Introduction

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.

Different types of programming languages evolved during time using various paradigms like procedural, object-oriented or functional. Object-oriented programming is

Keywords

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

Existing Systems

1. *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* 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* 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.

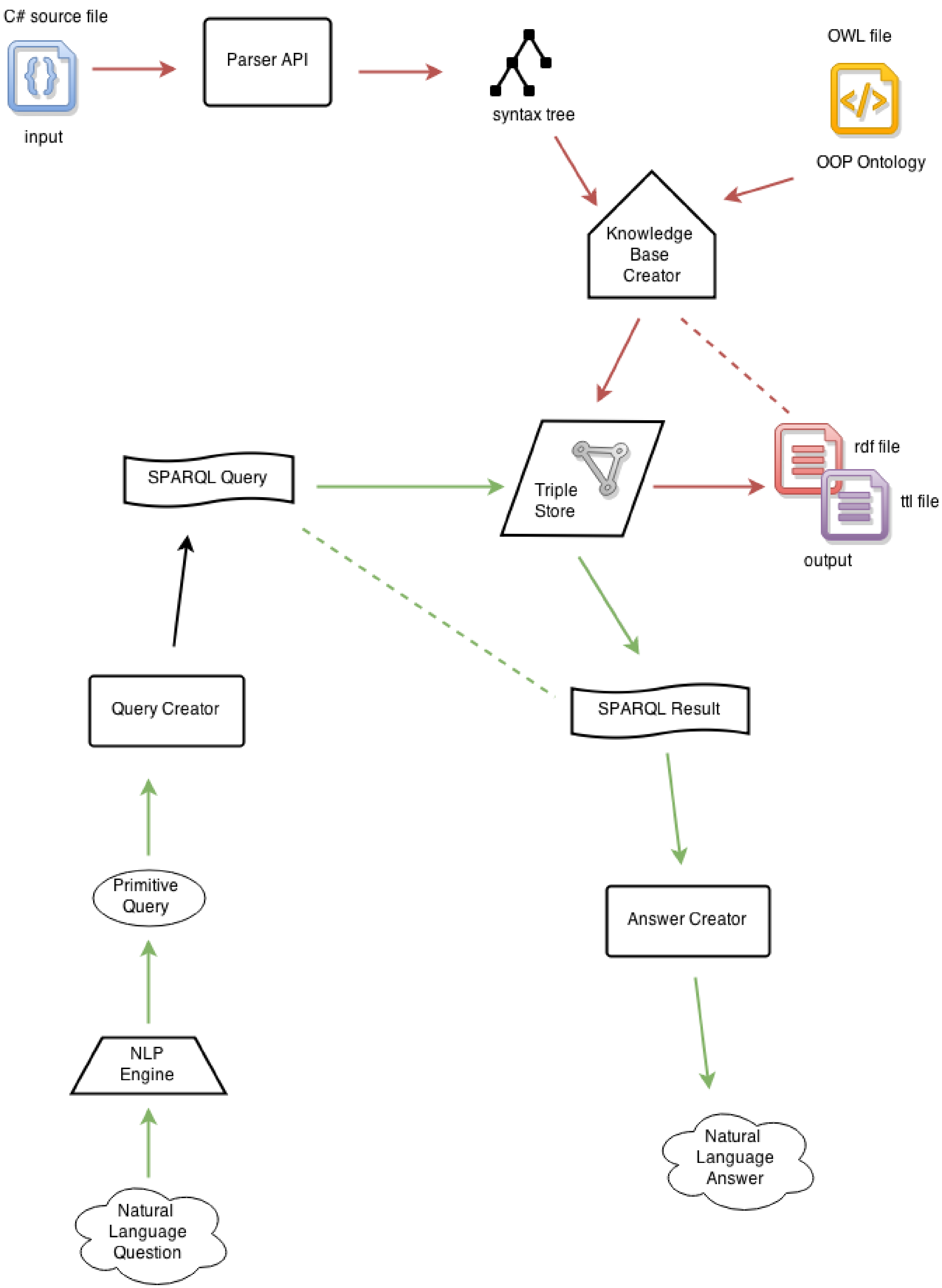
4. *Ion Smeureanu, Bogdan Iancu - Source Code Plagiarism Detection Method Using Protégé Built Ontologies.* In this paper *the authors demonstrate how source code plagiarism could be detected with the help of an ontology which models software programs.*

System Overview

The proposed system 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.


C# source file
input
Parser API
syntax tree
OWL file
OOP Ontology
Knowledge Base Creator
Triple Store
rdf file
ttl file
output
SPARQL Query
Query Creator
Primitive Query
NLP Engine
Natural Language Question
SPARQL Result
Answer Creator
Natural Language Answer

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