International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Automatic Source Code Summarization using Semantics and Stereotypes

Chitti Babu.K, Sethukarasi.T

Abstract: ***In Software industries developers and testers spend most of the time in code maintainability than in developing or testing the code. Even after spending lot of time in analyzing the code the final analysis will serve only 50% of their purpose. The analysis involves many complex tasks like code comprehension, code extensibility and code portability. The traditional approach is to analyze the code manually, which is a tedious and time consuming task. Existing techniques does not provide the required summary, most of them are complex and they are not in natural language format. In this paper we propose a novel summarization technique that summarizes the source code similar to natural language text. The proposed system provides summary is based on the semantic content in the source code, in case the source code does not contain semantic contents it uses the java stereotypes to get the semantic content. The proposed system delivers the summaries which simplifies the task of the developer and provides the required summary.***

Keywords : Source code summary, source code entities.

I. INTRODUCTION

In the software development cycle one of the major time consuming tedious task is software maintenance. In the software maintenance process, developers and testers need to analyze the code to make the required changes. Even today this analysis is done manually which is a time consuming process. Recent studies have shown that developers spend more time reading and navigating the code than writing it [1-2]. The manual documentation of source code is a time consuming process and it must be updated regularly. Some languages like java come with document generators. Javadoc is one such tool which provides the basic documentation to a java source code. But the descriptions provided by such tools are not hundred percent useful for the developer or tester. They only provide the syntactical description with little semantic descriptions.

There are few existing approaches for summary generation which are based on skimming, reading entire code and folding.

Manuscript published on November 30, 2019.
* Correspondence Author
Chitti babu K*, Department of Information Technology, R.M.K.Engineering College, Chennai, India.
Email: kchittibabucse@gmail.com
Sethukarasi.T, department of Computer Science and Engineering, R.M.K Engineering College, Chennai, India.
Email: tsk.cse@rmkec.ac.in

TASAL [3] is technique which is based on folding approach. It folds away the least salient collisions allowing developer to focus on the main functionality of the code. Few summarizations are centered at a specific source code entity like methods or loops or classes. Automatic source code summarization of java methods [7], where summarization is generated for java methods based on the context of the methods. Loop summarization[8-9] where the summarization of loop statements in the entire source code is done. In this paper we propose a novel summarization technique for source code which satisfies the needs of the developer and tester. In the proposed approach summary is generated based on the semantic content in the source code and stereotypes. Here a novel approach is used to normalize the source code before its given for summary generation, so that the summary generation process can be quick and accurate. The main contributions of this paper are as follows

- A novel approach for generating the summaries of source code,
- Gives natural language like summaries even if there is no semantic content.
- It works for multiple languages like C, C++ and Java.
- The summaries provide useful contextual information about the entities in the program.

This paper is organized into five different sections the first section gives the basic introduction to summarization of source code, section two gives an overview of existing approaches to source code summarization, section three describes the proposed system and section four presents the conclusion highlighting the proposed system.

II. EXISTING

TASAL [3] folds the least salient regions allowing the developer to concentrate on the most important regions. It uses language model to determine the least significant regions in the code. It provides an interface where the user or developer can specify the level of folding.

There are few automatic summarization techniques likes [7] source code summarization of context for java methods, its based on the contextual data about java methods. It extracts the keywords and generates the ranks for the keywords based on this information natural language like summaries are generated. Main problem with this approach is its complexity, generates summaries for only methods and its inability to generate summaries for other modern languages like C and C++.

Summarization of source code fragments based on crowd sourcing features [8], here data-driven crowd enlistment for feature extraction is done. The feature selection analysis is done to find the discriminate selected features. At last SVM and NB classifiers trained on code fragments produce the require summaries.

Path dependencies [9] were used to generate summaries for loop statements. The loop statements were converted into Control Flow graph (CFG) and a Path dependency automaton (PDA) is generated to identify the dependencies. The PDA is traversed to generate the required summaries.

Loop summarization based on invariants [10], where the state and transition of invariants is used to generate summaries for loop statements. This approach uses an abstract model of a program with respect to a given abstract interpretation by replacing loops and function calls in the control flow graph by their symbolic transformers. The main problem with existing methods is as follows

- They work for few entities (methods, class, loops) in the code but not all the entities.
- They does not support multilingual, they work for one specific language.
- They are not light-weight, the approaches are very complex and costlier.

III. PROPOSED METHOD

In this paper we propose a novel approach for generating natural language like summaries for software artifacts. The approach is semantic level summarization which is based on the semantic content in the source code. Semantic type summarization is simple mechanism which is based on the semantics of the source code. The common semantic content any source code is comment lines. The proposed approach involves four basic phases as shown in Fig-1.

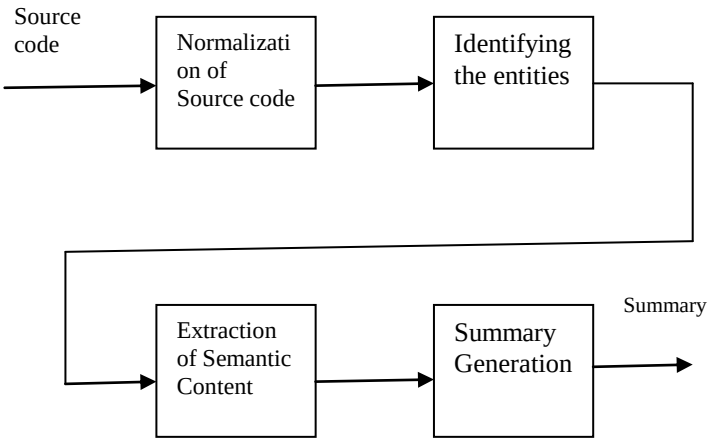

Fig-1: Semantic Based Summarization

A. NORMALIZATION OF SOURCE CODE

A raw source code is given as input to this phase and its produces a normalized source code as output. In this phase two tasks are done one is de-duplication of code and addition of stereotypes. Let's consider a java file say input.java is given as input to the normalization phase. Duplicate codes in source code reduce the maintainability of the source code so they must be removed.

To detect the code clones so many techniques are there one such technique is CCFinder [4]. In this paper we assume that one such detection technique is used and the duplicate codes are identified. The normalization algorithm removes such duplicates from the source code. In the next phase the classes and methods which do not include any semantic content (comment line) are identified and stereotypes are included for such classes and methods similar to a JSummarizer[5] and JSterocode [6].

For a java file after first level of normalization, semantic content and stereotypes are identified in the second phase of normalization as shown in algorithm-2. Algorithm-2 normalizes a java source code by generating stereotypes if the input file does not contain comments. The first step is to identify the number of classes and methods in the source file as shown in line-no 1. The next step is identifying whether there is semantic content for the identified classes and methods, if semantic content is not there than stereotypes are generated as shown in line-no 2 to 4.

For languages like C, C++ we propose an algorithm to generate the stereotypes as shown in algorithm-3 and 4. In java stereotypes for a class it identifies the members in the class and what is the basic task of those entities. Similarly for languages like C and C++ the proposed algorithm generates the stereotype which gives the basic description of a particular function in source code.

Algorithm-1:Normalization_deduplication (input.java)

Input: A java/C/C++ source file

Output: de duplicated Source file

1. Find the LOC of the source code say input.java
2. Identify the no of segments/methods in the source code.
3. **If** $segment_i = segment_j$

   Remove $segment_i$ from source code.
4. **End if**
5. Find the LOC of $segment_i$
6. Sourcecode LOC=Sourcecode LOC- LOC of $segment_i$

The stereotype tells what parameters the function is taking as input and what values the function is returning. Algorithm-3 generates the stereotypes required for a C++ source file. The first step is to identify the classes and private, public data members of the class as shown in line-no 1 & 2.

The next step is to identify any constructors and destructors present in the class as in line-no 3. After this step the stereotypes are generated for the classes, methods, constructors and destructors as in line-no 4 to 6.

For methods a special classification is done as accessors or mutator methods. Algorithm-4 generates the stereotypes required for a C source file. The first step is to identify the number of functions in the input file as shown in line-no 1.

Once the functions are identified the stereotypes are generated by identifying the type, number of input parameters, function name, return type of the function and all these details are included in the stereotype as shown in line-no 3 to 5.

Algorithm 2:Normalization_ stereotypes (input.java)

Input: a source code file

Output: Semantic contents for classes & methods in input source file.

1. Identify the no of classes and methods in the source code.
2. **If** $class_i$ does not contain semantic content generate stereotypes for $class_i$
3. **End if**
4. **If** $Method_i$ does not contain semantic content generate stereotypes for $method_i$
5. **End if**

B. EXTRACTION OF SEMANTIC CONTENT

In this phase the comment lines are identified , in general modern languages we have two types of comment lines namely single line comment which start with // and multiple comments which start with /* and end with */.

Once the comment lines are identified the semantic content in this comment lines is extracted and place in summary. In case there are no comments the stereotypes are identified, they generally start with @stereotype as illustrated in the algorithm-5. The main purpose of algorithm-5 is to extract the entities and the semantic or stereotype contents from the input source file. The first step is to read every token from the input source file if the token is "/*" or "//" the semantic content after these symbols is extracted and placed into the two dimensional array description as shown in line-no's 1 to 2. The description array is a two dimensional array containing the entity name in the 0^{th} column and the its summary in the 1^{st} column. If the token extracted is "@" symbol then the stereotype description after the symbol @ is extracted and placed as summary in the 1^{st} column of the description array as shown in line-no's 14 to 19.

Algorithm 3: Normalization_ stereotypes (input.cpp)

Input: A C++ source file
Output: Source file with stereotypes

1. Identify the number of classes and methods
2. Identify the private and public data members of the class
3. Identify the constructors, destructors and methods in the class.
4. Generate a stereotype for class giving the description of the members in the class.
5. Generate a stereotype for every accessor and mutator methods.
6. Identify whether its accessor or mutator.

C. SUMMARY GENERATION

In the summary generation phase two types of summary are generated, one is based on the semantic content for that particular entity and the second is based on the stereotype. The semantic contents normally resides in the comment lines of the code when a particular entity is identified its preceding line gives the semantic content of that entity. This semantic content is included as summary. If for a particular entity no semantic content exists then stereotype summary is generated. In the stereotype summary the class stereotype and method stereotypes are used, in the class stereotype the class description, accessors, mutators information is provided. In method stereotype the method signature, whether its an accessor or mutator method, parameters and return type information. Table & 2 shows the format of two levels of summary generated by the proposed system for the source snippet shown in Fig-2, these summaries help the developer or tester to understand the code more precisely.

Algorithm 4: Normalization_ stereotypes (input.c)

Input: A C source file
Output: Source file with stereotypes

1. Identify the number of functions in the input file
2. Generate a stereotype for every functions using the below steps
3. Identify the number of input arguments to the function.
4. Identify the return type value of the function.
5. Include the details of step-3 and 4 in the stereotype of the function.
6. Repeat steps 2 to 5 for every method in the input file.

The summary generated by proposed system look like as in table-1 and 2. The generated summary is similar to natural language text which is useful for the developer in analyzing the code.

Table-1 : Format of Summary using Semantics

Entity Name	Entity Type	Line No	LOC	Semantic Summary
Student	Class	3	19	Class representing student object
getdata	Method	8	7	Method to read student details

Table-2 : Format of Summary using Stereotypes

Entity Name	Entity Type	Line No	LOC	Summary
Student	Class	3	19	Class description: class student Stereotype: entity Text: It includes both accessors & mutators
getdata	Method	8	7	Sterotype: getter Text: it returns the student data No of parameters: 0 Return type: string

Algorithm 5: Extraction (input.java)

Input: A java source code file

Output: An array containing only comments

1.Read a single token from the source code and store in Token[i].

2.**if** Token[i]==//

3.**While** Token[i]!=\0 **do**

4.description[j][i]=Token[i]

5.i ← i+1

6. **end While**

7. j← j + 1

8.**if** Token[i]==/*

9.**While** Token[i]!=*/ **do**

10. description[j][i]=Token[i]

11. i← i+1

12. **end While**

13. j← j + 1

14. **if** Token[i]==@

15. **While** Token[i]!=*/ **do**

16. description[j][i]=Token[i]

17. i← i + 1

18. **end While**

19. j← j + 1

20. **end if**

```
1  import java.util.*;
2  /* Class representing student details*/
3  class Student
4  {
5      int rollno;
6      String name;
7      // Method to read Student details
8      public void getdata()
9      {
10         System.out.prinln("Enter name and Rollno");
11         Scanner sc=new Scanner(System.in);
12         name=sc.next();
13         rollno=sc.nextInt();
14     }
15     //Method to display student deatils
16     public void putdata()
17     {
18         System.out.println("Name :: "+name);
19         System.out.println("Rollno:: "+rollno);
20     }
21 }
```

Fig-2: Sample code in java representing Student details

The proposed approach helps the developer or tested to identify the code snippet to be modified very efficiently in less amount of time than the existing approaches.

IV. RESULTS AND DISCUSSION

The proposed system was tested with five set of simple problems namely Student information system, Employee information system, Inventory information system, vehicle management system and Electricity Bill Calculation these repository was collected as an assignment given to a set of students. In this section we present the results of the problem "Student Information System"(SIS), the project is implement using java language, the directory structure is shown in Fig-3 The software solution contains four packages namely entity, exception, main and service, each package contains a set of classes as shown in table-3.

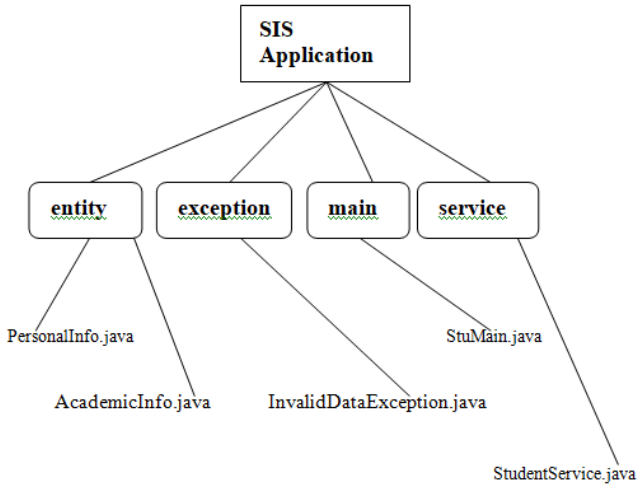

Fig-3: Directory Structure of SIS application

Table-3 : Package details of SIS Application

Package Name	No. of Classes	Class names
entity	2	PersonalInfo, AcademicInfo
exception	1	InvalidDataException
main	1	StuMain
service	1	StudentService

Each of the classes mentioned in table-3 contain a set of data members and methods as shown in table-4. The proposed approach is tested on the SIS software application, here all the methods in the PersonalInfo class contain semantic content and all the methods in the AcademicInfo class does not contain the semantic content.

Table-4 : Methods in each class of SIS application

Class name	Data members	methods
PersonalInfo	name,age, gender, mobileno.	public int getName() public void setName(String nam) public int getAge() public void setAge(int no) public int getGender() public void setGender(String gn) public int getMobileno() public void setMobileno(int no) public PersonalInfo(name,age,gender, mobilenumber)
AcademicInfo	rollno, branch, marks[], rank.	public int getRollno() public void setRollno(int no) public Strng getBranch() public void setBranch(String br) public int [] getMarks() public void setMarks(int m1.int m2,int m3,int m4,int m5,int m6) public int getRank() public void setRank(int r) public int calculateResult (marks, rollno, branch) public AcademicInfo(rollno,branch)
InvalidDataException	--	public String toString()
StuMain	rollno, rame, branch, age, marks[], moblieno.	public static void main(String[] args)
StudentService	--	public boolean validate (rollno, branch) public String calculateResult(marks[],rolln o) public int generateRank (totalmarks, rollno,branch)

So for the AcademicInfo class stereotypes are used to generate the summaries of all the methods inside this class. Table-5 show the semantic summaries generated for PersonalInfo class.

In SIS application AcademicInfo class does not conatin any semantic contents, so stereotypes are generated for all the methods in the AcademicInfo class as shown in table-6.

Table-5 : Summary generation using Semantics

Entity Name	Entity Type	Line No	LOC	Semantic Summary
PersonalInfo	Class	2	43	Class representing student personal information
public int getName()	Method	3	3	Method to get the student name
public void setName (String nam)	Method	7	3	Method to assign student name
public int getAge()	Method	11	3	Method to get the student age
public void setAge (int no)	Method	15	3	Method to assign student age
public int getGender()	Method	19	3	Method to get the student gender
public void setGender(String gn)	Method	23	3	Method to assign student gender
public int getMobileno()	Method	27	3	Method to get the student mobile number
public void setMobileno(int no)	Method	31	3	Method to assign student mobile number.
PersonalInfo (name, age, gender, mobilenumber)	constructor	34	6	Constructor to initialize the data members

The summaries generated by the proposed system were tested with a few set of students who were asked to update the given application, by using the proposed approach the students were able to identify the code snippets easily and modifications were done with ease when compared to the set of students who made the changes without using the proposed summaries.

Table-6: Summary Generation Using Stereotypes

Entity Name	Entity Type	L.no	LOC	Summary
AcademicInfo	Class	2	67	Class description: class student Stereotype: entity Text: It includes both
public int getRollno()	Method	3	3	Sterotype: getter Text: it returns rollno No of parameters: 0 Return type: int
public void setRollno (int no)	Method	7	3	Sterotype: Setter Text: it assigns student rollno No of parameters: 1
public String getBranch ()	Method	11	3	Sterotype: getter Text: it returns branch No of parameters: 0 Return type: String
public void setBranch(String br)	Method	15	3	Sterotype: Setter Text: it assigns student branch No of parameters: 1
public int [] getMarks()	Method	19	3	Sterotype: getter Text: it returns marks as an array No of parameters: 0
public void setMarks(int m1.int m2,int m3,int	Method	27	3	Sterotype: Setter Text: it assigns student rollno No of para: 1, Return type:
public int getRank()	Method	31	3	Sterotype: getter Text: it returns rank No of para: 0, Return type: int
public void setRank(int r)	Method	35	3	Sterotype: Setter Text: it assigns student rank No of para: 1, Return type: void
public int calculateResult (marks, rollno, branch)	Method	39	25	Sterotype: Normal method Text: it finds the result of the student. No of parameters: 2, Return
public AcademicInfo(rollno, branch)	Constructor	65	2	Sterotype: Constructor Text: it assigns values to rollno & branch. No of

V. CONCLUSION

In this paper a novel approach for source code summarization which is useful for software developers and testers is presented. This approach is based on the semantics in the code. The two level summaries generated by proposed system help the developer to understand the code in a more precise manner thereby reducing the maintenance cost of the software developer. Both the semantic level summary and stereotype summary help the developer or tester to analyze the code in a quick manner compared to the existing methods. The proposed approach results were presented using an SIS application from the results it's evident that the generated summaries were better than the existing contextual information.

REFERENCES

1. T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: a study of developer work habits," in 28th IEEE International Conference on Software Engineering,2006.
2. 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 Transactions on Software Engineering, vol. 32, pp. 971-987, 2006.
3. Fowkes, J., Chanthirasegaran, P., Ranca, R., Allamanis, M., Lapata, M., & Sutton, C. (2016). TASSAL: Autofolding for source code summarization. Proceedings - International Conference on Software Engineering, 649–652. http://doi.org/10.1145/2889160.2889171.
4. Toshihiro Kamiya,Shinji Kusumoto, Member and Katsuro Inoue, Member, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code",IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 28, NO. 7, JULY 2002.
5. Laura Moreno, Andrian Marcus, Lori Pollock, K. Vijay-Shanker, "JSummarizer: An Automatic Generator of Natural Language Summaries for Java Classes", 2013 IEEE.
6. Laura Moreno,Andrian Marcus, "JStereoCode: Automatically Identifying Methodand Class Stereotypes in Java Code",2012,ACM.
7. P. W. Mcburney and C. Mcmillan, "Automatic Source Code Summarization of Context for Java Methods," vol. 5589, no. c, pp. 1–18, 2015.
8. N. Nazar and H. Jiang, "Source code fragment summarization with small-scale crowdsourcing based features," pp. 1–14, 2016.
9. X. Xie, B. Chen, L. Zou, Y. Liu, W. Le, and X. Li, "Automatic Loop Summarization via Path Dependency Analysis," IEEE Trans. Softw. Eng., vol. 5589, no. c, pp. 1–21, 2017.
10. D. Kroening, N. Sharygina, S. Tonetta, A. Tsitovich, and C. M. Wintersteiger, "Loop summarization using state and transition invariants," pp. 221–261, 2013.

AUTHORS PROFILE

K. Chitti Babu*, is working as Assistant Professor in Department of Information Technology at R.M.K Engineering College, since June 2018. He is currently pursing Ph.D. in Anna University and his research is in Software Engineering. He has been in the teaching profession for the past 11 years. His area of interest includes Software Engineering and Deep Learning.

Dr. T.Sethukarasi*, Professor and Head, Department of Computer Science and Engineering, R.M.K Engineering College. She is having 24 years of experience in teaching field, she published more than 14 papers in various reputed journals. At present 11 scholars are doing research under her. Her area of interest is Data mining and Soft Computing.words.