2015 IEEE International Conference on Computational Intelligence & Communication Technology

Analysis of Testing Metrics for Object Oriented Applications

Rohit Beniwal
Division of Computer Engineering
Netaji Subhas Institute of Technology, NSIT
Delhi - 78, India
rohitbeniwal@yahoo.co.in

Abstract— **In today's era, most of the softwares are developed using Object Oriented Programming languages. Testing is the integral part of development process for ensuring the quality and reliability of the software. However, testing is needed to be measured in some order to really know its effectiveness. For measuring the effectiveness of the testing, testing metrics are used. This paper empirically explains the role of testing metrics on the applications or softwares that are developed using object oriented programming languages. To do this, testing metrics are applied on various projects in order to determine their advantages and usefulness in managing the software development process, software product and software project.**

Keywords- Analysis, object oriented applications, testing metrics.

I. INTRODUCTION

In today's era, most of the softwares are developed using Object Oriented Programming (OOP) languages. Testing is the integral part of development process for ensuring the quality and reliability of the software. However, testing is needed to be measured in some order to really know its effectiveness because we cannot control what we cannot measure [1]. Hence software metrics are used for measuring any characteristic or property of the software. Although software metrics are in use from last several decades [2], but only little work is done on implementing the testing metrics on softwares or applications that are developed using OOP languages in order to know their effectiveness. Softwares or applications that are developed using OOP languages are known as Object Oriented Applications (OOA).

Hence in order to know the effectiveness of testing metrics, I have implemented them on OOA. As far as taxonomy of metrics is concerned, there are three kinds of metrics: process metrics, product metrics and project metrics [3], [4]. Testing metrics for OOA are applied on all of them i.e. process testing metrics for OOA, product testing metrics for OOA and project testing metrics for OOA.

This paper is organized as follows: section II discusses the process testing metrics for OOA and their role; section III discusses the product testing metrics for OOA and their role; section IV discusses the project testing metrics for OOA and their role; section V discusses the conclusion of the research paper and section VI discusses some research directions for further work [4].

II. PROCESS TESTING METRICS FOR OOA

In this section I discuss what process testing metrics for OOA are used and what role do they play?

To do this, a project entitled 'SP/LX' is taken which is developed in java programming language and following analysis is done:

A. *Test Progress curve*

Test progress tracking is a crucial step for tracking and managing the software testing. To do the analysis of test progress tracking, I used a metric known as test progress curve [5] in which x axis represents the time and y axis represents the no. of test cases.

Now to use the metric successfully, it should contain the following information on the same graph:

1) No. of test cases that are planned over the time and that should be passed or become successful by the week end.

2) No. of test cases executed by the week end.

3) No. of test cases that are actually passed by the week end.

The objective of this metric is to show the difference between planned test cases that should be passed and actual test cases that are passed and if there is any difference between planning and actual activity, then take the appropriate action for the remedy. It is normally seen that due to 'time' and 'cost' constraint, generally testing is forbidden (it is either partially done or completely ignored). However, when this metric is in operation, it is difficult to ignore the problem and it is more likely that appropriate action is taken. This metric is used for better planning [5].

Now to explain it further consider fig. 1.

In the graph of fig. 1, firstly planned test cases, which need to be passed, are drawn over the time as indicated by the black line. The graph also contains two bars for each week end in which dark shaded bar is representing the no. of test cases attempted till that week whereas light shaded bar represents the actual no. of test cases that are passed till that week.

This project has total of 959 test cases which need to be successfully pass till the end of week 16.

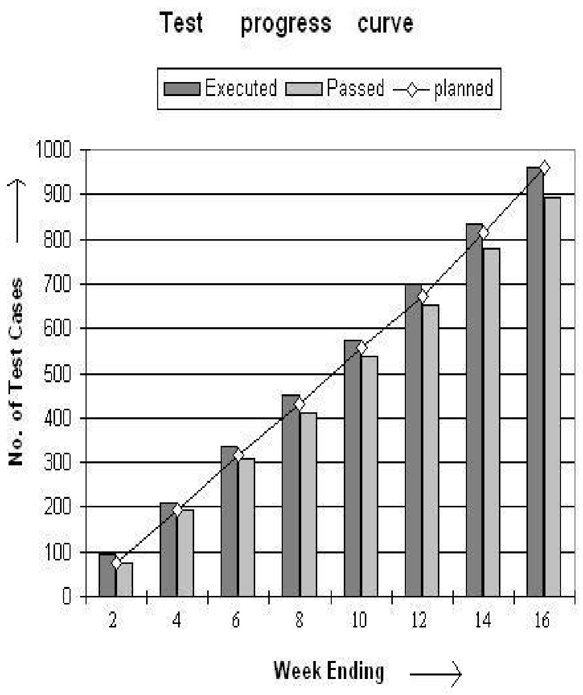

Fig.1 Test Progress Curve showing executed, passed and planned test cases

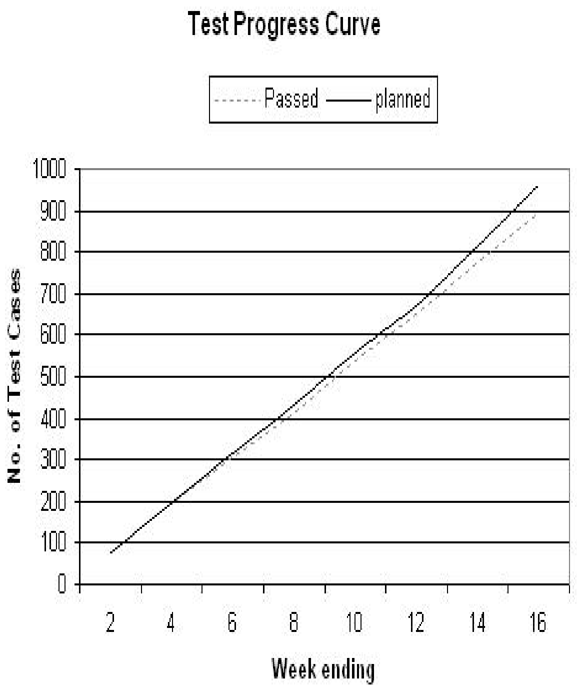

Fig. 2 Test Progress Curve showing passed and planned test cases

Now consider fig. 2.

From fig 1 and fig 2, it is obvious that at the end of 6th week, deviation between the planned and actual successful test cases occurs and this remains till the end of week 16. Earlier it is planned that all the test cases should be passed till the end of week 16, but we can clearly observe that only 892 out of 959 test cases are passed till 16th week. Actually it took one week more than planned to completely pass all the test cases i.e. 17 weeks.

Hence to do the remedy here I can say that we should take appropriate action as soon as possible when the deviation started i.e. from the week 6.

To get the clear picture of weekly progress, refer to table I.

One possible solution of above problem is that when the deviation started between planned and actual work, more staff can be allocated to complete the work in proper time or to analyse the cause of deviation and then take the appropriate action.

TABLE I. TEST PROGRESS CURVE BIWEEKLY DATA

S. NO.	**Week No.**	**Planned Tests Cases that should be passed (%)**	**Planned Tests Cases that should be passed (Quantity)**	**Executed test Cases (%)**	**Executed Test Cases (Quantity)**	**Actual Passed Test Cases (%)**	**Actual Passed Test Cases (Quantity)**
1	2	8	77	10	96	8	77
2	4	20	192	22	211	20	192
3	6	33	316	35	336	32	307
4	8	45	432	47	451	43	412
5	10	58	556	60	575	56	537
6	12	70	671	73	700	68	652
7	14	85	815	87	834	81	777
8	16	100	959	100	959	93	892

B. Problem Tracking Report (test defects) arrival over time

In software engineering, it is recommended that defect tracking should be done as a standard practice for better management of software testing. To do this I use a problem tracking report tool. It is also recommended that defect tracking should be done with tracking by test phases. Overall defect density doesn't supply much information; hence it is not much useful and cannot be used as a process metric. The defect arrival rate supplies more meaningful and useful information [5].

Now to use this metric successfully, it should contain the following information:

1) The defect arrival over time should be compared with the similar kind of project to do the analysis. Previously developed similar kind of project defect rate can be taken as the baseline data. If there is no baseline project, some assumption can be made at key point of scheduled time before starting the defect tracking.

2) X axis represents the time remaining to general availability of the product.

3) Y axis represents the no. of defects arrived over the time.

Now to explain it further, consider fig. 3.

In the graph of fig. 3, it contains two lines, one is dark broken line which served as a baseline data and other is light continuous line which serves as a current data. From this graph we can find that the peak time for defect arrival for baseline project is when 12 weeks are remaining for general availability and for current project, it is when 16 weeks are remaining. Baseline and current project are having maximum 40 and 44 defects. Both the baseline and current project decline to low or stable level before the general availability.

We can do the following analysis based on the above information: Initially there are a lot of defects found in current project as compared to baseline. Now the question arises that have we planned better, if yes then the quality perspective is positive, if not then the quality perspective is negative i.e. developers have developed a low quality product. After certain period current project generates fewer defects as compared to baseline. Here we need to find out that whether testing of current project is less effective, if no quality perspective is positive, if yes extra and effective testing is needed.

Now consider fig 4.

In the graph of fig. 4, it shows that as the product is close to general availability all kind of defects are decreasing, which is a good sign from quality and schedule perspective. Critical defects should be minimal in the testing phase when the product is close to general availability; otherwise it can lead to serious delay to our scheduled delivery. One more point need to remember is that overall critical defect line should be below major defect line, which in turn should be below minor defect line.

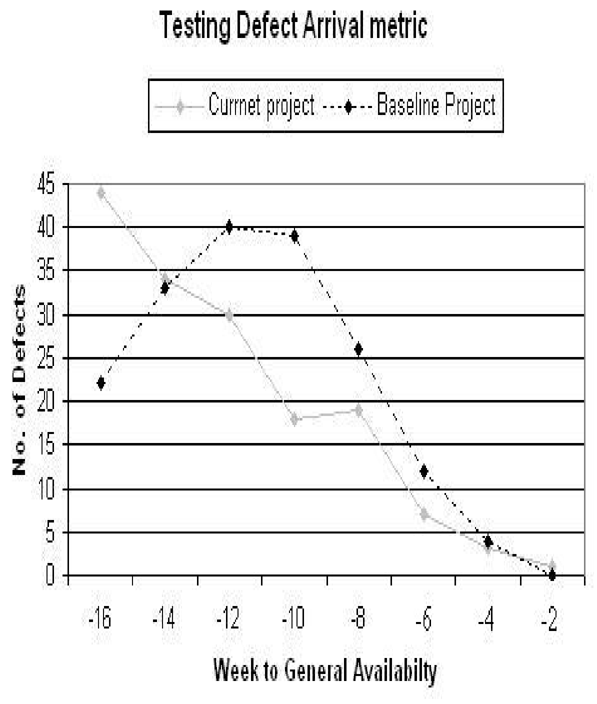

Fig. 3 Test Defect Arrival Metric

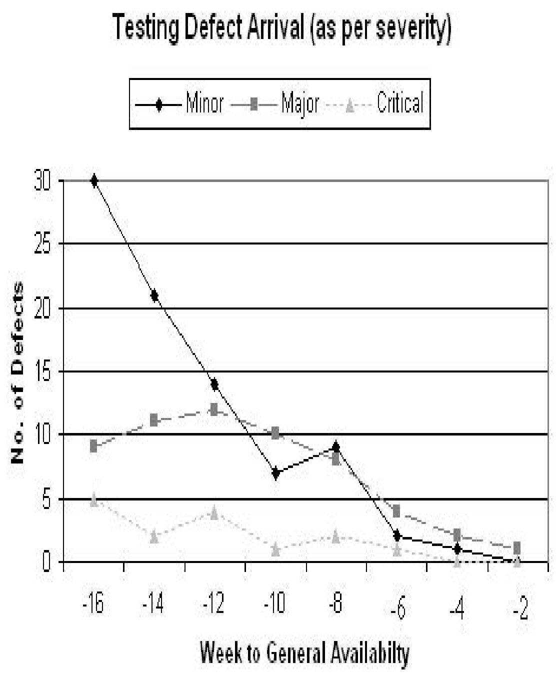

Fig. 4 Test Defect Arrival (as per severity)

To get the clear picture, refer to table II.

TABLE II. TEST DEFECTS ARRIVAL BIWEEKLY DATA

Weeks to General Availability	**Minor Defects**	**Major Defects**	**Critical Defects**	**Total Defects**
-16	30	9	5	44
-14	21	11	2	34
-12	14	12	4	30
-10	7	10	1	18
-8	9	8	2	19
-6	2	4	1	7
-4	1	2	0	3
-2	0	1	0	1
Total Defects	84	57	15	156

III. PRODUCT TESTING METRICS FOR OOA

In this section I discuss what product testing metrics for OOA are used and what role do they play?

To do this, a project entitled 'Rossiya Web' is taken which is developed in java programming language in duration of one year for aviation industry and following analysis is done:

A. *Test Coverage*

This metric is used to show that how much of the total code is tested [6].

$$\text{Test Coverage} = \frac{\text{No. of units tested}}{\text{Total size of the system}} * 100$$

Size can be in any unit (Kilo Lines of Codes or Function Point).

For our project it is 96.3% i.e. 96.3% units of the total code is analysed during the testing phase.

B. *Quality of Testing*

This metric is used to show the quality of testing [7].

$$\text{Quality of Testing} = \frac{\text{No. of defects found during testing}}{\text{No. of defects found during testing} + \text{No. of defects found after delivery of the product}} * 100$$

For our project it is 97.70% i.e. 97.70 % defects are found during testing phase while remaining 2.3% defects are found after delivery.

C. *Scale of Ten (for testing)*

This metric is used to show the rating of testing done by the testers .Rating is given in scale of 1to 10 [6], [8].

$$\text{Scale of Ten (for Tesitng)} = \text{Assesment of tesing by giving it rating in in scale of 1 to 10}$$

For our project testing is given the rating of 9 out of 10.

D. *Test Cost (in %)*

This metric is used to show that how much of total cost is devoted to the testing only [6].

$$\text{Test Cost} = \frac{\text{Cost of testing}}{\text{Total cost of the project}} * 100$$

For our project it is 20 %.

E. *Analysis*

From the above discussed metrics, following analysis can be done: It is given that the product is tested up to 96.3% which results in a testing quality of 97.7%. We should always focus, despite of any constraint (cost, schedule etc.), that our project should cover 100% test coverage area and it should be resulted in 100 % quality of testing. However, from the above data it is also obvious that more the test coverage, better the quality of testing it is. Our testing should always get a rating of 10 in Scale of Ten. We should not compromise the quality of testing at any cost.

One more analysis can be done here that only 20% of total cost is spent on the testing which doesn't result in 100% quality of testing. So we should try to increase the cost of testing so that it results in 100% quality of testing.

IV. PROJECT TESTING METRICS FOR OOA

In this section I discuss what project testing metrics for OOA are used and what role do they play?

To do this, a project entitled 'Silk Road' is taken which is developed in C# using .Net framework and following analysis is done:

A. *Defect per function point (per life cycle phase)*

This metric is used to find out that the no. of defects per function point in each life cycle phases [2], [9].

To explain it further consider fig 5.

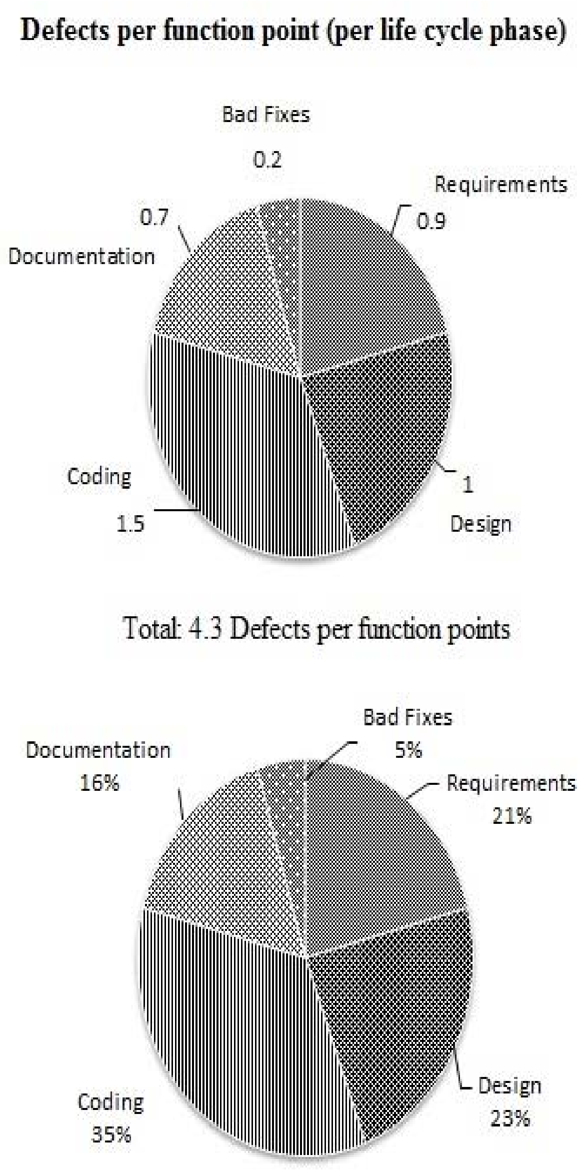

Fig. 5 Defects per function point (per life cycle phase)

It is obvious form the graph that coding contains maximum no. defects per function point. So in this particular project more focus should be done on coding phase in order to do the remedy. After that it comes to design, then requirements, then documentation and lastly on bad fixes.

B. Defect found per hour(per testing approach)

This metric is used to find out the no. of defects per hour in each testing approaches [2], [9].

To explain it further consider the following fig. 6

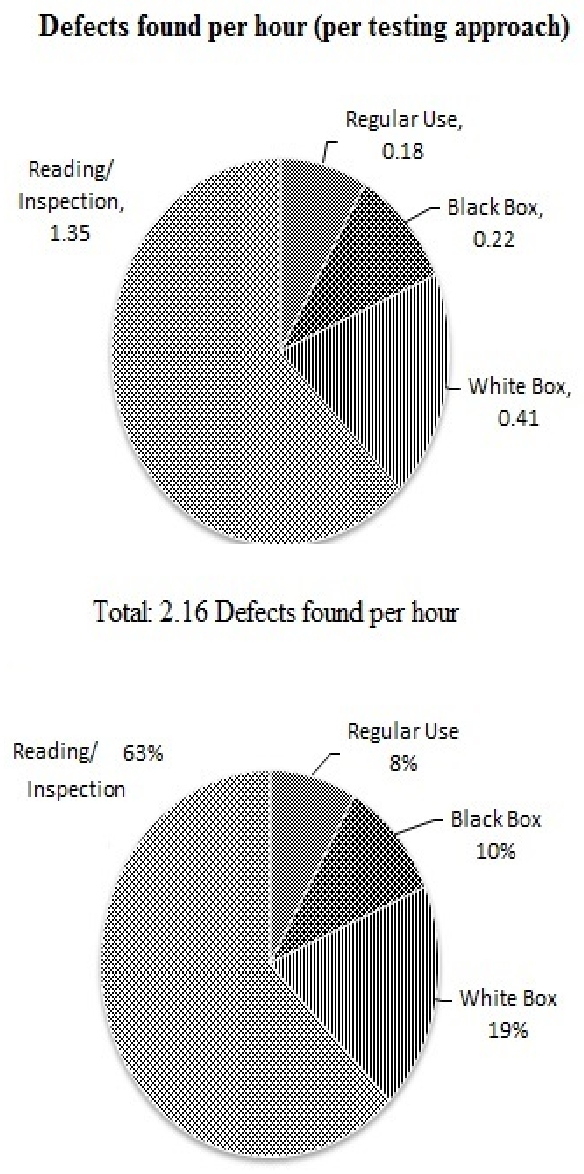

Fig. 6 Defects found per hour (per testing approach)

It is obvious that most of the defects are found through reading or inspections. So in this particular project if most of defect found per hour are maximum for reading or inspection, we should be more focused on this testing approach. After that it comes to white box, then black box and lastly on regular use.

V. CONCLUSIONS

I have implemented several metrics on various projects and found that testing metrics for OOA are useful at every stage of development. These metrics play a crucial role in managing the applications that are developed using OOP languages. These metrics are always having the advantage to the organization, which implements them in their regular practice.

From the above discussed metrics, following advantages can be summarized:

A. *In case of process testing metrics for OOA*

1) Whenever there is a deviation between planned and actual work, it can be easily discovered by the test progress curve metric.

2) No. of defects arrived over the time in the current project can be easily found by problem tracking report tool to compare it to any similar kind of baseline project.

B. *In case of product testing metrics for OOA*

1) Test coverage, quality of testing, scale of ten and test cost metrics are interlinked with each other and they can provide sufficient information about quality of testing done on the product.

C. *In case of project testing metrics for OOA*

1) Defect per function point (per life cycle phase) metric can easily explain that which phase of the project has maximum number of defects and when this information is available, focus can be shifted to that phase to do it more effectively.

2) Defect found per hour (per testing approach) metric can easily explain that which testing approach finds maximum number of defects and when this information is available, focus can be shifted to that approach to do it more effectively.

VI. FUTURE WORK

I have selected few testing metrics and found their advantages on OOA. Future work can include selecting more testing metrics for OOA and implementing them on number of different projects in order to know their advantages and effectiveness in managing the software development. Future work can also include to find that whether testing metrics for OOA shows the same result for similar kind of project under similar environment.

REFERENCES

[1] DeMarco, Tom, “Controlling Software Projects: Management, Measurement & Estimation,” Yourdon Press, New Jersey, 1982.

[2] N. Fenton and M. Neil “A Critique of Software Defect Prediction Research”, IEEE *Trans. Software Eng.*, vol. 25, No.5, 1999.

[3] Stephen H. Kan, *Metrics and Models in Software Quality Engineering*, 2nd Edition, Addison-Wesley Professional, September 2002.

[4] F. Brito e Abreu, R. Carapuca, “Candidate Metrics for Object-Oriented Software within a Taxonomy Framework” J. Systems Software,vol. 26, pp 87-96, 1994.

[5] S. H. Kan, J. Parrish, D. Manlove*, In-process metrics for software testing*, IBM Systems Journal, v.40 n.1, p.220-241, January 2001.

[6] Metrics for Evaluating Application System Testing. [Online]. Available: http://www.exforsys.com/tutorials/testing/metrics-used-in-testing.html

[7] Software Quality Metrics. [Online]. Available: http://it.toolbox.com/wiki/index.php/Software_Quality_Metrics

[8] All about Metrics Used by Software Testers. [Online]. Available: http://www.softwaretestinggenius.com/articaldetails.php?mode=details&qry=98&parent=86

[9] Measuring Defect Removal Accurately - Test Metrics Sidebar. [Online]. Available: http://beta.stpcollaborative.com/knowledge/276-measuring-defect-removal-accurately-test-metrics-sidebar