Vincati and GM Node Flip Metrics

Rishi Reddy Cheruku
Department of Computer Science, University Of Akron
rc81@zips.uakron.edu

Abstract – *Software metrics play very crucial role in every project. It helps in maintaining the quality, complexity of the project. Well-developed software will reduce the maintainability cost of the project. So calculating the software measure during the development stage itself it is a good practice. Calculating the metrics manually is a time consuming and can also be inaccurate in a complex situation.*
So selecting the right tool and right set of metrics also plays a crucial role. Within this paper, we start by giving an overview of the few metrics in particular, then we give a brief overview of the project and the metrics calculated from the projects and we continue further by analyzing the calculated metrics and suggestions with the projects. In summary, the paper aims at illustrating the metrics applied to the VINCATI and GM Node Flip projects and suggest few improvements in the project.

I. Introduction

Software Engineering, as a discipline, needs many of its metrics be applied in software measurements and demands a standard scale of measure against which many inferences and decisions could be made, from product to process metrics [1]. It is becoming important for many organizations to measure the process. Only by measuring the engineering process and its products can we quantitatively assess the merit of process improvements [2]. A metric is a measure to evaluate the software projects. Metrics will help in understanding the workflow and technical activities of a program. The measure of the metrics vary from those which measure the people involved in the process and the time taken to implement, testing to that of the software code. In this project I considered some quantitative measures to get great insights of project performance to characterize, to improve, to evaluate and to predict. Usually metrics include internal attributes like Cost and Effort, lines of code and few external measures like complexity, maintainability, efficiency and reliability.

Truly speaking, lines of code (LOC) differ for each project and can be considered as function of complexity because it is dependent on the programmer and language. So, LOC can implicitly be a size oriented metric. Moreover, external measures are considered productive as they define exact project performance. The most appropriate use of metrics can also be inferred when measures are no longer relevant. Moreover, we can replace them in order to achieve the goal and the make an environment changes around it. A better use of metrics and frequent inspecting the changes will ensure trends to accelerate and decelerate constantly for the success of project.

Also metrics help to predict the defects in code and used to determine the code quality. And also metrics analysis varies from being simple to being very complex i.e. depends on the type of the project and its processes. But it's always important to upgrade on the metrics used in the software development which improves the on time objective decision making. Metrics have a purpose and a place in organizations and teams. And now a day's organizations are proposing the metrics driven improvement programs during the project development. Ensuring these improvements will deliver the meaningful results and determining the effectiveness of the project. And, applying such kind of measures to my project helped me to understand the project benefits and challenges involved.

This paper discusses the results of the project which can be achieved by choosing appropriate metrics and making an analysis on those metrics.

II. Static Code Analysis Tools

Since the projects are developed in PL/SQL, The tools which are being used to calculate the metrics as ClearSQL and Code Analysis for Toad. ClearSQL is a code review and quality control tool for Oracle PL/SQL. It generates a set of quality control code metrics such as Cyclomatic Complexity, Maintainability Index, Halstead Volume, etc. to identify potential problem areas based on complexity, size and modularity [3]. We can also generate

Flowcharts, call trees for each method. All the metrics generated by the tool can be exported to various formats. We can get a 30 day trial version of this tool and can be downloaded from dell software's.

Code Analysis is another tool developed by Toad. It is automatic analysis and code review tool which helps to ensure that the performance, maintainability, and reliability of code meets and exceeds the best standards. It also gives us the Create, Retrieve, Update and Delete (CRUD) matrix which helps to identify any tables which are not used, as well as tables which may be used heavily, and could therefore be a performance bottleneck [4]. We can get a 30 day trial version of this software and can be downloaded from dell software's.

III. Software Metrics

The software development fields as advanced and calculating metrics manually is a time consuming process and can also be inaccurate. Now days there are good number of tools which calculate the metics on fly. The following are list of some software metrics used to measure the software complexity:

Lines of Code (LOC):

It is a software metric used to measure the size of a software program by counting the number of lines in the program. This is used to predict the efforts that will be required to develop a program. 1000 lines of code are represented as KLOC. Few efforts that can be calculated are

The below differentiate the possible issue for the calculated statement metrics [4].

- 0 - 100 Small: Further functional decomposition probably unnecessary, cohesion probably not an issue
- 101 - 250 Medium: Probable candidate for functional decomposition, some cohesion improvements may exist.
- > 250 Large: Likely candidate for functional decomposition, numerous cohesion improvements probably exist

The LOC is easy to understand, fast to count, and independent of the program language [7].

Halstead volume:

The Halstead complexity measurement is used to measure a program module's complexity directly from source code, with emphasis on computational complexity [4]. The measures were developed by the late Maurice Halstead as a means of determining a quantitative measure of complexity directly from the operators and operands in the module [5]. Among the earliest software metrics, they are strong indicators of code complexity. Because they are applied to code, they are most often used as maintenance metric.

The Halstead measures are based on four scalar numbers derived directly from a program's source code:

n1 = the number of distinct operators
n2 = the number of distinct operands
N1 = the total number of operators
N2 = the total number of operands

Then, the program volume V is defined as

$$V= (N1+N2)\log_2(n1 + n2)$$

Halstead's volume V describes the size of the implementation of an algorithm. The Halstead's effort measure predicted that it would take longer to produce the initial part of the program than the entire program, and it's doing so raises serious questions about its use as a syntactic complexity measure [6].

Table 1. Lists the Halstead volume and complexity evaluation for each value assigned [8].

Halstead Volume	Complexity Evaluation
0 -1000	An average programmer should be able to comprehend and maintain this code
1001-3000	More senior skills most likely required to comprehend and maintain this code
>3000	Candidate for re-design or re-factoring to improve readability and maintainability

Cyclomatic complexity:

Cyclomatic complexity is the one of the most widely used software metrics which gives an analysis of software attributes like reliability and development effort. Given the increasing costs of software development, McCabe considered that a 'mathematical technique that will provide a quantitative basis for modularization and allow us to identify software modules that will be difficult to test or maintain' was required [9]. Cyclomatic complexity is also referred to as program complexity, or as McCabe's complexity. The Cyclomatic complexity is widely used metric and it is independent of the language or the language formats it being used.

Cyclomatic complexity measures the amount of decision logic in a single software module. It is used has two related purposes in the First, it gives the minimum number of recommended tests to be performed on the software. Second, it is used throughout the software test life cycle, beginning with design where the projects high level design is being implemented, to keep software reliable, testable, and manageable. It is mainly based on the structure of software's control flow graph.

McCabe Cyclomatic complexity directly maps to the bug count, so having a class with very high complexity value will definitely have bugs in it. So if while coding if you are not sure whether the method is large enough the McCabe will let you know whether the method is large or small. So this can be used as a diagnostic. It gives you the area of the code where you need to spend more time on testing. It can be used to get the review on the code as its being developed, if the value cross the threshold then it can be reviewed. It is used to get how many test cases should be covering a method.

The implicit model appears to be that the decomposition of a system into suitable components (or modules) is the key issue [9]. The decomposition should be based upon ease of testing individual components. Testing difficulty is entirely determined by the number of basic paths through a program's flow graph [9]. A large number of programs can been measured, and ranges of complexity can be analyzed for various versions of the software. The resulting measure can be used in development and maintenance, to develop estimates of risk, cost, or program stability. It is also analyzed that a correlation between the program's cyclomatic complexity and its error frequency. A low cyclomatic complexity values indicates that the program's understandability and indicates that it has less chances of risk than a more complex program.

It defines the complexity of a program and also measures the number of linearly independent paths which can be computed using control flow graph. It's easier to understand and need fewer efforts to modifications if we can have lower cyclomatic complexity.

It can be calculated as:

Cyclomatic complexity = E – N + P (for a directed graph)

Cyclomatic complexity = E – N + 2P (for a strongly connected graph)

where,

E = no of edges in a flow graph
N = no of nodes in a flow graph
P = no of nodes that have exit points (connected components).

Table 2. Illustrates the values for Cyclomatic Complexity and Risk Evaluation for the value. [8].

Cyclomatic Complexity	Risk Evaluation
1-10	A simple program, without much risk
11-20	A more complex program, with moderate risk
21-50	A complex, high risk program
>50	An un-testable program with very high risk

Maintainability Index:

The Maintainability Index (MI) is a Quantitative measurement of an operational system's maintainability is desirable both as an instantaneous measure and as a predictor of maintainability over time [8]. The MI of code can be tracked at regular bases and this helps in the development and maintenance of the projects as it progresses. The table 3 gives the possible evaluation rule for a given Maintainability index value. So after calculating the metric the table will give an scope of what issues would the project have like high maintainable means the project needs to be maintained daily and is very risk prone.

Table 3. Threshold values used for Maintainability Index

Maintainability Index	Maintainability Evaluation
0 - 64	Difficult to Maintain
64 – 84	Moderate Maintainability
>85	Highly Maintainable

Few other metrics used Toad Code Rating (TCR) will actually give a value which is calculated based on all the above metrics. Having a value 1 indicates the file is in good state. 2 indicate it is in average, 3 indicate it is below average, 4 indicate it is in bad state [8].

IV. VINCATI Automation

The vincati file is used to complete the VIN decoding by filling in missing data, to replace incorrect data, or to determine correct data when multiples are found in the decoding process. The VIN number is unique combination of number and alphabets which any vehicle has with which the type, year, catalog, model and manufacturer can be deduced. The Vincati Automation will automate this process to find the issues with decoding the VIN to the correct catalog. This process has also been extended to identify when a new model string code may need to be published in VINCATI data. The VINCATI automation tool will automate the entire process and display the issues with the VIN.

This tool has two versions where in the 1st version was developed only for North America operating unit to find the correct catalog and RPO codes. The 2nd version was to improve the performance by adding the indexes to the database and make the tool even include global data that is checking for issues with the VIN number for all the GM vehicles throughout world.

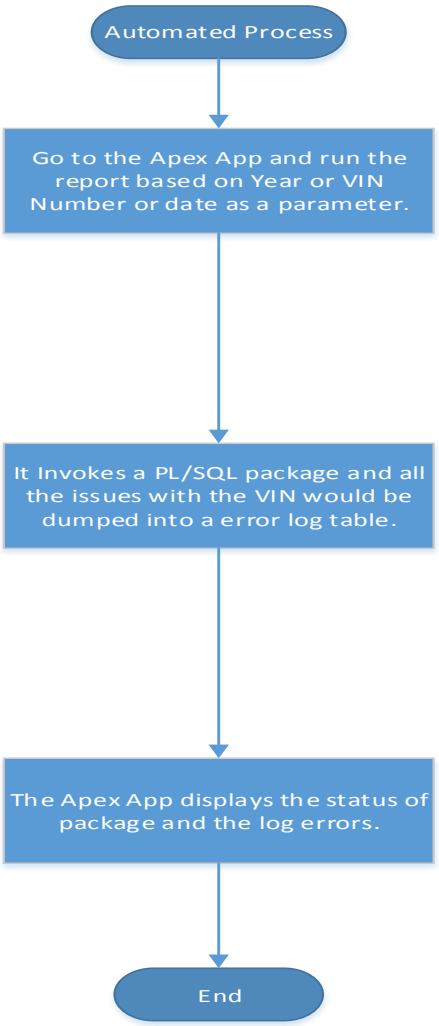

Figure 1. Illustrates the high level flow of data for the VINCATI Automation.

So we start with version one for the project. The project has three files the .sql file creates the front end creation. The package body and specification holds the logic for the backend. We analyze the metrics for the whole project.

In this section we have the following naming convention:
“CM” - Cyclomatic Complexity
“ST” - Statements
“TCR” - Toad code Rating
“HV” - Halstead Volume
“MI” - Maintainability Index
“GVAA”- GM_VINCATI_AUTOMATION_APEX.sql
“GVAS” - GM_VINCATI_AUTOMATION.pks

"GVAB" - GM_VINCATI_AUTOMATION.pkb

Table 4. Illustrates the metrics for the version 1 of the project VINCATI Automation.

Name	ST	TCR	HV	CM	MI
GVAA	2008	2	655.7	2	104.11
GVAS	0	1	1.00	1	100.00
GVAB	203	3	3014.52	20	64.63

The table gives the metrics for the VICNATI automation for all the files being used. Having the high statement value of shows that the methods are not interdependent. The Halstead's volume for the project is high which also makes it even tougher for the developer to refactor as the project size increases.

Table 5. Illustrates the metrics for all the methods in version 1 of project VINCATI Automation

File Name	ST	HV	CM	MI
Weighted mean	40	3015	23	62.62
GM_VINCATI_AUTOMATION	118	5626	34	40.99
ERROR_LOG_WRITE	7	309	2	109.20
VIN_DECODE_FUNCTION	11	775	2	97.10
VIN_CATALOG_FUNCTION	45	939	13	70.75
VIN_RPO_FUNCTION	32	707	6	79.35

The Halstead volume is very high for one of the method which as a good scope to refactor the method. Having too much code in a single method makes the code smell. This will hinder the comprehension and reuse of the method. We can refactor the method by breaking it to smaller methods that is by using extract method refactoring. The maintainability index is also more than the threshold value which makes the methods difficult to maintain as the method size increases.

Considering the 2nd version of the project. There were changes to the database to improve the performance of the project. So the indexes were added to the tables to fetch the data faster. A new method was added to increase to make the project work for global operating units and changes in other methods so they pull the global data. The performance did increase a lot previously fetch of 15 million records used to take 30 minutes after the increasing the performance it completes in 5 minutes.

Now we get metrics for the 2 version of the project and do analyze the metrics.

Table 6. Metrics calculated for the version 2 of the Project VINCATI Automation.

Name	ST	TCR	HV	CM	MI
GVAA	2096	2	645.36	2	104.68
GVAS	0	1	1.00	1	100.00
GVAB	244	3	2386.06	15	66.99

Having 2096 statements clearly shows that there are cohesion improvements. There are lots of duplicate operands uses in the package so that the terminology of the variables is consistent throughout the package. This increases the number of operands count.

Table 7. Metrics Calculated for all the methods on version 2 of the ds in project.

File Name	ST	HV	CM	MI
Weighted mean	40	2386	15	66.99
GM_VINCATI_AUTOMATION	103	4303	21	47.58
ERROR_LOG_WRITE	7	309	2	109.20
VIN_CATALOG_FUNCTION	65	1413	17	61.75
VIN_RPO_FUNCTION	40	857	9	74.05
VIN_DECODE_FUNCTION	11	775	2	97.10
VIN_DATASOURCE_FUNCTION	18	121	5	98.09

We analyze the methods after calculating the metrics. The GM_VINCATI_AUTOMATION method has large number of operands and operators which has a very good scope to improve the readability and maintainability of the method. The Maintainability index of the method is below the threshold values

which make it difficult for any developer to understand. The method could be refactored by extracting it to smaller methods by having the same functional logic but with smaller methods.

Comparing the version 1 with version 2:

There were definitely few improvements when two versions of the projects are compared.

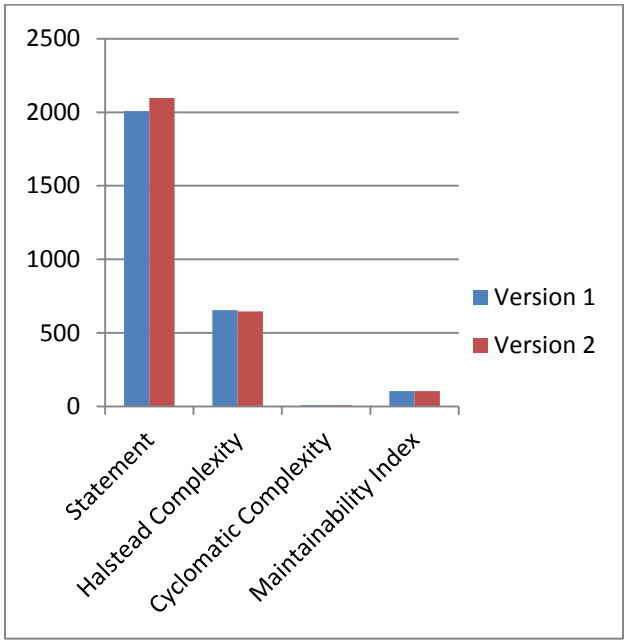

Figure 2. The above graph illustrates the sql file comparison between version 1 and 2.

The figure 1 gives the metrics compared for the project from version 1 and version 2. The numbers of statements are increased overall. As the package is modified to even handle the global data which increases the number of parameters too. This is kind of code smell which can be improved by replacing the parameter with a method. The halstead complexity, cyclomatic complexity and MI are same between the two versions.

The Figure 2 illustrates the package comparison between the two versions. It clearly shows that the halstead complexity has decreased a lot from the version 1 but it is still higher than the threshold value. The maintainability index and cyclomatic complexity are same.

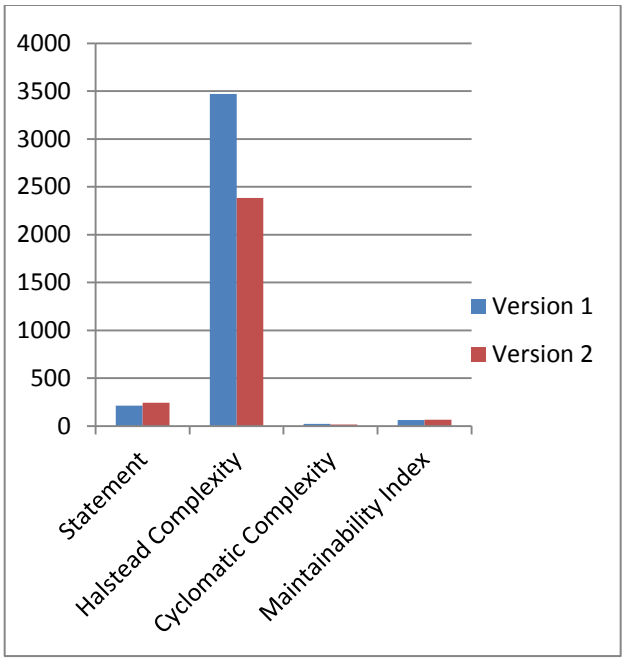

Figure 3. The above graph illustrates the package comparison between version 1 and 2.

Though the performance of the version 2 of the project as increased but there are many more changes that can be done like refactoring the methods and parameters which will reduce the complexity of the methods and I strongly believe that the performance will be improved further.

V. GM Node Flip

The GM Flip Flop where the GM team has two servers one used internally on which all the changes would be pushed and on other the entire GM users use. So at some point of time they want to swap the servers. For this a GM Node Flip app was built to swap the servers by the team themselves previously which was done by the DBA.

In this section we have the following naming convention:

“CM” - Cyclomatic Complexity
“ST” - Statements
“TCR” - Toad code Rating
“HV” - Halstead Volume
“MI” - Maintainability Index
“GNFA” - GM_NODE_FLIP_APEX.sql
“GNS” - GM_NODE_FLIP.pks
“GNB” - GM_NODE_FLIP.pkb

Table 8. Metrics calculated for the GM Node Flip project.

Name	ST	TCR	HV	CM	MI
GNFA	1991	2	1346.28	2	95.75
GNS	0	1	1.00	1	100.00
GNB	45	2	830.04	16	81.97

Having 1991 statements clearly shows that there are numerous cohesion improvements.
Having a Halstead volume 1346.28 shows that we can refactor the file to improve the readability and maintainability.

Table 9. Metrics calculated for various methods in the project.

File Name	ST	HV	CM	MI
Weighted mean	22	830	16	81.97
LIST_ACTIVE_APPLICATION_NODE_V	11	82	3	108.55
SET_SCHEMA_NODE	34	1072	20	72.99

The SET_SCHEMA_NODE method can be tuned little more to reduce the complexity.

VI. Analysis based on the Metrics

The projects has few long methods, few code smells in the methods. This can be resolved by refactoring the method that is extracting the methods. With this we can reduce the complexity of the code. The Halstead volume for few methods is very high which directly indicates that the method is too complex and needs to be refactored. So in general the GM_VINCATI_AUTOMATION method in both the versions has high complexity and needs to be refactored.The sql file has high cohesion in both the versions GM_VINCATI_AUTOMATION_APEX.sql there are many independent anonymous blocks which need to be interlinked so that the cohesion can be improved among these methods.

VII. Conclusion

Thus, using efficient software tools and project related metrics will improve software quality, increase the productivity and reduce the overall development time. Well maintained code will always be easier to read and maintain. So metrics will play a lead role in determining the project performance and can be effective in any organization to predict the success of project. Re-structuring the methods with more complexity will improve the readability of the code. Preferably the metrics that I used in this project are helpful in determining the project success along with statistics.

VIII. References

[1]. Justus, S.; Iyakutti, K., "Measurement Units and Scales for Object-Relational Database Metrics," *Emerging Trends in Engineering and Technology, 2008. ICETET '08. First International Conference on*, vol., no., pp.514, 518, 16-18 July 2008

[2]. Brooks, C.L.; Buell, C.G., "A tool for automatically gathering object-oriented metrics," Aerospace and Electronics Conference, 1994. NAECON 1994., Proceedings of the IEEE 1994 National , vol., no., pp.835,838 vol.2, 23-27 May 1994

[3]. "ClearSQL 6.9", http://www.myclearsql.com/ , (Accessed: 3 December 2014)

[4]. "Code Analysis of Toad", http://www.toadworld.com/ (Accessed 30 November 2014)

[5]. Halstead, M.H., Elements of Software Science, Elsevier, New York, 1977.

[6]. Rajaraman, C.; Lyu, M.R., "Reliability and maintainability related software coupling metrics in C++ programs," *Software Reliability Engineering, 1992. Proceedings. Third International Symposium on* , vol., no., pp.303,311, 7-10 Oct 1992

[7]. Sheng Yu; Shijie Zhou, "A survey on metric of software complexity," *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on* , vol., no., pp.352,356, 16-18 April 2010

[8]. "Description about the metrics used in Code Analysis tool", http://documents.software.dell.com/, (Accessed: 4 December 2014)

[9] Shepperd, M., "A critique of cyclomatic complexity as a software metric," *Software Engineering Journal*, vol.3, no.2, pp.30,36, Mar 1988