CodeOntology: Querying Source Code in a Semantic Framework

Mattia Atzeni and Maurizio Atzori

University of Cagliari
Math/CS Department
Via Ospedale 72, 09124 Cagliari (CA), Italy
ma.atzeni12@studenti.unica.it
atzori@unica.it

Abstract. Code reuse, code querying and computer aided programming are some of the main research challenges in software engineering. Therefore, we have introduced *CodeOntology* as an approach to leverage recent advances in the Semantic Web area and the impressive amount of open source code freely available online, to provide a semantic view of software systems by extracting structured information from source code and by performing named entity disambiguation on the comments provided within the code, in order to link the corresponding entities to pertinent DBpedia resources. In this paper, we focus on the expressiveness of this framework by showing how CodeOntology can be used for static code analysis, semantic component search and code reuse.

Keywords: SPARQL, RDF, OWL, Ontology, Programming Languages

1 Introduction

Recent research in software engineering is focusing towards graph-based approaches to model software architecture and software process. For instance, in [1] and [2], software is modeled as a directed multigraph, thereby enabling the collection and maintenance of the architectural knowledge in respect to both software and software process. However, the Semantic Web technology stack already provides flexible and expressive standards to represent structured information in a format that is easy to query and automatically process. Therefore, in [3] we have introduced *CodeOntology*, as an attempt to provide a semantic view of software systems, by leveraging the Semantic Web technology stack. CodeOntology includes two main contributions: *(i)* an OWL 2 ontology modeling object-oriented code constructs and *(ii)* a parser which relies on Spoon [4] to serialize Java source code or bytecode into RDF triples, thereby creating a queryable RDF representation of source code. Furthermore, CodeOntology makes use of TagMe [5] to automatically add links to DBpedia [6] by disambiguating named entities found within the comments available in the source code.

Details about the implementation of the parser and the design of the ontology have been provided in [3]. In this paper, we focus on a practical demonstration about the level of expressiveness that can be achieved using CodeOntology. We dig into more details about the structure of the data sets generated by the parser and we present extended results and experiments by providing some simple SPARQL queries showing how CodeOntology can be used for static code analysis and component search and reuse.

2 Data Set from OpenJDK

The parser provided by CodeOntology is capable of analyzing the structure of Java projects to generate RDF triples. The input of this process can either be *(i)* a text file containing Java source code, *(ii)* the root directory of a Java project or *(iii)* a JAR file aggregating many Java class files. The output mainly contains triples about structural information common to all object-oriented programming languages, like class hierarchy or the RDF serialization of the underlying structure of each class. The parser has been successfully applied to the OpenJDK 8 source code[1], generating a data set which consists of about 2 million RDF triples. Figure 1 shows some information about the structure of this data set.

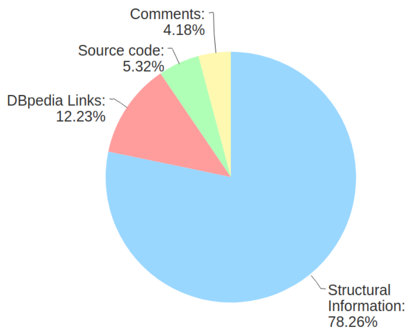

Fig. 1. Structure of the data set extracted from OpenJDK.

As we can see, actual source code as literals and literal comments are less than 10% of the total number of extracted RDF triples, while structural information about source code covers almost the 80% of the data set. This knowledge base is available at `https://doi.org/10.5281/zenodo.818116` and can be queried at `http://codeontology.org/sparql`.

The parser supports both Maven and Gradle projects. This allows to download the dependencies of the input project in the form of JAR files that can be optionally analyzed and serialized into RDF triples. The parser, along with a detailed tutorial about how to apply it on different kinds of Java projects, is available on GitHub[2].

[1] `http://openjdk.java.net/`
[2] `https://github.com/codeontology/parser`

3 Queries over Source Code

CodeOntology allows to leverage a powerful language like SPARQL to run highly expressive queries over source code. When a method m references another resource r, then the parser is able to serialize this information into a triple of the form: m *woc:references* r. We can use this property to easily select recursive methods by means of the following simple query.

```
SELECT ?method
WHERE {
  ?method a woc:Method ;
    woc:references+ ?method .
}
```

A more interesting example which makes use of the same property is given by the following query, which selects the classes that turn out to be the most referenced ones by the methods of the other classes.

```
SELECT ?class (COUNT(DISTINCT ?anotherClass) as ?count)
WHERE {
  ?class a woc:Class .
  ?method a woc:Method ;
    woc:isDeclaredBy ?anotherClass ;
    woc:references ?class .
  FILTER (?class != ?anotherClass)
}
GROUP BY ?class
ORDER BY DESC(?count)
```

Unsurprisingly, the most referenced class in OpenJDK is the `java.lang.String` class, followed by the classes `java.lang.Object` and `java.io.IOException`.

Another important use case of CodeOntology is undoubtedly the semantic retrieval of software components. For instance, we can exploit DBpedia links to select all methods computing the cube root of a parameter of type double.

```
SELECT ?method
WHERE {
  ?method a woc:Method ;
    woc:hasParameter/woc:hasType woc:Double ;
    dul:associatedWith dbpedia:Cube_root .
}
```

The execution of this query against the data set extracted from OpenJDK yields two methods, namely the method `cbrt(double)` declared by the class `java.lang.Math` and the method `cbrt(double)`, declared by the class `java.lang.StrictMath`. Another example is given by the following query, which selects all resources associated with public-key cryptography and, in particular, with RSA.

```
SELECT ?r
WHERE {
  ?r dul:associatedWith
    dbpedia:Public-key_cryptography ,
    dbpedia:RSA_\(cryptosystem\) .
}
```

CodeOntology allows also to run other interesting queries for different purposes, such as detecting the implementation of a specified design pattern or computing software metrics, like the well-known CK metrics [7]. Table 1 lists some more examples, which are available at `http://codeontology.org/examples`.

Sort classes by the number of subclasses
Select classes implementing the Singleton pattern
Select classes implementing the Builder pattern
Select classes implementing the Factory pattern
Select methods to read/write Zip files
Select methods to read an image at a specified URL

Table 1. Query examples available at `http://codeontology.org/examples`.

4 Conclusions and Demo Showcase

CodeOntology is an open community-shared resource which aims at enabling the RDF representation, from coarse to fine grain, of the structure and relations found within source code. This way, it is possible to precisely search specific software components using expressive SPARQL queries, some of which will be showcased during the demo. A video showing how to build and use the parser, as well as the execution of some query examples, is available at: `https://www.youtube.com/watch?v=bd6pvUDy8kA`.

Acknowledgments. This work was supported in part by a 2015 Google Faculty Research Award and Sardegna Ricerche (*project OKgraph*, CRP 120).

References

1. Dabrowski, Robert and Stencel, Krzysztof and Timoszuk, Grzegorz. In: Software Is a Directed Multigraph. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
2. Dabrowski, Robert and Stencel, Krzysztof and Timoszuk, Grzegorz: Software is a directed multigraph (and so is software process). CoRR **abs/1103.4056** (2011)
3. Atzeni, M., Atzori, M.: CodeOntology: RDF-ization of Source Code. In: The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, Springer International Publishing (2017)
4. Pawlak, R., Monperrus, M., Petitprez, N., Noguera, C., Seinturier, L.: Spoon: A library for implementing analyses and transformations of java source code. Software: Practice and Experience (2015)
5. Ferragina, P., Scaiella, U.: Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. CIKM '10, ACM (2010)
6. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal
7. Chidamber, S.R., Kemerer, C.F.: Towards a metrics suite for object oriented design. SIGPLAN Not. **26**(11) (November 1991) 197–211