Volume 4, 2020 ISSN: 2617-4537
DOI:https://doi.org/10.31058/j.data.2020.42001

A Review on Software Testing Metrics

Bindia Tarika[1*]

[1] Computer Science & Engineering, PTU, Punjab, India

Email Address
bindiatarika11@gmail.com (Bindia Tarika)
*Correspondence: bindiatarika11@gmail.com

Received: 1 December 2019**; Accepted:** 25 February 2020**; Published:** 24 March 2020

Abstract:
Software testing metrics or software test measurement is the quantitative indication of extent, capacity, dimension, amount or size of some attribute of a process or product. A measurement is an manifestation of the size, quantity, amount or dimension of a particular attributes of a product or process. Software measurement is a titrate impute of a characteristic of a software product or the software process. It is an authority within software engineering. This paper describes various software testing metrics with their needs, benefits. It also define the types of metrics to assure software quality.

Keywords:
Software, Metric, Measurement, Process, Testing

1. Introduction

Software Testing Metric is defined as a quantitative measure that helps to estimate the progress, quality, and health of a software testing effort. A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute.

Software testing metrics provide quantitative approach to measure the quality and effectiveness of the software development and testing process. It helps the team to keep a track on the software quality at every stage in the software development cycle and also provides information to control and reduce the number of errors. It allows the stakeholders to measure the efficiency of the team and accelerates application delivery.

The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.

Within the software development process, there are many metrics that are all related to each other. Software metrics are related to the four functions of management: Planning, Organization, Control, or Improvement.

1.1. Benefits of Software Metrics

The goal of tracking and analyzing software metrics is to determine the quality of the current product or process, improve that quality and predict the quality once the software development project is complete. On a more granular level, software development managers are trying to:

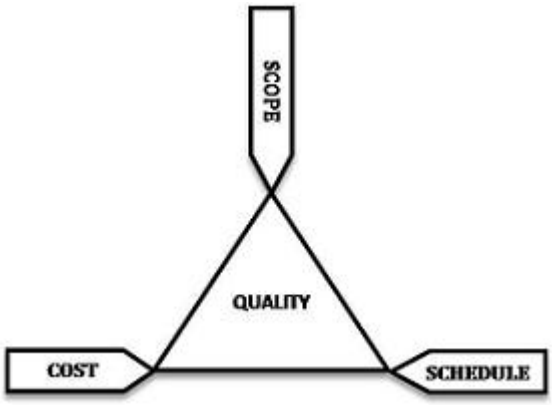

Figure 1. *Software Metrices.*

a. Increase return on investment (ROI)

b. Identify areas of improvement

c. Manage workloads

d. Reduce overtime

e. Reduce costs

These goals can be achieved by providing information and clarity throughout the organization about complex software development projects. Metrics are an important component of quality assurance, management, debugging, performance, and estimating costs, and they're valuable for both developers and development team leaders: Managers can use software metrics to identify, prioritize, track and communicate any issues to foster better team productivity. This enables effective management and allows assessment and prioritization of problems within software development projects. The sooner managers can detect software problems, the easier and less-expensive the troubleshooting process.

Software development teams can use software metrics to communicate the status of software development projects, pinpoint and address issues, and monitor, improve on, and better manage their workflow.

Software metrics offer an assessment of the impact of decisions made during software development projects. This helps managers assess and prioritize objectives and performance goals.

1.2. Why to Measure Software Quality

a. To evaluate the quality of the current product or process

b. To improve quality of a product /process by continuous monitoring

c. Take decisions based on analysis

1.3. How to Track Software Metrics

Software metrics are great for management teams because they offer a quick way to track software development, set goals and measure performance. But oversimplifying software development can distract software developers from goals such as delivering useful software and increasing customer satisfaction.

Of course, none of this matters if the measurements that are used in software metrics are not collected or the data is not analyzed. The first problem is that software development teams may consider it more important to actually do the work than to measure it.

It becomes imperative to make measurement easy to collect or it will not be done. Make the software metrics work for the software development team so that it can work better. Measuring and analyzing doesn't have to be burdensome or something that gets in the way of creating code. Software metrics should have several important characteristics. They should be:

a. Simple and computable

b. Consistent and unambiguous (objective)

c. Use consistent units of measurement

d. Independent of programming languages

e. Easy to calibrate and adaptable

f. Easy and cost-effective to obtain

g. Able to be validated for accuracy and reliability

h. Relevant to the development of high-quality software products

This is why software development platforms that automatically measure and track metrics are important. But software development teams and management run the risk of having too much data and not enough emphasis on the software metrics that help deliver useful software to customers.

1.4. Why Do You Need Software Testing Metrics?

Software Testing Metrics are useful for evaluating the health, quality, and progress of a software testing effort. Without metrics, it would be almost impossible to quantify, explain, or demonstrate software quality. Metrics also provide a quick insight into the status of software testing efforts, hence resulting in better control through smart decision making. Traditional Software testing metrics dealt with the ones' based on defects that were used to measure the team's effectiveness. It usually revolved around the number of defects that got leaked to production named as Defect Leakage or the Defects that were missed during a release, reflects the team's ability and product knowledge. The other team metrics was with respect to percentage of valid and invalid defects. These metrics can also be captured at an individual level, but generally are measured at a team level.

Software Testing Metrics had always been an integral part of software testing projects, but the nature and type of metrics collected and shared have changed over time. Top benefits of tracking software testing metrics include the following:

a. Helps achieve cost savings by preventing defects

b. Helps improve overall project planning

c. Facilitates to understand if the desired quality is achieved

d. Enforces keenness to further improve the processes

e. Helps to analyze the risks associated in a deeper way

f. Helps to analyze metrics in every phase of testing to improve defect removal efficiency

g. Improves Test Automation ROI over a time period

h. Enforces better relationship between testing coverage, risks and complexities of the systems

1.5. Divisions of Common Software Testing Metrics

Some of the most common software testing metrics that can be used for both automation and for software testing in general are given below. These are useful in the broader spectrum of software testing.

The general software testing metrics are divided into the following three categories:

a. Coverage: It refers to the meaningful parameters for measuring test scope and test success

b. Progress: Deals with the parameters that help identify test progress to be matched against success criteria. This metrics is collected iteratively over time and measures metrics like Time to fix defects, Time to test, etc.

c. Quality: is used to obtain meaningful measures of excellence, worth, value, etc. of the testing product and it is difficult to measure it directly

1.6. Need of Software Measurement

Software is measured to:

a. Create the quality of the current product or process.

b. Anticipate future qualities of the product or process.

c. Enhance the quality of a product or process.

d. Regulate the state of the project in relation to budget and schedule.

1.7. Classification of Software Measurement

There are 2 types of software measurement:

1. Direct Measurement: In direct measurement the product, process or thing is measured directly using standard scale.

2. Indirect Measurement: In indirect measurement the quantity or quality to be measured is measured using related parameter i.e. by use of reference.

2. Classification of Software Metrics

There are 3 types of software metrics:

i. Process Metrics: Process metrics pay particular attention on enhancing the long term process of the team or organisation.

ii. Product Metrics: Product metrics are used to evaluate the state of the product, tracing risks and undercovering prospective problem areas. The ability of team to control quality is evaluated.

iii. Project Metrics: Project matrix is describes the project characteristic and execution process.

a. Number of software developer

b. Staffing pattern over the life cycle of software

c. Cost and schedule

d. Productivity

Figure 2. *Types of Metrices.*

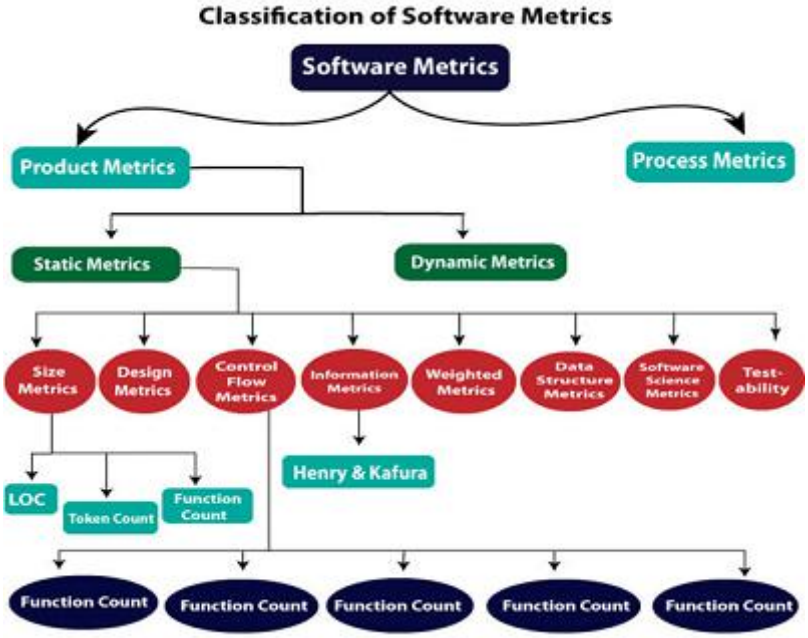

Figure 3. *Classification of Metrices.*

3. Types of Metrics to Assure Software Quality

The three types of metrics you should collect as part of your quality assurance process are: source code metrics, development metrics, and testing metrics.

3.1. Source code metrics

These are measurements of the source code that make up all your software. Source code is the fundamental building block of which your software is made, so measuring it is key to making sure your code is high-caliber. Look closely enough at even your best source code, and you might spot a few areas that you can optimize for even better performance.

When measuring source code quality make sure you're looking at the number of lines of code you have, which will ensure that you have the appropriate amount of code and it's no more complex than it needs to be. Another thing to track is how compliant each line of code is with the programming languages' standard usage rules. Equally important is to track the percentage of comments within the code, which will tell you how much maintenance the program will require.

3.2. Development Metrics

These metrics measure the custom software development process itself. Gather development metrics to look for ways to make your operations more efficient and reduce incidents of software errors.

Measuring number of defects within the code and time to fix them tells you a lot about the development process itself. Start by tallying up the number of defects that appear in the code and note the time it takes to fix them. If any defects have to be fixed multiple time then there might be a misunderstanding of requirements or a skills gap – which is important to address as soon as possible.

3.3. Testing Metrics

These metrics help you evaluate how functional your product is.

There are two major testing metrics. One of them is "test coverage" that collects data about which parts of the software program are executed when it runs a test. The second part is a test of the testing itself. It's called "defect removal efficiency" and it checks your success rate for spotting and removing defects.

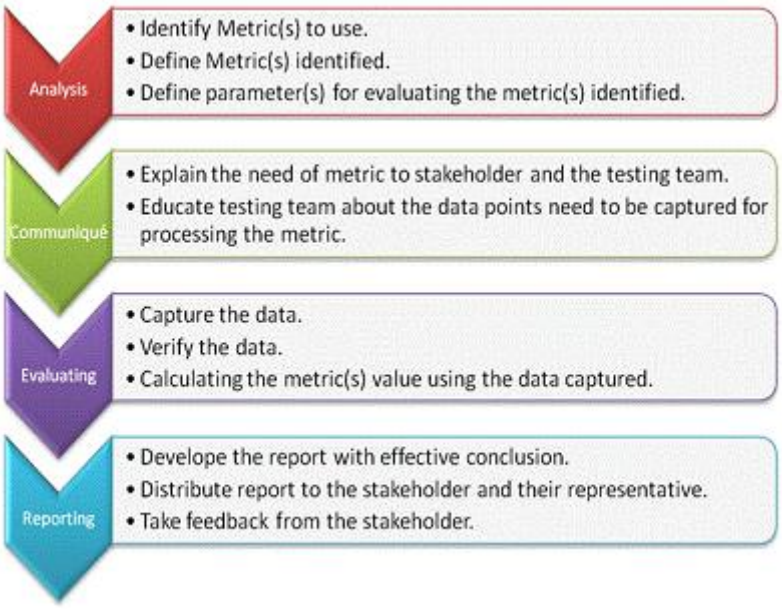

Figure 4. *Test Metrics Life Cycle.*

The more you measure, the more you know about your software product, the more likely you are able to improve it. Automating the measurement process is the best way to measure software quality – it's not the easiest thing, or the cheapest, but it will save you tons of cost down the line.

Table 1. *How to calculate test metric.*

SR NO.	STEPS TO TEST METRICS	EXAMPLE
1	Identify the key software testing processes to be measured	Testing progress tracking process
2	In this Step, the tester uses the data as a baseline to define the metrics	The number of test cases planned to be executed per day
3	Determination of the information to be followed, a frequency of tracking and the	The actual test execution per day will be captured by the test manager at the end of

	person responsible	the day
4	Effective calculation, management, and interpretation of the defined metrics	The actual test cases executed per day
5	Identify the areas of improvement depending on the interpretation of defined metrics	The Test Case execution falls below the goal set, we need to investigate the reason and suggest the improvement measures

4. Examples of Software Metrics

There is no standard or definition of software metrics that have value to software development teams. And software metrics have different value to different teams. It depends on what are the goals for the software development teams.

As a starting point, here are some software metrics that can help developers track their progress.

4.1. Agile Process Metrics

Agile process metrics focus on how agile teams make decisions and plan. These metrics do not describe the software, but they can be used to improve the software development process.

4.2. Lead Time

Lead time quantifies how long it takes for ideas to be developed and delivered as software. Lowering lead time is a way to improve how responsive software developers are to customers.

4.3. Cycle Time

Cycle time describes how long it takes to change the software system and implement that change in production.

4.4. Team Velocity

Team velocity measures how many software units a team completes in an iteration or sprint. This is an internal metric that should not be used to compare software development teams. The definition of deliverables changes for individual software development teams over time and the definitions are different for different teams.

4.5. Open/Close Rates

Open/close rates are calculated by tracking production issues reported in a specific time period. It is important to pay attention to how this software metric trends.

4.6. Production

Production metrics attempt to measure how much work is done and determine the efficiency of software development teams. The software metrics that use speed as a factor are important to managers who want software delivered as fast as possible.

4.7. Active Days

Active days are measure of how much time a software developer contributes code to the software development project. This does not include planning and

administrative tasks. The purpose of this software metric is to assess the hidden costs of interruptions.

4.8. Assignment Scope

Assignment scope is the amount of code that a programmer can maintain and support in a year. This software metric can be used to plan how many people are needed to support a software system and compare teams.

4.9. Efficiency

Efficiency attempts to measure the amount of productive code contributed by a software developer. The amount of churn shows the lack of productive code. Thus a software developer with a low churn could have highly efficient code.

4.10. Code Churn

Code churn represents the number of lines of code that were modified, added or deleted in a specified period of time. If code churn increases, then it could be a sign that the software development project needs attention.

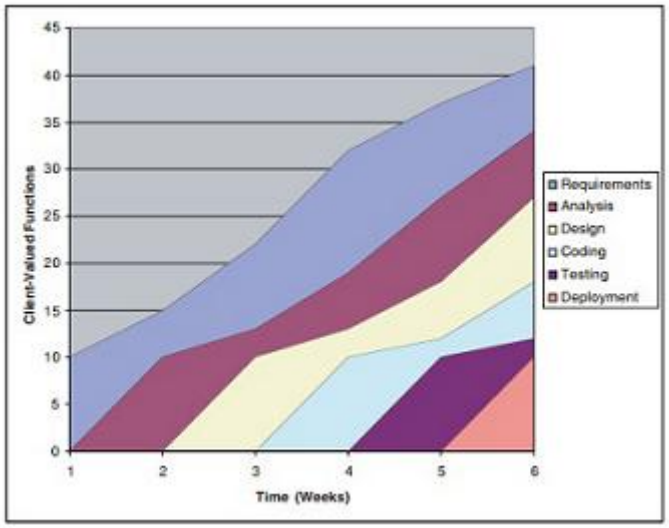

Figure 5. *Life Cycle.*

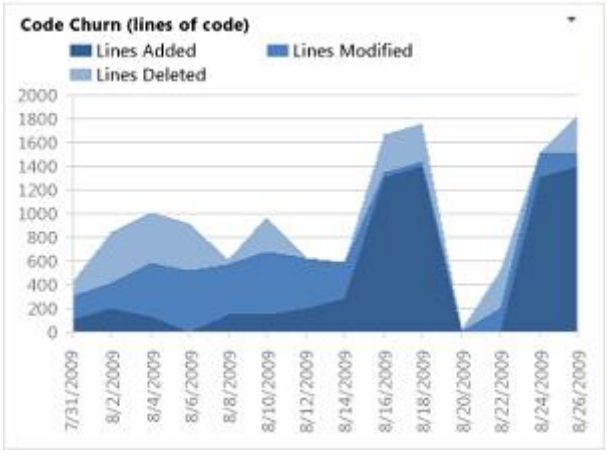

Figure 6. *Code Churn.*

4.11. Impact

Impact measures the effect of any code change on the software development project. A code change that affects multiple files could have more impact than a code change affecting a single file.

4.12. Mean Time Between Failures (MTBF) and Mean Time to Recover/Repair (MTTR)

Both metrics measure how the software performs in the production environment. Since software failures are almost unavoidable, these software metrics attempt to quantify how well the software recovers and preserves data.

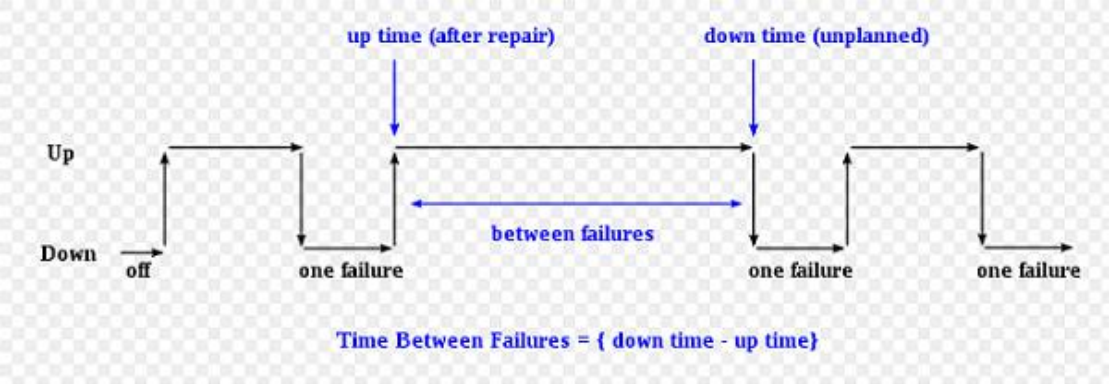

Figure 7. *Time Between Failure.*

4.13. Application Crash Rate (ACR)

Application crash rate is calculated by dividing how many times an application fails (F) by how many times it is used (U).

ACR = F/U

4.14. Security Metrics

Security metrics reflect a measure of software quality. These metrics need to be tracked over time to show how software development teams are developing security responses.

4.15. Endpoint Incidents

Endpoint incidents are how many devices have been infected by a virus in a given period of time.

4.16. Mean Time to Repair (MTTR)

Mean time to repair in this context measures the time from the security breach discovery to when a working remedy is deployed.

4.17 Size-oriented Metrics

Size-oriented metrics focus on the size of the software and are usually expressed as kilo lines of code (KLOC). It is a fairly easy software metric to collect once decisions are made about what constitutes a line of code. Unfortunately, it is not useful for comparing software projects written in different languages. Some examples include:

a. Errors per KLOC

b. Defects per KLOC

c. Cost per KLOC

4.18. Function-oriented Metrics

Function-oriented metrics focus on how much functionality software offers. But functionality cannot be measured directly. So function-oriented software metrics rely on calculating the function point (FP) — a unit of measurement that quantifies the business functionality provided by the product. Function points are also useful for comparing software projects written in different languages.

Function points are not an easy concept to master and methods vary. This is why many software development managers and teams skip function points altogether. They do not perceive function points as worth the time.

4.19. Errors Per FP or Defects Per FP

These software metrics are used as indicators of an information system′ s quality. Software development teams can use these software metrics to reduce miscommunications and introduce new control measures.

4.20. Defect Removal Efficiency (DRE)

The Defect Removal Efficiency is used to quantify how many defects were found by the end user after product delivery (D) in relation to the errors found before product delivery (E). The formula is:

DRE = E / (E+D)

The closer to 1 DRE is, the fewer defects found after product delivery.

5. Metrics Lifecycle

5.1. Analysis

a. Identify and define the metrics

b. Define parameters for evaluating the metrics

5.2. Communicate

c. Explain the need and significance of metrics to stakeholders and testing team

d. Educate the testing team about the data points need to be captured for processing the metric

5.3. Evaluation

e. Capture the required data

f. Verify validity of the data captured

g. Calculate the metrics value

5.4. Reports

h. Develop the report with effective conclusion

i. Distribute the reports to the stakeholders, developer and the testing team

j. Take feedback for further improvements

6. Conclusion

Using all above software metrics one can achieve the following:- Measuring the size of the software quantitatively, Assessing the level of complexity involved, Assessing the strength of the module by measuring coupling, Assessing the testing techniques, Specifying when to stop testing, Determining the date of release of the software, Estimating cost of resources and project schedule. Software metrics help project managers to gain an insight into the efficiency of the software process, project, and product. Also, when metrics are applied in a consistent manner, it helps in project planning and project management activity.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this article.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

References

[1] Srinivasan, K.P.; Devi, T. Software Metric Validation Methodologies in Software Engineering. *International Journal of Software Engineering & Applications (IJSEA)*, 2014, 5(6), 87-102.

[2] Amrit, D.; Amrinder, S. Analysis of Software Metrics for Bubble Sort and Selection Sort. *International Journal of Computer Applications & Information Technology,* 2012, 1(1), 1-3.

[3] Manik, S.; Gurdev, S. Analysis of Static and Dynamic Metrics for Productivity and Time Complexity. *International Journal of Computer Applications,* 2011, 30(1), 0975-8887.

[4] Manik, S.; Gurdev, S. Predictive Metric- A Comparative Study. *International Journal of Computer Science and Technology (IJCST)*, 2011, 2(1).

[5] Amjan, S.; Reddy, C.R.K. Damodaram, A. 2012. Object Oriented Software Metric and Quality Assesment: Current State of the Art. *International Journal of Computer Applications.* 2012, 37(11), Corpus ID: 4695174.

[6] Rani, G.; Paramvir, S. 2014. Dynamic Coupling Metrics for Object Oriented Software Systems- A Survey. *ACM SIGSOFT Software Engineering Notes,* 2014, 39(2).

[7] Vivanco, R.A.; Pizzi, N.J. Identify Effective Software Metrics using Genetic Algorithms. In *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*, Montreal, Quebec, Canada, INSPEC Accession Number: 7992264.

[8] Deepak, A.; Pooja, K.; Alpika, T.; Shipra, S.; Sanchika, S. Software Quality Estimation through Object Oriented Design Metrics. *International Journal of Computer Science and Network Security*, 2011, 11(4).

[9] Bansiya, J.; Davis, C.G. A Hierarchical Model for Object-Oriented Design Quality Assessment. *IEEE Transactions on Software Engineering*, 2002, 28(1), 4-17.

[10] Sanjeev, D.K. Software Metrics – A Tool for Measuring Complexity. *International Journal of Software and Web Sciences,* Corpus ID: 16335170. Available online: https://www.semanticscholar.org/paper/Software-Metrics-%E2%80%93-A-Tool-for-Measuring-Complexity-Dhawan/2eb56d288a6226872bc346adb31ff763770f8166 (accessed on 30 December 2019).

[11] Neelamegam, C.; Punithavali, M. A survey on object oriented quality metrics. *Global journal of computer science and technologies*, 2011; pp. 183-186.

[12] Vivienne, L.; Christopher, C. Principal Components of Orthogonal Object-Oriented Metrics*).* Available online: https://www.semanticscholar.org/paper/Principal-Components-of-Orthogonal-Object-Oriented-Laing-Coleman/10260d67fa5ed4ecea9dd4398c3f236c52cabf6b (accessed on 30 December 2019).

[13] Amit, S.; Sanjay, K.D. Comparison of Software Quality Metrics for Object-Oriented System. *IJCSMS International Journal of Computer Science & Management Studies*, 2012, 12. Available online: https://www.semanticscholar.org/paper/Comparison-of-Software-Quality-Metrics-for-System-Sharma-Dubey/d9895e36e85b3bc495a37ba7d49e36a98fb75f93 (accessed on 30 December 2019).

[14] Mrinal, S.R.; Arpita, M.; Sanjay, K.D. Survey on Impact of Software Metrics on Software Quality. *International Journal of Advanced Computer Science and Applications*, 2012, 3(1), 137-141.

[15] Sonal, C.; Gagandeep, K. Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development. *International Journal of Advanced Computer Science and Applications,* 2013, 4(9), 161-164.