International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Code Knowledge Acquisition for Knowledge Management Trajectory Framework

M.A. Krishna Priya, Justus Selwyn

Abstract: ***Knowledge and its related techniques cater the need of many tools and frameworks for software development, testing and management. Accurate usage of code according to the software requirement is prevalent in this regard, as inappropriate development of code with conceptual misunderstanding might lead to waste of time and effort. This paper describes a new approach to automate code acquisition task. This gives the advantage of using the coding knowledge that is already encoded which in turn reduces developer s task and their knowledge and effort could be utilized for further enhancement of their work. Using the existing coding knowledge eradicate the errors which are introduced at the time of naïve development activity. Furthermore, this technique can be utilized by software development applications. In this paper, a new knowledge execution technique to acquire code knowledge from software application is implemented. Code extraction gathers code segments from projects organize and logs it for future use. Here, Source Code Extraction algorithm is described that captures code.***

Keywords: Code segments, Knowledge Management, Source code extraction, Code extractor

I. INTRODUCTION

Acquisition of code logs for knowledge management in software organizations is very crucial because history gives high stories of best and worst practices which gives insight experience for individuals from which people can learn dos and don'ts. Question arises whom are all ready for KM contribution, Developers are keen in their day to day activity at last no contribution happens and here our algorithm works for KM. There are three kinds of developers, first kind is novice developers; they lag in work since they are new, they need support, through KM they can utilize the code knowledge, next knowledge hoarders, they don't share their knowledge means they don't contribute for KM, that's why code knowledge is captured and stored for KM and the third type of developers voluntarily lag their task in order to get the overtime benefit which is loss for the organization, since our KM supports them with proactive suggestions to complete the task, they have no other go and developers has to finish their work with in the allocated time period. These knowledge transfer techniques are highly beneficial for software organization in terms of cost. Knowledge management trajectory framework is followed in this paper which presents knowledge transfer techniques to provide knowledge solutions for a particular problem scenario; software developers can choose the best solution in their software development activity. This framework consists of three layers called knowledge harvesting, knowledge mapping and knowledge sharing.

A. Knowledge harvesting:

Through knowledge harvesting techniques tacit knowledge is made more explicit. This identifies and extracts people's knowledge and explicitly stores it into documents which are shared with others easily. Knowledge from various resources is harvested. Knowledge base is a foundation for a knowledge based system. This contains domain knowledge in a representation which is suitable for a particular system. Knowledge is represented in a required format in knowledge base according to its usage. Here, codes written by experienced software developers are acquired. Initially, knowledge acquisition happens through capturing code data and code data is stored as code logs. Knowledge base is created with different code attributes and extracted codes are stored. Knowledge extraction algorithms could be utilized for acquiring the code patterns. To implement knowledge extraction systems different knowledge transfer techniques can be followed. This implementation acquires reliable code knowledge using conditional rules that identifies the desired code patterns. Software organizations use various software languages; each of them has their own uniqueness which is reflected in its code formats, developing extraction rules for individual software application is tedious. In order to experiment simple extraction methods are used in this algorithm. Source Code Extraction (SCE) algorithm is synthesized for the act of knowledge harvesting. Harvesting code knowledge from various software projects for future use is very mandatory for software development organizations. In this paper, an approach is presented to store code knowledge from various project applications that is code knowledge is harvested as code logs which is the base for the creation of solution base.

Manuscript published on 30 September 2019
* Correspondence Author
M.A. Krishhna Priya, Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, India. Email: priya_krishi@yahoo.co.in
Justus Selwyn, Professor, School of Computer Science and Engineering,, VIT University, Chennai, India. Email: justus.s@vit.ac.in

The approach uses context features extracted from the files. It also leverages the redundancy across multiple files to harvest code knowledge accurately. These code sets are parsed and used to improve the coding efficiency of a developer in solving requirement scenarios.

Through this structured code data knowledge base is built which in turn will be utilized to provide logical inference in solving programming tasks.

B. Knowledge mapping:

Knowledge mapping technique maps explicit information and tacit knowledge. Knowledge base serves as a knowledge resource. The important task is to supply the required knowledge by identifying it from the available resource to the developers. Identifying the appropriate code knowledge according to the requirement is a great challenge because conceptual understanding of every code piece is very much essential. In this layer, solutions base is constructed with appropriate code.

C. Knowledge sharing:

Sharing of knowledge happens through different applications. Many applications are available nowadays. Amidst of the development the developer has to come out of his/her current application and then he/she has to explore various options according to the need. But by using this KM framework, the developer will not get the chance to search because every code piece is available in solution base. They just need to assemble those code fragments. This layer shares coding knowledge proactively. This can be used with any software application.

II. PROCEDURE FOR CODE KNOWLEDGE ACQUISITION PAPER SUBMISSION

A. Proposed Architecture

Source code extraction has five main phases that identify system directories, check for directory accessibility, file discovery, extract source code, and store code logs. Fig. 1 explains the architecture pictorially. First phase identifies and selects all the directories available in the organization's server or from the cloud. Second phase checks for the accessibility of the directories, whether the selected directories contain system files or programming files. Third phase does file discovery based on the date validation and file extensions. Date validation checks the last accessed and modified file using system date. If a particular file is either created or modified on today's date then the system checks for the required file extensions. In fourth phase source code sets are extracted from the selected files. Finally in fifth phase, source code sets are stored as code logs.

B. Proposed SCE algorithm

Source Code Extraction (SCE) from various files is possible by implementing this algorithm. Here, Procedure 1 is explained with simple steps. The process starts with the 'dir' string array, get the subdirectories from each 'dir' subscripts till it reaches end node. Every directory is verified for its accessibility whether the sub directories are system directories. If a directory is accessible then get all files from the particular directory. In a directory all files are identified by its last modified date. In case if the current date is equal to last modified date then that the file's extension is stored in a variable. And a counter check is made to find whether the extension matches with any of the required file format if so then read the file name and content. Save it in code log. This procedure continues till the end of 'dir' subscript for the extraction of source code.

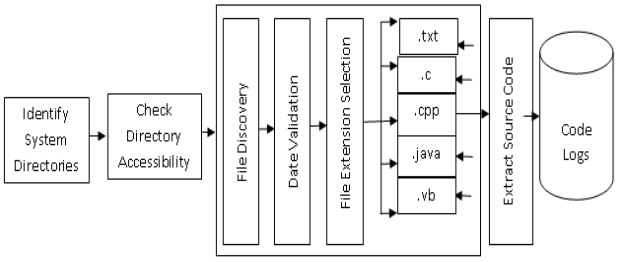

Fig. 1.Architecture of source code extraction

Procedure 1: Source code extraction

Begin:

1. Dir() = GetDirectories from First Directories
2. For Each D In Dir
3. If D <> Valid Directories
4. files() = GetFiles D
5. For Each File In files
6. mdate = File date
7. If tdate = mdate Then
8. ex = GetExtension(File)
9. If ex = ".txt" Or ex = ".cpp" Or ex = ".vb" Or ex = ".c" Or ex = ".docx" Or ex = ".java" Or ex = ".m" Then
10. File = file.filename
11. Dir1() = Sub Directory(D)
12. For Each s Dir1
13. Dim files1() = GetFiles(s)
14. For Each File In files1
15. mdate = File1 date
16. If tdate = mdate Then
17. ex = GetExtension(File1)
18. If ex = ".txt" Or ex = ".cpp" Or ex = ".vb" Or ex = ".c" Or ex = ".docx" Or ex = ".java" Or ex = ".m" Then
19. File = file1.filename
20. Repeat the process till end of system directories

1) Selection of directories

Whole set of directories from the system drives are identified and stored in to Dir () array, each directory is stored under one subscript, it continues till the end of directory Dir(n). The initial process consists of storing n number of directories. Subsequently for each directory, it's subdirectories are identified and stored in to Dir1() array variable.

2) Accessibility of directories

Read each directory from Dir() and Dir1() . Every directory is checked for its accessibility whether it has read only accessibility or it possesses read/write accessibility. If the selected directory has read only access, then it is skipped otherwise the directory information is retrieved from Dir() array. This process is repeated for all the subdirectories which are in Dir1() array.

3) File discovery

In this stage, system checks for main and subdirectories, then it identifies and selects the files based on the file extensions for example '.txt', '.cpp', '.vb', '.c', '.docx', '.java' and '.m'. The files are text, C, C++, Visualbasic, document and Matlab files. These selected files are stored in to files() array. Every file's creation date is been identified and checked whether it is created on today's date or on other dates. If the date of creation of a file or last date of modification of a file is not matching with today's system date then those files are ignored otherwise the files are stored in to files() array. Process identifies the main and subdirectories, here, Table I displays the process data, which is synthesized, the syntax extraction column states the corresponding code/file retrieval, second column displays the selected file extensions for the discovery of files and the third column shows the date on which files were identified and discovered.

Table I : File discovery from directories

Code Extractions	Extensions	Created Date
class CheckEvenOdd	.java	28/08/2018
public static void main(String args[])	.java	28/08/2018
System.out.println("Enter an Integer number:");	.java	28/08/2018
Scanner input = new Scanner(System.in);	.java	28/08/2018
num = input.nextInt();	.java	28/08/2018
if (num % 2 == 0)	.java	28/08/2018
System.out.println("Entered number is even");	.java	28/08/2018
System.out.println("Entered number is odd");	.java	28/08/2018
int n, reverse = 0, t;	.c	28/08/2018
printf("Give number check for palindrome \n");	.c	23/08/2019
scanf("%d", &n);	.c	28/08/2018
t = n;	.c	28/08/2018
while (t != 0)	.c	28/08/2018
reverse = reverse * 10;	.c	28/08/2018
reverse = reverse + t%10;	.c	28/08/2018
t = t/10;	.c	28/08/2018
if (n == reverse) printf("%d number.is a palindrome \n", n);	.c	23/08/2019
printf("%d number isn't a palindrome.\n", n);	.c	23/08/2019

4) Code extractions

This Code Extractor (CE) works as a background process. This tool monitors the project repository and its files, which are saved recently. Code Extractor listens and extracts the code on day to day basis. Programmer's activity is continuously monitored by CE. Code is extracted from the programmer's application only after he/she saves it. Files are tracked, code sets from the software application's work area is extracted.

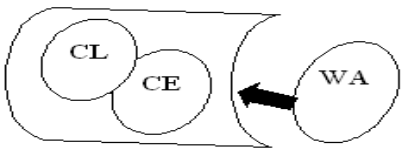

Fig. 2..Code extractor(CE), Code Logs(CL) and Work Area (WA)

As it is shown in Fig. 2 this process silently extracts source code from the work area of an application domain and stores the source code sets in code log. Code log consists of multiple code segments.

5) Storage of code logs

Code Logs (CL) are assessed and evaluated manually by knowledge champions. Source codes statements are identified based on certain keywords for instance class, new, double and return, etc. The extracted statements are stored in to a database according to the file ID. This is to submit source code sets to a particular storage medium called database. Further work is to proceed with the database through which independent help can be provided to the software developers who are in need.

III. RELATED WORK

Symbolic execution is used in the context of program verification [1] uses the idea of using symbolic execution to extract a function from C code. Approach described in [2] uses model extraction of GUI programs from hand-held devices and compares extracted models with the expected models. There are existing approaches [3], [4] that use program synthesis based approach to generate bit-vector formulas representing instruction semantics. Rather than searching the functions in a large search space, the search space is confined manually by specifying semantics of basic instructions [4]. SMT solvers, STP [5] and Z3 [6], are used in many popular symbolic execution systems [7]-[14]. SMT solvers, with support for various theories arrays match closely with the semantics of C language.

Product life-cycle management is a software framework, or a kind of guideline to approach digital manufacturing, which is the highest level of recently known and applied ways of manufacturing automation. This product life-cycle management features [15] the importance of reuse and recycling. This model shows how PLCM is related to primary and secondary effects of sustainability, i.e. losses generated in order to be sustainable [16]. Important components are presented as Product is the core, containing data, technology, methods, tools and their programming support as Computer Aided Engineering, Computer Aided Process Planning in general Computer Aided Solutions, Product Data Management etc. The up most layer is the Management with services, marketing, general design, engineering, procurement, manufacturing, etc.; Between product and management there is the lifecycle with conceive, develop, realize and use[17].

IV. EXPERIMENTAL RESULT

This experiment shows the execution progress on basis of time. Table II indicates source code extraction method's performance with the captured records that are used by the training module. The first column shows the total time. Second column shows the difference between existing and proposed method s success path. Positive path= No of positive * No. of negative / No. of negative. Third column denotes failure paths explored at the time of execution.

Negative path= No of positive * No. of negative / No. of positive. The fourth column shows the coverage obtained, expressed in terms hits while visiting file nodes. Coverage=No. of files identified/ No. of files accessed. The fifth column shows the usage of virtual memory. Usage of memory is measured. Memusage as MemorymappedFile, Memusage= MemorymappedFile.CreatefromFile(filename).

Total Time (Sec.)	Success Path		Failure Paths		Coverage		Virtual Memory Use	
	Existing (M)	*Proposed (M)*	*Existing (M)*	*Proposed (M)*	*Existing (%)*	*Proposed (%)*	*Existing (MB)*	*Proposed (MB)*
0.00	0	0	0	0	0	0	17	15
0.51	4.5	5	18.1	16	13	15	20	18
0.89	5.8	6	28.4	20	15	17	20	18
1.27	7.1	7.5	38.6	36	17	19	20	19
1.65	7.3	7.5	49.9	47	17	19	21	16
2.04	7.6	7.8	61.2	57	19	21	20	18
3.17	8.5	9	94.9	92	21	23	18	21
4.88	9.0	10	127.6	124	37	39	19	22
6.09	11.2	13	156.3	150	49	50	19	16
7.31	17.4	18	223.3	221	66	67	18	16
8.53	17.6	20	248.0	240	69	70	19	17
9.75	18.1	19	268.5	265	70	72	19	16
10.42	19.2	20	287.5	280	79	80	19	16
11.55	23.5	21	316.9	312	87	88	18	14
12.22	24.3	26	336.0	324	91	93	18	15
13.11	25.6	27	360.2	356	100	102	17	16

Table- II: Performance of SCE

The metric values of success path, failure path, coverage, memory occupied illustrates the efficient performance of this technique when compared with the existing one. The proposed method addresses common file discovery, efficiently tracks the files based on date and file extensions and extracts source code. Fig. 3 explains the performance comparison of existing and proposed method.

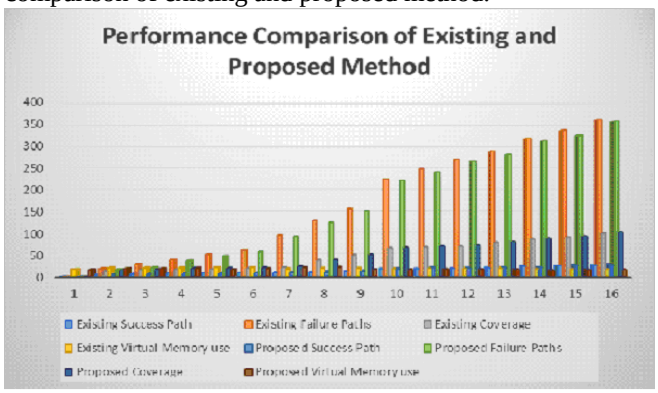

Fig. 3.Performance comparison of existing and proposed method

This method has reduced manual work when compared with existing approaches. The proposed SCE yields 13.5 % positive paths, 5.0% negative paths and 96.1 % coverage. SCE algorithm's average time of execution is less than 5.8 seconds.

V. CONCLUSION

This SCE algorithm works well to extract source codes based on certain keywords and also there is a plan to extend and improve the algorithm with wide range of keywords to cater the need of the knowledge management framework. The constraint of this work is that, some of the extracted source code sets might be associated with some other set of codes. In order to address this, the intention is to extend the algorithm by identifying the individual requirements and its matching code pattern along with its status in terms of efficiency in solution base would provide solution services to the software developers.

REFERENCES

1. Aizatulin M., Gordon A. D., Jurjens J. Extracting and verifying cryptographic models from c protocol code by symbolic execution,Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, Illinois, USA, 17-21 October 2011, pp. 331–340.
2. Wang S., Dwarakanathan S., Sokolsky O., Lee I. High-level model extraction via symbolic execution,Technical Report,No. MS-CIS-12-04, Department of Computer and Information Science, University of Pennsylvania, 2012, pages 29.
3. Godefroid P.,Taly A. Automated synthesis of symbolic instruction encodings from I/O samples,Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation, Beijing, China, 11-16 June 2012, pp. 441–452.
4. Heule S., Schkufza E., Sharma R., Aiken A. Stratified synthesis: Automatically learning the x86-64 instruction set,Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, Santa Barbara, CA, USA, 13-17 June 2016, pp. 237–250.
5. Ganesh V., Dill D. L. A decision procedure for bit-vectors and arrays, In Proceedings of the 19th International Conference on Computer Aided Verification, Berlin, Germany, 3-7 July 2007, pp. 519–531.
6. De Moura L., Bjørner N. Z3: An efficient SMTsolver,Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, Vol 4963, Springer, 2008, pp. 337–340.
7. Sen K., Marinov D., Agha G. CUTE: a concolic unit testing engine for C,Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFTInternational Symposium on Foundations of Software Engineering, Lisbon, Portugal, 5-9 September 2005, pp. 263–272.
8. Godefroid P., Klarlund N., Sen K. DART: directed automated random testing,Proceedings of the 2005 ACM SIGPLANConference on Programming Language Design and Implementation, Chicago, IL, USA, 12-15 June 2005,pp 213–223.
9. Cadar C., Dunbar D., Engler D. KLEE: unassisted and automatic generation of high-coverage tests for complex system programs, Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, San Diego, California, 8-10 December 2008, pp. 209–224.
10. Brumley D., Jager I., Avgerinos T., Schwartz E. J. Bap: a binary analysis platform, In Proceedings of the 23rd International Conference on Computer Aided Verification, Snowbird, UT, 14-20 July 2011, pp. 463–469.
11. Cadar C.,Engler D. Execution generated test cases: How to make systems code crash itself, Proceedings of the 12th International Conference on Model Checking Software, San Francisco, CA, 22-24 August 2005, pp. 2–23.
12. Godefroid P., Levin M. L., Molnar D. A. Automated whiteboxfuzz testing. Proceedings of 16th Annual Network and Distributed Security Symposium,Vol. 8, San Diego, California, USA, 10-13 February 2008, pp. 151–166.
13. Pasareanu C. S.,Rungta N. Symbolic path finder: Symbolic execution of Java bytecode, In Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, 20-24 September 2010, pp. 179–180.

14. Chipounov V., Georgescu V., Zamfir C., Candea G. Selective symbolic execution,Workshop on Hot Topics in System Dependability, Estoril, Portugal, 30 June 2009, pages 1–6.
15. Kovács G. L., Kopácsi S., Haidegger G., Michelini R. Ambient intelligence in product lifecycle management, Engineering Application of Artificial Intelligence, Vol. 19, No. 8, 2006, pp. 953–965.
16. Leitol Cs. Resource and cost efficient selective collection, Pollack Periodica, Vol. 9, Supplement 1, 2014, pp. 43–54.
17. Kovács George L, Petunin Alexander. An Information technology view of manufacturing automation - product life-cycle management, Pollack Periodica, Vol. 11, No. 2, pp. 3–14 (2016)
18. Building Large Knowledge Bases by Mass Collaboration Matthew Richardson mattr@cs.washington.edu Pedro Domingos pedrod@cs.washington.edu Department of Computer Science and Engineering University of Washington Seattle, WA 98195-2350
19. M.A. Krishna Priya, Justus Selwyn.KM Trajectory Schema Service Frame Work for Software Development Organizations, International Journal of Engineering &Technology, 7 (3.12) (2018) 616-620

AUTHORS PROFILE

M.A. Krishna Priya obtained her Master of Computer Applications from Bharathiar University, Coimbatore, India Her previous appointment includes lecturer in Karpagam College of Engineering, Coimbatore, and worked as a Software Engineer at Antares Technologies Pvt Ltd, Coimbatore, India. Currently, She is working as Assistant Professor in Hindustan College of Arts Science, Chennai (Affiliated to University of Madras, Chennai, India).. Her current research interests are Knowledge Management, Information Retrieval Systems

Dr. Justus Selwyn is a PhD in computer science, specialized in software engineering and knowledge engineering. His specializations include Software Engineering, Metrics, Object-Relational modeling, and knowledge engineering. He is heading the software & knowledge engineering research group and is working as Associate Professor at the School of Computing Sciences and Engineering, VIT University, Chennai. He is the co-founder of a software consulting company, Inclefs, which is into product engineering and development. He has research publications in reputed journals and has regular consultancy works with tie-ups with software companies in Chennai.