

Test Metrics in Iterative Model of Development

Sanjay B. Dange
Datamatics Limited, Nashik, Maharashtra, India

Datamatics Limited
Nashik, Maharashtra, India
sanjay_dange@datamatics.com

Abstract:

Test metrics are important indicator of efficiency & effectiveness of Software testing process. There are several test metrics identified as part of the overall testing activity in order to track and measure the entire testing process. These test metrics are collected at each phase of the testing life cycle. Their analysis is done and appropriate process improvements are determined. Measurements, data collection, analysis & improvement measures are parallel activities to testing life cycle activities.

When a testing team determines exactly what can be learned from each test and manages it according to the user experience, testing time can be spent wisely. Working on metrics development early in the process is a key, as it will impact the types of metrics needed to examine and ensure performance. Also, this provides the testing team with the time needed to draw up an efficient test plan, gather resources needed, and schedule the testing. Involvement of all stakeholders like developers, client representatives will give sharpness to outputs of metrics.

How to measure a test process is a required competence for an effective software test manager. Effective test process measurement is useful for designing and evaluating a cost-effective test strategy. Although existing research has addressed many metrics for those purposes, most metrics are used independently.

This paper summarizes a test process measurement and how it can be made more effective in *Iterative model of development*. Using analysis and measurement as drivers of the enhancement process is one major advantage of iterative software development. It provides support for determining the effectiveness of the processes and the quality of product. It allows one to study, and therefore improve and tailor, the processes for the particular environment. How to apply analysis done in previous iterations for improvement of testing process in next iteration? This is core area of discussion in this paper. Further discussions about implementation problems, how to apply test wisdom learned in previous iteration for metrics improvement, and cost saving due to measurement activities and metrics are presented.

A section of this paper will discuss improving metrics to provide more immediate, real-time feedback to testers on the quality of testing during each test phase, instead of waiting until the release. In order to accomplish this, team will have to focus on defining and developing “incremental” quality, productivity, cost, and effectiveness metrics for the development and test process.

Keywords:

Test Metrics, Iterative Model, Defect Categorization, Process Improvement,

1 Introduction

Using analysis and measurement as drivers of the enhancement process is one major difference between iterative enhancement and other models of development. It provides support for determining the effectiveness of the processes and the quality of product. It allows one to study, and therefore improve and tailor, the processes for the particular environment. Data collected by Test Metrics in previous iteration can be used to improve future work estimates, redefine the focus areas and to improve efficiency.

While the need of metrics has been recognized, implementation of structured measurement program is still a pain area.

2 Basic Process

Appropriate process for use of test metrics is vital to achieve adequate quality. Basic steps in process of implementation of metrics are:

- a. Identify the pain area or goals.
- b. Define metrics for each goal.
- c. Set up procedure to collect relevant data.
- d. Educate stakeholders
- e. Collect data
- f. Verify & Analyze Data
- g. Communicate results to all stakeholders
- h. Apply results for further improvement

Identification of pain area or problem area is an important step. In this step, risk analysis is done for the future iteration. The objective of Risk Analysis is to identify potential problems that could affect cost or outcome of the project.

Based on identified pain areas appropriate *metrics should be defined*. Base metrics will be used for data that can be captured directly. Base metrics like # of Test cases executed/passed/failed/deferred etc. are used. Similar metrics for time & defects are also used. Calculate Metrics or Derived Metrics convert the base metrics data into more useful information. Test metrics are meaningful if they provide objective feedback to the project team regarding any of the development processes from analysis, to coding, to testing. Hence defining metrics those will provide objective result is important.

Next step is to *setup a procedure to collect data*. In this step, means of collecting data are decided. The collected data should be relevant, valid, and measurable, which can be used for analysis to derive conclusions about process effectiveness. Collecting irrelevant data leads to wastage of effort, time and team starts losing faith in need of implementation of metrics.

Communicating metrics implementation plan to all stakeholders is very important. Unless the team members (test team as well as other entities) keep deliverables updated, data collected by metrics will not be valid. Many times, members of a project team intuitively understand that changes in the development lifecycle could improve the quality and reduce the cost of a project, but they are unwilling to implement changes without objective proof of where the changes should occur. Hence it is important to spread awareness amongst stakeholders about objectives of metrics implementation.

Collecting Data & Analyzing it can be carried out using number of tools starting from Microsoft Excel to more specialized tools like McCabe QA, LDRA Test bed, Functional Point Analysis tools & Rational Suit. This stage should be carried out with minimum manual intervention as manual recording of data is subject to bias (deliberate or unconscious), error, omission, and delay. Therefore is automatic data capture desirable, and sometimes also essential. Derived/Calculated metrics are calculated in this stage.

Communicating results of metrics implementation should be communicated to all stakeholders periodically. Results should be in the format that is easily readable and consistent in its format. Analysis reports should include graphs highlighting the information based on which further decisions are expected.

Applying Results to achieve improvement should be done in diplomatic fashion. The results should not be used as measure of performance of individuals. Instead they should be used in constructive manner to achieve continuous improvement. Before taking decision, feedback should be taken from concerned members of team. In this stage it is necessary to revisit effectiveness of defined metrics. Metrics should be evaluated and restructured to get effective results in next iteration of development.

3 Test Metrics Iterative Model

Implementation of test metrics in Iterative model has become comparatively difficult due to time constraints, variation in focus areas & variation in client requirements. Effective implementation of test metrics requires quick actions, accurate selection of metrics and collective commitment. Further the time and efforts spent on metrics collection should be optimum. If more time and effort is spent on this activity, it is likely that metrics will be perceived as futile activity. This generally leads in lost faith of project management and finally abandoning of metrics collection.

4 Case Study

The rest of this paper will discuss a case study on implementation of test metrics in project 'X' that is being developed in six iterations.

4.1 Application

Software under consideration is a reengineering project with vast number of functionalities in securities market. Project was divided into six iterations starting with basic modules gradually developing into more complex processes. Existing COBOL application was a baseline for development and new system will be a web based system developed in J2EE, Struts.

4.2 Scope and Strategy

In first phase extensive analysis of existing application and COBOL code was carried out to write the requirements for new application. The system testing strategy was agreed upon that was supposed to cover all functional requirement and few non-functional requirements. Since the iterative model was agreed upon, it was decided to have test metrics in place.

4.3 Test Metrics Implementation

Metrics for collection of data for Test case preparation and test cases execution were decided. Defects were analysed according to Action taken, Injection Phase, Detection Phase, Priority and Category. The tool used for defect tracking provided data for Defect Prevention analysis. Since the data could be exported to excel sheet, analysis was made simple by using macros and templates.

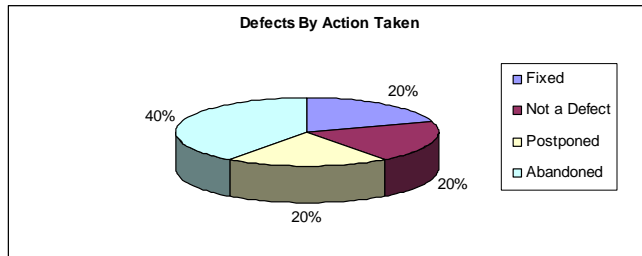


Fig.1: Defects by Action Taken

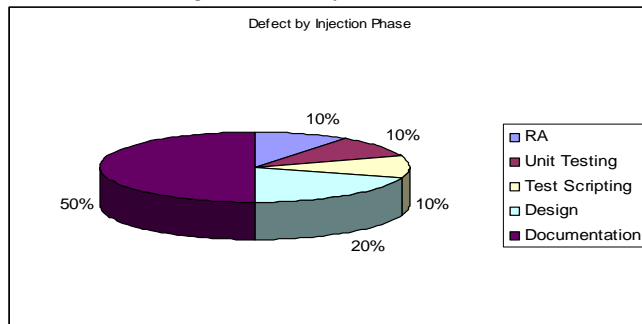


Fig.2: Defects by Injection Phase

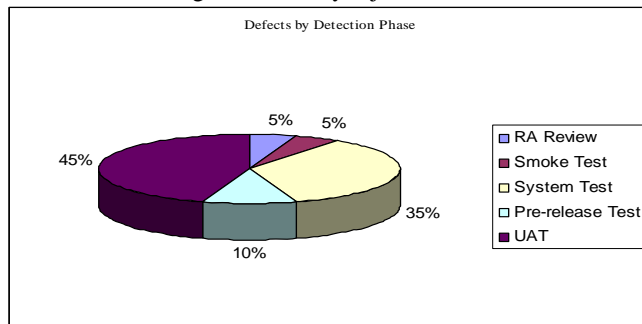


Fig.3: Defects by Detection Phase

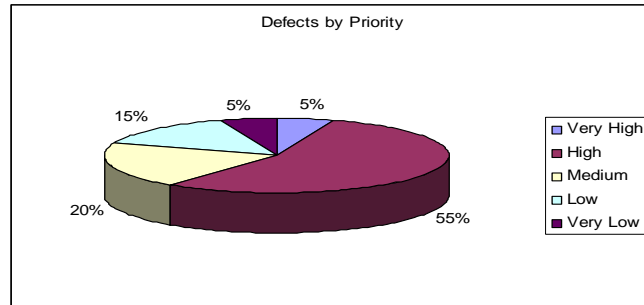


Fig.4: Defects by Priority

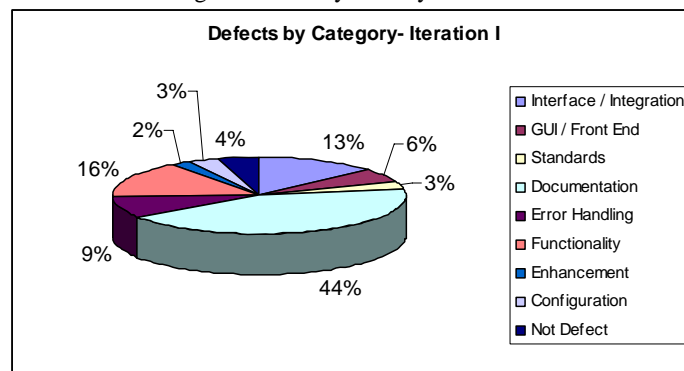


Fig.5: Defects by Category- Iteration I

4.4 Analysis and Decisions

Based on analysis of data collected in first iteration, following decisions were taken:

1. It was concluded from 'Defects by Action taken' graph that eighty percent of defects were either postponed, abandoned or marked as 'No Defects'. Only twenty percent were fixed. The prominent reason obvious from other graphs is 'Lack of understanding of functionalities and their interdependence'. Based on this primary conclusion, resources were encouraged to attend maximum knowledge transfer sessions conducted by business analysts. Recordings of these sessions were preserved at a central repository for easy access.
2. Another conclusion that was made regarding phase of detection of defect was very high percentage of defects was leaked to UAT. Most of these defects were related to integration testing. Hence it was concluded that in coming iterations, integration testing will be a focus area. Metrics were decided for collecting data related to integration testing.
3. Based on analysis of defect injection phases, it was prominent that most defects were due to incorrect requirement documents. The efforts graph showed that lot of effort was spent on rework for fixing of defects which

could have found in functional requirements. Based on this conclusion, a FRS creation process was revised.

4. Percentage of defects found in RA review process was small. In order to eliminate maximum defects at requirement level, it was necessary to have strong review process. Decision was taken to have senior level review along with peer reviews. Testers were encouraged to do comparison testing between existing application and new application so that conditions missed by COBOL miners can be found out.
5. Since the future iterations were supposed to have processes based on modules developed in first iteration, integration testing will be prime focus.
6. The percentage of rejected defects was high in first iteration. After doing analysis of these defects, it was concluded that many defects were due to incorrect data. Since all processes updating database were under construction, people were creating test data from backend. In the process, some invalid data was injected. This data was a major culprit in defect coming up in system testing. It was decided to clean up database and create test data from functionalities, wherever possible.

4.5 Process Improvement in Iteration II

Decisions based on analysis of data collected in first iteration were fruitful. Defects in 'Documentation' category reduced by 27%. There was reduction of approximately 18% in defects leaked to UAT.

In spite of implementation of metrics for focusing integration related defects, the percentage of defects reported in that category remained same. Although the defects in integration of one section of application were reduced, those in other section increased.

Following are the representative graphs for results achieved in second iteration.

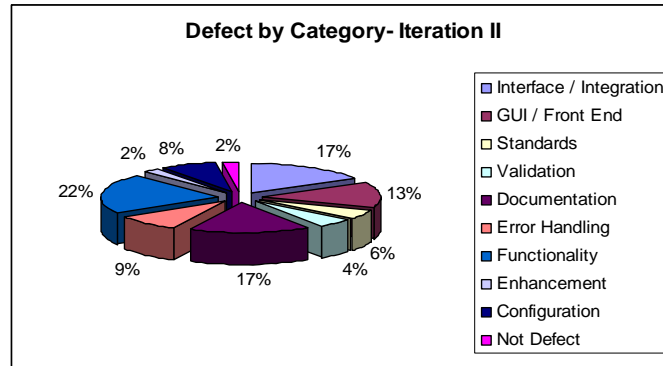


Fig.6: Defects by Category- Iteration II

Refinement of requirement review process was reflected in increase in percentage of defects detected in RA review phase.

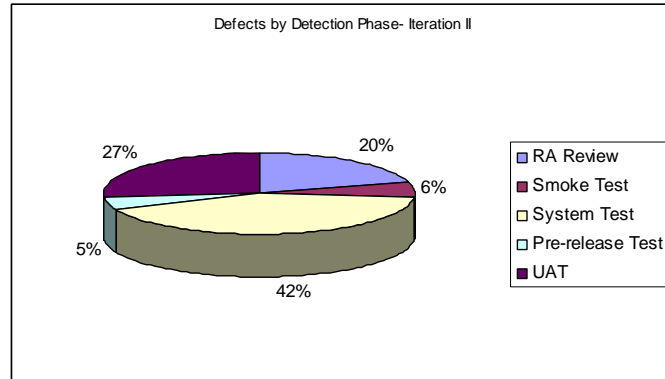


Fig.7: Defects by Category- Iteration II

4.6 Story Continues....

Due to encouraging results of Test metrics implementation in first and second iteration, project management kept strong encouragement for the initiative. In spite of increased complexity and dependencies in further iterations, interface/integration related defects were reduced considerably. Reduction in defects reported in UAT resulted in customer satisfaction and client agreed on increasing efforts for metrics implementation.

5 Conclusions

Test Metrics implementation in Iterative development should be based on following:

- Invest less time and efforts for metrics implementation by measures like Automation of Data collection and analysis.
- Accurate selection of metrics and data point to avoid collection of data that is of no use.
- As requirements change frequently in iterative development and new risks are identified, introduce new metrics or reform the existing metrics whenever required.
- Quick decisions and their implementation in order to get proper results.
- Total involvement of project management and other stakeholders.
- Timely and frequent communication with stakeholders.
- Avoid using metrics for evaluation of individual performance.
- Challenges are never ending but measurements and preventive actions will always help getting proper results.

References:

- [1]. Peter Kulik, Catherine Weber, Software Metrics Best Practices – 2001, March 2002, <http://visualbasic.ittoolbox.com/pub/PK043002.pdf>
- [2]. Pusala Ramesh, Infosys 'Operational Excellence through Efficient Software Testing Metrics' 'operational-excellence.pdf'
- [3]. Peter Eriksson, 'Metrics Tools' <http://www.users.abo.fi/kaisa/EO.pdf>.
- [4]. Stale Amland, 'Risk Based Testing and Metrics', EuroSTAR'99 www.amland.no/WordDocuments/EuroSTAR99Paper.doc
- [5]. Ed Carroll, Agilis Solutions. 'A Software Metrics Case Study.Doc' www.agilissolutions.com/frontarticles/doc/Software%20Metrics.doc
- [6]. Butcher, Munro, Kratschmer, 'Improving Software testing via ODC: Three case studies' IBM System Journal, Volume 41, No. 1, 2002.