ANALYZING SOURCE CODE IDENTIFIERS FOR CODE REUSE

Ponnampalam Pirapuraj

158021B

Thesis submitted in partial fulfilment of the requirements for the

Degree of Master of Science (Part Time) (By Research)

Department of Computer Science & Engineering

TH 3497 + CD ROM

University of Moratuwa

Sri Lanka

October 2017

004 "17"
004 (043)

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement the material previously submitted for a Degree or Diploma in the other University or institute of higher learning and to the best of my knowledge and belief it does not contain the material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Candidate

..11-10-2017....... ....[illegible]..........

Date P.Pirapuraj

The above candidate has carried out research for the Masters thesis under my supervision.

Supervisor

11. 10. 2017 .... ***UOM Verified Signature***

Date Dr. Indika Perera

Acknowledgements

I would like to take this opportunity to acknowledge and extend my heartfelt gratitude to my supervisor Dr. Indika Perera for his long support and guidance throughout this project work, He has given me many stimulating suggestions and his valuable comments have helped me to solve most of the problems during the project development. Many thanks goes out to our MSc research degree coordinator Dr. A. Shehan Perera for his dedication and valuable comments and suggestion as a progress review panel board member. Thanks to another progress review panel board member Dr. Lochandaka Ranathunga, for his valuable suggestion and comments. Thanks to all the lecturers at the Faculty of Computer Science and Engineering, University of Moratuwa, for their valuable advice. Also thanks to my parents, my wife, and brothers, and friends to give support and motivation.

Abstract

Today a massive amount of source code is available on the Internet and open to serve as a means for code reuse. Developers can reduce the time cost and resource cost by reusing these external open source code in their own projects. Even though a number of Code Search Engines (CSE) are available, finding the most relevant source code is often challenging. In this research, we proposed a framework that can be used to overcome the problem faced by developers in code searching and reusing. The framework starts with the software architecture design in XML format (Class Diagram), extracts information from the XML file, and then based on the extracted information, fetches relevant projects using three types of crawler from GitHub, SourceForge, and GoogleCode. We will have a huge amount of projects by downloading process using the crawlers and need to find most relevant projects among them.

In this research, we particularly focus on projects developed using Java language. Each project will have a number of .java files, and all files will be represented as Abstract Syntax Trees (AST) to extract identifiers (class names, method names, and attributes name) and comments from the .java files. Then, on one hand, we will have the identifiers which are extracted from the XML file (Class diagram), and the other hand the identifiers and the action words (verbs) extracted from downloaded projects. Action words are extracted from comments using Part of Speech technique (POS). These two group of identifiers need to be analyzed for matching, if the identifiers are matched, an amount of marks will be given to these identifiers, likewise marks will be added together and then if the total marks is greater than 50%, the .java file belongs to these identifier will be suggested as relevant code. Otherwise, synonyms of the identifiers will be discovered using WordNet, and the matching process will be repeated for the synonyms. For the composite identifiers, camel case splitter is used to separate these words. If the programmers do not follow camel case naming convention, N-gram technique is used to separate these word. The Stanford Spellchecker is used to identify abbreviated words. Evaluation of our developed framework resulted in 95.25% of average accuracy of four subsystem [project downloader (100%), identifier analyzer (94%), word finder (87%), and comments analyzer (100%)] accuracy.

Keywords— Software Architecture, WordNet, N-gram technique, Part of Speech Tagging, Camel Splitter, and Abstract Syntax Tree.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CSE	Code Search Engine
AST	Abstract Syntax Tree
DTW	Dynamic Time Warping
API	Application Programming Interface
SVN	Subversion (Source code Management)
XML	Extensible Markup Language
DOM	Document Object Model
NLP	Natural Language Processing
TDCS	Test Driven code Searching
SME	Small and Medium Enterprise
AQE	Automatic Query Expansion
SVG	Scalable Vector Graphics
SSI	Structure Semantic Indexing
JSON	JavaScript Object Notation
MIS	Method Invocation Sequence
AST	Abstract Syntax Tree