AUTOMATICALLY GENERATING DESCRIPTIVE SUMMARY COMMENTS FOR C LANGUAGE CODE AND FUNCTIONS

Kusuma M S[1], K Srinivasa[2]
[1] M.Tech Student, [2]Assistant Professor
Email: kusumams92@gmail.com[1], srinivas.karur@gmail.com[2]

Abstract
Good comments for a program or project can help programmers and the beginners quickly understand what program or snippet of code does, providing a way for program understanding and maintaining software. Few software projects manually comment the code; other way is to automatically generate them. One way to overcome and guard against rudimentary comments is to automatically generate them. In this paper we propose a method to automatically generate descriptive summary comments for C language code and functions using content selection and text generation. In content selection we process using method for selecting the important code statements to be included in the summary comment. For selected statements, text generation method tells how to generate the content in natural language sentences.
Index Terms: Program analysis, Content generation, Text generation.

I. INTRODUCTION

Natural language processing (NLP) is the way that a computer program understands human language. NLP is a component of artificial intelligence (AI). NLP involves challenges such as natural language understanding that is making computers to obtain a meaning from human input i.e. natural language input, and also involves a challenge of natural language generation i.e. generating natural language from a machine code.

Generating natural language i.e. comments for a machine code helps the developers to understand the working of the code easily without having to analysis the entire code. A comment also helps developers to understand only relevant part of code needed during software evolution. The high level overview of methods functionality [1] helps programmers to understand the particular snippet of code without having to read the entire code or function. Descriptive summary comment helps programmers and even beginners to understand the required code or function easily.

Descriptive summary comments can be written orally by the developers or automatically generated. Orally i.e. manually writing comments is tedious and time consuming and difficult to keep up to date as changes evolves. The developer need to keep in track of changes in code and update the summary comments. By automatically generating summary comments by using the suitable technique reduces the overhead of developers to maintain it manually.

In this paper we propose a method to automatically generate descriptive summary comments for C code and functions. The method takes needed snippet of C code or function as input and output's a descriptive summary comment. Automatically generating descriptive summary comments involves program analysis, content selection and text generation. Content selection involves selecting the statements of C code to be included in summary comment. For the selected statements, text generation involves a way to express the content in natural language sentence.

II. RELATED WORK

There are few related works which are closest to our proposed method to automatically generate summary comments for C language.

Giriprasad et al [4] has proposed a tool to generate natural language comments for java methods automatically by using SWUM. Automatic generation of natural language summaries for java language using stereotype proposed by L. Moreno et al uses the stereotype to select the important statements of the method and then use the technique of Giriprasad Sidhara for text generation. Sana Makil et al [13] has proposed a method for evaluating a software word usage for C++ which uses SWUM for generating phrase.

We have used the above similar technique to automatically generate descriptive summary comments for C Language code and functions (method).As per our knowledge this is the first technique proposed for C language.

.

III. PROPOSED WORK

We propose three main methods to automatically generate descriptive summary comments for C language code: (1) Program analysis, which involves identifying identifiers, variables, functions and control flow of the code. (2) Content selection, which involves selecting the statements of code to be added in summary comments. (3) Text generation, which involves generation of natural language sentences and combining different phrases obtained to provide good summary comments for given code. Fig. 1 illustrates the proposed method to automatically generating descriptive summary comments for C language code and functions

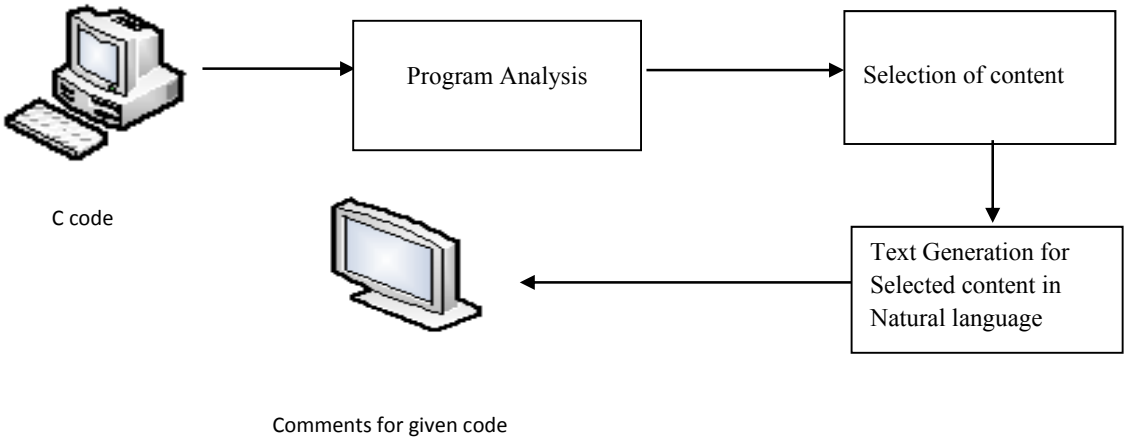

Fig. 1. Block Diagram of Proposed System

3.1. *Program Analysis*

Program analysis involves analysis of given snippet of code or function line by line to identify the identifiers, functions, condition statements and computational statements of the code. Once the different parts of code are identified, next we select the important content needed based on the priority and generate text to obtain descriptive summary comments for C language code and functions.

3.2. *Content Selection*

In our proposed method C code or function may contain multiple lines of statements. Depending upon the information obtained in program analysis, content selection is made based upon the priority we have proposed. In content selection process priority will be given for the below snippet of the code based on the order in which they are listed (1) User defined function. (2) Condition statements and looping statements (3) Inbuilt functions. (4) User defined identifiers and inbuilt identifiers (5) Computational statements. Beyond the above defined priorities functions which are called many times and functions called within other functions are given more preference than other functions in content selection for generation of high level overview of the code [10]. One or more statements of code are selected in content selection. In addition we use the S-unit selection technique [4] of prior work to remove redundancy.

Let us consider below the different cases which we come across during content selection:-

Case1: - Code containing user defined function

Consider the Fig. 2 code snippet, the line 4 “ele_pos = queue_insert (item)” statement is selected for generating summary comment, as

per the priority we have defined in content selection method. Depending upon code one or more statements is selected for generating summary comments.

```
1   void queue_info(){
2      int ele_pos, item;
3      printf("Enter element to be added");
4      scanf("%d",&item);
5      ele_pos= queue_insert(item);
6      printf("%d",ele_pos);
7   }
```

Fig.2: C language code

Case2: - Code containing only library functions

Consider the Fig. 3 code snippet, here there is no properly named identifiers, only the inbuilt library function "printf" gives some meaning, so based on priority of content selection inbuilt library function "printf" is selected.

```
1   void main(){
2      printf("hello world");
3   }
```

Fig.3: C language code with only inbuilt function

Case 3: - Code containing conditional statement

In the Fig. 4 C code snippet as per the priority defined, we select the line 5 statement i.e. condition statement and the action performed by the condition i.e. line 6.

```
1   void fun(){
2      int const=10;
3      int n;
4      scanf("%d",&n);
5      if(n<const)
6         printf("%d",n);
7      else
8         printf("null");
9   }
```

Fig.4: C language code with condition statement

Case 4: - Code containing only computation

Consider the Fig. 5 code snippet, here there is no functions or identifiers for content selection. In such worst case the computational statements are selected, i.e. line 3 along with line 2 is selected.

```
1.  void main{
2.  int i, n, s;
3.     for(i=1; i<n; i++){
4.        s=n*i;
5.     }
6.  }
```

Fig .5: C language code with only computation

3.3. Text Generation

Once the important statements needed for summary comment is obtained from the content selection process, next step is text generation to obtain summary comment in natural language sentence i.e. converting the selected content into natural language. In our proposed text generation method we first split the identifiers into component words based on camel case, for example infoListAdded is splited as "info list added". If camel case is not followed completely, we define the rules based on prepositions and verbs for splitting identifiers by the idea obtained from thousands of examined C code. Consider the example "INTtoDEC", here it is splited as "INT to DEC".

Once the identifiers, function names are splited, next step is expanding them and obtaining the text in high level overview. Different rules are followed for text generation for functions and normal code. Then we define the English grammar rule to identify the noun, verb and preposition and generate the natural language phrases by adding additional phrases to make meaningful phrase [15]. If more phrases are obtained for given snippet of code we combine them to obtain single meaningful summary comments.

For functions after splitting next, we identify action, object and parameters for text generation. Consider the example as shown in Fig. 6 "queue_insert (item)", the action is insert, parameter is item and secondary argument is queue. Next we generate phrases i.e. Action (c1), parameter (p), object (c2) with additional words added as per the English grammar rules (g) if necessary. The text generated here is "insert item into queue".

In general for functions:-

r=c1_c2 (p) (function)

c1 (g) p (g) c2 (g) r (text generated)

If parameters or object are not present in the function then only action is used for text generation.

For inbuilt library functions we have defined a set of word phrases, i.e. printf as display or print, scanf as input or accept , sqrt as square root, strcpy as copy string etc. For inbuilt function we use this defined set of word phrases

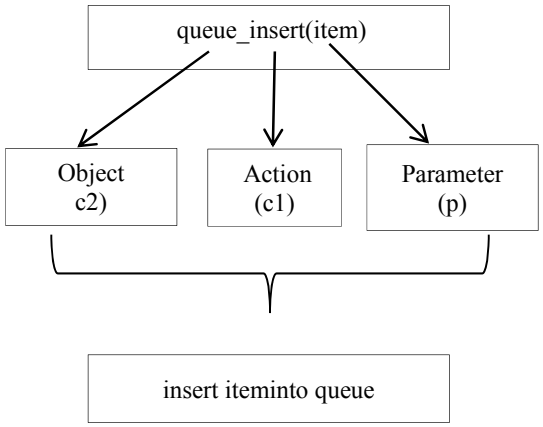

Fig.6: Text generation for function

along with English grammar rules for text generation. Consider the example inbuilt function "printf("hello world"), here action is printf, parameter is hello world, therefore the text generated is "display hello world".

For the conditional statements switching statements and looping statements we determine the condition and the action performed. For example in Fig. 4 condition is n<const and action is printf statement, i.e. line 5 and 6 respectively, by smoothening both the statements the phrase generated is "Display n if it is smaller than constant".

For other selected statements of code for text generation after the splitting of identifiers, next step is using the defined grammar rules for generating the text phrases to obtain suitable summary comments as shown in TABLE. 1. In worst case if no identifiers or function or any meaningful word is not present in code we use the notation for the computation done in code and generate summary comment in the form of notations.

Table-1 : Generation of phrases for selected statements

Content Selected	*Text generation (Phrases)*
sum = n+ m	To find sum of two variables
for(i=1;i<n;i++) s = n * i;	Compute n * i by incrementing i such that i is smaller than n

IV. EXAMPLE

Consider the following C code in Fig. 7 to illustrate our proposed method. Initially program analysis is done to identify identifiers, variables, functions of the code. In content selection process we select the important content from multiple lines of code based on the priority we have defined in the section 3.2 and redundancy is removed. In the Fig 7 the function signature and the method call (i.e. line number 4 and 6) , then function called in line 1, later condition statement in line 9 will be selected.

Once these contents are selected the natural language text generation is done using defined grammar rules, and

```
1.  void queue_info () {
2.      int a, key;
3.      Printf("Enter the key element");
4.      scanf("%d",&key);
5.      a=search(key);
6.  }

7.  int search (int key) {
8.      int cur=0;
9.     while(cur!=null) {
10.      if(key==cur->info) return cur;
11.      else cur=cur->link;
12.    }
13. return null;
```

Fig .7: C language code

different rules defined in section 3.3 for functions, condition statements and computation statements in text generation.

For the selected statements i.e. Function void "queue_info ()", the action is into and secondary argument is queue. For the function int search(int key), the action is search and parameter is key. For the condition statement "if(key==cur->info) return cur", the condition is to find if key is equal to current information and if equal return current element. The phrases are obtained as shown in TABLE. 2.

As there are multiple phrases obtained for single snippet of code, the text is generated by combining the phrases obtained to single

meaningful phrase. Hence the automatic summary comment generated for above code is "search the key element if it is in the queue and return the key".

Table -2: Phrases Generation

Priority	*Content Selected*	*Phrases*
1	Signature:- search(key); Call:- int search(int key)	Search the key
2	void queue_info ()	Information of the queue
3	if (key==c>info) return cur	Is key present in the queue return the current element

V. CONCLUSION AND FUTURE WORK

We have proposed a method to automatically generate descriptive summary comments for C language code and functions. We have used program analysis, content selection and text generation technique for generating summary.

In future we would like to implement the proposed system with still better text generation method. We would also like to evaluate the performance on comment generation by testing on different C language code and functions.

REFERENCES

[1] E.Hill. L. Pollock, and K. Vijaya-Sanker. "Automatically Capturing Source Code Context of NL-Queries for Software Maintenance and Reuse." In Intl Conf on software Engineering (ICSE). 2009.

[2] E.Hill, Z.P. Fry, H,Boyd, G.Sridhara, Y.Novikova, L. Pollock, and K. Vijay-Shanker. "AMAP: Automatically Mining Abbreviation Expansion in Programs to Enhance Software Maintenance Tools." Intl Working Conf on Mining Software Repositories (MSR).2008

[3] E.Hill. "Integrating Natural Language and Program Structure information to improve Software Search and Exploration." Ph.D Dissertation, university of Delaware.2010.

[4] Giriprasad Sridhara, Emily Hill, Divya Muppaneni, Lori Pollock and K.Vijay Shanker. "Towards Automatically Generating Summary comments for java Methods ." In IEEE/ACM intl conf on Automated Software Engineering. 2010.

[5] .L. Moreno, J. Aponte, S. Giriprasad, A. Marcus, L. Pollock, and K. Vijay-Shanker. "Automatic Generation of Natural Language Summaries for Java Classes." In proceedings of the 21st international Conference on Program Comprehension, ICPC. 2013.

[6] Paul W. Mc Burney and Collin Mc Millan. "Automatic Documentation Generation Via Source Code Summarization of Method Context." Department of Computer Science and Engineering. University of Notre Dame, in USA. 2014.

[7] Giriprasad Sridhara, Lori Pollock and K. Vijay-Shanker."Generating Parameter Comments and Integrating with Method Summaries." Department of Computer and Infromation Sciences. University of Delaware. Newark, DE 19719 USA.

[8] Bill Shannon,"C Style and Coding Standards for SunOS." By Sun Microsystems, Inc. 1993

[9] A.V. Aho, M.S Lam, R.Sethi, and J.D. Ullman "Compilers: Principle Technique, and Tools." (2nd edition). Addision-Wesely Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[10] W.k. Chan, H.Cheng, and D.Lo. "Searching connected api subgraph via text phrases." In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundation of Software Engineering. FSE 12,pages 10:1-10:11, New York, NY, USA ,2012. AMC..

[11] R. Jackenededoff. "Semantic Structures." MIT Press, Cambridge, MA, 1990.

[12] Y.S Maarek, D. M. Berry, and G.E. Kaiser. "An information retrival Approch for automatically constructing software libraries." IEEE Transactions on Software Engineering, 17(8):800-813, 1992.

[13] Paul W. Mc Burney and Collin Mc Millan. "Automatic Documentation Generation Via Source Code Summarization of Method

Context." Department of Computer Science and Engineering. University of Notre Dame, in USA. 2014.

[14] Sana Malik, Emily Hill, Lori Pollock and K. Vijay Shanker.,"Evaluating a software word Model for C++" August 17,2009.

[15] Emily Hill, Lori Pollock and K. Vijay-Shanker. "Automatically Capturing Source Code Context of NL-Queries for Software Maintenance and Reuse."Department of Computer and Information Sciences University of Delaware Newark, DE 19716 USA.