A GQM APPROACH OF SOFTWARE ANALYSIS TO PREDICT MAINTAINABILITY

Sai Nidhin Kaminipaty	970831-5495	kasd17@student.bth.se	50%
Venkata Manoj Kumar Mandalaneni	961104-2517	mbvq17@student.bth.se	50%

Abstract—Software maintainability is adaption of software to meet the new requirements of customers. Maintainability is the one of the main attributes that is used for determining how ease is a software product can be modified and maintained. A maintainable software means a software which can be used for correcting defects, improving the performance and for enhancing other quality attributes. The reason for considering maintainability as significant factor is because more budget is required for software development in which the high priority is given to maintenance. Therefore, maintainability of the software plays a major role in overall software development. Now the present goal of our study is to extract the selected metrics in GQM framework of various Jedit versions concerned to the maintainability. A GQM framework is used in our empirical study which lists the goals, research questions for the metrics considered.

Keywords: Maintainability, GQM frameworks, Metrics, Tool IntelliJ IDEA, Metric Reloaded.

Introduction

Maintenance plays a vital role for overall development of software. Maintainability of software product refers to how ease a software product can be modified and maintained. Maintainability of the software is used to for maintaining the software to use for longer periods. Maintainability is the quality associated with the software product whereas maintenance is considered as a part of SDLC process [1]. In software development life cycle (SDLC) maintenance is given much priority and it is a critical phase where several modifications are made in software product. The modifications include correcting the defects, Improving the performance. If the complexity of the software product increases there is a need to predict and measure maintainability of software as it reflects the costs, as much cost is included by maintenance [2]. Software maintainability is a significant external attribute for determining the quality of software. To know the external attributes, we should use internal attributes. It can be done by evaluating the internal attributes and analyzing their values. The values are the metrics that can be obtained for the selected software using the metric extraction tools like Jhawk, IntelliJ IDEA, stan, CKJM etc.

In this project as described, we have to evaluate systematically the metrics of the JEDIT which is a open source text editor for programmers. The metrics to be evaluated are selected by GQM based framework. This is an empirical study which means the research with evidence that includes direct and indirect observation or experience. For this study we select the 11 versions of Jedit mentioned in the project and a case study is made on the modules to determine how maintainability is affected. Therefore, we obtain the selected metrics to identify the values and to analyze metrics of various Jedit software system for code structure, size, complexity and understandability.

RESEARCH METHODOLOGIES

The methodologies of our research include study, metric extraction tools, Statistical measures and visualization methods.

I)Study:

The study selected for the empirical study is case study by which we can concentrate on a unit or a group of test cases by using both qualitative and quantitative analysis. case study is used for combining the various aspects to relate for evaluating the goal. There are various approaches to perform a case study. The reason for selecting the case study is because it is flexible and provides a clear insight of the of our study. The metrics are evaluated at package level of jedit in this study.

ii)Metric extraction tools:

we have used metrics reloaded plugin to obtain the metrics at package level of various jedit versions. It is an open software metric tool. It is used for code inspection that detect and correct anomalous code in our project. It can find various problems like locate dead codes, finding bugs, spell mistakes and overall code structure.

The reason for selecting metrics reloaded is because it is easily integrating with intellij IDEA. It is available free open source tool present in IntelliJ IDEA plugin and is easy to install.

Hence the tool is ease of use to obtain various metrics that are selected in our empirical study. The metrics obtained from the tool metrics reloaded plugin are suitable for case study considered for our empirical study. The metrics are LCOM, LOC, Ca, Ce, CBO, CLOC, WMC, Average cyclomatic complexity and comments ratio. The tool also provides us with several options to calculate such as martin packaging metrics, complexity metrics, dependency metrics etc.

iii)Statistical measures and visualization methods:

In our project we have considered line graphs, bar graphs for the selected metrics of research questions in GQM tree. The graphs of internal attributes are used to measure the external attribute maintainability of software system under study. After measuring the maintainability, it is used for identifying the modules that are poor to maintain. The bar graphs are made for V(G), CBO, LCOM, WMC, CLOC and Comment Ratio. The line graphs are made for metrics Ca, Ce and LOC. These graphs are used to analyze the maintainability of the versions. The modules that are difficult to maintain can be identified.

Goal question metric (GQM) tree

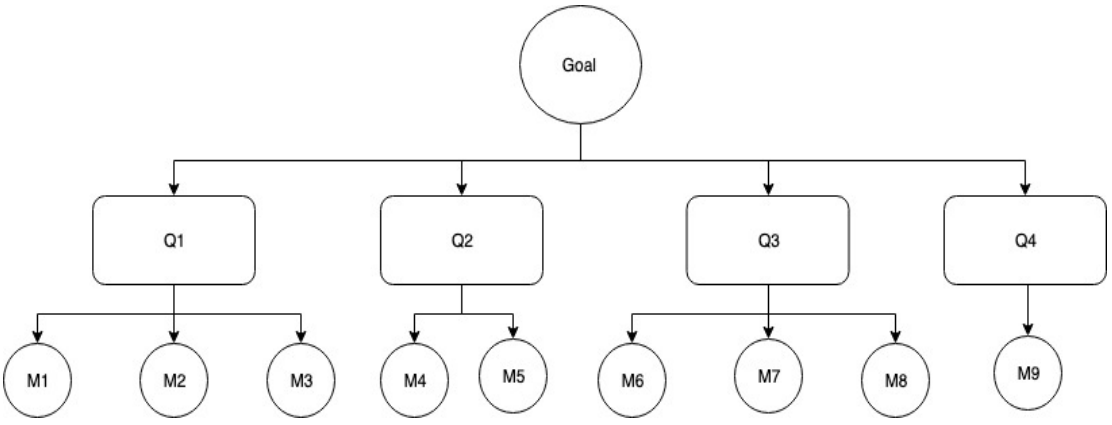
The overall goal of the study is to identify modules that are difficult to maintain.

Goal: To evaluate the maintainability of Jedit versions.

To identify the modules that are difficult to maintain.

Goal question metric (GQM) is an approach by which we can achieve our goal in a specific manner. It contains the questions formulated to achieve the goals. By answering the questions using suitable selected metrics our goals can be achieved. The following figure represents our project GQM tree as shown below:

GQM TREE



GQM Table

GOAL
To identify those jEdit modules that would be more difficult to maintain

QUESTIONS	
Q1	What are the Jedit modules that are frequently modified?
Q2	What are the Jedit modules that are having high complexity?
Q3	What are the Jedit modules that are highly coupled?
Q4	What are the Jedit modules that are with high cohesion?

METRICS	
M1	Lines of Code (LOC)
M2	Count Lines of Code (CLOC)
M3	Comments Ratio (CR)
M4	Average cyclomatic complexity (V(G))
M5	Weighted Methods per Class (WMC)
M6	Coupling Between Objects (CBO)
M7	Afferent Coupling (CA)
M8	Efferent Coupling (CE)
M9	Lack of Cohesion in Method (LCOM)

Description of selected metrics

These metrics made us to answer the questions we did according to the GQM tree. The literature is written below:

LOC: lines of code shows that how many lines of source code is there in our code and namespace, class and methods in our code. It also can used to find out the size of code unit and the size of the project.

CLOC: cloc counts the blank lines, comment lines, and physical lines of source code in many programming languages. It has many features to make it easy to use.

WMC: weighted methods per class it is sum of CC of all methods in class. It shows us that high of WMC. It reflects effort and time in maintaining the code. [3]

CBO: Coupling between the objects. The dependency between the objects in class. The highest value indicates that the objects are highly critical for modifications and it increases the maintenance effort. [4]

Ca: Afferent coupling it measures the classes that depends on the classes inside of packages. It is propositional to modify so maintainability increasing is declined. [5]

Ce: Efferent coupling it measures the total number of classes inside the component which are depends on the classes outside of component. The other name of this is Outside Dependencies.

LCOM: Lack of Cohesion Between methods. It is used to measure the cohesion between the classes and methods. It is useful for to measure the extent of relation between methods. Low cohesion metrics is difficult to maintain and test the code. [5]

V(G): Average cyclomatic complexity is used to define the structural complexity of the code. It can be calculated by doing the average of number of distinct paths in flow of the program. If average cyclomatic complexity has higher value it will be less maintainable because of more code coverage.

Comment Ratio: It is the ratio that Calculates the ratio of lines of comment code to total lines of code for the project. Lines of whitespace are not counted for purposes of this metric.

Relating the metrics to internal attributes

The internal attributes are described below:

Understandability: It measures the coupling between the classes and methods. The high maintainability can be achieved by decreasing the coupling value. [6]

Complexity: The maintainability is directly proportional to complexity. If the complexity is increased the cost, effort and time also increases. It increases the cost of maintenance of the software. [7]

Size: It is used to estimate the code size in the project. It can be known by counting the lines of code (LOC) and CLOC. If greater the size it indicates more effort is needed for maintenance. The changes can be identified by counting the lines of code. [5]

Structure: cohesion metrics are used to define the structure. If the cohesion value is high, then it is easily maintainable. If the cohesion value is low, then it is difficult to maintain.

Selection of scales:

Questions	**Entity**	**Attributes**	**Type of Attribute**	**Metrics**
Q.1 What are the Jedit modules that are frequently modified?	Packages	Size	Internal	LOC, COM_ RAT, CLOC
Q.2 What are Jedit modules that are having high complexity?	Packages	Complexity	Internal	V(G), WMC
Q.3 What are the Jedit modules that are highly coupled?	Packages	Understandability	Internal	CBO, Ca, Ce
Q.4 What are the Jedit modules that are with high cohesion?	Packages	Structure	Internal	LCOM

The metrics LOC, CLOC, CBO, Ca, Ce, WMC, V(G) and LCOM are of absolute scale while comment ratio (COM_ RAT) is the only one which is a ratio scale type.

Related work:

In this project we have consider Jedit versions which is an open source software for this study. GQM framework is selected for this empirical study which consists of Goal, Question and metrics. GQM frame work is used to identify the attributes, entities and mapping the attributes with the suitable metrics. A case study is also done on the selected metrics using metric extraction tools. The tool we considered is IntelliJ IDEA with Metric Reloaded Plugin. The selected suitable metrics are extracted, and the graphs are obtained for various packages that are present in JEDIT versions.

In the final part of the report the extracted metrics are analysed to know the maintainability of the packages present in Jedit versions. By this we identified the packages that are difficult to maintain.

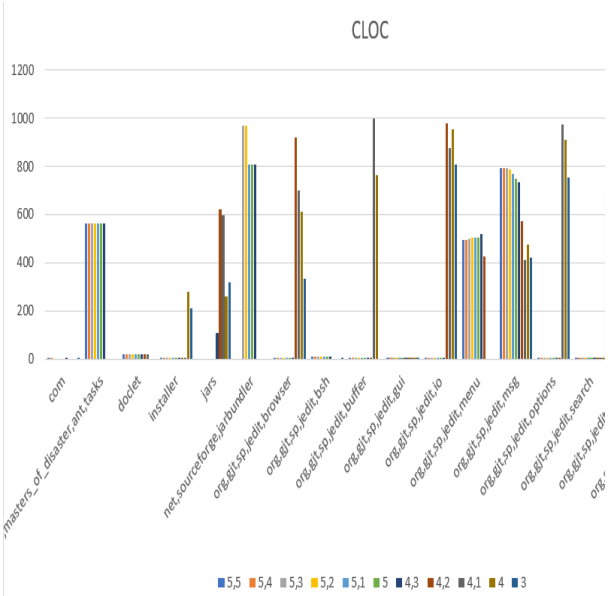
Results and analysis:

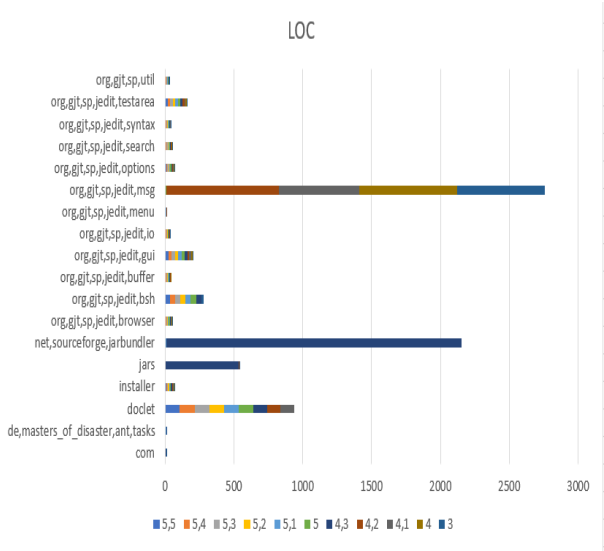
By analysing the metrics of the packages to the corresponding Jedit versions using IntelliJ IDEA with the Metrics Reloaded Plugin. We have answered the research questions of the GQM tree. As the all packages considered for all the Jedit versions. The metrics will decide the maintainability. By this we can identify the modules that are difficult to maintain. Therefore, the research questions are answered using the suitable metrics.

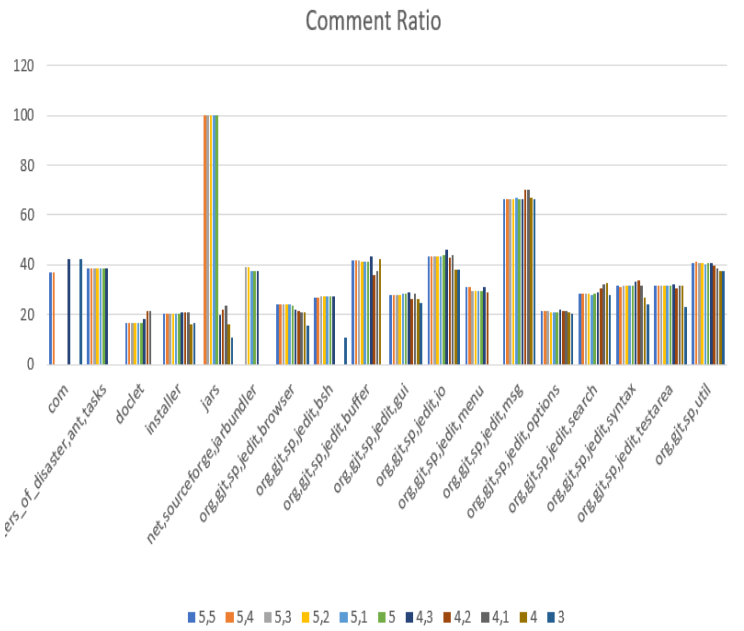
1. **What are the Jedit modules that are frequently modified?**

To identify the modules that are frequently modified. We have considered the metrics of Lines of code (LOC), Comment ratio

(COM_RAT) and Comment lines of Code (CLOC) increasing the values of these metrics states that the modules are modified. The following figures represents the graphical results of the metric data. Shown below:







For LOC the distortion of the lines represents the modifications of the packages. For CLOC and Comment Ratio the changes in the bar values represents the modifications made for the selected packages.

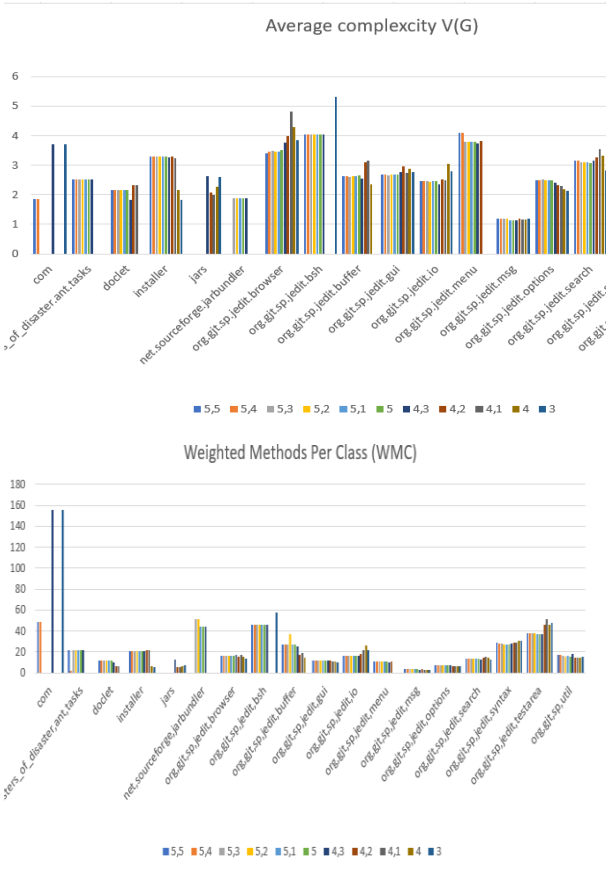
The results are given in table 1:

Module	**Changes in frequency level**
Com	Very Less
de.master-of-disaster.ant. tasks	Very Less
Docklet	High frequently
Installer	Less frequently
Jars	High frequently
Net.sourceforge. jarbundler	High frequently
Org.gjt.sp. jedit.browser	Less frequently
Org.gjt.sp.jedit. bsh	High frequently
Org.gjt.sp.jedit. buffer	Less frequently
Org.gjt.sp.jedit. gui	High frequently
Org.gjt.sp.jedit. io	Less frequently
Org.gjt.sp.jedit. menu	Very less frequently
Org.gjt.sp.jedit. Msg	High frequently
Org.gjt.sp.jedit. option	Less frequently
Org.gjt.sp.jedit. search	Less frequently
Org.gjt.sp.jedit. syntax	Less frequently
Org.gjt.sp.jedit. textarea	Less frequently
Org.gjt.sp.util	Very Less frequently

2. What are the Jedit modules that are having high complexity?

The Jedit modules that are having high complexity can be known using the selected suitable metrics average cyclomatic complexity V(G), WMC (Weighted methods per class). High complexity of the module requires more effort and cost to maintain. Higher the average cyclomatic complexity V(G) modifications of the code take large time. Similarly, higher the value of WMC it becomes difficult to maintain as the weighted methods per each class increases.

The following figures represents the graphical representation of V(G) and WMC



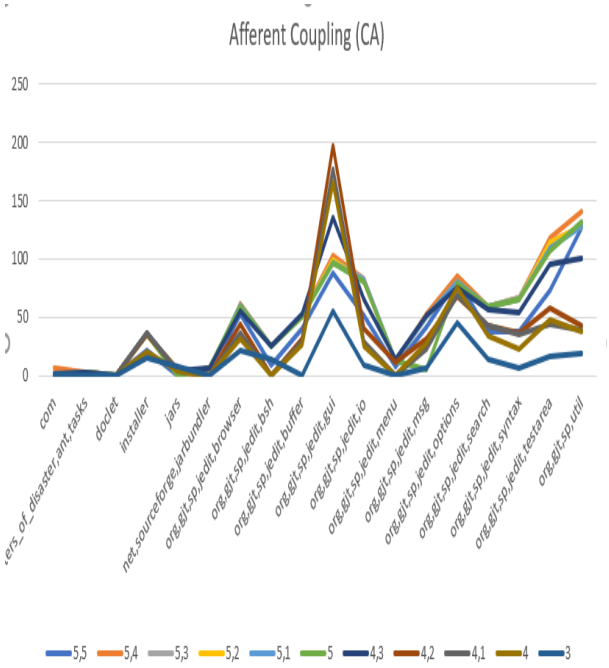
The results are given in table 2:

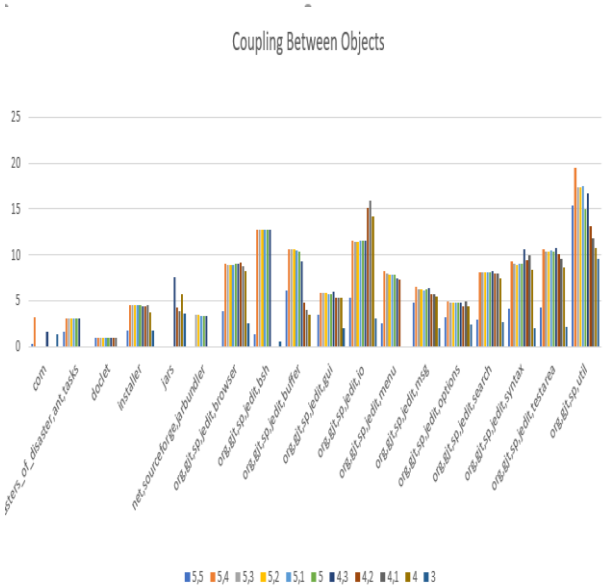
Module	Complexity rating level
Com	17
de.master-of-disaster.ant. tasks	14
Docklet	13
Installer	5
Jars	16
Net.sourceforge. jarbundler	18
Org.gjt.sp. jedit.browser	1
Org.gjt.sp.jedit. bsh	6
Org.gjt.sp.jedit. buffer	11
Org.gjt.sp.jedit. gui	9
Org.gjt.sp.jedit. io	10
Org.gjt.sp.jedit. menu	8
Org.gjt.sp.jedit. Msg	15
Org.gjt.sp.jedit. option	12
Org.gjt.sp.jedit. search	4
Org.gjt.sp.jedit. syntax	2
Org.gjt.sp.jedit. textarea	3
Org.gjt.sp.util	7

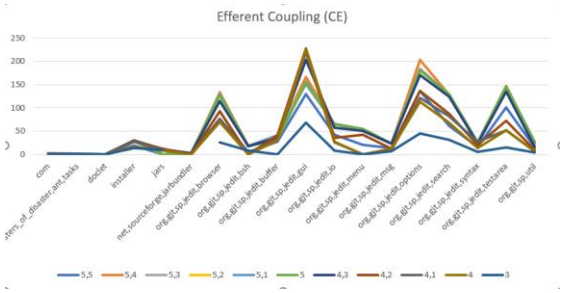
3. What are the Jedit modules that are highly coupled.

To identify the Jedit modules that are highly coupled we had considered the metrics CBO, Ca and Ce. If the coupling between objects (CBO) increases there will be more dependencies which reflect the decreasing maintainability. Similarly, increasing in the values of Afferent Coupling (Ca) and Efferent Coupling (Ce) decreases the maintainability.

The following figures represents the graphical representation of CBO, Ca and Ce







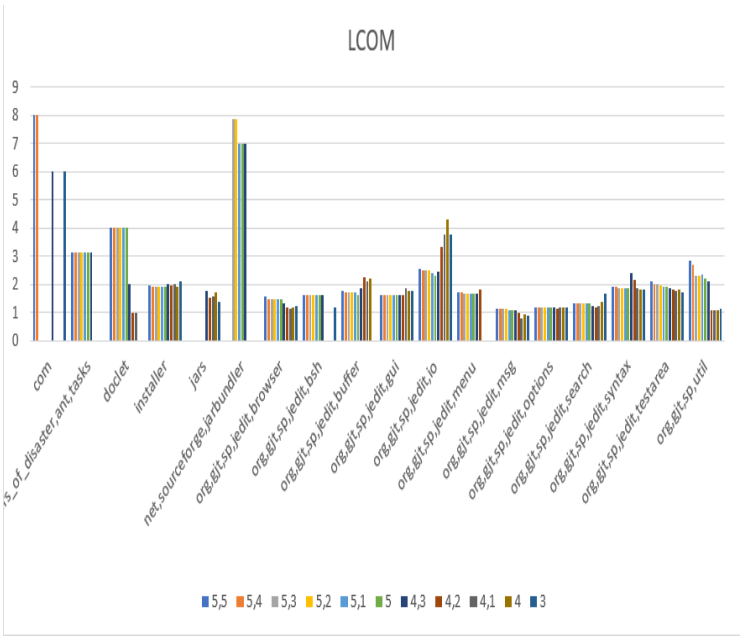
The results are given in table 3:

Module	**Rating**
Com	18
de.master-of-disaster.ant. tasks	15
Docklet	17
Installer	13
Jars	14
Net.sourceforge. jarbundler	16
Org.gjt.sp. jedit.browser	5
Org.gjt.sp.jedit. bsh	7
Org.gjt.sp.jedit. buffer	6
Org.gjt.sp.jedit. gui	11
Org.gjt.sp.jedit. io	2
Org.gjt.sp.jedit. menu	10
Org.gjt.sp.jedit. Msg	9
Org.gjt.sp.jedit. option	12
Org.gjt.sp.jedit. search	8
Org.gjt.sp.jedit. syntax	4
Org.gjt.sp.jedit. textarea	3
Org.gjt.sp.util	1

4. What are the Jedit modules that are with high cohesion?

To identify the modules that are having high cohesion. The selected metric is Lack of Cohesion Methods (LCOM). Cohesion metrics are used to define the internal attribute structure. It is used for measuring the extent of relation between the methods. If the cohesion value is high, then it is easily maintainable. While low cohesion results in less maintenance.

The following figure represents the graphical representation of LCOM shown below:



The results are given in table 4:

Module	Rating
Com	4
de.master-of-disaster.ant. tasks	5
Docklet	3
Installer	6
Jars	18
Net.sourceforge. jarbundler	1
Org.gjt.sp. jedit.browser	12
Org.gjt.sp.jedit. bsh	16
Org.gjt.sp.jedit. buffer	10
Org.gjt.sp.jedit. gui	11
Org.gjt.sp.jedit. io	2
Org.gjt.sp.jedit. menu	14
Org.gjt.sp.jedit. Msg	17
Org.gjt.sp.jedit. option	15
Org.gjt.sp.jedit. search	13
Org.gjt.sp.jedit. syntax	7
Org.gjt.sp.jedit. textarea	9
Org.gjt.sp.util	8

Reflections:

From this project we found the modules that are difficult to maintain using the GQM empirical study which consists of metric case study. There are five modules that are selected according to analysis of metrics from the research questions.

The modules are that are difficult to maintain are *org.gjt.sp.jedit.browser, org.gjt.sp.jedit.bsh, org.gjt.sp.jedit.textera.*

Conclusion

In this project the work we done is to know the modules that are difficult to maintain of the Jedit versions. The latest 11 versions of open software of Jedit are considered. For each of the version we had evaluated the metrics which we have selected. For this to obtain the metrics we had chosen the IntelliJ IDEA tool and Metric Reloaded Plugin. We framed the GQM which is known as Goal Question Metrics in that pattern we have done our work. To extend this work we can set up of more goals and analysis can be performed to the Jedit versions.

References

[1] "Virtual Machinery - Sidebar 3 - WMC, CBO, RFC, LCOM, DIT, NOC - 'The Chidamber and Kemerer Metrics.'" [Online]. Available: http://www.virtualmachinery.com/sidebar3.htm. [Accessed: 22-May-2018].

[2] G. Kaur and D. Sharma, "A Study on Robert C. Martin's Metrics for Packet Categorization Using Fuzzy Logic," Int. J. Hybrid Inf. Technol., vol. 8, no. 12, pp. 215–224, Dec. 2015.

[3]A. Kaur, K. Kaur, and K. Pathak, "A proposed new model for maintainability index of open source software," in Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014, pp. 1–6.

[4] S. Burger and O. Hummel, "Applying maintainability-oriented software metrics to cabin software of a commercial airliner," Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR, pp. 457–460, 2012.

[5] S. K. Dubey and A. Rana, "Assessment of maintainability metrics for object-oriented software

system," ACM SIGSOFT Softw. Eng. Notes, vol. 36, no. 5, pp. 1–7, 2011.

[6] A. Shafiabady, M. N. R. Mahrin, and M. Samadi, "Investigation of software maintainability prediction models," Int. Conf. Adv. Commun. Technol. ICACT,

vol. 2016–March, pp. 783–786, 2016.

[7] C. van Koten and A. R. Gray, "An application of Bayesian network for predicting object-oriented software maintainability," Inf. Softw. Technol., vol. 48, no. 1, pp. 59–67, Jan. 2006.

[8] A. Kaur, K. Kaur, and K. Pathak, "Software maintainability prediction by data mining of software code metrics," 2014 Int. Conf. Data Min. Intell. Comput. ICDMIC 2014, pp. 1–6, 2014.

[9] "IntelliJ IDEA: the Java IDE for Professional Developers | JetBrains." [Online]. Available:

https://www.jetbrains.com/idea/. [Accessed: 25-

May2017].

[10] "MetricsReloaded :: JetBrains Plugin Repository." [Online]. Available:

https://plugins.jetbrains.com/plugin/93metrics reloaded. [Accessed: 25-May-2017].

Appendix