Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Software trustworthiness 2.0—A semantic web enabled global source code analysis approach

Iman Keivanloo, Juergen Rilling*

Department of Computer Science, Concordia University, Montreal, Canada

ARTICLE INFO

Article history:
Received 11 October 2012
Received in revised form 29 June 2013
Accepted 27 August 2013
Available online 4 September 2013

Keywords:
Semantic Web
Source code analysis
Linked data
Global source code analysis

ABSTRACT

There has been an ongoing trend toward collaborative software development using open and shared source code published in large software repositories on the Internet. While traditional source code analysis techniques perform well in single project contexts, new types of source code analysis techniques are ermerging, which focus on global source code analysis challenges. In this article, we discuss how the Semantic Web, can become an enabling technology to provide a standardized, formal, and semantic rich representations for modeling and analyzing large global source code corpora. Furthermore, inference services and other services provided by Semantic Web technologies can be used to support a variety of core source code analysis techniques, such as semantic code search, call graph construction, and clone detection. In this paper, we introduce SeCold, the first publicly available online linked data source code dataset for software engineering researchers and practitioners. Along with its dataset, SeCold also provides some Semantic Web enabled core services to support the analysis of Internet-scale source code repositories. We illustrated through several examples how this linked data combined with Semantic Web technologies can be harvested for different source code analysis tasks to support software trustworthiness. For the case studies, we combine both our linked-data set and Semantic Web enabled source code analysis services with knowledge extracted from StackOverflow, a crowdsourcing website. These case studies, we demonstrate that our approach is not only capable of crawling, processing, and scaling to traditional types of structured data (e.g., source code), but also supports emerging non-structured data sources, such as crowdsourced information (e.g., StackOverflow.com) to support a global source code analysis context.

1. Introduction

Given the complex nature of software systems, there has been little progress made in providing more general and, reusable source code analysis services. As discussed in Binkley (2007), source code analysis techniques include two major parts: (1) an internal representation for the data (i.e., in-memory objects) and (2) an analysis algorithm (i.e., analysis knowledge). Due to limitations of existing source code analysis infrastructures (e.g., compilers, parsers, and modeling approaches) both, internal representations and analysis knowledge, are usually proprietary to a given analysis task/context by optimizing the analysis performance rather than focusing on reuse. There have been many applications for source code analysis in the software engineering domain, including structural search, pattern recovery, clone detection, and security concern analysis. However, their success has been mostly dependent on the ability to extract and represent the embedded semantics of source code. Many tools and techniques have been developed to address context specific source code analysis objectives (e.g., points-to analysis (Milanova et al., 2005; Hirzel et al., 2007), dependency analysis (Mancoridis et al., 1999; Robillard, 2008), flow analysis (Choi et al., 1993a), call graph construction (Rountev et al., 2004a), program slicing (Weiser, 1982a), and impact analysis (Ryder and Tip, 2001)). The level and scope of these approaches vary, ranging from those considering only the behavior of individual statements and declarations, to those that include the complete source code of a program. A common objective for these existing analysis techniques has been to analyze compilable source code of individual projects to derive specific analysis results that are as complete and precise as possible. In this paper, we focus our discussion on how Semantic Web technologies can become the foundation for both sharing data and analysis results as part of a global source code analysis infrastructure which might evolve and replace in the long-term existing code sharing and analysis techniques.

Semantic Web has already been widely accepted as a defacto standard approach in many domains for knowledge modeling, sharing and, integration (e.g., bioinformatics (Shadbolt et al., 2006;

* Corresponding author. Tel.: +1 514 848 2424.
E-mail addresses: i_keiv@cse.concordia.ca (I. Keivanloo), juergen.rilling@concordia.ca (J. Rilling).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.08.030

Lambrix, 2005)). However, it still lacks the same acceptance in the software engineering community, where its use has been mainly limited to the conceptualization of a domain of discourse (Knublauch, 2004). Similar to Web 2.0 promoting a collective intelligence that can be defined by harnessing the network effect to build applications that improve the more people use them. We do believe that source code analysis applications built on such open and decentralized networks can also lead to new concentrations of power and knowledge. In this research, we not only illustrate how Semantic Web technology can play an enabling technology role for modeling knowledge and results, but also can provide services to be harness by the community. In this paper we introduce our SeCold project (Keivanloo and Rilling, 2012), an open data software repository which provides (1) a model to share information (data) and (2) integrates analysis results (knowledge) in a global context and (3) provides some core analysis services which are scalable to large source code repositories. Through several case studies, we demonstrate how such a Semantic Web based global source code analysis approach can contribute toward improving the trustworthiness of software systems across project boundaries. Furthermore, the case studies not only illustrate that our Semantic Web-based approach (SeCold) is capable of crawling, processing and scaling to traditional types of structured data (e.g. source code) but also supports emerging data sources, such as crowdsourced information sources (e.g., StackOverflow.com). For the case studies, we combine an Internet-scale dataset (SeCold) with Semantic Web enabled global source code analysis services provided by our infrastructure to infer knowledge from crowdsourced information resources to support a more global approach to analyzing the trustworthiness of software systems. While the case studies illustrate the flexibility, scalability of our Semantic Web enabled approach they also illustrate, how our infrastructure can provide a first step toward a novel proactive approach to improve the trustworthiness of software systems.

The remainder of the paper is organized as follows. In Section 2, we introduce global source code analysis and some of its key properties. Section 3 introduces both research challenges and potential research contributions for the global source code analysis domain. Section 4 presents research background related to Semantic Web and Linked Data and introduces our SeCold project, a linked data repository which provides the foundation for core analysis services is explained in Section 5. Section 6 introduces some core services such as SE-Clone and SE-Search which, when combined with our linked data repository and Semantic Web technologies form the core for many other types of global source code analysis. Two case studies are presented in Section 7, illustrating how our semantic Web enabled approach can improve the trustworthiness of software systems. Section 8 concludes and summarizes our work and provides a discussion on our long-term vision on how Semantic Web can become an integrated part of the source code analysis and software engineering domains.

2. Background

2.1. Semantic web and linked data

The Semantic Web was introduced to address existing data ambiguity issues on the Web, by making Web content machine understandable (Shadbolt et al., 2006). The Semantic Web allows knowledge to be formally represented using logics such as Description Logic (DL). DL can describe a domain in terms of concepts (classes), roles (properties, relations) and individuals (Baader et al., 2003). Today, DL has become a cornerstone of the Semantic Web for describing ontologies, due to its decidability property. The Web Ontology Language (OWL) (OWL, n.d.) is one of the basic knowledge representation languages used for describing ontologies. Other major languages are RDF (RDF, n.d.-a) and RDFS (RDF, n.d.-b).

The Semantic Web community already provides a significant body of tools and techniques to create necessary infrastructures for dealing with inconsistent and incomplete data. There have also been significant advances in the persistent triple storage technology, improving their scalability, performance, and usage (e.g., (DBp, n.d.)). SPARQL, a widely accepted query language for triple stores (Kiefer et al., 2007), provides functionalities similar to SQL. Based on the formal definition of semantics using DL, Semantic Web reasoners (OWL, n.d.) can infer logical consequences from asserted statements.

Linked data (Berners Lee, n.d.), can be considered a by-product of the Semantic Web, was introduced to ease both data sharing and integration in distributed environments and is considered to be superior to XML-based approaches (Milanova et al., 2005; Würsch et al., 2010). Linked data main objective is on publishing structured data in RDF using URIs rather than focusing on ontological representations or inference. Linked Data best practices have led to the extension of the Web within a global data space allowing data from diverse domains, such as online communities, statistical and scientific data to be connected. Furthermore, Linked Data can be interpreted by humans and machines for mining, searching, and analysis purposes. A key requirement to enable the global sharing is that each entity in the domain of discourse must have its own unique identifier (UID) in the form of a URI (Uniform Resource Identifier). Note that we use the terms UID, URI, and URL interchangeably throughout the paper. Facts about the *resource* (i.e., entity) are represented using RDF *statements*, with each fact being a triple of subject, predicate, and object. To make this information inter-linkable and online, linked data mandates that URLs must be *dereferencable*. That is, clients (i.e., humans and machines) must be able to fetch resource related data via its URL (with the http://prefix). Using a HTTP header, a client specifies the desired output format: HTML or RDF/XML. A response must comprise the following: (1) *Description* and *backlinks* that contain all triples that have the URL as the subject and object. (2) Equivalent URLs that point to the same entity (i.e., `owl:sameAs`). Developers (i.e., third-party applications) can now readily benefit from the resulting linked datasets, which are based on a common data model, by supporting new types of applications.

2.2. Source code analysis

Source code is a set of instructions represented in a programming language (e.g., Java and C#) to control hardware. The instructions are usually written using a series of digits, letters, and special symbols. To meet different programming constraints (e.g., processing limitation and language complexity), several types of code analysis tasks are required (Scott, 2005) (Fig. 1) to extract meaning from high-level code and transform it to machine understandable instructions.

Lexical analysis is performed by a *scanner*, which accepts a sequence of characters and groups them into *tokens*. These tokens constitute the input for the *parser* and the syntactic analysis performed by it. Parsers use the programming language syntax to generate a parse tree as their output. A parse tree is then further analyzed by semantic analyzer, which applies the programming language semantics to produce and validate the program based on an abstract syntax tree (AST) and annotated abstract syntax tree (AAST). Figs. 3 and 4 show the corresponding AST and AAST for the following Java code fragment (Fig. 2). Depending on the language semantics, different types of analysis are performed during this semantic analysis, including for the example the assignment of defined variable types and the resolution of fully qualified name in the AST and AAST.