A Survey on Research of Code Comment Auto Generation

Lele Zhao\(^1\), Liping Zhang\(^1\) and Sheng Yan\(^1\)

\(^1\) College of Computer Science and Technology, Inner Mongolia Normal University, Hohhot, Inner Mongolia, 010022, China

*Corresponding author’s e-mail: zhaolele9@163.com

Abstract. A code comment is an explanation of the code. As an important part of software development process, code comment plays an important role in software maintenance, software reuse and other fields. Automatic generation of code comment aims to study the automatic method of comment generation so as to improve the efficiency of software development. This paper summarizes the related concepts and researches of code comment, discusses the automatic generation technology of code comment and the evaluation of comment quality, introduces the application of code comment generation in the software domain, and finally analyzes the challenges of automatic generation of code comment, and looks forward to the future research in this field.

1. Introduction

With the continuous expansion of software code size, how to help developers understand, write or maintain code in the process of software development has become an important task in the field of software engineering. Code comment is a brief readable natural language description of source code, which plays an important role in software reuse and software maintenance. Software reuse can improve the efficiency and quality of software development [1]. In the process of software reuse, multiplexers need to understand the functions of software codes, but many software lack documents that can describe the functions of software [2]. During software maintenance, developers often cannot understand all the code for the system, and when tasks change, they must explore the code in order to locate the parts associated with the change task. In the process of reading and exploring the code, developers often prefer to skim the code only by reading method signatures or comment to reduce the workload. By reading code comment, developers can review historical code implementation ideas or learn implementation details of other open source projects, or speed up the learning process when learning a new development language.

While code comments are popular in the software development process, they are time consuming and laborious to write, and often face problems of loss, inaccuracy, or outdated, and developers must spend more time reading code in detail to gain an understanding of the code as a whole. These difficulties in creating comments led researchers to try to design an automated comment generation method instead of writing comments manually. Automatic generation of code comment is of great help to software developers and maintainers. Therefore, it is of great significance to study the method of automatic generation of code comment to reduce the difficulty and cost of software maintenance and software reuse.
2. Related concept

2.1. Definitions
Code comment is the explanation or hint of a statement, program segment, function, etc. [3], which is the explanation of the code, and can improve the readability of the program code. Code comment describes the logic and function behind the code in the form of natural language, which is a communication bridge between the programmer and the reader.

2.2. Types
Code comment can be classified in several ways [4]. One category is divided into prologue and functional comment according to their content. The main contents of prologue comment include module interface, data description and module function. The main contents of functional comment include function of program segment, function of statement and state of data. The other category is classified into document comment, block comment, and line comment.

2.3. General framework for code comment generation
In recent years, researchers have carried out research on the automatic generation of code comment from various aspects, namely, automatic generation of natural language description of a given code fragment [5]. The general framework of automation method includes comment reuse and comment extraction. The comment reuse technology matches the code to be commented from the existing code fragments and obtains the comments therein, and uses optimization technology to output the original comments as target comments, as shown in figure 1. Comment extraction technology extracts the structure and semantic information of the code, and then constructs the model, combining the natural language processing technology with the grammatical structure to combine these words into statement and output them, as shown in figure 2.

![Figure 1. General framework for code comment reuse.](image1)

![Figure 2. General framework for code comment extraction.](image2)

2.4. Evaluation index
Different comment generation methods need to be customized and improved according to the size, structure and programming language of the code. In order to verify the effect of comment generation model, the following evaluation indexes can be used as a reference and guidance for continuous improvement of methods:

- **Accuracy**: Whether the code comments accurately represents the operation of the code, the important information of the code should be reflected in the comments.
- **Adequacy**: Evaluate whether the comments contain the core functions implemented by the code and fully cover the information elements.
- **Conciseness**: Concise comments are more readable and understandable than wordy comments, and conciseness evaluates whether the comments are clearly expressed, logically coherent and thematic.
3. Techniques for automatic generation of code comment

These techniques can be broadly divided into several strategies: rule-based, data-based, deep learning based, topic model based, machine translation based, and other approaches.

3.1. Rule-based method

The rule-based Method generates accurate comments by selecting details that are closely related to the source code structure and then formulating rules to closely match the given source code.

Sridhara et al. propose a technique for automatically generating comments for Java methods [6], which selects statements from the signature and body of a given method, places keywords into predefined templates, and converts them into readable natural language comments. In order to generate smoother and more concise method comments, Sridhara et al. propose another method [7], which aims to identify which statement blocks constitute a high-level operation, automatically identify code fragments in the method to realize high-level abstract operation. Haiduc et al. believe that skimming code and reading code are too extreme. Although skimming is fast, it will lead to misunderstanding, while detailed reading code is time-consuming [8]. They describe a method that combines word location information with text retrieval technology based on automatic text summary to automatically generate code summary. Moreno L et al., based on the work of Sridhara et al. [6], propose a technology that can automatically generate natural language comments for Java classes [9]. Using the information of class and method prototypes, combined with the lexicon tool, heuristic selection of information to be included in the comments was used to generate the comments. The focus of this method is on what the class covers and what it does. Wang et al. propose a code summary method based on Syntactic Analysis -- SA technology [10]. This method uses part of speech tagging to identify the noun set in the code, and then corrects the errors that may be introduced in the stage of part of speech tagging through block analysis, and then selects some words with the highest weight from them to generate the code summary after noise reduction.

These rules-based methods of synthesizing natural language sentences directly from code elements were the first successful methods of automatic comment generation. However, these methods lack portability and flexibility and can only generate comments for specific code structures. If new rules appear in the source code that you have never seen before, you have to manually update the rules and sentence templates, which is a manual effort.

3.2. Data-based method

Wong et al. propose a method to automatically generate comment by mining data from StackOverflow, a large Q&A website [11]. This method takes the code and the corresponding natural language description from StackOverflow, matches it with similar code snippets, and uses natural language processing to refine the description and output it as a comment. But it is weak in generalization and can only generate a limited number of comments, because if it is code that is not discussed in the Q&A site, the method cannot generate comments for code snippets, resulting in limited production. Therefore, the research team propose an improved method to automatically generate code comment by analyzing existing software repositories [12]. Firstly, the similar code segments between two open software repositories are identified by cloning detection technology, and the existing code comments are used to describe the target code and the related description statements are processed by natural language processing technology to generate comment.

Data-driven technologies are more flexible than rule-based technologies, reducing the labor involved in creating templates. If you want to improve the accuracy of your system, you just need to increase the amount of "training data" used to build it. However, this method largely retrieves existing comment. If the comment describing the existing code does not exist in the training data, it cannot generate an accurate comment and thus faces the problem of "data sparsity".
3.3. Deep learning-based method
Deep learning is an end-to-end method that automatically learns the features contained in the program code based on the existing programs and their related data.

Ying et al. propose a feasibility study of a method for generating code fragment summaries [13]. Using a supervised machine learning method, extract significant lines of code and summarize code snippets to generate summaries based on whether or not lines of code should be categorized in summaries. Iyer et al. propose an attention-based recursive neural network model called CODENN, which generates descriptive summaries for C# fragments and SQL query statements [14]. This method extracts the source code fragment from StackOverflow and predicts code comment by processing the code as plain text. Allamanis et al. use attention-based convolutional neural network to generate short summaries of similar names by combining words in the source code [15]. It predicts the name of the subroutine from the body of the subroutine, and these summaries average three words, and shorter comment may not express the functionality of a given snippet. Based on the program comment model based on seq2seq [16], Hu et al. further integrate the API sequence called by the code and input the knowledge contained in the API invocation sequence into the network model to assist the generation of comment.

The method based on deep learning achieves good results, but how to choose the appropriate neural network and tune is a challenge for researchers, and the structural information which is important to the programming language is easy to be ignored in the model training.

3.4. Topic model-based method
Some work at other levels of abstraction, such as a topic model, which is a statistical model in which words are related to other words based on their co-occurrence in the document.

McBurney et al. used the topic model to select keywords and themes as abstracts of source code [17]. They organize the topics in the source code into a hierarchy, based on the programmer's mental process of reading the code first to perform the highest level of functionality, and then to read how the lower level of functionality supports the higher level of functionality. Li W P et al. propose an automatic generation method of code topic summary [2]. LDA topic model is used to mine the theme describing the function of code. LexRank algorithm is used to extract the text content of theme-related document to generate a short topic summary.

These topic model-based methods provide a new perspective on the automatic generation of code comments, but the topic model treats documents as a set of words and ignores structural information in the code, such as program language syntax, functions, or method calls [18], resulting in comments that are only words, not phrases or sentences.

3.5. Machine translation-based method
Some researchers believe that the automatic code generation comment is essentially a translation task from source code to natural language description.

Hu et al. [19] were the first to use the machine translation model to solve the source code summary task, proving that the source code summary task is almost like machine translation. Zheng et al. use seq2seq neural machine translation model based on attention mechanism to generate code comments [20]. An attention module named code attention is proposed, which can convert code into comments by taking advantage of the domain features of code fragments, such as symbols and identifiers. Xu et al. propose a structure-aware dual-encoder model [21], which combines sequence encoder and tree encoder, fuses code abstract syntax tree information, and converts source code into accurate and readable natural language comments.

Machine translation is the process of converting one natural language (source language) into another (target language) by using a computer [22]. Machine translation-based method take code as source language and comment as target language, and use machine translation model to realize comment generation. These methods have achieved good results based on large data input.
4. Code comment quality evaluation

Although there are many methods that provide a variety of comment automatically generate tools, but the quality of these tools to generate comments tend to be restricted by technology and researchers of subjective factors and uneven, assess the quality of code comments, including understanding what type of information is important, what information is irrelevant or even misleading, in order to avoid this kind of information contained in the tool.

Khamis et al. propose an effective and automated method to evaluate the quality of inline documents [23]. Using a set of heuristics aimed at language quality and consistency between source code and its comments, similar comments in text are considered of higher quality. Steidl et al. propose an automated quality evaluation method based on four factors: consistency, usefulness, completeness and relevance [24]. The method estimates these factors by calculating the textual similarity between the comments and the code and the length of each comment. Gao et al. propose a quality evaluation framework that comprehensively consider objective quality attributes and subjective quality attributes, and conduct a comprehensive quality evaluation of source code analysis comments [25]. Source code analysis comment is an important output of software analysis task.

5. Code comment related research

In order to realize automatic code comment generation, researchers have studied code comment from various angles, providing support and guidance for automatic comment generation.

5.1. Code comment classification

Code comments enhance the readability of the code, but not all comments have the same target and target audience, so categorizing code comments can help you understand the purpose and habits of developers writing comments during development, and thus provide guidance for the automatic generation of code comments.

Pascarella et al. study how six different Java OSS projects classify source code comments using code comments [26]. Haouari et al. investigate comments habits of developers, study comments from both quantitative and qualitative perspectives, and propose a comment classification method based on comments analysis [27]. Steidl et al. propose a semi-automated method for quantitative and qualitative evaluation of comment quality by classifying comment in seven high-level categories [28]. It shows how to evaluate the quality aspects of the model by providing metrics that fit a particular category.

5.2. Consistency between code and comment

With the continuous evolution of software systems, code comments will gradually lose their consistency with the code. Many researchers focus on the issue of whether source code and related comments will change along with the evolution history of software systems to provide guidance for the design of comment generation tools.

Paul W. M et al. propose an empirical study to test source code summaries prepared by authors, readers and automatic source code summarization tools [29]. The text similarity between source code and source code summaries was tested using the Short Text Semantic Similarity (STSS) metrics. Fluri et al. describe a way to map code and comments to observe their co-evolution over multiple versions [30]. They study three open source systems, describing how comments and code have co-evolved over time. Tan et al. propose a method called @tcomment to test Javadoc comments [31], especially method attributes regarding null values and associated exceptions.

5.3. Other research

There are other attempts at code comment generation. Rodeghero et al. believe that the quality of code comment depends on the keywords selected from the code, so they improve the algorithm of keyword extraction through the eye tracking research on programmers [32]. In addition, there are also researches on the decision support method of code comment. Huang et al propose the decision support method of code comment [33]. This method extracts code semantic and structural features (abstract
6. Conclusion
The application fields of code comment generation include software maintenance, software reuse, code search, etc. Software maintenance refers to changes according to the requirements or hardware environment for the application of some or all of the modified [5], by reading the code comments, the maintenance personnel can reduce the time spent reading code. Software reuse is refers to the reuse of existing software products, including all kinds of documents and code, reduce the cost of software development and maintenance [5]. Before code reuse, if developers can use comment to read the code, will be of great help for the developers work. High-quality code comments can also be used for code search. Code search is queries entered by users in natural language, matches code fragments satisfying query conditions in the code base through various retrieval algorithms [38]. Due to the particularity of language code, the program expression may be related to natural language is different, common keyword search doesn't work, developers are hard to search code directly, but will comment as the search input statement, can realize fuzzy semantic search, keyword matching.

The research on the automatic generation, consistency changes, classification and quality evaluation and so on are the rich research achievements, but now the literature view, code comment generation technique is also facing a series of challenges, such as the accuracy is not high, application field of the single and so on. Both automatically generated and manually written code comments, there are problems such as irregular comment format, unreasonable comment location, and mismatch between comment content and code. These problems can be caused by a variety of factors, ranging from carelessness on the part of the developer to bugs in the automatic generation of code comments. In addition, the existing research has problems such as fuzzy way and single angle, and the results are not accurate or efficient enough. There is also a lack of effective tools to recommend and manage comments. Most of the previous studies were based on the collection of some attribute metrics of code comments for evaluation, and few related to the actual changes of software code, and relatively few researches on the evolution of code comments.

With the development of software, code comment plays an increasingly important role in the software development process, which is conducive to reducing the cost of software development and improving the quality of software products. This paper discusses the research status of code comment based on the related concepts of code comment and comment generation. Through the classification and summary of the existing code comment generation technology, this paper introduces and analyzes the comment generation technology, and on this basis reveals the challenges to be solved in the automatic code comment generation technology, so as to provide some references for relevant researchers and engineers.

Acknowledgments
Our thanks to the National Natural Science Foundation: 61462071 and Inner Mongolia Natural Science Foundation: 2018MS06009 for our support.

References
[1] YANG F Q, MEI H, LI K Q. Software Reuse and Software Component Technology[J]. Chinese Journal of Electronics,1999,27(2):68-75. (in Chinese).
[2] Li W P, Zhao J F, Xie B. Summary Extraction Method for Code Topic Based on LDA[J]. Computer Science, 2017(04): 42-45. (in Chinese)

[3] Yang, Bai & Liping, Zhang & Fengrong, Zhao. (2019). A Survey on Research of Code Comment. 45-51. 10.1145/3312662.3312710.

[4] Yu H, Li B, Wang P, Jia D, Wang Y. Source code comments quality assessment method based on aggregation of classification algorithms [J] //Journal of Computer Applications, 2016, 36(12), 3448-3453. (in Chinese).

[5] Jin Z, Liu F, Li G. Program comprehension: Present and future. Ruan Jian Xue Bao/Journal of Software, 2019, 30(1):110-126 (in Chinese).

[6] Sridhara G, Hill E, Muppaneni D, et al. Towards automatically generating summary comments for Java methods[C]// ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010.

[7] Sridhara G, Pollock L, Vijay-Shanker K. Automatically detecting and describing high level actions within methods[C]// International Conference on Software Engineering. 2011.

[8] Haiduc S, Aponte J, Moreno L, et al. On the Use of Automated Text Summarization Techniques for Summarizing Source Code[C]// Reverse Engineering. 2010.

[9] Moreno L, Aponte J, Sridhara G, et al. Automatic generation of natural language summaries for Java classes[C]// 2013 21st International Conference on Program Comprehension (ICPC). IEEE, 2013.

[10] WANG J S, XUE X S, WENG W. Source code summarization technology based on syntactic analysis[J]. Journal of Computer Applications, 2015, 35(7):1999-2003. (in Chinese).

[11] Wong E, Yang N J, Tan N L. AutoComment: Mining question and answer sites for automatic comment generation[C]// 2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE). IEEE Computer Society, 2013.

[12] Wong E, Liu TY, Tan L. Clocom: Mining existing source code for automatic comment generation. In: Proc. of the 22nd Int’l Conf. on Software Analysis, Evolution, and Reengineering. 2015. 380–389.

[13] Ying A T T, Robillard M P. Code fragment summarization[C]// Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, 2013.

[14] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In ACL, pages 2073–2083, 2016.

[15] Allamanis M, Peng H, Sutton C. A Convolutional Attention Network for Extreme Summarization of Source Code[J]. 2016.

[16] Hu X, Li G, Xia X, Lo D, Lu S, Jin Z. Summarizing source code with transferred API knowledge. In: Proc. of the 27th Int’l Joint Conf. on Artificial Intelligence. 2018.

[17] Mcburney P W, Liu C, Mcmillan C, et al. Improving topic model source code summarization[J]. Proceedings of the 22nd International Conference on Program Comprehension - ICPC 2014.

[18] Liang Y, Zhu K Q. Automatic Generation of Text Descriptive Comments for Code Blocks[J]. 2018.

[19] Hu X, Wei Y, Li G, et al. CodeSum: Translate Program Language to Natural Language[J]. 2017.

[20] Zheng W, Zhou H Y, Li M, et al. Code Attention: Translating Code to Comments by Exploiting Domain Features[J]. 2017.

[21] Xu S F, Pan W T, Xiong Y, Zhu Y Y. Code Comment Auto Generation based on Structure-aware Dual-encoder[J/OL]. Computer Engineering:1-7[2019-07]. (in Chinese).

[22] Oda Y, Fudaba H, Neubig G, et al. Learning to Generate Pseudo-Code from Source Code Using Statistical Machine Translation (T)[C]// 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). ACM, 2015.
[23] Khamis N, René Witte, Rilling J. Automatic Quality Assessment of Source Code Comments: The JavadocMiner [C]// Natural Language Processing and Information Systems, 15th International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, 2010. Proceedings.

[24] Steidl D, Hummel B, Juergens E. Quality analysis of source code comments. In: Proc. of the 21st Int’l Conf. on Program Comprehension. 2013. 83–92.

[25] Gao X W, Du Jing, Wang Qing. Quality Evaluation Framework of Source Code Analysis Comments[J].Computer Systems,2015,24(10):1-8.(in Chinese).

[26] Pascarella L, Bacchelli A. Classifying Code Comments in Java Open-Source Software Systems[C]// Ieee/acm, International Conference on Mining Software Repositories. IEEE Computer Society, 2017:227-237.

[27] D. Haouari, H. Sahraoui, and P. Langlais. How good is your comment? a study of comments in java programs. In Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, pages 137–146. IEEE, 2011.

[28] D. Steidl, B. Hummel, and E. Juergens. Quality analysis of source code comments. In Program Comprehension (ICPC), 2013 IEEE 21st International Conference on, pages 83–92. IEEE, 2013.

[29] Mcburney P W, Mcmillan C. An empirical study of the textual similarity between source code and source code summaries[J]. Empirical Software Engineering,2016.

[30] Fluri B, Michael W, Gall H C. Do Code and Comments Co-Evolve? On the Relation between Source Code and Comment Changes[C]// Conference on Reverse Engineering. IEEE, 2007.

[31] Tan S H, Marinov D, Tan L, et al. @tComment: Testing Javadoc Comments to Detect Comment-Code Inconsistencies[C]// 2012 IEEE Fifth International Conference on Software Testing, Verificaiton and Validation. IEEE Computer Society, 2012.

[32] Rodeghero P, Mcmillan C, Mcburney P W, et al. Improving automated source code summarization via an eye-tracking study of programmers[C]// 2014.

[33] Huang Y, Jia N, Zhou Q, Chen XP, Xiong YF, Luo XN. Method combining structural and semantic features to support code commenting decision. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2226−2242 (in Chinese).

[34] Loyola P, Marrese-Taylor E, Matsuo Y. A Neural Architecture for Generating Natural Language Descriptions from Source Code Changes[J]. 2017.

[35] R. P. Buse and W. R. Weimer. Automatic Documentation Inference for Exceptions. Intl Symp on Software Testing and Analysis (ISSTA), 2008.

[36] Zhang J, Zhang C, Xuan JF, Xiong YF, Wang QX, Liang B, Li L, Dou WS, Chen ZB, Chen LQ, Cai Y. Recent progress in program analysis. Ruan Jian Xue Bao/Journal of Software, 2019,30(1):80 109 (in Chinese).

[37] S. Zhang, C. Zhang, and M. D. Ernst. Automated documentation inference to explain failed tests. In Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE ’11, pages 63–72, Washington, DC, USA, 2011. IEEE Computer Society.

[38] Zhang F Y, Peng X, Chen C, et al. Research on code analysis based on deep learning[J]. Computer Applications and Software, 2018, v.35(06):15-23+28. (in Chinese).