**1. Introduction**

**1.1 Motivation**

Computer programs have become the most frequently used tools in our modern society. Nowadays, they are present at large scale in industry, covering multiple areas. As complex software systems are built at a fast pace, they need to remain maintainable through time. For this reason, software quality must be at its highest level, yet in most cases, it decreases as the systems are getting bigger.

Testing the code is the way for assuring the required functionality from the perspective of the users. From the programmers point of view, the code needs to be clean and easy to extend or reuse. Design patterns, coding standards, static code analysis are software engineering methodologies serving such a purpose.

Another key aspect of software development is the use of version control systems in order to keep track of changes and make possible for teams to collaborate. They also provide a general view on the projects and backup service as well.

But none of the above mentioned techniques address the problem of retrieving meta information from the code, in a semantic manner. Large software projects involving thousands of source code files would be easier to understand, control and extend if they would be complemented by a solid information retrieval system. The first step to achieve this is to create a graph oriented knowledge base from the entities present in code. This can be built on top of specific ontologies, using a convenient format like the resource description framework (RDF). Having the knowledge base in place, it would be easy to query the system (e.g. SPARQL) about its interacting components and services. Going further, an answering mechanism could be applied for enabling natural language questions on the knowledge base.

**1.2 Keywords**

C#, static analysis, ontology, RDF, triple store, linked data, natural language, SPARQL

**2. Theoretical Notions**

……………………………………….

**2.1 Definitions**

………………………………………………

**3. System Overview**

**3.1 The proposed system**

The proposed system (*Figure 1)* is designed to follow two main ideas.

First, there is the process of extracting structured data from C# source files, based on an existing specific ontology (e.g. object-oriented programming ontology) and store that data in a way such that it can be easily retrieved later. In this case, the data will be saved in a triple store.

Second, there is the process of retrieving data from the store, having hierarchical levels of querying. Whilst SPARQL queries are used for this purpose at the lowest level, at the highest level, the goal is the use of natural language questions. In the middle, annotated questions are used, based on NLP techniques.

**3.2 Existing Systems**

Some of the existing approaches that are based on similar ideas are mentioned below. They address singular or specific problems, so for the proposed system, the intent is to adapt, extend and combine some of the ideas, in order to achieve the goal.

**3.2.1 For the first part of the system**

*1. RDF Coder*

*RDF Coder* is a tool able to generate RDF models of code libraries, entirely written in Java. *RDF Coder* can be used to perform multi-level code inspection, create code dependency graphs and generate custom documentation. Currently is supported the Java language only at version *1.5*.

This library can be used by Java programmers as a tool to deal with huge classpaths, to find relationships across classes and objects. *RDF Coder* can be also used to develop more complex analysis tools leveraging on the flexibility of the RDF related technologies.

*2. Fuzzy Ontology Framework*

*Fuzzy Ontology Framework* is a library that helps to integrate a fuzzy ontology with object-oriented programming (OOP) classes written in .NET. It is a hybrid integration, i.e. some OWL concepts can be mapped directly to OOP classes, yet most OWL concepts are derived just from OOP instance properties, with no direct mapping to a .NET class. Hence the OOP instance-OWL concept(s) mapping can evolve dynamically in the course of time.

The implementation currently supports FuzzyOWL2 ontologies together with FuzzyDL reasoner to infer affiliation of OOP instances to particular OWL concepts. It can be however easily modified to support any fuzzy ontology notation as well as any fuzzy reasoner.

*3. SCRO*

*SCRO* is created to support major Software Understanding tasks by explicitly representing the conceptual knowledge structure found in source code.

*SCRO* captures major concepts of object-oriented programs and helps understand the relations and dependencies among source code artifacts. Supported features include, encapsulation, inheritance (subclassing and subtyping), method overloading, method overriding, and method signature information.

*4. Ion Smeureanu, Bogdan Iancu -* *Source Code Plagiarism Detection Method Using Protégé Built Ontologies.* In this paper *the authors demonstrate how source code plagiarism could be detected with the help of an ontology which models software programs.*

**3.2.2 For the second part of the system**

*1. Treo*

The main ideas behind this system are:

- Entity Recognition and Pivot Determination through Entity Search

- Query Syntactic Analysis: Partial Ordered Dependency Structure (PODS) Determination

- Spreading Activation using Semantic Relatedness

The algorithm first determines the *key entities* present in the natural language query. They can be potentially mapped to instances or classes in the Linked Data Web. The entity search engine receives the *key entities* and resolves *pivot entities* (URIs) in the Linked Data Web. The query is then analyzed and parsed to obtain *partial ordered dependency structure (PODS).* This is the reduced representation of the query targeted towards maximizing the matching probability between the structure of the terms present in the query and the subject, predicate, object structure of RDF.

The *spreading activation search* takes the URIs of the pivots and the PODS structure, and thus, starting from the pivot node, the algorithm navigates through neighboring nodes in the Linked Data Web computing the *semantic relatedness* between query terms and vocabulary terms in the node exploration process. The navigation process builds the answer of the query. The algorithm returns a set of ranked triple paths determined by the navigation from the pivot entity to the final resource representing the answer, ranked by the average of the relatedness scores over each triple path. Answers are displayed to users using a list of triple paths merged in a graph after a simple post-processing phase.

*2. TBSL*

The ideas of this system are:

- SPARQL Template from question

- Mapping between NL expressions to the domain vocabulary

The *TBSL* approach to question answering over RDF data relies on a parse of the question to produce a SPARQL template that directly mirrors the internal structure of the question and that, in a second step, is instantiated by mapping the occurring natural language expressions to the domain vocabulary.

The user's input is a natural language question which is processed by a POS tagger. The result is the semantic representation of the natural language query, based on lexical entries that are created using a set of heuristics. In the next step, this is converted into a SPARQL query template which contains slots: missing elements of the query that have to be filled with URIs. The URIs are determined using sophisticated entity identification approaches, based on string similarity as well as on natural language patterns which are compiled from existing structured data in the Linked Data cloud and text documents.

*3. Squall2sparql*

SQUALL (Semantic Query and Update High-Level Language) is a Controlled Natural Language (CNL) for English that has full compliance with Linked Open Data (LOD), and covers nearly all features of SPARQL 1.1, for both queries and updates. The advantage of CNLs is to provide a natural language syntax while retaining the precision and lack of ambiguity of formal languages like SPARQL. The main drawback of CNLs is that users have to learn the language and its disambiguation rules. The system *squall2sparql2* is a Web application that supports the translation from SQUALL to SPARQL, as well as the direct querying of SPARQL endpoints, like DBpedia. Given a SQUALL sentence, the system first translates it into an intermediate logical representation using a *Montague* grammar. The intermediate representation is then translated into SPARQL, simply mapping logical constructs to combinations of SPARQL constructs. The produced query can then be sent to any SPARQL endpoint, and results returned.

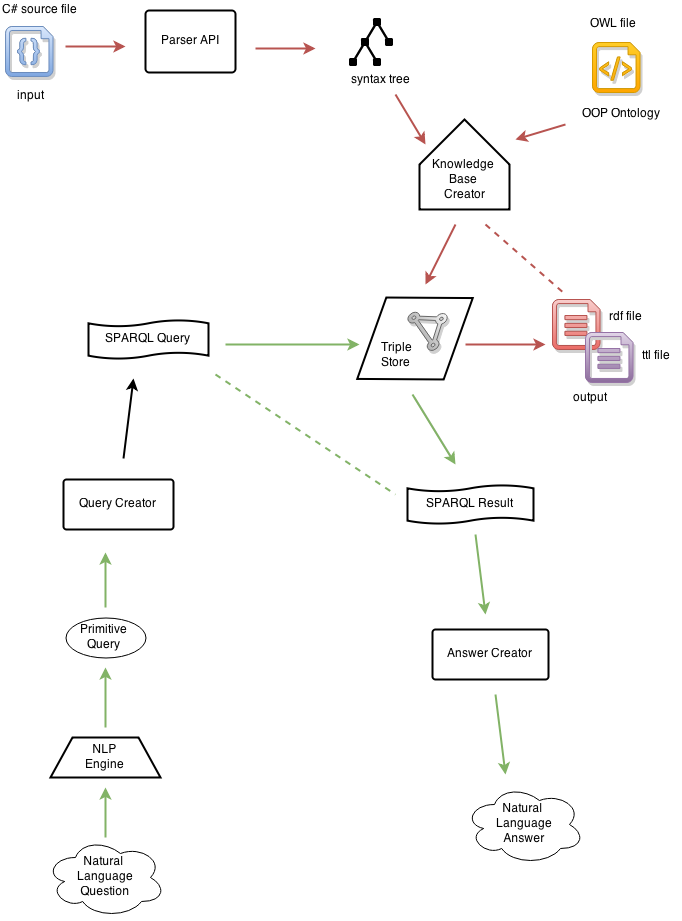
*4. GFMed*

The main ideas of this system are:

- use of grammars created in *Gramatical Framework*

- use of medical datasets such as *SIDER*, *Diseasome* and *DrugBank*

*GFMed* system is focused on medical area questions and knowledge base.It consists mainly of a *Gramatical Framework* grammar for the application domain given by *SIDER*, *Diseasome* and *DrugBank* datasets. There is a minor preprocessing of questions and postprocessing of translation results, mainly in order to deal with structures involving numeric values, e.g. values for water solubility, or free text, like different names of foods. The system uses an abstract grammar and two concrete grammars. For each concrete grammar, lexicons derived from *SIDER*, *Diseasome* and *DrugBank* were generated. Both English and SPARQL grammars are based on the datatsets terminology.

Figure 1: General scheme of the proposed system

**References**

1. <https://code.google.com/p/rdfcoder/>

2. <http://www.codeproject.com/Articles/348918/Fuzzy-Ontology-Framework>

3. <http://www.cs.uwm.edu/~alnusair/ontologies/scro.html>

4. [http://revistaie.ase.ro/content/67/07%20-%20Smeureanu,%20Iancu.pdf](http://revistaie.ase.ro/content/67/07 - Smeureanu, Iancu.pdf)

5. <https://www.deri.ie/sites/default/files/publications/freitas_qald_2011_0.pdf>

6. <http://www.edwardcurry.org/publications/SIGIR13_Demo_freitasPS.pdf>

7. <http://svn.aksw.org/papers/2013/KESW_AutoSparqlTbsl_Demo/public.pdf>

8. <http://ceur-ws.org/Vol-1179/CLEF2013wn-QALD3-Ferre2013.pdf>

9. <http://ceur-ws.org/Vol-1180/CLEF2014wn-QA-Marginean2014.pdf>