UNIVERSITI PUTRA MALAYSIA

SOURCE CODE ANALYSIS EXTRACTIVE APPROACH TO GENERATE TEXTUAL SUMMARY

KAREEM ABBAS DAWOOD
GS45730

FSKTM 2017 3

SOURCE CODE ANALYSIS EXTRACTIVE APPROACH TO GENERATE TEXTUAL SUMMARY

By

KAREEM ABBAS DAWOOD

GS45730

Thesis submitted to the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, University Putra Malaysia, in Fulfilment of the Requirements for the Degree of Computer Science/Software Engineering

Jun, 2017

APPROVAL

This thesis report is submitted to the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, and has been accepted as partial fulfillment of the requirements for the Master's Degree of Computer Science/Software Engineering.

The members of the Examination Committee are as follows:

_____

Supervisor
Dr. KHAIRONI YATIM BIN SHARIF
Senior Lecturer/ Head of Software Engineering Research Group
Department of Software Engineering and Information System
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia

_____

Examiner
Dr. NORHAYATI BINTI MOHD ALI
Computer Science and Information Technology
Universiti Putra Malaysia

DECLARATION

I hereby declare that this thesis, submitted to the Department of Software Engineering and Information System as a fulfillment of the requirements for the master's degree in Computer Science/Software Engineering has not been previously, and is not concurrently, submitted for any other degree at university Putra Malaysia and other universities. I also certify that the work described here is entirely my own except for citations, which have been duly acknowledged.

_____

KAREEM ABBAS DAWOOD

GS45730

Date: Jun, 2017

DEDICATION

To soul of my mother, my wife, and my family

ACKNOWLEDGEMENTS

In the name of Allah the most Beneficent and the most Merciful, first and foremost, I would like to express gratitude to Allah Almighty, for endowing me the opportunity, patience and guidance to complete this thesis successfully.

Secondly, I would like to express deep gratitude to my supervisor, Dr. Khaironi Yatim Bin Sharif, who has been my greatest guidance and support during this thesis. With his experience and advice I have been achieved the goals of this research work. I would also express thanks to postgraduate coordinator Dr.Norhayati Binti Mohammad Ali for giving the guidance and invaluable help.

I am also profoundly grateful to my colleagues and friends for their motivation and moral support.

Finally, I would like to thank my family for their continuous supports throughout the process. Their concerns have been the greatest inspiration for me to sail through the phases of stress in doing this thesis.

ABSTRACT

Nowadays, obtain program features becomes a hot issue in source code comprehension. A large amount of efforts spent on source code understanding and comprehension to develop or maintain it. As a matter of fact, developers need a solution to rapidly detect which program functional need to revise. Hence, many studies in this field are concentrating on text mining techniques to take out the data by source code analysis and generate a code summary. However, in this thesis, we attempt to overcome this problem by propose a novel approach (Abstract Syntax Tree with predefined natural language text Template (AST-W-PDT)) to generates human readable summaries for Java methods role. The thesis describes how we developed a tool that the java source code can be summarized from the methods role. In evaluating our approach, we found that the automatically generated summary from a java class 1) is helpful to the developers in order to understand the role of the methods and will be useful, and 2) the automatically generated summary is precise.

ABSTRAK

Pada masa kini, mendapatkan ciri program menjadi isu panas dalam pemahaman kod sumber. Banyak usaha yang dibelanjakan untuk pemahaman dan pemahaman kod sumber untuk membangun atau memeliharanya. Sebagai hakikatnya, pemaju memerlukan penyelesaian untuk mengesan dengan pantas mana program berfungsi untuk mengkaji semula. Oleh itu, banyak kajian dalam bidang ini menumpukan pada teknik perlombongan teks untuk mengambil data dengan analisis kod sumber dan menghasilkan ringkasan kod. Walau bagaimanapun, dalam tesis ini, kami cuba mengatasi masalah ini dengan mencadangkan pendekatan baru (Abstrak Syntax Tree dengan Templat teks bahasa asal yang telah ditetapkan) (AST-W-PDT) untuk menghasilkan ringkasan manusia yang dapat dibaca untuk peranan kaedah Java. Tesis ini menerangkan bagaimana kita membangunkan alat bahawa kod sumber java dapat diringkaskan dari peranan kaedah. Dalam menilai pendekatan kami, kami mendapati ringkasan yang dihasilkan secara automatik dari kelas java 1) membantu pemaju untuk memahami peranan kaedah dan akan berguna, dan 2) ringkasan yang dihasilkan secara automatik adalah tepat.

TABLE OF CONTENTS

LIST OF TABLES

Table No. **Page**

LIST OF FIGURES

Page

GLOSSARY OF TERMS

AST-W-PDT	Abstract Syntax Tree with Predefined natural language Template
AST	Abstract Syntax Tree
TF-IDF	Term Frequency-Inverse Document Frequency
SWUM	Software Word Usage Model
JDT	Java Development Kit
API	Application Programming Interface

CHAPTER I

INTRODUCTION

1.1 BACKGROUND

Software engineering is the process of analyzing software systems in order to improve the efficiency [1] . This process can be explained as supplying recommendation, illustration and providing reports for enhancing the performance of a particular system. To do so, a comprehensive analysis should be concentrated on the significant features shown in the source code of the system [2]. Analyzing these features within the code provides valuable understanding of the intention of the code which facilitate the process of re-use and modification that would be performed on such code.

Developer spends a large proportion of their time reading and navigating source code in order to comprehend it. However, studies of program comprehension consistently find that developers would prefer to focus on small sections of code during software maintenance, and try to avoid comprehending the entire system [3] . The result is that developers skim source code, for example by reading only method signatures or important keywords, to save time. Skimming is valuable because it helps programmers quickly understand the underlying code, but the drawback is that the knowledge gained cannot easily be made available to other programmers. An alternative to skimming code is to read a summary of the code. A summary consists of a few keywords, or a brief sentence, that highlight the most important functionality of the code[3]. Hence, developers need software documentation.

However, documentation is expensive to produce and maintain, and often becomes outdated over time. Developers often lack the time and resources to write documentation. Therefore, automated solutions are desirable [4]. One solution is to use simple textual descriptions of source code entities that developers can grasp easily, while capturing the code semantics precisely[5] . As a consequence, developers can review software systems quickly and decide which entities to analyze and modify. A few works already proposed to generate code summaries by adapting text summarization techniques[6][7].

Recent research has made inroads towards automatic generation of text summarization in natural language descriptions of software[8][9][10][11][12]. In particular, work by Sridhara (2010) can form natural language summaries of Java methods[11]. Then summaries can then be aggregated to create the software's documentation. Although these techniques are already enough to provide good summaries by finding suitable keywords (lexical information), they may present some limitations related to support method role. This role is not considered by adapting existing text summarization techniques to source code, if do so we thought that the summary will be more readable, understandable, and accurate.

In fact, these techniques focus mainly on lexical dimension (e.g., Latent Semantic Indexing [13], term frequency-inverse document frequency(tf-idf) [8], Vector Space Mode [8], etc.) to detect relevant terms. One strategy was using Software Word Usage

Model (SWUM) for text generation is to define templates of natural language sentences, and use the output from SWUM to fill these templates [7].

In this research a novel approach is proposed for automatic summary generation taking into consideration both lexical and methods role information. We hypothesize that existing summary generators would be more effective if they included information from the data within the methods that describe method role (like method signature, variables, invocation, and method return value). We define "more effective" in term of programmers find the generated summaries to be more helpful to convey the most important aspects of its intended functions.

Our novel approach works by collecting data from the methods (like method signature, variables, invocation, and method return value), and then using these data with predefined natural language template to describe the role of methods. We use Abstract Syntax Tree (AST), to identify and extract the data that we need to include it in source code summaries. However, there are multiple differences between the regular text and the source code. In the source code, the multi-word identifiers are written without a space between them, instead several strategies can be used. First, it may be divided using special characters such as 'Employee-Name' or 'Employee_Name' [14]. Second, it may be written using 'CamelCase' approach, this approach aims to capitalize the first letter of the first words and the first letter of the second word without spacing (e.g. EmployeeName) [15]. In this study the camel case and underscore case was used to

manipulate the splitting the identifier into single words. Our system then generates a readable English description of role for each method in a Java source code.

To test our hypothesis, we introduce a novel approach to automatically generate source code summary that includes methods role. Then we perform a case study to evaluate the source code summaries generated by our approach with perspective of novice developers who is expected to perform some specific maintenance tasks. Specifically, we contribute the following:

- A novel approach is introduced for generating source code summaries, taking into consideration both identifiers and method role information. Our approach is different from previous approaches in that we summarize the role of methods as readable English text by using Abstract Syntax Tree Parser (ASTPareser) in addition to predefined natural language template.
- A complete implementation of our novel approach for Java methods.

1.2 SIGNIFICANCE OF STUDY

With the dramatic evolution of software engineering, tremendous amount of software nowadays is being modified, changed and improved chronically. This continuous changing requires understandable developer who can treat the source code. The developer should know what the source code is intended to do by each included function. However, dealing with a large-scale source code would significantly hinder the process of modification by the user. Therefore, developer will tend to search manually

on the desired portions that wanted to be modified. Meanwhile, the manual searching would be tedious and time consuming especially when there are thousands of lines. Therefore, Source Code Analysis Extractive Approach to Generate Summarization has been proposed and implemented for this purpose in order to facilitate the process of extracting code summary.

1.3 PROBLEM STATEMENT

Recently, information retrieval has been applied on software engineering applications in order to enhance the productivity of systems. This can be shown by allowing the developer to search within the source code for specific portions. Program comprehension is essential for code maintenance and evolution activities. It saves time and efforts of developers who want to perform any code changes [11]. Programmers need software documentation. However, documentation is expensive to produce and maintain, and often becomes outdated over time. Programmers often lack the time and resources to write documentation. Therefore, automated solutions are desirable [2]. Textual summaries for source code provide great help to code understanding activities. Such process depends mainly on the natural language processing techniques by utilizing the syntax and semantic of the words. The most important semantics that have been addressed is the identifiers. Such identifiers can provide a big picture of the whole source code. This can facilitate the process of code summary. The problem under consideration in this research is how to support program understanding efforts for the source code.

1.4 RESEARCH OBJECTIVES

The research objectives of this study are illustrated as follows:

- To design and implement source code summarizer tool that it conveys the most important aspects of its intended function.
- To evaluate the efficiency of the approach for generating source code summary from the perspective of novice developers who is expected to perform some specific maintenance tasks.

1.5 RESEARCH APPROACHE

Table 1.1: shows the link between the research problems, research objectives, and research approaches. There are three main approaches that are used in this research as the following:

- Critical data analysis using literature review: through the revision of the literature related to our research, the main idea is the source code summarization and comprehension. Also, the most suitable methods of source code summarization could be analyzed in order to make the summary more readable, understandable, and not missing essential information.
- Quantitative data collection using a questionnaire: via this approach, the primary data is collected, where a Likert scale is used to determine the answers. The aim of this approach is to evaluate the efficiency of the approach for generating code summary from the perspective of novice developers who is expected to perform some specific maintenance tasks. The quantitative data collected from 15

postgraduate students from the Faculty of Computer Science and Information Technology at University Putra Malaysia (UPM).

- Quantitative empirical evaluation: this approach is used in order to evaluating the quality of generated summary.

Table 1.1: Linking between research directions

Research Problem	Research Objectives	Objectives Activities	Approach
The problem under consideration in this research is how to support program understanding efforts for the source code. Automated solutions are desirable. And Textual summaries for source code provide great help to code understanding activities. Such process depends mainly on the natural language processing techniques by utilizing the syntax and semantic of the words.	1-**To** design and implement source code summarizer tool that it conveys the most important aspects of its intended function. 2-**To** evaluate the efficiency of the approach for generating code summary from the perspective of novice developers who is expected to perform some specific maintenance tasks.	<u>Phase 1:</u> To conducts a comprehensive literature review for the code summarization by identifying the problem, tools and techniques used for this problem.	Literature Review
		<u>Phase 2:</u> To implement an effective source code summarizer tool that it conveys the most important aspects of its intended function.	Implementation and Collect the Data
		<u>Phase 3:</u> To evaluate the efficiency of the approach for generating code summary from the perspective of novice developers.	Quantitative Evaluation

1.6 RESEARCH SCOPE

This study aims to propose a source code analysis extractive approach to generate summarization. The type of analysis used to focus on entities and sub-entities, mainly the role of methods through signature and invocation. The source code of each method is analyzed to extract a textual summary about its role. The summary of a method will generate from its contents. The contents include method name, parameters, local variables, return value types, and methods' invocations.

1.7 EXPECTED RESULTS

The expected results of this study are:

1. Automated summarizer that helps developers in term of reducing the effort and time spend to reading and navigating source code in order to comprehend it.
2. Evaluate and verify the finding by comparing the generated summaries to summaries written manually by experts. And /or a group of programmers judge the generated summaries and determine if it is readable, understandable ,and not missing essential information [7][8].

1.8 THESIS ORGANIZATION

This thesis composed of five chapters that are being described as follows:

Chapter 1 provides the headlines of this research where the background of the study, problem statement, objectives, scope, and expected result are being discussed.

Chapter 2 conducts a comprehensive literature review for the code summarization by identifying the problem, tools and techniques used for this problem. In addition, this chapter aims to concentrate on the method role.

Chapter 3 discusses the implementation of the proposed approach (Research Methodology) by discussing how the ASTParser with predefined natural language template (AST-W-PDT) is used to present novel approach that use java method to generate the summary, this approach is differ from the other by summarizing the role of each method in the provided java code.

Chapter 4 discusses experiment setting in which the mechanism of attaining the results be identified; and analyzes the experimental results that have been obtained by the proposed method. Consequentially, the results are analyzed technically in terms of the effectiveness.

Chapter 5 provides the final conclusion of the research in which a summary of the whole thesis is being described. In addition, an emphasis of the research contribution is also provided. Finally, the future directions that could be inspired by this research is described.

1.9 SUMMARY

This chapter has provided the outline of the research in which the background of the study, problem statement, research objectives, and research architecture are being described properly. Next chapter will discuss the literature review by describing the related work and their techniques in more details.

REFERENCES

[1] B. Du Bois, “Towards an Ontology of Factors Influencing Reverse Engineering,” 2005.

[2] S. W. Thomas, “Mining software repositories using topic models,” *Proceedings of the 33rd International Conference on Software Engineering*. pp. 1138–1139, 2011.

[3] P. Rodeghero, C. Liu, P. W. McBurney, and C. McMillan, “An Eye-Tracking Study of Java Programmers and Application to Source Code Summarization,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 11, pp. 1038–1054, 2015.

[4] P. W. McBurney, “Automatic Documentation Generation via Source Code Summarization,” *Proc. - Int. Conf. Softw. Eng.*, vol. 2, pp. 903–906, 2015.

[5] S. Haiduc, J. Aponte, and A. Marcus, “Supporting program comprehension with source code summarization,” *2010 {ACM}/{IEEE} 32nd {International} {Conference} {Software} {Engineering}*, vol. 2, no. May 2016, pp. 223–226, 2010.

[6] C. Science and M. Studies, “J-Summarizer,” vol. 7782, pp. 59–62, 2016.

[7] P. W. McBurney and C. McMillan, “Automatic Source Code Summarization of Context for Java Methods,” *IEEE Trans. Softw. Eng.*, vol. 42, no. 2, pp. 103–119, 2016.

[8] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, “Automatic generation of natural language summaries for Java classes,” *IEEE Int. Conf. Progr. Compr.*, pp. 23–32, 2013.

[9] H. Burden, “Natural Language Generation from Class Diagrams Categories and Subject Descriptors.”

[10] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, “AUSUM: Approach for Unsupervised Bug Report Summarization,” *Proc. ACM SIGSOFT 20th Int. Symp. Found. Softw. Eng. (FSE ’12)*, p. 11:1-11:11, 2012.

[11] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, “Towards automatically generating summary comments for Java methods,” *Proc.*

IEEE/ACM Int. Conf. Autom. Softw. Eng. - ASE '10, p. 43, 2010.

[12] G. Sridhara, L. Pollock, and K. Vijay-Shanker, "Generating parameter comments and integrating with method summaries," *IEEE Int. Conf. Progr. Compr.*, pp. 71–80, 2011.

[13] Y. Liu, X. Sun, X. Liu, and Y. Li, "Supporting program comprehension with program summarization," *2014 IEEE/ACIS 13th Int. Conf. Comput. Inf. Sci. ICIS 2014 - Proc.*, pp. 363–368, 2014.

[14] E. Enslen, E. Hill, and L. Pollock, "Mining Source Code to Automatically Split Identifiers for Software Analysis ∗," pp. 71–80, 2009.

[15] D. Lawrie and D. Binkley, "Expanding Identifiers to Normalize Source Code Vocabulary," 2011.

[16] Chitti babu K, Kavitha C., and SankarRam N, "Entity based source code summarization (EBSCS)," *2016 3rd Int. Conf. Adv. Comput. Commun. Syst.*, pp. 1–5, 2016.

[17] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, "An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks," *IEEE Trans. Softw. Eng.*, vol. 32, no. 12, pp. 971–987, 2006.

[18] D. Latoza, Thomas, G. Venolia, and R. Deline, "Maintaining mental models: a study of developer work habits," *Proc. 28th Int. Conf. Softw. Eng.*, pp. 492–501, 2006.

[19] J. Starke, C. Luce, and J. Sillito, "Searching and skimming: An exploratory study," *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 157–166, 2009.

[20] I. Mani, M. T. Maybury, and M. Sanderson, "Advances in Automatic Text Summarization," *Comput. Linguist.*, vol. 26, no. 2, pp. 280–281, 1999.

[21] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: A taxonomy and survey," *J. Softw. Evol. Process*, vol. 25, no. 1, pp. 53–95, 2013.

[22] K. Chen and V. Rajlich, "Case Study of Feature Location Using Dependency Graph," pp. 241–249, 2000.

[23] R. Koschke and J. Quante, "On dynamic feature location," *Proc. 20th IEEE/ACM*

Int. Conf. Autom. Softw. Eng. SE - ASE '05, pp. 86–95, 2005.

[24] D. Poshyvanyk, Y.-G. Gueheneuc, A. Marcus, G. Antoniol, and V. Rajlich, "of Methods Based on Execution Scenarios and Information Retrieval," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 420–432, 2007.

[25] G. Gay, S. Haiduc, A. Marcus, and T. Menzies, "On the use of relevance feedback in IR-based concept location," *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 351–360, 2009.

[26] A. Marcus, A. Sergeyev, V. Rajlich, and J. I. Maletic, "An Information Retrieval Approach to Concept Location in Source Code," *Proc. 11th Work. Conf. Reverse Eng. (WCRE '04)*, pp. 214–223, 2004.

[27] E. Hill, L. Pollock, and K. Vijay-Shanker, "Automatically capturing source code context of NL-queries for software maintenance and reuse," *Proc. - Int. Conf. Softw. Eng.*, pp. 232–242, 2009.

[28] D. Shepherd, Z. P. Fry, E. Hill, L. Pollock, and K. Vijay-Shanker, "Using natural language program analysis to locate and understand action-oriented concerns," *Proc. 6th Int. Conf. Asp. Softw. Dev.*, pp. 212–224, 2007.

[29] U. Hahn and I. Mani, "of Automatic Researchers are investigating summarization tools and methods that," *Comput. 33.11*, no. November, pp. 29–36, 2000.

[30] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, "On the use of automated text summarization techniques for summarizing source code," *Proc. - Work. Conf. Reverse Eng. WCRE*, pp. 35–44, 2010.

[31] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," *Proc. 36th Int. Conf. Softw. Eng.*, pp. 390–401, 2014.

[32] R. P. L. Buse and W. R. Weimer, "Automatic documentation inference for exceptions," *Proc. 2008 Int. Symp. Softw. Test. Anal. (ISSTA '08)*, p. 273, 2008.

[33] E. Wong, "Mining Question and Answer Sites for Automatic Comment Generation by," pp. 562–567, 2014.

[34] S. Rastkar, G. C. Murphy, and A. W. J. Bradley, "Generating natural language summaries for crosscutting source code concerns," *IEEE Int. Conf. Softw.*

Maintenance, ICSM, no. Section II, pp. 103–112, 2011.

[35] C. Ludwig, “Text Retrieval,” vol. 24, no. 5, pp. 1–21, 2007.

[36] D. Lawrie and D. Binkley, “Expanding identifiers to normalize source code vocabulary,” *IEEE Int. Conf. Softw. Maintenance, ICSM*, pp. 113–122, 2011.

[37] N. Dragan, M. L. Collard, and J. I. Maletic, “Automatic identification of class stereotypes,” *IEEE Int. Conf. Softw. Maintenance, ICSM*, 2010.

[38] “Abstract Syntax Tree.” [Online]. Available: http://www.eclipse.org/articles/Article-JavaCodeManipulation_AST/. [Accessed: 09-Apr-2017].

[39] D. Freitag, “Machine Learning for Information Extraction in Informal Domains,” *Mach. Learn.*, vol. 39, no. 2/3, pp. 169–202, 2000.

[40] K. Spärck Jones, “Automatic summarising: The state of the art,” *Inf. Process. Manag.*, vol. 43, no. 6, pp. 1449–1481, 2007.