A Use Case for Controlled Languages as Interfaces to Semantic Web Applications

Pradeep Dantuluri, Brian Davis, Siegfried Handschuh

Digital Enterprise Research Institute
National University of Ireland Galway
pradeep.varma, brian.davis, siegfried.handschuh@deri.org

Abstract

Although the Semantic web is steadily gaining in popularity, it remains a mystery to a large percentage of Internet users. This can be attributed to the complexity of the technologies that form its core. Creating intuitive interfaces which completely abstract the technologies underneath, is one way to solve this problem. A contrasting approach is to ease the user into understanding the technologies. We propose a solution which anchors on using controlled languages as interfaces to semantic web applications. This paper describes one such approach for the domain of meeting minutes, status reports and other project specific documents. A controlled language is developed along with an ontology to handle semi-automatic knowledge extraction. The contributions of this paper include an ontology designed for the domain of meeting minutes and status reports, and a controlled language grammar tailored for the above domain to perform the semi-automatic knowledge acquisition and generate RDF triples. This paper also describes two grammar prototypes, which were developed and evaluated prior to the development of the final grammar, as well as the Link grammar, which was the grammar formalism of choice.

1. Introduction

The Semantic web[1] aims to simplify the process of building knowledge-based applications by enabling a web of interoperable and machine-readable data. This is done by formalizing the descriptions of the structure and semantics of the data available on the web. The linked data initiative[2] is a positive step in that direction, exposing huge amounts of data for further analysis and use by other applications. However creating and exposing linked data is a task that requires thorough knowledge of various technologies. A solution to this is to create technologies which would enable the average internet user to annotate and embedd data in his/her own textual resources. This paper aims to explore the possibility of using controlled natural languages (hereby referred to as CNL) as an interface to semantic web applications, specifically targeting the domain of project documents like meeting minutes, status reports, etc. The major goal was to enable novice users to author and annotate text documents using a controlled language. Furthermore, these documents can be parsed to extract the implicit knowledge contained, due to the enforcement of a fixed grammar and vocabulary. The authors have previously used this approach to build an annotation tool along with prototypes of the grammar and ontologies for the meeting minutes domain (Davis et al., 2009).

CLANN (Controlled Language for ANNotation)[3] builds on the experiences gained from the previous work, by incorporating redesigned versions of the grammar and the domain ontology. CLANN is designed to be an end-to-end semantic web application complete with a domain ontology, a persistent layer based on RDF[4] and a user interface for editing and authoring documents. The domain was expanded to include all the documents in a project specific setting (for example, meeting minutes, status reports, etc).

The main contributions of the paper include an designing an ontology for the domain of project documents and the design and implementation of the CLANN grammar and its previous prototypes.

2. Controlled languages as an interface

Rolf Schwitter[5] defines a CNL as ,

> *"Controlled Natural Languages are subsets of natural languages whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity."*

Traditionally, CNLs fall into two major categories: those that improve the readability for human readers, in particularly for non-native speakers, and those that improve the computational processing of a text. CNLs have already been applied to ontology authoring and population (Smart, 2008Unpublished). Previous work by the authors (Davis et al., 2009) focussed on applying CNLs to semantic annotation. The process of semantic annotation according to (Handschuh, 2005) involves addition or association of semantic data or meta-data to the content, according to an agreed-upon ontology.

3. The CLANN grammar

CLANN is designed to facilitate knowledge capture from every-day, repetitive, domain-specific texts. To better explore the applicability, two independent prototypes of the grammar were developed(Davis et al., 2009) focusing alternatively on usability and expressivity. This work has

[1] http://www.w3.org/2001/sw/

[2] http://linkeddata.org/

[3] In this document *CLANN* refers to the annotation platform as well as controlled grammar

[4] Resource Description Framework - http://www.w3.org/RDF/

[5] http://sites.google.com/site/controllednaturallanguage/

Table 1: Excerpt of CLANN I grammar with examples

Sentence Pattern	Example	Parsed pattern
<NP><VP><NP>(<PP>+)	Ambrosia to submit "her PhD Proposal" during "the next week".	(Ambrosia <NP>) (to submit <VP>) (her PhD Proposal <NP>) (during (the next week <NP>)<PP>).

Table 2: Excerpt of CLANN II grammar with examples

Sentence Pattern	Example
<text>[is a <Class>].	Dirk[is a Person]. Creates an object of the class Person with label *Dirk*. Note the the label is taken from the document content.
<text>[is a subclass of <Class>].	Proposal[is a subclass of Document] or [Proposal is a subclass of Document]. Creates a new class with label *Proposal* as a subclass of the class *Document*.
<text>[<property> <object>].	Dirk[toComplete "PhD Proposal"] or [Dirk toComplete "PhD Proposal"]. Creates a triple which links the instances of *Dirk* and *PhD Proposal* with the property *toComplete*.

eventually led to the CLANN grammar, which is essentially a merge between the former two grammars, incorporating most of the advantages, albeit a few changes.

CLANN-prototype1 is designed with a major focus on usability. Each sentence adhered to one of the syntactic rules and used a lenient vocabulary. This domain vocabulary was derived by corpus analysis using Word Smith tools [6] on the document corpus. This ensured that most of the sentences resembled normal English sentences. An example of such a syntactic construct is given in Table 1. CLANN-prototype1 is grammatically lax in comparision to typical CNL approaches to knowledge creation.

A modified shallow parser, built in GATE[7] using JAPE[8] rules, is then used to parse the text, extract the knowledge and instantiate the ontology.

CLANN-prototype2 was designed with a major focus on expressivity. It differs from the conventional notion of CNL, whereby the entire document is written in CNL, rather it allows the user to add snippets of CNL text, enclosed in "[ ]", to the document or associate them to a particular text in the document. These snippets should adhere to a *Subject-Verb-Object* syntax, where the subject is either specified in the snippet or taken from the free text. The vocabulary for CLANN-prototype2 also includes the vocabulary of the ontology, thereby allowing the user to represent any kind of relational meta-data. This approach was inspired by the CLOnE[9] Language (Funk et al., 2007). The syntactic constructs of CLANN-prototype2 along with examples are shown in Table 2.

The finer details of design and implementation of these grammars are described in (Davis et al., 2009). Both grammars use a common template, described above, which is initially parsed to extract the inherent meta data of the document (in this case, meeting minutes). A detailed comparision of the two approaches is shown in the Table 3.

The CLANN grammar is essentially a combination of the previous prototypes, incorporating the snippets of CLANN-prototype2 into controlled text of CLANN-prototype1. This is currently in the development. For other sample documents, demos and information please refer to the project home page[11]. The CLANN grammar makes use of three diffrent techniques to encode knowledge into the text. They are the *templates*, the *CL snippets* (controlled language snippets) and *controlled english*. Each of these are detailed below along with examples.

3.1. Templates

Conventionally, work on semantic annotation focused on two-step approaches where the authoring of a document has to precede the annotation of the same. This problem can be overcome by adopting a *latent annotation*(Davis et al., 2009) approach by the use of controlled languages, which merges both the authoring and annotation steps into one. The information is encoded in the restricted vocabulary and grammatical structure of the controlled language. However, preliminary evaluations suggest that annotating every piece of information using a CNL makes the task quite verbose, thereby demotivating the users. A simple solution to this is to supplement the CNL by using templates which encode implicit domain information, an example of which is shown below.

[6] http://www.lexically.net/wordsmith/version5/index.html

[7] General Architechture for Text Engineering - http://www.gate.ac.uk

[8] Java Annotation Patterns Engine - http://gate.ac.uk/sale/tao/splitch8.html#chap:jape

[9] CLOnE - Controlled Language for Ontology Editing

[11] http://smile.deri.ie/projects/clann

Table 3: Comparision of CLANN1 and 2 grammars

Feature	CLANN1	CLANN2
Pre-requisite knowledge for the user	Does not require any knowledge of ontologies, basic training enough to get started	Requires the knowledge of the domain ontology and the priciples behind it.
Domain dependence	The syntactic rules and vocabulary are heavily domain-dependent	Syntactic rules are domain-independent but the vocabulary is dependent on the ontology
Expresiveness	Not very expressive, can only generate ABox[10] statements from the restrictive syntax	Very Expressive, can generate both TBox and ABox statements
Ease of Use	Consice and easy to write, provided you know the grammar	Potential to become very verbose, owing to the expressiveness needed.

```
Project Name:  <String>
Group Name:  <String>
Date:  <Date>
Chair:  <String>
Attendees:  <String>(,<String>)+
Scribe:  <String>
Action Item:<String>:<String>(:<String>)?
(<CNL))+
Agenda:  <String> (<CNL>)+
Poll:<String>:<String>
(<String>:<String>)+
```

These templates were constructed by analysing a collection of in-house meeting minutes and status reports of the Nepomuk project[12]. This approach combines the benefits of the two by using templates for mundane information annotation and CNL for other non-mundane information, consequently minimising the effort and enhancing the user experience.

3.2. CL Snippets

Controlled Language snippets are used to explicity add annotation to words in the text. Snippets of text (enclosed in "[ ]") can be appended to words in the document to explicitly add annotations to the document. These snippets should adhere to a *subject-verb-object* syntax, where the subject is either specified in the snippet or taken from the free text. An example of the snippets is shown below.
Let us consider the sentence below

```
Brian went to Dublin for the weekend.
```

More information about "Brian" and "Dublin" can be added by using CL Snippets.

```
Brian[is a Person] went to Dublin[is a
City] for the weekend.
```

Further new resources and links between existing resources can be defined.

```
[City is a subclass of Place].
[Brian livesIn Dublin].
```
[13]

[12] http://dev.nepomuk.semanticdesktop.org/wiki/WikiStart

[13] assuming *livesIn* is a property between Person and City already defined in the domain ontology

```
Brian[is a Person] went to Dublin[is a
City] for the weekend.
```

The vocabulary for CLANN-prototype2 also includes the vocabulary of the ontology, thereby allowing the user to represent any kind of relational meta-data. This approach was inspired by the CLOnE Language (Funk et al., 2007). The syntactic constructs of CLANN-prototype2 along with examples are shown in Table 2.

3.3. Controlled English

The controlled english, developed during the process, forms the core of the CLANN grammar. This is inspired mostly from the efforts of CLANN-prototype1. The structure of sentences of the controlled english are restiricted by a simple set of desgin principles described below.

- Every sentence should be *declarative* and in *active voice*.
- Every verb used should be either an *infinitive* (to - verb : to denote tasks) or *simple past tense* (to denote reports)
- Every sentence should follow the *Subject-Verb-Object* pattern.
- Only *prepositional clauses* (prep NP : "by next week", etc) can be used to append nouns.
- Include *"snippets"* inline to add extra annotations
  - *Pradeep[is a Student]...*

The above guidelines are only easy-to-read restrictions of the grammar/ for a more complete specification fo the grammar please refer to http://smile.deri.ie/projects/clann

4. Implementation of the Grmammer

The previous prototypes of the grammar were developed using JAPE rules in GATE. JAPE(Java Annotation Patterns Engine) provides finite state transduction over annotations based on regular expressions(Cunningham et al., 2000).[14]

[14] Elaborating JAPE is out of scope of this paper. For more information on the subject the reader is referred to http://gate.ac.uk/sale/tao/splitch8.html#chap:jape

The ease of writing rules in Jape coupled with the extensive support of the GATE platform immensely helped in developing and testing rapid prototypes of the grammar. However adding support for auto-completion, based on Jape rules proved to be much harder. So we decided to explore other grammar formalisms which woud take advantage of the restricted vocabulary of the grammar to produce efficent parses along with support for auto-completion.

Various grammar formalisms have been used over the years for understanding natural language. Phrase structure grammars (PSG), the most widely used formalism, model the inherent structure of the sentences of a language by breaking it into different phrases. They belong to the class of generative grammars and are composed of a set of productions or rules which break-up the sentences into meaningful phrases. Dependency grammars(DG), however, concentrate on the links between words without paying attention to the word order. Structure of a sentence is not broken down into phrases, but determined by adding relations between a head word and its dependent words. There have been many variations of gramamar formalisms that stemmed out of both PSGs and DGs. The next few sections describe one such variation of the dependency grammar, the Link grammar, and justifies its selection.

4.1. Link Grammar

Link grammars, introduced by (Sleator et al., 1991), are a variation of dependency grammars. Similar to DGs, the link grammars use realtions between words to generate a structure for a sentence. However, unlike DGs, the links also encode information about directionality and distance. Moreover, they do not enforce a head-dependent relationship like the DGs.

(Sleator et al., 1991) defines link grammar as follows:

> *A sequence of words is a sentence of the language if there is a way to draw links between words in such a way that*
>
> - *the linking requirements of all the words are satisfied,*
> - *the links do not cross, and*
> - *the words form a connected graph*

The linking requirements of each word are specified as a dictionary, which forms the basis of the link grammar. Each entry in the dictionary consistes of a word or a group of words belonging to the same grammatical category, appended on the right-hand-side with its linking requirements. The linking requirements are a series of connectors joined by the logical operators *&* and *or*. Each connector denotes the type and direction of the link. It is a label followed by +/- . + denotes a link to the right and - denotes a link to the left. For illustration purposes, an example of a sentence parsed using a very simple link grammar is provided in Figure 1, and an explanation of the same is provided below.

The *D+* connector on the word *the* denotes that *the* is expecting a *D* link to its right. So It can connect to any word which has a *D-* connector, which, in this case, is either *boy* or *apple*. The word *ate* has an *&* operand on *S-* and *O+*.

words	linking requirements
the	D+
small	A+
ate	S- & O+
boy apple	(A- & D- & S+) or (D- & O-)

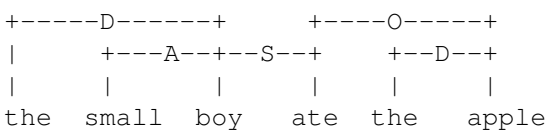

Figure 1: A sample Link grammar and parse structure.

This means, for the word *ate* to be part of a valid sentence, it should connect to both an *S* connector to its left and an *O* connector to its right. The case for the nouns *boy* and *apple* is more interesting. They have two expressions joined by the *or* operand. On closer observation, the first one, *(A- & D- & S+)*, models the behavior of a subject noun and the second one, *(D- & O-)*, models that of an object noun. The reader should also note that the order of the connectors is also valuable. The expression *(A- & D- & S+)* also declares the order of linking. So an *A* link should be made to a word closer than the *D* link. This is illustrated in the parse structure shown in Figure 1.

4.2. Why Link Grammar?

The main design principles for the CLANN grammar are ease of use and the ability to extend the ontology. However, the develpoment of the grammar posed different challenges. One major priority was to extract RDF tripples from the sentences. This works very well with the link grammar parse, because the the tripples can directly be extracted by mapping the links. In the example shown in Figure 1, the tripple *boy ate apple*, can be easily extracted from the left and right links of the word *ate*. This is not the case with phrase structure grammars, where extracting dependencies requires detailed ananlysis of the tree structure.

Another major priority was to develop an intelligent editor on top of the grammar, which supports auto-suggestion and sentence-completion. An intuitive editor which assists the user while writing the CNL sentences, goes a long way in helping him to quickly learn the restrictions of the grammar. This requires an ability to predict text and check the grammatical correctness of partial sentences. The dictionaries of the link grammar provide valuable information about all the words of the language, which can be exploited for the purpose.

5. The PDO ontology

The domain of meeting minutes and status reports was used to engineer an ontology for the purpose of knowledge management. The initial CLANN prototype was bootstrapped using the the Nepomuk(Decker, 2006) ontologies[15] and later extended by MEMO(Meeting Min-

[15] http://www.semanticdesktop.org/ontologies/

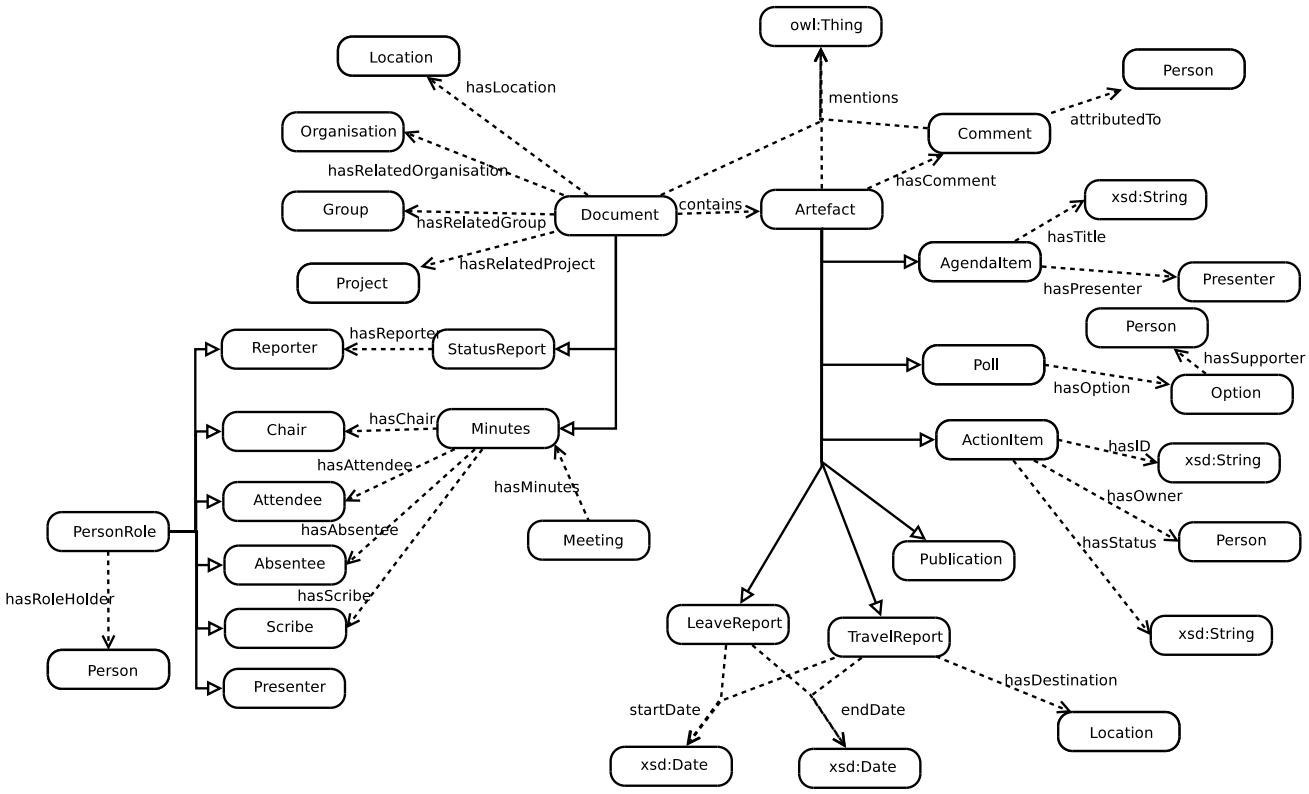

Figure 2: Overview of the PDO ontology

utes Ontology). However, the MEMO ontology was only used as a proof-of-concept implementation of the domain. Later, this was completely redesigned and a new ontology PDO (Project Document ontology) was developed in accordance with proper ontology design principles, specifically the METHONTOLOGY approach outlined by (Fernández-López et al., 1997). The PDO ontology, described using RDFS[16] and OWL-DL[17], models the inherent structure and concepts of various documents in a project-specific setting, like meeting minutes, status reports etc. The scope of this ontology was limited to modelleing the discourse structure of various project documents like meeting minutes, status reports, final reports, deliverables, etc. The content of these documents is not modelled, in order to make the ontology very flexible and interoperable. Care was taken to ensure that other domain ontologies can be easily linked. So, for instance, a meeting minute note might talk about anything from software projects to movie reviews but still be modelled by the ontology, while using the respective domain ontolgies of software projects and movies.

A pictorial representation of the ontology is shown in Figure 2. *Document* is the central class which is subclassed by *Minutes* and *StatusReport*. *Artefact* is the main placeholder class for various arefacts contained in a document, like *AgendaItem*, *Poll*, *ActionItem*, *TravelReport*, etc. A partial instantiation of the ontology is described in Turtle[18] syntax in the Figure 3. For a complete specification of the PDO ontology please refer to `http://ontologies.smile.deri.ie/pdo#`.

6. Conclusion and Future Work

In conclusion, we have set out to explore the possibility of using controlled languages as interfaces to semantic web applications. We decided to narrow down the domain to meeting minutes and status reports, and designed an ontology representing the domain. We have developed the CLANN system for knowledge acquisition along with the CLANN grammar and experimented with a few flavours of the grammar. Furthermore, we are currently working on implementing the ideas presented for the CLANN3 grammar.

Future work will involve developing a smart CNL text editor for novice users, as well as packaging the CLANN module as an extension to the Semantic MediaWiki. We also plan to evaluate our work against othe semantic wikis, to judge the feasibility of the approach.

Acknowledgements

The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

[16]RDFSchema `http://www.w3.org/TR/rdf-schema`

[17]OWL (Web Ontology Language) has three flavours, OWL Lite, OWL DL and OWL Full. `http://www.w3.org/TR/owl-features/` OWL DL is named so becase of its correspondence with Description Logics. This flavour of OWL is neither too strict nor too simple.

[18]Turtle is an easy, human-readable serialization of RDF. `http://www.w3.org/TeamSubmission/turtle/`

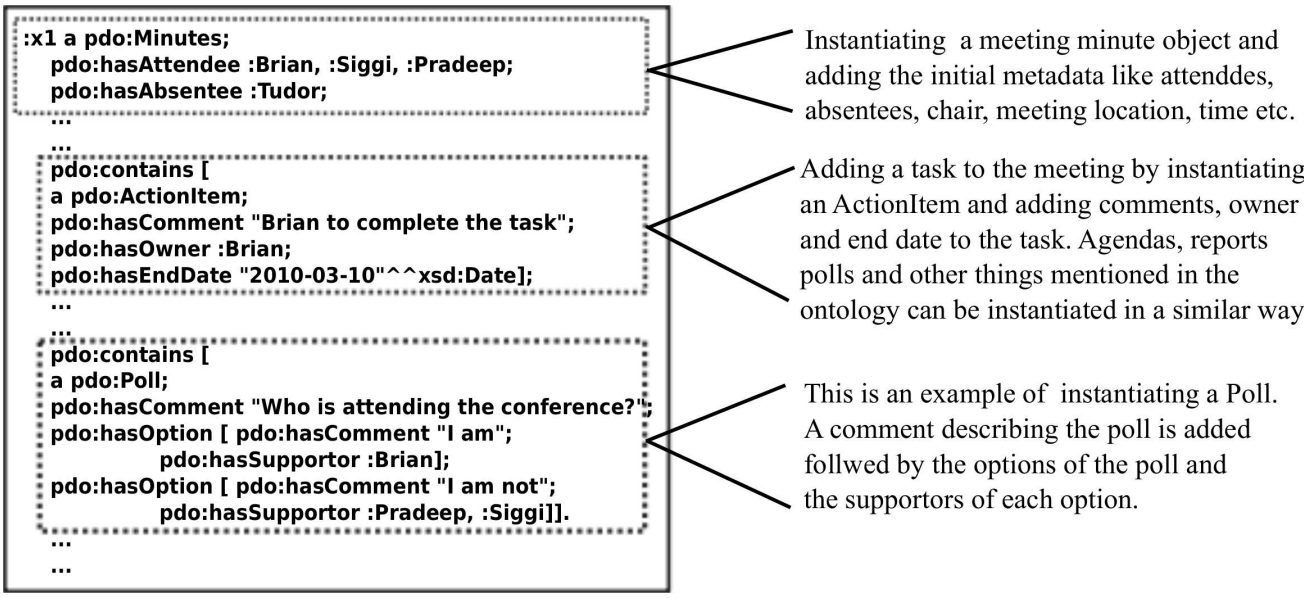

Figure 3: Using the PDO Ontology

7. References

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November.

Brian Davis, Pradeep Varma, Siegfried Handschuh, Laura Dragan, and Hamish Cunningham. 2009. On designing controlled natural languages for semantic annotation.

S. Decker. 2006. The social semantic desktop: Next generation collaboration infrastructure. *Information Services and Use*, 26(2):139–144.

Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. 1997. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, Stanford, USA, March.

A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, and S. Handschuh. 2007. Clone: Controlled language for ontology editing. In *ISWC/ASWC*, pages 142–155.

Siegfried Handschuh. 2005. *Creating Ontology-based Metadata by Annotation for the Semantic Web*. Ph.D. thesis.

Daniel D. K. Sleator, C Fl Daniel Sleator, and Davy Temperley. 1991. Parsing english with a link grammar. In *In Third International Workshop on Parsing Technologies*.

P. R. Smart. 2008,(Unpublished). Controlled natural languages and the semantic web. Technical report, School of Electronics and Computer Science, University of Southampton.