Enabling Semantic Search in Large Open Source Communities

Gregor Leban(✉), Lorand Dali, and Inna Novalija

Jožef Stefan Institute, Jamova Cesta 39, 1000 Ljubljana, Slovenia
{gregor.leban, lorand.dali, inna.koval}@ijs.si

Abstract. This paper describes methodology used for building a domain specific ontology. Methods that allow automatic concept and relation extraction using domain-related glossaries are presented in this research. The constructed ontology contains vocabulary related to computer science and software development. It is used for supporting different functionalities in the ALERT project, which aims to improve software development process in large open source communities. One of the uses of the ontology is to provide semantic search functionality, which is a considerable improvement over the keyword search that is commonly supported.

1 Introduction

Open source communities and software development organizations in general are often using several different communication channels for exchanging information among developers and users. Beside a source code management system (SCMS), these software developing communities also frequently use an issue tracking system (ITS), a forum, one or more mailing lists and a wiki. Each of these channels typically serves a different purpose. Issue tracking system allows the users of the software to report to the developers issues they encountered or to suggest new features. Forums and mailing lists have a similar purpose which is to allow open discussions between the members of the community. Wikis are commonly used as platforms for providing software documentation, user guides and tutorials for the users.

A problem that is common for the present day and that is becoming more and more troubling for large and medium size open source communities is information overload. Users are generating large amounts of information on different communication channels and it is difficult to stay up-to-date. For illustration, consider the KDE community [1], which is offering a wide range of open source products. In April 2012 KDE had approximately 290.000 bug reports in their ITS, 126.000 posts on their forum and 163 active mailing lists where according to one of the KDE developers between 30–80 emails are exchanged per day.

Providing help in processing and managing such large amounts of information is one of the main goals of the ALERT[1] project [2]. ALERT is a European project that aims to develop a system, which will be able to help users (especially developers) in large open source communities. The system, once finished, will be able to collect and

[1] ALERT is acronym for Active support and reaL-time coordination based on Event pRocessing in open source software development.

© Springer-Verlag Berlin Heidelberg 2015
E. Simperl et al. (Eds.): ESWC 2012 Satellite Events, LNCS 7540, pp. 291–300, 2015.
DOI: 10.1007/978-3-662-46641-4_22

process all the posts (emails, issues, forum posts, etc.) that are generated in different communication channels used by the community. The information will be processed and stored in a way so that it will provide support for different features of the system.

The use case partners in the project are KDE, OW2 [3] and Morfeo [4]. At the beginning of the project they provided a set of most wanted features that should be supported by the ALERT system. One of the features that were identified as very important was advanced search functionality. Search that is supported on communication channels such as ITS and forums is only a simple Boolean keyword search. A fundamental problem with keyword search is that different people use different words (synonyms) to refer to the same concept. Therefore not all posts that discuss the same concept can be retrieved by a simple keyword-based search.

As an improvement to keyword search we would like to provide in the ALERT system *semantic* search. What we mean by semantic is that the search is performed using the actual concept that the search term represents. Consider, for example, that the user performs a search for term "dialog". The concept of the dialog can be represented also with other terms, such as "window" or "form". Instead of returning the results that directly mention "dialog" we therefore also want to return results that contain any of the term synonyms. Additionally, since search is based on the actual concepts we can also exploit the fact that concepts can be related to each other. When searching for one concept we can therefore also consider including results about some closely related concepts. In KDE domain, for example, searching for "email client" should also return posts containing concept "KMail" which is the KDE's email application.

In order provide semantic search we have to use an ontology. Each class in the ontology should represent a concept that can have one or more labels (synonyms). When a new post is created in one of the communication channels, the ALERT system annotates or tags it with the concepts that are mentioned in the post. These annotations are stored in a knowledge base, which allows us then to quickly find all posts tagged with a particular concept.

The main question that needs answering is what ontology should be used for annotating the posts. Since we are providing support for software developing communities the important concepts that should be annotated are the ones related to computer science and software development. Since we were not able to find any such existing ontology we had to construct it ourselves. The process that we used to construct such an ontology is the main contribution of the paper. The steps in the process are general and can be reused also for constructing other domain specific ontologies.

The remainder of this paper is organized as follows. Section 2 provides details of the methodology used for building the Annotation ontology. In Sect. 3 we describe how the ontology can be used in ALERT to provide the semantic search functionality. Section 4 describes related work and Sect. 5 provides the conclusions.

2 Building the Annotation Ontology

In this section we describe the process that was used in order to construct the ontology used for annotating the posts. The process consists of two main steps – (a) identifying the computer science specific terminology that we wish to represent in the ontology, and (b) constructing the relations between the concepts.

2.1 Creating Ontology Concepts

As stated before, the concepts that we wish to have in the Annotation ontology are related to computer science and software development. In order to obtain a relevant set of terms we searched online for glossaries related to computer science. The two web sites that we found to be most up-to-date and relevant for our purpose were *webopedia.com* [5] and *whatis.techtarget.com* [6]. For each of the terms we were also able to obtain a description of the term which in most cases contained links to several related terms. To identify terms especially related to software development we used the *stackoverflow* website [7], which is a Q-A system with more than 2.5 million questions related to software development. *Stackoverflow* contains an up-to-date list of tags that are used to tag the questions. Most popular tags together with their descriptions were also included in the starting set of concepts.

After obtaining the set of terms, our first goal was to merge synonyms. Merging of terms was performed in two ways. First way was using a synonym list that we were able to obtain from the *stackoverflow* website and which contained around 1,400 synonym pairs. The second way was by using the term descriptions and searching in them for patterns such as "X also known as Y" or "X (abbreviated Y)". In cases when such patterns were identified, terms X and Y can be considered as synonyms and be represented as the same concept. In this way we obtained a set of concepts where each concept has one or more labels that represent the concept.

In the next step we wanted to link the concepts to corresponding Wikipedia articles. This allows us to obtain more information about the concepts and potentially also extend the ontology with new related concepts. By using a semi-automatic approach, we make the repetition of the process relatively easy to do, such that future updates of the ontology are not too costly. By identifying a corresponding Wikipedia article we are also able to implicitly create links to well-known knowledge bases which are extracted from Wikipedia, such as DBpedia, Yago and Freebase.

Our approach for mapping concepts to Wikipedia articles has several steps. First, we link the concept labels to Wikipedia articles. We do this by automatically matching the labels to the titles of articles to see which article corresponds to each label. In this process, the following two challenges were identified:

a) The article with the matching title is a disambiguation page i.e. a page containing links to pages which each describe one of the meanings of the concept. For example *TTL* is mapped to a page which contains links to *Time to Live*, *Transistor Transistor Logic*, *Taiwan Tobacco and Liquor,* etc.
b) Some of the computer science concepts are so frequently used in common language that they are not considered ambiguous. In this case a computer science concept can be mapped on Wikipedia to something completely unrelated to computer science. An example of such a concept is *ant* which in computer science refers to *Apache ant*, a software tool for automatic build processes, but is mapped to the Wikipedia article about the ant insect.

The disambiguation pages were not difficult to identify since they typically contain phrases such as '*X may mean…*' or '*X may refer to…*', '*X is an abbreviation and can*

refer to...'. We have defined rules to automatically match these patterns and exclude disambiguation pages from further analysis.

After mapping labels to the corresponding Wikipedia pages we used the content of these pages to identify new terms which were not covered by the glossaries. To do this, we only used the first paragraph of each article, which usually gives a short definition of the term. Often it also contains links to articles describing closely related concepts. We used the articles linked in the first paragraph as candidates for new terms and sorted them by their frequency. We expect that if an article was linked to by many articles that we know are about computer science, then this article is very likely about a computer science concept as well. Based on this assumption, titles of the frequently appearing articles were added to the ontology as new concepts.

After obtaining the final set of concepts we also wanted to organize the concepts into a hierarchy of categories. For this purpose we used text mining techniques and in particular the OntoGen [8] toolbox which interactively uses k-means clustering [9] to group the concepts into a hierarchy and extracts keywords to help the user in assigning a name to each category. In this way we were able to semi-automatically define 31 categories such as "operating systems", "programming languages" and "companies".

2.2 Creating Relations Between the Ontology Concepts

An important part of the ontology are also relations between the concepts. With regard to our task of semantic search, the relations allow us to expand the search to also include closely related concepts.

To create the relations between the concepts we can start by using the information that was available in the online glossaries. As we mentioned, the descriptions of the terms usually contained several links to other related terms. These links can be used to automatically create relations between the corresponding ontology concepts. Since hyperlinks don't contain any additional semantic information about the type of relation we can only use them to create some general kind of relation between the concepts. In our ontology we represented them using a *linksTo* relation.

In order to obtain more specific and usable relations we decided to apply natural language processing (NLP) techniques on term descriptions with the goal of identifying semantic relations between the concepts. Consider, for example, the following sentence from the "C#" concept description:

> "C# is a high level, general-purpose object-oriented programming language created by Microsoft."

If "C#" and "Microsoft" are concepts in the ontology then it is possible using NLP techniques to identify that the verb connecting the two concepts is "created by". The task of creating relations between concepts can then be reduced to defining a mapping from verbs to appropriate relations.

A detailed list of steps involved in creating the relations is as follows. The input to the procedure was the list of ontology concepts and all the descriptions of the concepts. First we identify in the descriptions sentences that mention two or more concepts. Next we use Stanford parser [10] to generate a dependency parse of the sentence.

The dependencies provide a representation of grammatical relations between words in a sentence. Using the dependency parse and the co-occurring ontology concepts, we can extract the path from one ontology concept to another one. As a next step, we used Stanford Part-Of-Speech (POS) tagger [11] to tag the words in the sentence. Of all the tags we are only interested in the verb (with or without preposition) that connects the two concepts. As a result we can obtain triples, such as:

XSLT, used by, XML schema
WSDL, describes, Web Service
Microsoft, created, Windows
Apple Inc., designed, Macintosh.

In the next step we use WordNet [12] and group the obtained verbs into synsets (synonym sets). From all the sentences we obtained verbs that can belong to around 750 different WordNet synsets. Of all these synsets we only considered those that can be mapped to relations *isPartOf, hasPart, creator* and *typeOf*. We chose to include these relations because they are mostly hierarchical and can be used to expand the search conditions. WordNet synsets that were used to obtain these relations were:

- *isPartOf* and *hasPart* relations were obtained from "include" and "receive have" synsets
- *creator* relations were obtained from "make create", "form constitute make", "implement", "construct build make", "produce bring forth", "introduce present acquaint", "make do" and "plan project contrive design" synsets
- *typeOf* relations were obtained from "establish base ground", "include", "exist be" and "integrate incorporate" synsets.

In addition to these relations we also included a few other types of relations:

- *subclass* and *superclass* relationships have been obtained by using the OntoGen [8] text mining tool
- *sameAs* relationships provide links to the identical Wikipedia and DBpedia resources.
- *linksTo* relations were used for all relations that we extracted from term descriptions but were not mapped to some more specific type of relation (like *isPartOf*, *creator*, etc.).

2.3 Filtering and Publishing the Ontology as RDF

Before the ontology was finished we wanted to make sure that it doesn't contain any unnecessary concepts. By checking the terms on the online glossaries we noticed that some of them are obsolete and therefore irrelevant for our ontology. To determine if a concept is relevant or not we decided to again use the stackoverflow website. For each concept we searched in how many questions the concept is mentioned. If the concept was mentioned in less than 10 questions we decided to treat it as irrelevant and we removed the concept and its relations from the ontology. The value 10 was chosen experimentally by observing which concepts would be removed at different thresholds. An example of a concept that was removed by this procedure is HAL/S (High-order Assembly Language/ Shuttle) which was found only in one question on the *stackoverflow* website.

The final version of the generated ontology contains 6,196 concepts and 91,122 relationships and is published in the Resource Description Framework (RDF) format. In Fig. 1 we show an example of an ontology concept for the Central Processing Unit. It contains two labels, a description of the concept, links to equivalent Wikipedia and DBpedia resources and several relations to related concepts.

3 Using the Annotation Ontology for Semantic Search

The created ontology can be used to annotate all the posts that are generated in the communication channels monitored by the ALERT system. When a new post is created we annotate it with concepts that are mentioned in the text. We do this by checking the

```
<rdf:Description rdf:about="http://ailab.ijs.si/alert/resource/r18328">
        <ailab:highlights>central processing unit</ailab:highlights>
        <ailab:highlights>CPU</ailab:highlights>
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r14315" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r14039" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r16273" />
        <owl:sameAs rdf:resource="http://dbpedia.org/resource/Cpu" />
        <owl:sameAs rdf:resource="http://dbpedia.org/resource/CPU" />
        <owl:sameAs
rdf:resource="http://en.wikipedia.org/wiki/Central_processing_unit" />
        <owl:sameAs
rdf:resource="http://dbpedia.org/resource/Central_processing_unit" />
        <owl:sameAs rdf:resource="http://en.wikipedia.org/wiki/Cpu" />
        <owl:sameAs rdf:resource="http://en.wikipedia.org/wiki/CPU" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r18514" />
        <rdfs:comment>The central processing unit, or "processor". The thing
thatll eventually *execute* all that code youre writing. The central processing unit
(CPU) is the portion of a computer system that carries out the instructions of a
computer program, and is the primary element carrying out the computer's func-
tions. The central processing unit carries out each instruction of the program in
sequence, to perform the basic arithmetical, logical, and input/output operations
of the system. This term has been in use in the computer industry at least since the
early 1960s. The form, design and implementation of CPUs have changed dramat-
ically since the earliest examples, but their fundamental operation remains much
the same. Source: Wikipedia</rdfs:comment>
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r18328" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r16984" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r16766" />
        <ailab:linksTo rdf:resource="http://ailab.ijs.si/alert/resource/r12379" />
        <rdfs:label>central processing unit</rdfs:label>
        <rdfs:label>CPU</rdfs:label>
</rdf:Description>
```

Fig. 1. Ontology concept for Central Processing Unit

labels of the concepts and determining if any of them appears in the text. The post with its annotations is then stored in the Knowledge base and can be used when performing the search.

Since the ALERT project is still in progress we currently only have a preliminary version of the search interface. A screenshot of the interface is shown in Fig. 2. The search form is located in the top left corner and allows the user to specify a rich set of search conditions. Beside the keyword search the conditions can also include:

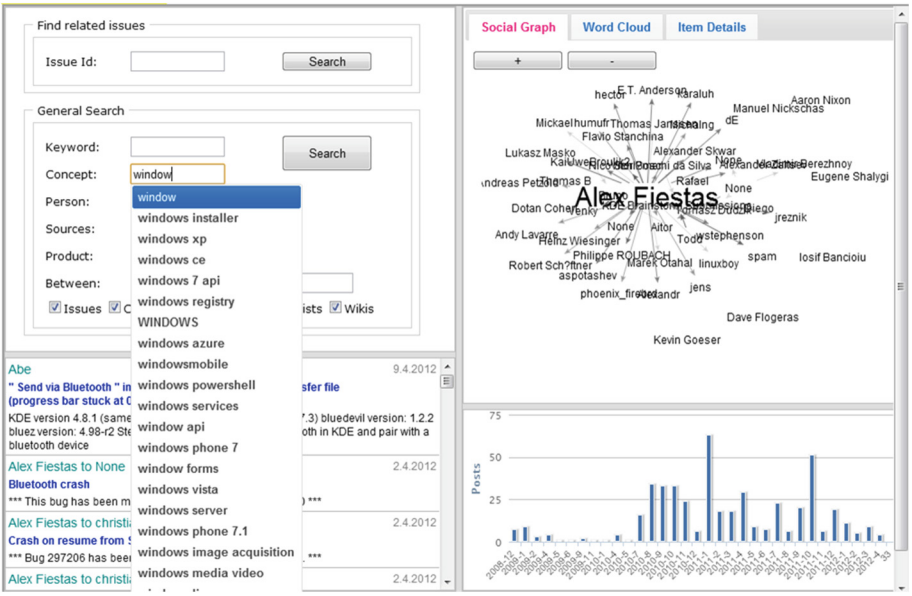

Fig. 2. Search interface provided by the ALERT system

- Concepts. The user can specify a concept from the Annotation ontology in order to find posts that are annotated with this concept.
- Authors of the posts. All posts from the communication channels have authors and they can be specified as a condition.
- Source code (files, classes, methods). By monitoring source code management systems used by the community we are aware of all the files, classes and methods developed in the project. Because the information is stored in the knowledge base we can easily find all the source code commits where a particular file/class/method was modified.
- Product. Issue tracking systems allow the issues to be assigned to particular products and components. When searching for issues we can limit the results only to issues that belong to a particular product or component.
- Time constraints. All posts have an associated time stamp that represents the time when the post was created. The search interface allows us to limit the search to a particular time period that we are interested in.

– Filtering by post type. Each post in the ALERT system is assigned a type based on the source where the post comes from (issues, emails, forum posts, etc.). The user can specify what type of posts he would like to see in the list of results.

After performing the search, the list of posts that match the query is displayed below the search form. Each item in the list contains the author of the post, the creation date, the subject and a short snippet of the post. Along with the list of results, the system also provides two visualizations of the results. Social graph of the people involved in the resulting posts is displayed on the right side of the screen. It shows who is corresponding with whom and highlights the most active people. Below the social graph is the timeline visualization that shows the distribution of results over time. It is an important aggregated view of the results since it can uncover interesting patterns. An example of such interesting pattern would be a spike in the number of submitted issues for a particular product. Such a pattern can be used to determine which source code commits should be analyzed to find the cause of the issues.

4 Related Work

The automatic and semi-automatic ontology learning methods usually include a number of phases. Most approaches define the set of the relevant ontology extension sources, preprocess the input material, build ontology according to the specified methodology, evaluate and reuse the composed ontology.

While developing ontologies, it is important to follow a number of ontology design criteria. Gruber [13] defines the following design criteria for ontology developing: Clarity, Coherence, Extendibility, Minimal encoding bias and Minimal ontological commitment. Uschold and King's method [14], Grüninger and Fox's methodology [15], METHONTOLOGY [16], On-To-Knowledge [17] represent the classic methodologies to ontology creation.

As Reinberger and Spyns [18] state, the following steps can be found in the majority of methods for ontology learning from text: collecting, selecting and preprocessing of an appropriate corpus, discovering sets of equivalent words and expressions, establishing concepts with the help of the domain experts, discovering sets of semantic relations and extending the sets of equivalent words and expressions, validating the relations and extended concept definition with help of the domain experts and creating a formal representation.

As suggested in [19], ontology learning from text is just one phase in the methodology for semi-automatic ontology construction preceded by domain understanding, data understanding and task definition and followed by ontology evaluation and ontology refinement.

In our approach we have utilized the traditional steps for ontology development, like terms extraction, synonyms extraction, concepts definition, establishment of concept hierarchies, relations identification [20].

Fortuna et al. [8] developed an approach to semi-automatic data-driven ontology construction focused on topic ontology. The approach combines machine learning and text mining techniques with an efficient user interface. The domain of interest is

described by keywords or a document collection and used to guide the ontology construction. OntoGen [8] uses the vector-space model for document representation. In current work, the tool has been utilized for defining the hierarchical relationships between concepts.

Learning relations in the ontology was addressed by a number of researchers. Taxonomic relations have been extracted by Cimiano et al. [21]. Moreover, Cimiano et al. [22] suggested a method of learning concept hierarchies from text based on Formal Concept Analysis. Maedche and Staab [23] contributed to the approach, which allowed discovering conceptual relations from text.

In our work of building a domain specific ontology, we not only define ontology concepts, but also to specify the possible binary relations between the concepts using NLP techniques.

A number of researchers used ontologies for different tasks in software domain. Nallusamy et al. [23] describes how ontologies can be applied for software re-documentation. Rene Robin et al. [24] have designed an ontology suite for software risk planning, tracking and control.

5 Conclusions

In this paper we have proposed an approach for building a domain specific ontology related. The methods for concept and relation extraction have been suggested and applied in order to build an ontology related to computer science and software development. The generated ontology is used in the ALERT project among other things to provide semantic search functionality. The advantages of the semantic search over keyword search are (a) the avoidance of issues with synonyms, and (b) the ability for expanding the search by including related concepts in the search. The current version of the ALERT system provides a preliminary interface for performing the semantic search by entering the concept name. In future we plan to improve the interface to allow the user also to extend the search to related concepts.

Acknowledgements. This work was supported by the Slovenian Research Agency, European Social Fund and ALERT (ICT-2009.1.2).

References

1. KDE. http://www.kde.org
2. ALERT project. http://www.alert-project.eu
3. OW2. http://www.ow2.org
4. Morfeo project. http://www.morfeo-project.org
5. Webopedia. http://www.webopedia.com
6. Computer Glossary, Computer Terms. http://whatis.techtarget.com
7. Stack Overflow. http://www.stackoverflow.com
8. Gopal, D., Wang, Q., Gupta, G., Chitnis, S., Guo, H., Karshmer, A.I.: Winsight: Towards completely automatic backtranslation of nemeth code. In: Stephanidis, C. (ed.) HCI 2007. LNCS, vol. 4556, pp. 309–318. Springer, Heidelberg (2007)

9. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297. University of California Press
10. de Marneffe, M.-C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: LREC 2006 (2006)
11. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 63–70 (2000)
12. WordNet. http://wordnet.princeton.edu
13. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum Comput Stud. **43**, 5–6 (1995)
14. Uschold, M., King, M.: Towards a methodology for building ontologies. In: Proceedings of Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95 Canada (1995)
15. Gruninger, M., Fox, M.: The role of competency in enterprise engineering. In: Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, IFIP (1994)
16. Corcho, Ó., Fernández-López, M., Gómez-Pérez, A., López-Cima, A.: Building legal ontologies with METHONTOLOGY and WebODE. In: Benjamins, V., Casanovas, P., Breuker, J., Gangemi, A. (eds.) Law and the Semantic Web. LNCS (LNAI), vol. 3369, pp. 142–157. Springer, Heidelberg (2005)
17. Sure, Y., Studer, R.: On-To-Knowledge Methodology - Final Version, On-To-Knowledge deliverable D-18, Institute AIFB. University of Karlsruhe (2002)
18. Reinberger, M.L., Spyns, P.: Unsupervised text mining for the learning of DOGMA-inspired ontologies. In: Buitelaar, P., Handschuh, S., Magnini, B. (eds.) Ontology Learning from Text: Methods. IOS Press, Evaluation and Applications (2005)
19. Grobelnik, M., Mladenic, D.: Knowledge discovery for ontology construction. In: Davies, J., Studer, R., Warren, P. (eds.) Semantic Web Technologies: Trends and Research in Ontology-Based Systems, pp. 9–27. Wiley, Chichester (2006)
20. Buitelaar, P., Cimiano, P., Magnini, B. (eds.): Ontology Learning from Text: Methods, Evaluation and Applications. IOS Press, Amsterdam (2005)
21. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. J. Artif. Intell. Res. **24**, 305–339 (2005)
22. Cimiano, P., Pivk, A., Schmidt-Thieme, L., Staab, S.: Learning taxonomic relations from heterogeneous evidence. In: Proceedings of ECAI 2004, Workshop on Ontology Learning and Population (2004)
23. Maedche, A., Staab, S.: Discovering conceptual relations from text. In: Horn, W. (ed.) ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, 21–25 August 2000, pp. 321–324. IOS Press, Amsterdam (2000)
24. Nallusamy, S., Ibrahim, S., Mahrin, M.N.: A software redocumentation process using ontology based approach in software maintenance. Int. J. Inf. Electron. Eng. **1**(2) (2011)
25. Rene Robin, C.R., Uma, G.V.: Design and development of ontology suite for software risk planning, software risk tracking and software risk control. J. Comput. Sci. **7**(3), 320–327 (2011)