doi:10.3233/ATDE230582

Research on Software Fault Feature Data Extraction Method for Software Fault Prediction Technology

Ran YAN [a,1], Meichen WANG [a], Zhaowei XU [a] and Kai ZHANG [a]
[a] *China Shipbuilding Software Quality & Reliability Testing Center, Beijing, China*

Abstract. The correlation between software failure characteristics and software failure directly determines the predictive performance of the failure prediction model. The extraction of software fault features is crucial for building equipment software fault prediction models, and is an important process to ensure accurate prediction. However, the software fault feature data extraction method is often complicated to use and has no pertinence to the selected software fault feature data, and it takes a lot of time to complete the extraction steps. This paper summarizes software metrics and software defect types based on research at home and abroad, and selects software metrics and software defect types that are suitable for equipment software. Using regular expression technology and CSV technology research the automatic extraction way of software fault features, and finally constructs a fault data set that can be used for software fault prediction models.

Keywords. Software fault feature data, data extraction, software metrics, software defect types, Element measurement

1. Introduction

For software failure prediction models, the independent variables of the model are software failure characteristics (or software metrics), and the dependent variables of the model are software defects (such as defect type, defect tendency, etc.). The prediction model mines and analyzes the historical defect data (including the metric value of each module of the software and the defect status of the module) to establish the mapping relationship between the software metric and the software defect, and then analyzes the defect status of the new software module. The correlation between software fault features and software faults directly determines the predictive performance of the fault prediction model. If the correlation between the software failure characteristics and the software defect is weak, the prediction performance of the prediction model is definitely not ideal. Furthermore, if the selected software fault features have a strong correlation with software defects, but there is little software defect data, resulting in insufficient model training, the performance of the prediction model cannot be guaranteed either. Therefore, the extraction of software fault features is crucial for building software fault prediction models.

[1] Ran YAN, Corresponding author, China Shipbuilding Software Quality & Reliability Testing Center, Beijing, China; E-mail: yanran_buaa@163.com.

2. Research Status

Software metrics play an important role as model independent variables in software fault prediction models. IEEE gives the definition of software metrics in "Standard for Software Quality Metrics Methodology": "A metric is a function whose input is software data and output is a single value. It can be used to explain the degree to which attributes affect software quality."[1]. According to IEEE Std. 1061-1992, software metrics are quantitative measurements of attributes that affect software quality.
Software metrics can be classified into three categories: product metrics, process metrics, and project metrics (such as CPU usage, memory usage). Among them, the first two types of metrics are most widely used in software fault prediction models.

2.1. Software Product Metrics

According to different programming languages, software product metrics can be divided into method-level metrics for process software and class-level metrics for object-oriented software. For process-oriented software, a method (or function) is usually called a software module. The measurement object is a single function, and the measurement element is called a method-level element. For object-oriented software, a class is usually called a software module, and the measurement object is each class. At this time, the measurement element is called a class-level measurement metrics (or object-oriented metrics).

Note that for software product metrics, in addition to the most commonly used method-level metrics and class-level metrics, there are several references in the literature that refer to file-level metrics [2], package-level metrics [3], and some other product metrics, for example, cascading style sheets metric [4], code smell metrics and so on. The following mainly introduces the commonly used method-level metrics of process-oriented software and class-level metrics of object-oriented software.

2.1.1. The Method-level Metrics of Process-oriented Software

According to the literature survey results, for process-oriented software, it is generally considered that a function/method is a module. Therefore, the value of the metric is counted in units of functions, and the prediction objects also refer to functions. Table 1 shows the commonly used method-level metrics.

Table 1. Commonly used method-level metrics.

Class	Metric
McCabe	Cyclomatic Complexity, Essential Complexity, Module Design Complexity
Halstead	Unique Operators, Unique Operands, Total Operator, Total Operands, Length, Program Volume), Program Level, Programming Effort, Programming Time
Line count	Physical Lines of Code, Lines of Comment, Lines of Blank, Lines of Mixed Code and Comment

2.1.2. The Class-level Metrics of Object-oriented Software

For object-oriented software, a class is generally regarded as a software module. Class-level metrics are also known as object-oriented metrics. The metric object of each class-level metric element is for each class, and the model prediction object is also each class. The most commonly used object-oriented metric elements in the fault data set of

public object-oriented open source software include: Chidamber-Kemerer (CK) metrics, QMOOD metrics, and other object-oriented metrics [5]. Table 2 shows the commonly used class-level metrics.

Table 2. Commonly used class-level metrics

Class	Metric
CK Metric Suite	Weighted Methods per Class, WMC
	Depth of Inheritance Tree, DIT
	Number of Children, NOC
	Coupling between object classes, CBO
	Response for a Class, RFC
	Lack of Cohesion in methods, LCOM
QMOOD Metric Suite	Number of Public Methods, NPM
	Data Access Metric, DAM
	Measure Of Aggregation, MOA
	Measure of Functional Abstraction, MFA
	Cohesion Among Methods, CAM
Tang Metric Suite	Inheritance Coupling, IC
	Coupling Between Methods, CBM
	Average Method Complexity, AMC
Jureczko Metric Suite	Lines of Code, LOC
	Maximum of Circle Complexity, MAX_CC
	Average of Circle Complexity, AVG_CC
Martin Metric Suite	Afferent Couplings, CA
	Efferent Couplings, CE
Henderson Metric Suite	Lack of Cohesion in Methods

2.2. Software Process Metrics

Software failures are not only related to code, but also closely related to code modification, developers and other elements in the software development process.

Graves proposed several code modification metrics to predict the number of software failures [6]. The author found that the code modification metrics are more conducive to predicting the number of failures than traditional code metrics. Hassan proposed History Complexity Metrics (HCM) by measuring the complexity of code modification, and conducted experiments on six failure data sets. The results show that HCM can be well used for software failure prediction [7]. The researchers designed 8 relative code churning metrics. The experimental results show that the relative code churning metrics are very beneficial to the prediction of module failure density. In addition, 18 kinds of code modification metrics are designed for software failure prediction according to the modification times, modified lines of code and other information. Some researchers [8] proposed Developer Micro Interaction Metrics (DMIMs) for software fault prediction by measuring the interaction information of software developers, and found in the experiment that the combination of developer interaction micro-interaction metrics Traditional class-level metrics can significantly improve the predictive performance of failure propensity models. Some researchers have tried to use the dependencies between modules to predict software module failures. For the first time, they proposed a metric based on the complexity of program module dependencies as an independent variable to predict the number of software failures.

2.3. Software Defect Classification

The purpose of software defect classification is to form multi-dimensional classification information for defect management and data support for defect

measurement analysis. At present, there is no unified standard for software defect classification. The existing software defect classification methods can be classified into three categories: defect classification methods proposed by famous scholars, classification methods proposed by large research institutions, and classification methods published in national standards.

3. Overall Research Proposal

Through investigation and reference to authoritative documents at home and abroad, the set of candidate defect types of equipment software is determined, and on this basis, the defect types that meet the characteristics of equipment software defects are screened.

Then, according to the selected equipment software measurement metrics set, use the software static analysis tool, *Testbed*, to statically scan the given equipment software project to obtain a static analysis report containing the measurement element information of each module of the software. Since the report is in HTML format , you need to use regular expression technology to extract the metric value of each module from the analysis report. At the same time, according to the test report provided by the tester, the defect information statistics table of each module of the software can be obtained through analysis and statistics. The name of the software module is the key word. The equipment software module measurement metadata table and the equipment software module defect statistics table can be correlated and matched, and then the equipment software fault data set can be obtained.

4. Researching on Extraction Method of Software Fault Feature Data

4.1. Selection of Software Fault Characteristic Data

4.1.1. Software Metrics

By analyzing the historical data of software failures, it is found that the software metrics are strongly correlated with software failures or defects, and are easily obtained in software development units and software evaluation units, as shown in table 3.

Table 3. Common Software Metrics.

Metrics	Scope	Metrics	Scope
CK Metric Suite		Procedure Exit Points	Procedure
Essential Knots	Procedure	Executable ref. Lines	File
Essential Cyclomatic Complexity	Procedure	Number of Procedures	File
Knots	Procedure	Total source Lines	File
Cyclomatic Complexity	Procedure	Total Comments	Procedure
Total Classes Declared	Class	Number of Globals	Procedure
Executable reformatted Lines	Procedure	Fan in	Procedure
Number of Basic Blocks	Procedure	Fan out	Procedure
Unreachable Lines	File	Invocation Path	Procedure
Number of Loops	Procedure	Procedure Formal Parameters	Procedure
Procedure Entry Points	Procedure	Global Variables	Procedure

4.1.2. Classification of Software Defect Types

Table 4. Software defect classification.

Metrics	Scope
requirement	Software Requirements Specification
programming	System design or software design, software architecture
procedure	software program
code	Irregular coding
orthers	Documents such as software plans, requirements, tests, users, operation manuals, user manuals or data files, etc.

According to the description in GJB2786A, the defect types are classified by the nature of defects, and the categories are requirement, programming, procedure, code and others, as shown in table 4. The subcategory is table 5.

Table 5. Subcategory of software defect.

Metrics	Scope	Metrics	Scope
Requirement	Requirement is incomplete, requirements/design is inconsistent		Unused procedural parameter
	Requirement is redundant		Array bound exceeded
	Requirement is unnecessary		The parameters of shifts in the prohibit shift operation is negative
	Requirement is incorrect		
	Requirement is ambiguous		Unsigned expression negated
	Requirement is unverifiable		Equality compare of floating point
	Requirement identification is incomplete		
Programming	Design is incomplete		Divide by zero
	Module planning is unreasonable	code	Reformated lines
	The allowance design does not meet the requirements		More than 7 parameters in Procedure
	Interface design is incorrect		Memory not freed after last reference
	Algorithm design is incorrect		
	Data structure design is unreasonable		Assignment operation in expression
	Interface design is unreasonable		
	Design identification is incorrect		Assignment operation in boolean expression
			Go to detected
	Function not implemented		Specification/standard is understood inaccurate
	Performance not achieved		
	Parameter initialization is not		Test file is incorrect
	Performed/Parameter initialization error		Data base/data file is incorrect
	Control flow error		Manual is incorrect
	Logical error		The tracking relationship is incorrect
Procedure	Identifier redefinition	others	
	Assignment error		Paragram comment is inaccurate
	Interface parameters do not match		
	Interface definition error		
	Interface failure/Exception handling error		
	Algorithm is error		
	Code is unreachable		

Considering the defects caused by many codes due to irregular coding format or non-compliance with coding rules in the software, this paper increases the defect categories of coding classes, referring to the content of GJB8114 "C/C++ Language Programming Security Subset", and adding mandatory rules to defects.

4.1.3. The Software Failure Metrics (Fault Features) and Software Defect Type Data

According to the actual software testing project, through the research and analysis of the authoritative literature related to the software measurement element, combined with

the standard, and the analysis of the characteristics of the equipment software code, table 6 is the adopted software failure metrics.

Table 6. Software failure metrics (failure features).

Metrics	Type	Metrics	Type
Executable reformatted Lines	Numeric	Lines, Code Comments/Exe. Lines	Numeric
Number of Basic Blocks	Numeric	Knots	Numeric
Average Length of Basic Blocks	Numeric	Cyclomatic Complexity	Numeric
Procedure Entry Points	Numeric	Essential Knots	Numeric
Procedure Exit Points	Numeric	Essential Cyclomatic Complexity	Numeric
Total Comments	Numeric	Procedure Structured (SPV)	Numeric
Comments in Headers	Numeric	Number of Loops	Numeric
Comments in Declarations	Numeric	Depth of Loop Nesting	Numeric
Comments in Executable Code	Numeric	Number of Order 1 Intervals	Numeric
Blank Lines	Numeric	Maximum Interval Nesting	Numeric
Executable reformatted Lines	Numeric	Reducible (Intervals)	Numeric
Total Comments/Exe. Lines	Numeric	Globals in Procedure	Numeric
Header Comments/Exe. Lines	Numeric	Fan in	Numeric
Declaration Comments/Exe	Numeric	Fan Out	Numeric

Through the investigation and analysis of authoritative documents related to software defect types, as well as the analysis of the characteristics of equipment software defects, the classification standards for equipment software defect types are shown in table 7.

Table 7. Adopted software defects types.

Defect Type	
Code is unreachable	Requirement is incorrect
Interface parameters do not match	Manual is incorrect
Function not implemented	Control flow error
Interface failure/Exception handling error	Interface design is incorrect
Parameter initialization is not performed/Parameter initialization error	Divide by zero
Logical error	Pointer assignment to wider scope
Handing boundary is incorrect	Equality comparison of floating point
Requirement is ambiguous	Assignment operation in expression
Design is incomplete	Assignment operation in boolean expression
Memory not freed after last reference	Goto detected
requirements/design is inconsistent	

A module has different types of failures. Supposing a module has *k* types of failure, denoted as $F_1, F_2, ..., F_k$, and the number of each fault is $n_1, n_2, ...n_k$.

- Select the fault type with the largest number of faults as the fault type of the module, and the number of faults is $n = \max\{n_1, n_2, \cdots, n_k\}$。
- If there are two or more fault types with the largest number of faults, one of the fault types will be randomly selected as the fault type of the module.

When the number of faults is counted, the number of faults of this module is $n = \sum_{i=1}^{k} n_i$.

4.2. Extract Software Fault Feature Data

In the research process, it is found that the measures related to software code quality introduced can be obtained from the tool software, such as *Testbed, Guest, etc*., Testbed tool has been widely used internationally, so we can obtain these measures from the HTML file generated by the Testbed tool. Figure 1 shows the block diagram of software fault feature data extraction.

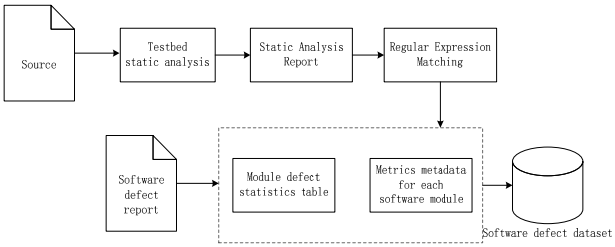


Figure 1. Principle of Software Fault Feature Data Extraction.

Regular expression technique is often used to retrieve and replace text that conforms to a pattern, or rule. Regular expressions are a logical formula for string operations, which using some specific characters defined in advance and the combination of these specific characters, to form a Rule String. This Rule String is used to express a filtering logic for strings. A regular expression is a text pattern that describes one or more strings to match when searching for text. The reason why regular expression technology can be used here is the HTML-formatted report file that should be generated for Testbed, where all the data is identified and described in a specific format. Regular expression techniques can be implemented in a variety of coding languages, such as Matlab, C/C++, etc.

After obtaining the metric metadata through regular expressions, we need to save the data output as a CSV file. CSV read and write technology is used here. CSV (Comma-Separated Values) files are one of the most popular text file formats for programs to share information and exchange data. Its files store tabular data (numbers and text) in plain text. Finally, the module name is used as the keyword, and the matching metric metadata table and the failure data table or the test report are associated.

Software metrics and fault data extraction steps are as follows:

- Use software testing tools, such as Testbed, to obtain static analysis reports in HTML format.
- Determine the software metrics to be extracted, and generate automatic extraction tools based on regular expression technology and CSV extraction technology to extract the software metrics.

Using the software module name field as the keyword, the two data tables can be matched and merged to obtain the final software failure data set.

4.2.1. Get the Static Analysis Report of the Software in HTML

Given the source code of a software project, first use Testbed for static analysis of the source code. Testbed is a professional software testing tool launched by LDRA company, which is powerful, comprehensive and easy to use, not only suitable for host

platform software testing, but also for embedded software testing, and has been successfully used in software testing departments of major research institutions at home and abroad. Testbed tool generates a static analysis report in HTML format, which can be obtained using a browser as shown in figure 2.

As shown in figure 2, the report records in detail the metric metadata of each module of the analyzed software. In the figure, Complexity Metrics represents the metric group name, which includes specific metrics such as Knots, Cyclomatic Complexity, etc. tsdview .cpp represents the file name, and the first column in the table indicates the names of the modules of the file. The green number indicates the value of the corresponding metric. The letter P in parentheses after the number indicates Past, meaning that the corresponding metric value does not exceed the threshold specified by the tool.

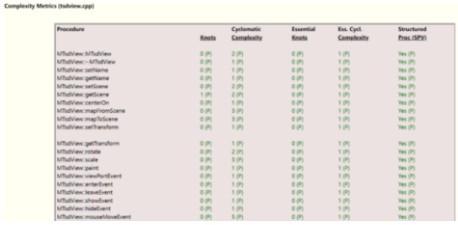


Figure 2. Metric metadata for modules in the Testbed static analysis report

Figure 3. HTML format analysis report source code

The source code of the above HTML file can be viewed directly using any text viewing tool, as shown in figure 3. Obviously, all data is uniformly identified and described in a specific format. Therefore, regular expressions can be used to extract the metric metadata required for this project.

4.2.2. Automatic Extraction Tool for Software Metrics and Software Failure Data Based on Regular Expressions

(1) Extract Metric Information Using Regular Expression Technique

Regular expressions are often used to retrieve and replace text that conforms to a pattern (rule). Regular expressions are a logical formula for string operations, that is, using some specific characters defined in advance, and the combination of these specific characters, to form a "rule string", this "rule string" is used to express a filtering logic for strings. A regular expression is a text pattern that describes one or more strings to match when searching for text. The reason why regular expression technology can be used here is the HTML-formatted report file that should be generated for Testbed, where all the data is identified and described in a specific format.

After obtaining the metric metadata, you need to save the data output as a CSV file. CSV read and write technology is used here. CSV (Comma-Separated Values) files are one of the most popular text file formats for programs to share information and exchange data. Its files store tabular data (numbers and text) in plain text. Plain text means that the file is a sequence of characters and contains no data that must be interpreted like binary numbers.CSV files consist of any number of records, separated by some kind of newline character; Each record consists of fields, and the separators between the fields are other characters or strings, most commonly commas or tabs.

Almost all programming languages support reading and writing CSV files, and most support reading and writing CSV files by directly calling internally defined functions. For example, the csvread function and the csvwrite function in MATLAB. Through programming, regular expressions can be used to extract the values of the metrics of each module of the software, as shown in figure 4. In the figure, the first column records the names of each module, and the remaining column headers represent the names of each metric. Each row in the table corresponds to a software module.

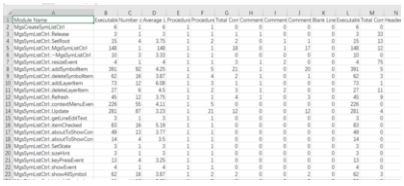
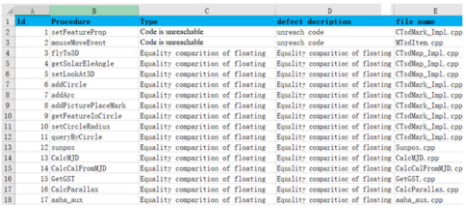


Figure 4. Module metric metadata extracted from a report file using regular expression techniques.



Id	Procedure	Type	defect decription	file name
1	setFeatureProp	Code is unreachable	unreach code	CTstMark_Impl.cpp
2	mouseMoveEvent	Code is unreachable	unreach code	WTstItem.cpp
3	flyTo3D	Equality comparition of floating	Equality comparition of floating	CTstMap_Impl.cpp
4	getSolarEleAngle	Equality comparition of floating	Equality comparition of floating	CTstMap_Impl.cpp
5	setLookAt3D	Equality comparition of floating	Equality comparition of floating	CTstMap_Impl.cpp
6	addCircle	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
7	addArc	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
8	addPicturePlaceMark	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
9	getFeatureInCircle	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
10	setCircleRadius	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
11	queryByCircle	Equality comparition of floating	Equality comparition of floating	CTstMark_Impl.cpp
12	sunpos	Equality comparition of floating	Equality comparition of floating	Sunpos.cpp
13	CalcMJD	Equality comparition of floating	Equality comparition of floating	CalcMJD.cpp
14	CalcCalFromMJD	Equality comparition of floating	Equality comparition of floating	CalcCalFromMJD.cp
15	GetGST	Equality comparition of floating	Equality comparition of floating	GetGST.cpp
16	CalcParallax	Equality comparition of floating	Equality comparition of floating	CalcParallax.cpp
17	asha_aux	Equality comparition of floating	Equality comparition of floating	asha_aux.cpp

Figure 5. Software test report.

(2) Associate metrics metadata and fault data to construct fault datasets

The principle is to use the module name as a keyword to correlate the matching metric metadata table and the failure data table, or test report. Specifically, after obtaining the metric metadata of each module of the software, the fault condition of each module is also required, that is, if there is a fault, the specific fault type needs to be known. For any software development enterprise, sufficient software testing is required before release, and software testers will compile software test reports, which include various description information related to software failures, such as the module in which the fault is located, the type of fault, and the test cases that stimulate the failure. Based on this information, the software module fault information statistics table shown in figure 5 can be obtained. As can be seen from the figure, the first column in the table is the fault ID, the second column is the name of the faulty module, the third column records the fault type, the fourth column is the description of the fault, and the fifth column is the file where the faulty module is located. Note that a software project typically contains multiple files, and a file contains multiple modules. The file name and module name together uniquely identify a specific module.

Obviously, both the software module metrics metadata table (shown in figure 2) and the software module failure statistics table (as shown in figure 5) contain software module name fields. Therefore, by using the software module name field as the keyword, the two data tables can be matched and merged to obtain the final software failure data set. The first column of the fault dataset is the name of each software module, the last column is fault type information, and the other columns represent individual metrics.

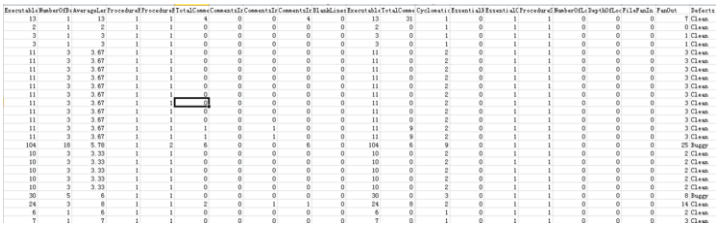


Figure 6. Software fault information table in CSV format.

Finally, the software module metric data table and the software module failure statistics table are combined into a CSV format table, as shown in figure 6.

4.2.3. Advantage

Integrating software fault information into CSV format has two advantages.

a) Almost all programming languages support the reading and writing of CSV files, and most of them support reading and writing CSV files by directly calling internally defined functions;
b) The integrated CSV file Id number, software measurement attribute value, fault type value can be directly converted into numerical type, and its file stores tabular data (numbers and text) in plain text, plain text means that the file is a sequence of characters, does not contain data that must be interpreted like binary numbers, providing good conditions for the call, reading, use and fast search of software failure prediction data.

5. Conclusion

Firstly, collect the failure data of various equipment software, and then analyze the characteristics of this data. Combined with domestic and foreign research, the software metrics and software defect types are summarized, and the software metrics and software defect types suitable for equipment software are selected. According to the above selected software metrics and defect types and the way to obtain data, a method of automatic extraction of software fault features is proposed by regular expression technology and CSV technology. Finally a fault dataset that can be used for software failure prediction model is constructed.

References

[1] Schneidewind N. IEEE Standard For A Software Quality Metrics Methodology Revision And Reaffirmation. In Proceedings of IEEE International Symposium on Software Engineering Standards. 1997; pp. 278-278.
[2] Yatish S, et al. Mining software defects: Should We consider affected releases. 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). 2019; pP. 654-665.
[3] Zhao Y, et al. Understanding the value of considering client usage context in package cohesion for fault-proneness prediction. Automated Software Engineering. 2016: 1-61. DOI: 10.1007/s10515-016-0198-6.
[4] Serdar BM and Diri B. Defect prediction for Cascading Style Sheets. Applied Soft Computing. 2016; DOI: http://dx.doi.org/10.1016/j.asoc.2016.05.038.
[5] Jureczko M and Spinellis DD. Using object-oriented design metrics to predict software defects. In DepCoS-RELCOMEX. 2010; pp. 69--81.
[6] Song Q, Guo Y and Shepperd M. A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Transactions on Software Engineering. 2019; 45(12): 1253-1269. DOI: 10.1109/TSE.2018.2836442.
[7] Torgo L, et al. SMOTE for regression. In Progress in Artificial Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg: Berlin, Heidelberg. 2013; p. 378-389.
[8] Lee T, et al. Developer micro interaction metrics for software defect prediction. IEEE Transactions on Software Engineering. 2016; 42(11): 1015-1035. DOI: 10.1109/TSE.2016.2550458.