[17]. The sample size ($n = 20$) satisfies the requirements for reliable SUS evaluations. [15] We recruited subjects with an industrial background (**I**) and participants with a research background (**R**) See (in Table 5) for details. In addition we attempted to control bias by selecting volunteers who were either:

- Research Assistants/Programmers/Post-Doctoral Researchers with an industrial background either returning (or new) to Academic Research respectively(**I**);
- Postgraduate Students who were new to the Semantic Web and unfamiliar with Ontology Engineering(**R**);
- Researchers from the E-learning and Sensor Networks lab but not from the Semantic Web Cluster(**R**);
- Researchers with no background in Natural Language Processing or Ontology Engineering(**R**); or
- Industrial Collaborators (**I**).

In all cases, we tried to ensure that participants had limited or no knowledge of GATE or Protégé. First, subjects were asked to complete the pre-test questionnaire, then they were permitted time to read the Protégé manual and Text Generator examples, and lastly they were asked to carry out each of the two task lists with one of the two tools. (Half the users carried out task list A with ROA and then task list B with Protégé; the others carried out A with Protégé and then B with ROA.) Each user's time for each task list was recorded. After each task list the user completed the SUS questionnaire for the specific tool used, and finally the comparative questionnaire. Comments and feedback were also recorded on the questionnaire forms.

[17] `http:www.deri.ie`

4.3 Quantitative findings

Table 2 summarizes the main measures obtained from our evaluation. We used SPSS[18] to generate all our statistical results. In particular the mean ROA SUS score are above the baseline of 65–70% while the mean SUS score for Protégé is well below the baseline. [20]. In the ROA/Protégé Preference (R/P Preference) scores, based on the comparative questionnaires, we note that the scores also favour on average ROA over Protégé. Confidence intervals are displayed in Table 3.[19]

Table 2. Summary of the questionnaire scores

Measure	min	mean	median	max
Pre-test scores	17	42	42	75
ROA SUS rating	48	74	70	100
Protégé SUS rating	10	41	41	85
R/P Preference	40	72	79	95

Table 3. Confidence intervals (95%) for the SUS scores

Tool	Confidence intervals		
	Task list A	Task list B	Combined
Protégé	28–55	29–51	32–49
ROA	63–77	69–84	68–79

We also generated Pearson's and Spearman's correlations coefficients.[21, 22] Table 4 displays the coefficients. In particular, we note the following results.

- The pre-test score has a weak negative correlations the with ROA task time.
- There are no correlations with pre-test score and the ROA SUS score.
- The pre-test score has a weak negative correlation with the Protégé SUS score.
- There are no correlations with pre-test score and the Protégé time.
- In previous results in comparing CLOnE and Protégé, the task times for both tools were more positively correlated with each other while in the case of ROA and Protégé, there correlation has being weakened by a significant

[18] SPSS 2.0, `http://www.spss.com`

[19] A data sample's *95% confidence interval* is a range 95% likely to contain the mean of the whole population that the sample represents. [21]

Table 4. Correlation coefficients

Measure	Measure	Pearson's	Spearman's	Correlation
Pre-test	ROA time	-0.41	-0.21	weak −
Pre-test	Protégé time	-0.28	-0.35	none
Pre-test	ROA SUS	-0.02	-0.00	none
Pre-test	Protégé SUS	-0.32	-0.29	weak −
ROA time	Protégé time	0.53	0.58	+
ROA time	ROA SUS	-0.65	-0.52	−
Protégé time	Protégé SUS	0.53	0.56	+
ROA time	Protégé SUS	-0.14	-0.10	none
Protégé time	ROA SUS	-0.02	-0.09	none
ROA SUS	Protégé SUS	0.04	-0.01	none
ROA SUS	R/P Preference	0.58	0.56	+
Protégé SUS	R/P Preference	-0.01	0.10	none

32% of its original value (of 78%) reported for CLOnE[1], indicating that the users tended not spend the equivalent time completing both ROA and Protégé tasks.

- There is a moderate correlation with Protégé task time and Protégé SUS scores.
- There strong negative correlation of -0.65 between the ROA task time and the ROA SUS scores. Our previous work reported no correlation between the CLOnE task time and CLOnE SUS time. A strong negative or inverse correlation implies that users who spent less time completing a task using ROA tended to produce high usability scores - favouring ROA. More importantly we noted associated probability reported by SPSS, was less then the typical 5% cut-off point used in social sciences. This implies there is a 5% chance that the true population coefficient is very unlikely to be 0 (no relationship). Conversely, one can infer statistically that for 19 out of 20 (95%)users with little or no experience in either NLP or Protégé who favour RoundTrip Ontology Authoring over Protégé also tend to spend less time completing Ontology editing tasks.
- The R/P Preference score correlates moderately with ROA SUS score similar to previous results but has no longer has a significant inverse correlation with Protégé SUS scores. The reader should note the R/P Preference scores favour ROA over Protégé.

We also varied the tool order evenly among our sample. As noted previously in [1], once again the SUS scores have differed slightly according to tool order (as indicated in Table 3). Previous SUS scores for Protégé tended to be slightly lower for B than for A, which we believe may have resulted from the subjects' decrease in interest as the evaluation progressed, While in previous results there was a decrease in SUS scores for CLOnE (yet still well above the SUS baseline), the SUS scores however, increase for ROA task B by Table 3 implying that if waning

interest was a factor in decreased SUS scores for CLOnE, it does not appear to be the case for ROA. What is of additional interest is that group **I**, subjects with industrial background scored on average 10% higher for both ROA SUS and ROA/Protégé which implies that Industrial collaborators or professionals with an Industrial background favoured a natural language interface over a standard Ontology Editor even more than Researchers.

Table 5. Groups of subjects by source and tool order

Source	Tool order		Total
	PR	RP	
R Researcher	5	7	12
I Industry	5	3	8
Total	10	10	20

Table 6. Comparison of the two sources of subjects

Measure	Group	min	mean	median	max
Pre-test	R	17	38	38	58
	I	17	47	50	75
ROA SUS	R	48	69	70	82
	I	65	80	80	100
Protégé SUS	R	10	30	28	52
	I	12	48	49	85
R/P Preference	R	40	68	72	88
	I	65	78	78	95

4.4 User Feedback

The test users also provided several suggestions/comments about ROA.

- "RoundTrip Ontology Authoring becomes much easier, once the rules are learnt". (This is very interesting considering that no syntax rules, extended examples or restricted vocabulary list were provided).
- Use of inverted commas should be used only once and afterwards, if same class /instance is reused, the system should automatically recognise it as the previous word.
- Many users suggested displaying the ontology pane on the right hand side of the text pane, where test users edit the text instead of moving between two separate panes.

– Some users suggested dynamic ontology generation, once a user finishes typing a sentence, in order to display the changes.
– Similar suggestions to the previous evaluation were provided for user autocompletion, syntax highlighting, options about available classes, instances or property names and keywords should be displayed, a similar concept to modern Word Processor or programming IDEs such as eclipse.
– Some test users with an industrial background demonstrated concern regarding scalability and ROA using with a larger business related ontology and suggest capabilities for verbalizing a portion of the ontology tree within the Ontology viewer, using text generation for subsequent editing.
– Some test users appreciated the singular/plural forms and sentence handling of ROA (e.g., study, studies).

5 Conclusion

Our user evaluation consistently indicated that our subjects found ROA(and continue to find CLOnE) significantly more usable and preferable than Protégé for simple Ontology editing tasks. In addition this evaluation differs, in that we implemented more tighter restrictions during our selection process, to ensure that users had no background in NLP or Ontology engineering. Furthermore, 40% of our subjects with an industrial background, tended to score ROA 10% higher then Researchers indicating that a Natural Language Interface to a Ontology Editor might be a preferred option for Ontology development within industry. The main research goal of this paper was to assess the effect of introducing NLG into the CLOnE Ontology authoring process to facilitate RoundTrip Ontology Authoring.

This evaluation differs from previous work [1] by two important factors: we *excluded* the CLOnE reference manual from the training material provided in the previous evaluation; and we introduced a Text Generator, verbalizing CLOnE text from a given populated Ontology and asked users to edit the Ontology, using the generated CLOnE text based on an example provided. We observed two new significant improvements in our results: the previous evaluation indicated a strong correlation between CLOnE task times and Protégé task times, this correlation has significantly weaken by 32% between ROA and Protégé task times. Hence, where users previously required the equivalent time to implement tasks both in CLOnE and Protégé, this is no longer the case with ROA (the difference being the text generator); and our previous evaluation indicated no correlation between either CLOnE/Protégé task times and their respective SUS scores. However, with ROA, we can now infer that 95% of the total population of naive users, who would favour RoundTrip Ontology Authoring over Protégé would also tend to spend less time completing Ontology editing tasks. We suspect that this is due to the reduced learning curve caused by the text generator. Furthermore, ROA tended to retain user interest, which CLOnE did not. We suspect that the absence of the need to refer to the CL reference guide was a factor in this. While Protégé is intended for more sophisticated knowledge

engineering work, this is not the case for ROA. Scalability was also an issue raised by our test subjects. It is possible that, authoring memory frequently used in translation memory systems or text generation from selective portions of the Ontology (using a Visual Resource) could help resolve this.

6 Continuing and future work

Several interesting and useful suggestions for improvements to ROA were made, many of which were already under development within the Nepomuk[20] (The Social Semantic Desktop) project. ROA has been ported to a Nepomuk-KDE [21] application, Semn[22] for Semantic Notetaking and will be targeted towards the task of semi-automatic semantic annotation. The ROA text generator was recently used in KnowledgeWeb[23] for the verbalization suggestions for semi-automatic ontology integration. Finally, ROA is being applied within the EPSRC-funded Easy project to create a controlled natural language interface for editing IT authorization policies (access to network resources such as directories and printers) stored as ontologies.

Acknowledgements

This research has been partially supported by the following grants: KnowledgeWeb (EU Network of Excellence IST-2004-507482), TAO (EU FP6 project IST-2004-026460), SEKT (EU FP6 project IST-IP-2003-506826, Líon (Science Foundation Ireland project SFI/02/CE1/1131) and NEPOMUK (EU project FP6-027705).

References

1. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: Clone: Controlled language for ontology editing. In: ISWC/ASWC. (2007) 142–155
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). (2002)
3. Schwitter, R.: Controlled natural languages. Technical report, Centre for Language Technology, Macquarie University (June 2007)
4. Adriaens, G., Schreurs, D.: From COGRAM to ALCOGRAM: Toward a controlled English grammar checker. In: Conference on Computational Linguistics (COLING'92), Nantes, France 595–601

[20] `http://nepomuk.semanticdesktop.org/xwiki/`
[21] `http://nepomuk-kde.semanticdesktop.org/xwiki/bin/view/Main/WebHome`
[22] `http://smile.deri.ie/projects/semn/`
[23] `http://knowledgeweb.semanticweb.org/`

5. Fuchs, N., Schwitter, R.: Attempto Controlled English (ACE). In: CLAW96: Proceedings of the First International Workshop on Controlled Language Applications, Leuven, Belgium (1996)
6. Fuchs, N.E., Kaljurand, K., Kuhn, T., Schneider, G., Royer, L., Schröder, M.: Attempto Controlled English and the semantic web. Deliverable I2D7, REWERSE Project (April 2006)
7. Hoefler, S.: The syntax of Attempto Controlled English: An abstract grammar for ACE 4.0. Technical Report ifi-2004.03, Department of Informatics, University of Zurich (2004)
8. Kaljurand, K., Fuchs, N.E.: Bidirectional mapping between OWL DL and Attempto Controlled English. In: Fourth Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro (June 2006)
9. Kuhn, T.: Attempto Controlled English as ontology language. In Bry, F., Schwertel, U., eds.: REWERSE Annual Meeting 2006. (March 2006)
10. Kaljurand, K., Fuchs, N.: Verbalizing OWL in Attempto Controlled English. In: Proceedings of OWL: Experiences and Directions (OWLED 2007). (2007)
11. Power, R., Scott, D., Evans, R.: What you see is what you meant: direct knowledge editings with natural language feedback. In Prade, H., ed.: 13th European Conference on Artificial Intelligence (ECAI'98). John Wiley and Sons, Chichester, England (1998) 677–681
12. Piwek, P.: Requirements definition, validation, verification and evaluation of the clime interface and language processing technology. Technical report, ITRI-University of Brighton (2002)
13. Bernstein, A., Kaufmann, E.: GINO—a guided input natural language ontology editor. In: 5th International Semantic Web Conference (ISWC2006). (2006)
14. Brooke, J.: SUS: a "quick and dirty" usability scale. In Jordan, P., Thomas, B., Weerdmeester, B., McClelland, A., eds.: Usability Evaluation in Industry. Taylor and Francis, London (1996)
15. Tullis, T.S., Stetson, J.N.: A comparison of questionnaires for assessing website usability. In: Usability Professionals' Association Conference, Minneapolis, Minnesota (June 2004)
16. Namgoong, H., Kim, H.: Ontology-based controlled natural language editor using cfg with lexical dependency. In: ISWC/ASWC. (2007) 353–366
17. Lopez, V., Motta, E., Uren, V.: Poweraqua: Fishing the semantic web. In: ESWC. (2006) 393–410
18. Funk, A., Davis, B., Tablan, V., Bontcheva, K., Cunningham, H.: Controlled language IE components version 2. Deliverable D2.2.2, SEKT (2006)
19. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory (March 2001)
20. Bailey, B.: Getting the complete picture with usability testing. Usability updates newsletter, U.S. Department of Health and Human Services (March 2006)
21. John L. Phillips, J.: How to Think about Statistics. W. H. Freeman and Company, New York (1996)
22. Connolly, T.G., Sluckin, W.: An Introduction to Statistics for the Social Sciences. Third edn. Macmillan (1971)