ISSN 2384-4418

Smart Convergence of Culture Technology Letters

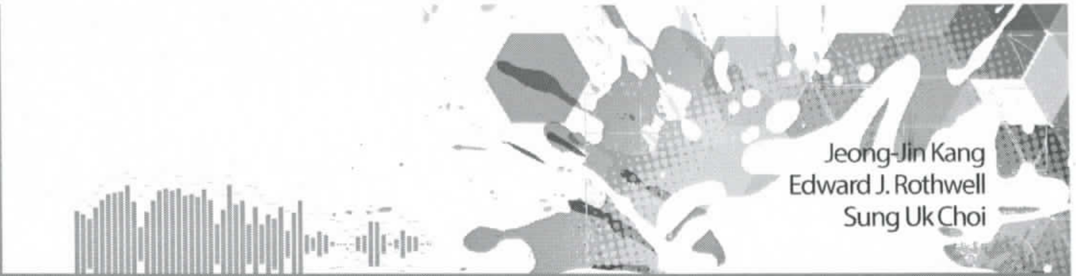
The SCCTL is committed to the publication of proceedings of Smart Convergence of Culture Technology. Its objective is to publish original researches in various areas of Smart Convergence. This will provide good chance for academia and industry professionals as well as practitioners to share their ideas, problems and solutions relating to the multifaceted aspects.
Research papers were strictly peer-reviewed by program committees to make sure that the papers accepted were high quality and relevant to the current and future issues and trends in Smart Convergence of Culture Technology.
The scope of SCCTL includes the entire area of smart technology and its convergence to other areas from the current and future trends. The language of publication is English.

http://ipact.kr/eng/

Kang et al.(Eds.)

SCCTL 05 Smart Convergence of Culture Technology

IPACT IIBC



Jeong-Jin Kang
Edward J. Rothwell
Sung Uk Choi

Smart Convergence of Culture Technology Letters SCCTL 05

Smart Convergence of Culture Technology

The 5th International Integrated Conference & Concert on Convergence, IICCC 2019 in conjunction with ICCPND 2019

Lotte Hotel Vladivostok, Vladivostok, Russia, July 23-28, 2019
Revised Selected Papers

IPACT The International Promotion Agency of Culture Technology

IIBC The Institute of Internet, Broadcasting and Communication

The Society of Convergence Knowledge

CONTENTS

An Automatic Visualization Approach for Vulnerability of Blockchain Code

Bo Kyung Park[1], Jihoon Park[2], Dohyoung Kim[3], Young B. Park[4] and R. Young Chul Kim[1*]

[1]SE Lab., Dept. Of Software and Communications Engineering, Hongik University, Sejong-City, Korea
[2]Telecommunications Technology Association, Mapo-gu, Seoul, Korea
[3]Innoblock, Gangnam-gu, Seoul, Korea
[4]Dept. Of Software Engineering, Dankook University, 152, Yongin-si, Gyeonggi-do, Korea
e-mail : park[1]@selab.hongik.ac.kr, [2]jh91082@tta.or.kr, [3]dohyoung.kim@ibpartners.io,
[4]ybpark@dankook.ac.kr, [1*]bob@hongik.ac.kr

Abstract

The global blockchain market is expected to grow more than 10 times over the next five years. The block chain paradigm will have popularly been adopting various fields such as finance, logistics, and medical care. However, no one has a technical verification of the reliability, scalability and stability of the blockchain code. To solve this problem, it will need how to identify the bad quality within blockchain code. In this paper, we propose a method to visualize vulnerability of blockchain code through static analysis. Through the proposed method, we solve the vulnerability problem of blockchain oriented software development (BOSD). This method can also improve the quality of the process.

Keywords: *Automatic Visualization, Vulnerability of Blockchain Code, Blockchain Software Quality, Code Quality Factors, Software Visualization, JavaParser*

1. Introduction

The global software market has grown to $ 1148.1 billion. The global software market is expected to further expand due to the effects of the fourth industrial revolution such as smart factories, AI, block chains, autonomous vehicles, and smart robot software. The era of the fourth industrial revolution is the intelligent information society. In this era, data generated through advanced information and communication technologies are merged with new ideas. In other words, converged data and ideas are expected to create new values. The software in the era of the fourth industrial revolution can change according to the intelligent, self-reliant capabilities and circumstances. In the future, software related to the fourth industry will be necessarily integrated with "blockchain technology."

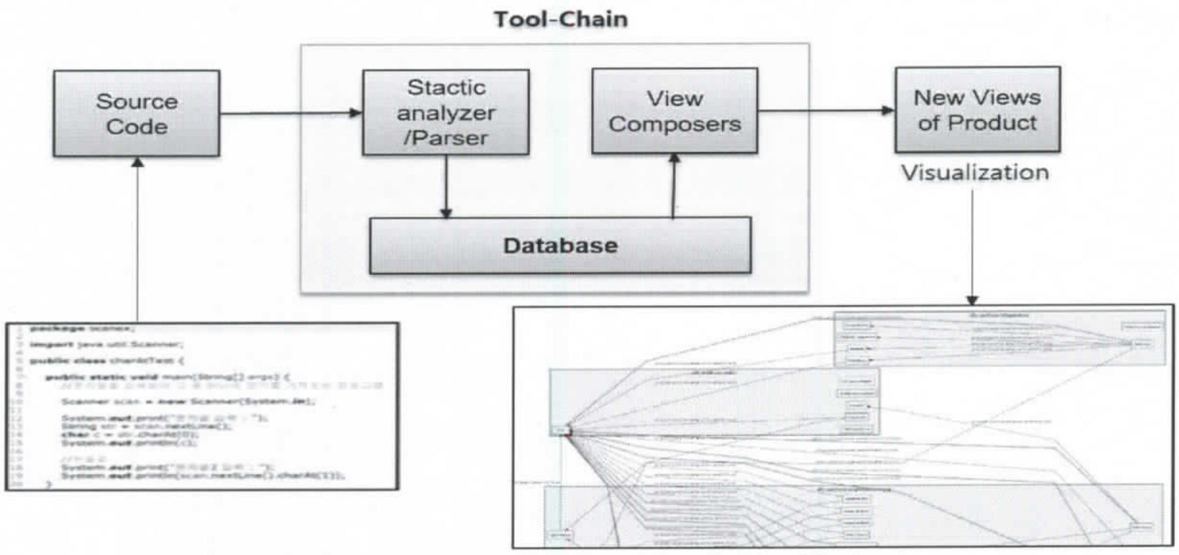
However, some scholars argue for critical opinions on the scalability and stability of the blockchain system. In order to apply the blockchain technology universally, it is necessary to verify the technology realistically. In order to realize high-quality blockchain software and satisfy quality, quality improvement technology of blockchain software is required.

This paper proposes the visualization mechanism for the blockchain code vulnerability visibility system through static analysis. The proposed framework is a static analysis of blockchain created with Java code. The proposed method visualizes the inside of the blockchain code. The advantages of this method are as follows: 1) identify the vulnerabilities inherent in the blockchain system, 2) apply the software engineering method to reduce time, effort and cost. We explain the visualization system of blockchain based java code. The static analysis tool is an open source, JavaParser. We also apply SQLite to the database and then automatically generate dotscript by combining the information stored in the database. Finally we create the visualization graph of code vulnerability with Graphviz. Through the proposed method, we solve the vulnerability problem of blockchain software development. This method will improve the quality of the process.

The composition of this paper is as follows. Chapter 2 introduces related works. Chapter 3 describes visualization method of the vulnerability of blockchain based code. Chapter 4 describes a case study. Finally, Chapter 5 mentions the conclusion.

2. Related Works

The previous tool-chain consists of a static analyzer (parser), database, and visualization tools. The static analyzer interprets the information in the source code. The code information is class, method, variable, structure, function, etc. The static analyzer generates the data. These data store the information needed for visualization in the database. To generate dotscript, stored information uses queries. Finally, the visualization tool generates the graph using the retrieved data. Figure 1 shows a structure of previous visualization mechanism.



```
package [illegible];

import java.util.Scanner;

public class charAtTest {

    public static void main(String[] args) {
        [illegible]

        Scanner scan = new Scanner(System.in);

        System.out.print("[illegible]");
        String str = scan.nextLine();
        char c = str.charAt(0);
        System.out.println(c);

        // [illegible]
        System.out.print("[illegible]");
        System.out.println(scan.nextLine().charAt(1));
    }
```

Figure 1. The Structure of Previous Visualization Mechanism

The existing coupling and cohesion is defined for procedural oriented language. According to Daniel Rodriguez, existing procedural oriented language has no concept of encapsulation, inheritance, polymorphism, etc. Object-oriented concepts are as follows: Objects, classes, components, packages, encapsulation, inheritance, polymorphism, etc. Thus, the existing coupling and cohesion should be redefined based on the characteristics of the object-oriented language.

Figure 2 shows the relationship between coupling and cohesion. Coupling and cohesion are always mentioned in the reusability of software engineering. Good software should have low cohesion and high cohesion. The left figure in Figure 2 is High Coupling, Low Cohesion. In this case, the complexity between the modules becomes extremely high. The right figure shows Low Coupling and High Cohesion. In this case, the complexity between the modules is lowered.

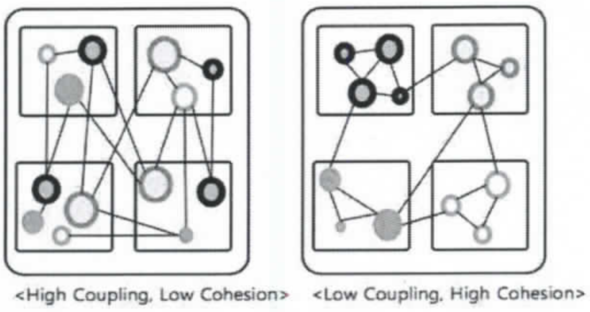


Figure 2. The Relationship between Coupling and Cohesion

3. Automatic Visualization between Coupling and Cohesion

The block chain system is implemented in various languages. The block chain system is implemented in Java. Go-based block-chain systems are often used in web applications. In this paper, we explain the method of code vulnerability visibility of blockchain system implemented in Java.

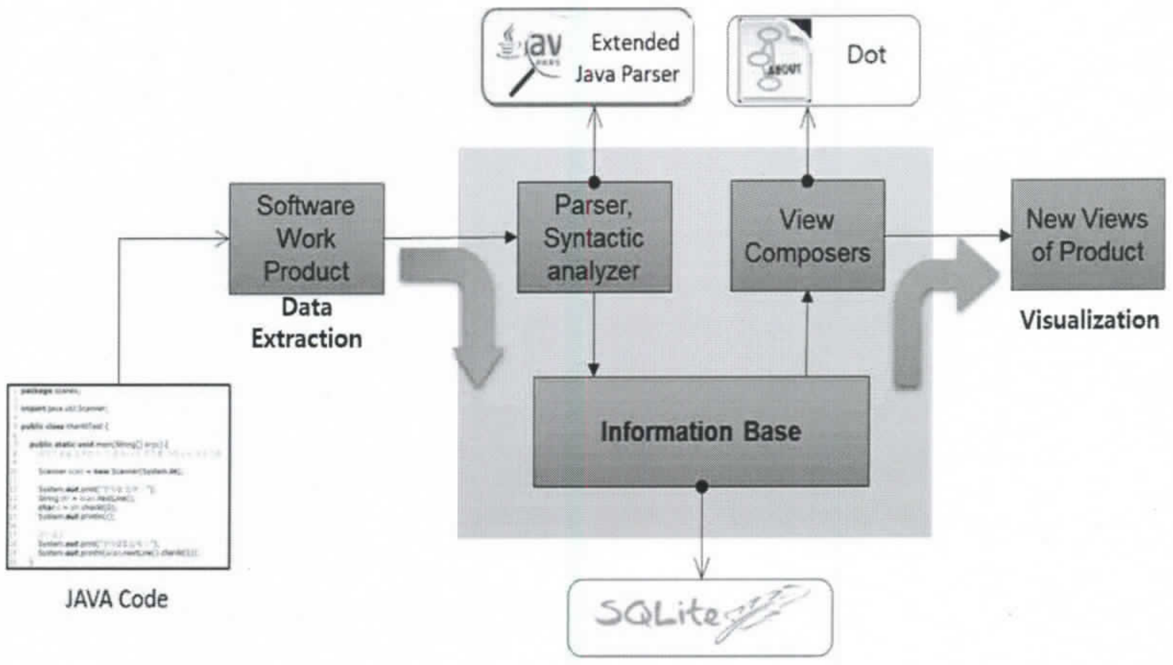


Figure 3. Automatic Visualization Framework of Java-based Blockchain Code

Figure 3 shows the automatic visualization framework of java-based blockchain code. This toolchain extracts information from Java code and generates a visualization graph. The proposed visualization framework is as follows: 1) the existing toolchain uses Source Navigator as a static analysis tool. The source Navigator parses the Java code and extracts the SNDB files. However, SNDB files are unreadable binary code. To make reuse easier, we improved JavaParser. The existing Source Navigator can analyze both C and Java. However, there are only six Java code files analyzed. The analyzed files are as follows: cl (Classes), in (Inheritances), iv (Instance variables), lv (Local variables), md (Method definitions). Source Navigator lacks data from static analyzed Java code. To solve this problem, we developed a new static analysis tool using

JavaParser. JavaParser parses into AST structure and extracts necessary analysis data. At this time, the analysis data is reconstructed for storage in the database. Figure 4 shows the structure of Java code analyzed by JavaParser.

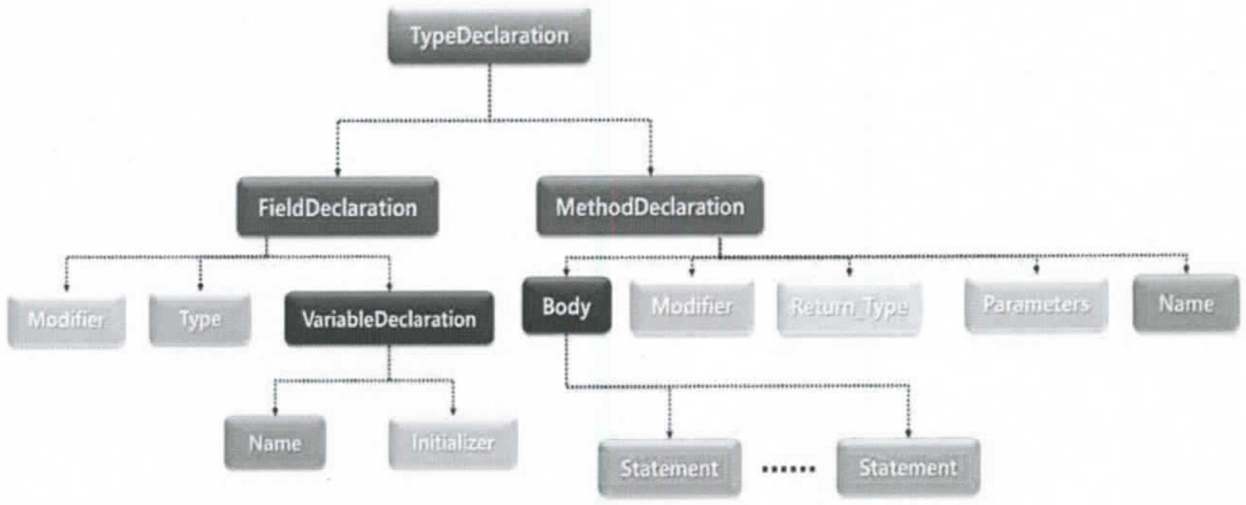


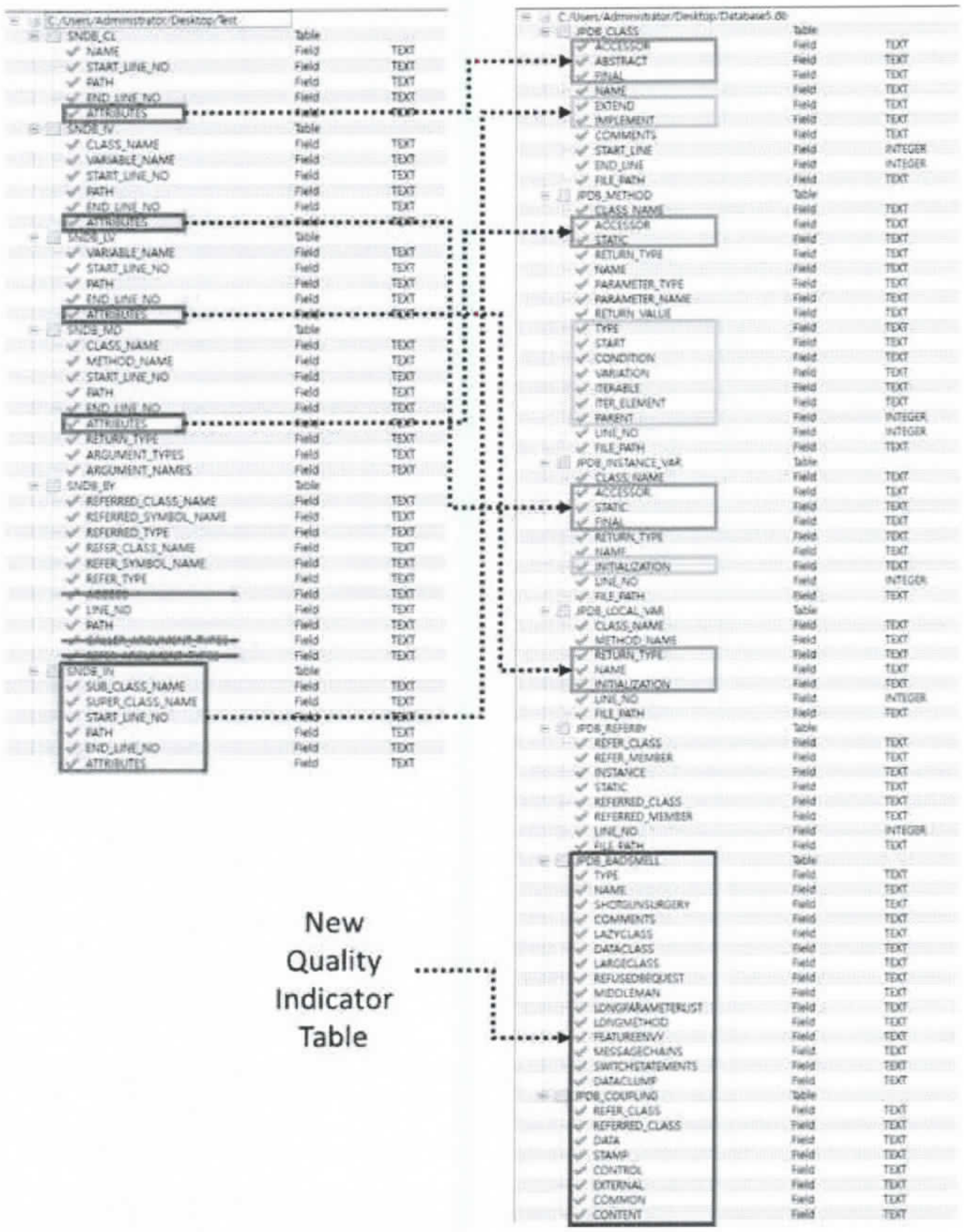
Figure 4. The AST Structure of Java Code

2) The analyzed results are stored into the database tables. We use the SQLite that is relatively light compared to other databases, extract information stored in the database, and write a DOT script. At this time, a query for extracting necessary data is prepared for each type. This data is generated by the DOT script. 4) We run this data in visualization tools. The analyzed code outputs the visualization graph.

4. Case Study

We expect to guild how to guarantee quality of blockchain based on our visualization mechanism. We analyze blockchain code through static analysis of Java code with our extended JavaParser. In the database, we extract the following information: 1) inheritance relationships in all statements, 2) Variable declaration, 3) method declaration, 4) call relation, etc.

Figure 5 compares the information extracted from the source navigator with the information extracted from the JavaParser. The left picture is the Source Navigator. The right figure is JavaParser. The squares on the left are extended to the squares on the right. ATTRIBUTES is divided into ACCESSOR, ABSTRACT, FINAL, STATIC, RETURN_TYPE, and INITIALIZATION. In addition, the inheritance information of the SNDB_IN table is added as a column in the JPDB_CLASS table. As a result, we reduced the relation of the database tables. In the figure on the right, the New Quality Indicator Table is the value of the quality index measured using the extended extracted information. Through the extraction results, we derived the improvement results: 1) the number of columns in the database has increased. 2) Visualization information was extracted specifically. 3) New Quality Indicator Table was created.



```
C:/Users/Administrator/Desktop/Test
  SNDB_CL                       Table
    NAME                        Field    TEXT
    START_LINE_NO               Field    TEXT
    PATH                        Field    TEXT
    END_LINE_NO                 Field    TEXT
    ATTRIBUTES                  Field    TEXT
  SNDB_IV                       Table
    CLASS_NAME                  Field    TEXT
    VARIABLE_NAME               Field    TEXT
    START_LINE_NO               Field    TEXT
    PATH                        Field    TEXT
    END_LINE_NO                 Field    TEXT
    ATTRIBUTES                  Field    TEXT
  SNDB_LV                       Table
    VARIABLE_NAME               Field    TEXT
    START_LINE_NO               Field    TEXT
    PATH                        Field    TEXT
    END_LINE_NO                 Field    TEXT
    ATTRIBUTES                  Field    TEXT
  SNDB_MD                       Table
    CLASS_NAME                  Field    TEXT
    METHOD_NAME                 Field    TEXT
    START_LINE_NO               Field    TEXT
    PATH                        Field    TEXT
    END_LINE_NO                 Field    TEXT
    ATTRIBUTES                  Field    TEXT
    RETURN_TYPE                 Field    TEXT
    ARGUMENT_TYPES              Field    TEXT
    ARGUMENT_NAMES              Field    TEXT
  SNDB_BY                       Table
    REFERRED_CLASS_NAME         Field    TEXT
    REFERRED_SYMBOL_NAME        Field    TEXT
    REFERRED_TYPE               Field    TEXT
    REFER_CLASS_NAME            Field    TEXT
    REFER_SYMBOL_NAME           Field    TEXT
    REFER_TYPE                  Field    TEXT
    [illegible]                 Field    TEXT
    LINE_NO                     Field    TEXT
    PATH                        Field    TEXT
    [illegible]                 Field    TEXT
    [illegible]                 Field    TEXT
  SNDB_IN                       Table
    SUB_CLASS_NAME              Field    TEXT
    SUPER_CLASS_NAME            Field    TEXT
    START_LINE_NO               Field    TEXT
    PATH                        Field    TEXT
    END_LINE_NO                 Field    TEXT
    ATTRIBUTES                  Field    TEXT
```

```
C:/Users/Administrator/Desktop/Database5.db
  JPDB_CLASS                    Table
    ACCESSOR                    Field    TEXT
    ABSTRACT                    Field    TEXT
    FINAL                       Field    TEXT
    NAME                        Field    TEXT
    EXTEND                      Field    TEXT
    IMPLEMENT                   Field    TEXT
    COMMENTS                    Field    TEXT
    START_LINE                  Field    INTEGER
    END_LINE                    Field    INTEGER
    FILE_PATH                   Field    TEXT
  JPDB_METHOD                   Table
    CLASS_NAME                  Field    TEXT
    ACCESSOR                    Field    TEXT
    STATIC                      Field    TEXT
    RETURN_TYPE                 Field    TEXT
    NAME                        Field    TEXT
    PARAMETER_TYPE              Field    TEXT
    PARAMETER_NAME              Field    TEXT
    RETURN_VALUE                Field    TEXT
    TYPE                        Field    TEXT
    START                       Field    TEXT
    CONDITION                   Field    TEXT
    VARIATION                   Field    TEXT
    ITERABLE                    Field    TEXT
    ITER_ELEMENT                Field    TEXT
    PARENT                      Field    INTEGER
    LINE_NO                     Field    INTEGER
    FILE_PATH                   Field    TEXT
  JPDB_INSTANCE_VAR             Table
    CLASS_NAME                  Field    TEXT
    ACCESSOR                    Field    TEXT
    STATIC                      Field    TEXT
    FINAL                       Field    TEXT
    RETURN_TYPE                 Field    TEXT
    NAME                        Field    TEXT
    INITIALIZATION              Field    TEXT
    LINE_NO                     Field    INTEGER
    FILE_PATH                   Field    TEXT
  JPDB_LOCAL_VAR                Table
    CLASS_NAME                  Field    TEXT
    METHOD_NAME                 Field    TEXT
    RETURN_TYPE                 Field    TEXT
    NAME                        Field    TEXT
    INITIALIZATION              Field    TEXT
    LINE_NO                     Field    INTEGER
    FILE_PATH                   Field    TEXT
  JPDB_REFERBY                  Table
    REFER_CLASS                 Field    TEXT
    REFER_MEMBER                Field    TEXT
    INSTANCE                    Field    TEXT
    STATIC                      Field    TEXT
    REFERRED_CLASS              Field    TEXT
    REFERRED_MEMBER             Field    TEXT
    LINE_NO                     Field    INTEGER
    FILE_PATH                   Field    TEXT
  JPDB_BADSMELL                 Table
    TYPE                        Field    TEXT
    NAME                        Field    TEXT
    SHOTGUNSURGERY              Field    TEXT
    COMMENTS                    Field    TEXT
    LAZYCLASS                   Field    TEXT
    DATACLASS                   Field    TEXT
    LARGECLASS                  Field    TEXT
    REFUSEDBEQUEST              Field    TEXT
    MIDDLEMAN                   Field    TEXT
    LONGPARAMETERLIST           Field    TEXT
    LONGMETHOD                  Field    TEXT
    FEATUREENVY                 Field    TEXT
    MESSAGECHAINS               Field    TEXT
    SWITCHSTATEMENTS            Field    TEXT
    DATACLUMP                   Field    TEXT
  JPDB_COUPLING                 Table
    REFER_CLASS                 Field    TEXT
    REFERRED_CLASS              Field    TEXT
    DATA                        Field    TEXT
    STAMP                       Field    TEXT
    CONTROL                     Field    TEXT
    EXTERNAL                    Field    TEXT
    COMMON                      Field    TEXT
    CONTENT                     Field    TEXT
```

Figure 5. The Comparing Source Navigator and JavaParser Extract Information

5. Conclusion

We apply the software engineering method to the block chain software. This paper propose a method to improve the complexity of blockchain software code. There is still a lack of research related to static analysis tools in blockchain codes. Through our visualization mechanism of reverse engineering, we visualized the complexity of the block-chain source code. This method allows static analysis of Java-based blockchain software source code. In addition, we developed static analysis tools suitable for object-oriented code analysis by expanding existing tools. Through a variety of quality indicators, blockchain developers can automatically visualize vulnerabilities in Java-based block-chain code. The advantages of the proposed method are as follows:

1) visualize the code inside. 2) Easily reduce the complexity of the code, and 3) prevent the possibility of bugs in the code. Future studies will analyze other languages used in Smart Contracts (e.g. Lua, Solidity, GO, etc).

Acknowledgement

This research was supported by Basic Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A3B03035421) and this research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2019-2017-0-01628) supervised by the IITP(Institute for Information & communications Technology Promotion).

Reference

[1] "The status of the global software market", SPRi, 2017, http://spri.kr/post/22205

[2] JaeHak, Jung, "Key to the era of the Fourth Industrial Revolution", Issue Report, NIPA, 2017

[3] Ji Hoon, Park, A Study on Automatic Visualization against Weakness and Vulnerability of Blockchain Code based on Static Analysis, Master's Thesis. Hongik University, Sejong-City, Korea, 2018.

[4] Porru, Simone, et al. "Blockchain-oriented software engineering: challenges and new directions." *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017.

[5] Jihoon, Park, Byungkook, Jeon, R. Young Chul Kim, Hyun Seung, Son, "Improving source code against bad-smell code patterns", *Journal of Engineering Technology*, Vol. *6, No. 2*, pp.503-516, 2018.

[6] Bo Kyung, Park, Ha Eun, Kwon, Hyun Seung, Son, Young Soo, Kim, Sang-Eun, Lee, R. Young Chul, Kim, "A Case Study on Improving SW Quality through Software Visualization", Journal of KIISE, Vol. 41, No. 11, pp.935-942, 2014.

[7] Sarker, Muktamyee. "An overview of object oriented design metrics." *From Department of Computer Science, Umeå University, Sweden* (2005).