



Protect what you value.

Test Execution through Metrics

Jayaprakash Prabhakar
QA Manager – Consumer Engineering



McAfee Inc

© 2007 McAfee, Inc.

2

Agenda

- What is Metrics?
- Flow of Metrics Collection
- Who are the interested Parties?
- When to collect Metrics?
- Various Metrics
- Conclusion



9/22/2007

Protect what you value.

What is Test Metrics?

3

A mechanism to know the effectiveness of the testing that

can be measured quantitatively
at every stage of test life cycle

It is a feedback mechanism to improve the testing /
development process that is being followed currently in the
organization / project.

McAfee

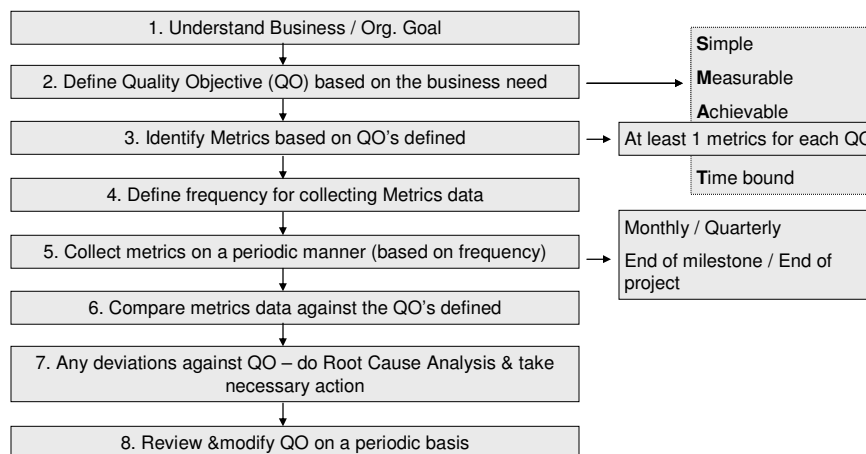
9/22/2007



Protect what you value.

8 Steps for Metrics Collection

4



McAfee

9/22/2007



Protect what you value.


5

Who are the interested?

Interested Parties	Purpose
Test Engineers	To understand their contributions to the project they work on.
Project Management / Test (Middle Mgmt)	To measure the health of the project on every milestone / phase of project till closure
Development Team (Middle Mgmt)	To know their contribution towards reducing defects and test productivity
Top Management	To know the function of test organization & progress towards achieving Organizational goal

When to collect metrics?
QA Metrics can be collected right from the beginning of the project till end of the project, in fact even after the release.

More examples are given in the following slides.



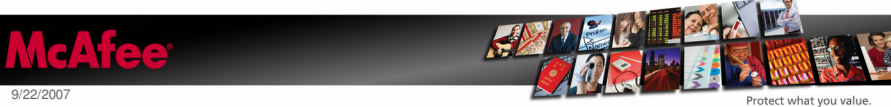
9/22/2007 Protect what you value.

6

Metrics – on Individual, Project & Organization

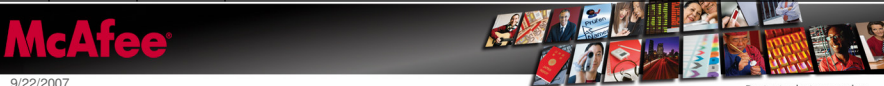
- ❖ **Metrics on individual**
To measure individual contributor's performance over a period
- ❖ **Project-level Metrics**
To know the health of the project based on the quality objectives defined for the project
- ❖ **Organizational Metrics**
To understand the overall organization's performance – based on the organizational quality objectives set

Note: The stated quality objectives in the below sections are only taken as example to illustrate the point and do not state recommended quality objectives for any group or organization to follow.




9/22/2007 Protect what you value.

Metrics on Individual 7				
Metrics	Formula	Quality Objective	Description	Frequency
Cost per bug (in \$value)	(Salary of Employee / Total number of Fixed Defects reported by employee)	Should not be more than \$7.	<p>Cost of the bug represents the \$ value spent as salary to each employee to find each bug.</p> <p>This metrics gives a fair idea about the money spent for each bug. Effectiveness of each resource will be reflected directly.</p> <p>Example: Annual Salary of X = \$ 5,000 No. of fixed bugs found by X = 500 Cost of the bug = 50,000 / 500 = \$10</p> <p>Annual Salary of Y = \$ 4,000 No. of fixed bugs found by Y = 600 Cost of the bug = 40,000 / 600 = \$6.7</p> <p>Resource Y is more effective than resource X.</p> <p>Note: This is the simplest form of finding the cost of bug. This doesn't include any other cost for the resources except salary. This comparison should be done ONLY for similar products (technology, quality of the product etc).</p>	Monthly / Quarterly / End of each Milestone
Average Weighted Defect Count (in #)	(S1*10 + S2*4 + S3*2 + S1*1) / Total Number of Raw Defects	Not less than 5	<p>AWDC represents the weighted average of severity of bugs. All the bugs are considered for this metrics.</p> <p>Example: Resource X: S1-4, S2-6, S3-15, S4-10 - Total-35 Resource Y: S1-11, S2-10, S3-4, S4-4 - Total-29</p> <p>AWDC of X = (4*10+6*4+15*2+10*1)/35 = 2.97 AWDC of Y = (11*10+10*4+4*2+4*1)/29 = 5.59</p> <p>Though the total number of bugs reported by Y are less, his/her contribution is quite high due to high severity bugs reported.</p>	Monthly / Quarterly / End of each Milestone

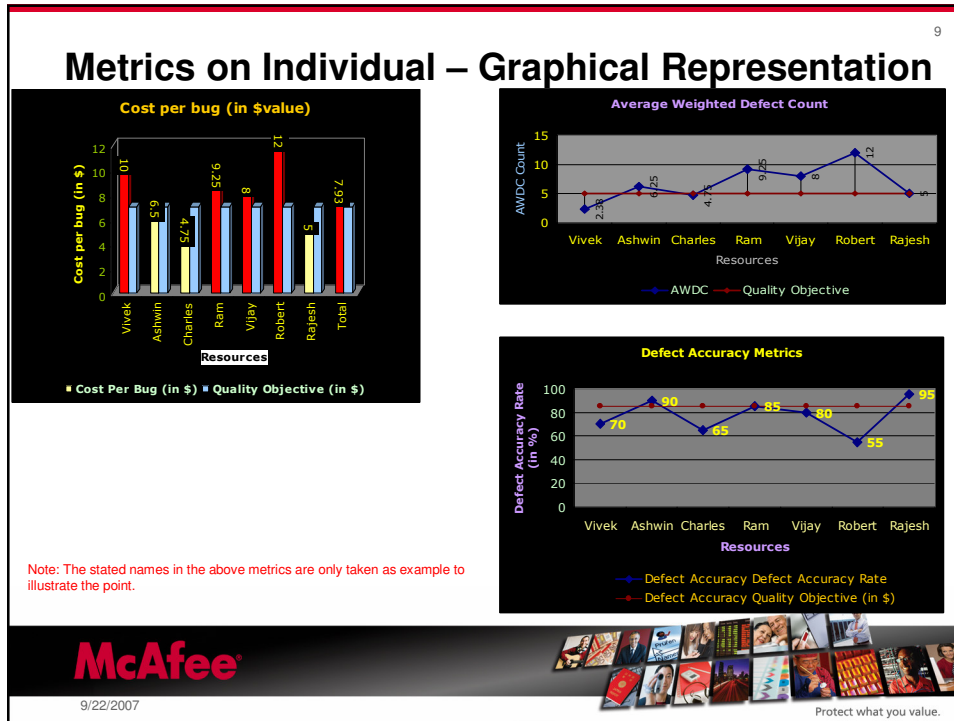


9/22/2007 Protect what you value.

Metrics on Individual – Contd. 8				
Metrics	Formula	Quality Objective	Description	Frequency
Defect Accuracy (in %)	[Total # of valid defects / Total # of raw defects] * 100	Not less than 85%	<p>This represents the % of valid defects reported by each resource.</p> <p>Invalid defects are high cost bugs, as there is no return on reporting these bugs.</p> <p>Total # of defects reported Resource X = 100 No. of valid defects (excluding Rejected & As-Designed) = 67 Defect Accuracy = 67 / 100 * 100 = 67%</p> <p>Total # of defects reported Resource Y = 70 No. of valid defects (excluding Rejected & As-Designed) = 60 Defect Accuracy = 60 / 70 * 100 = 85.71%</p> <p>Resource Y's defect accuracy is higher than Resource Y.</p>	Monthly / Quarterly / End of each Milestone
Defect Effectiveness (in %)	[Total # of fixed defects / Total # of raw defects] * 100	Not less than 70%	<p>Test Engineer's job is not only to find bugs, but also make sure that they are fixed. Its effective only when they are fixed. DE represents the % of fixed-defects for each Test Resource.</p> <p>Total # of defects reported by X = 70 Total # of fixed = 45 Defect Effectiveness = [45/70] * 100 = 64.29%</p> <p>Total # of defects reported by Y = 65 Total # of fixed = 50 Defect Effectiveness = [50/65] * 100 = 76.92%</p>	Monthly / Quarterly / End of each Milestone



9/22/2007 Protect what you value.



Project Level Metrics



Metrics	Formula	Quality Objective	Description	Frequency
Test Effort Variance Metrics (in %)	$\frac{[(\text{Total Actual Effort} - \text{Total Planned Effort}) / \text{Total planned effort on Testing}] * 100}{}$	Should not exceed more than 5%	This represents the effort overrun/underrun for each project with respect to testing. Project A Total planned Effort = 70 Total Actual Effort = 85 Effort Overrun = $\frac{(85-70)}{70} * 100 = 21.42\%$ Project B Total planned Effort = 115 Total Actual Effort = 120 Effort Overrun = $\frac{(120-115)}{115} * 100 = 4.35\%$	End of every milestone / Project
Schedule Variance Metrics (in calendar days)	The number of days delayed for each deliverable	Should not exceed more than 1 day	SV Metrics represents the total number of days delayed for each deliverable. Planned Delivery Date of Milestone 1 = 10-Aug-2007 Actual Delivery Date of Milestone 1 = 14-Aug-2007 Delay = 4 days	End of every milestone / Project
Test Coverage Metrics – Test Cases Planned Vs Executed (in %)	The total number of test cases planned Vs Executed for each milestone	More than 95% of test cases to be executed	This metrics gives a picture as to what is the total coverage of your testing. Deviation in this may lead to more bugs released to customer. Project A Number test cases planned for execution = 750 Number test cases executed = 725 Test Coverage = $\frac{725}{750} * 100 = 96.7\%$ Project B Number test cases planned for execution = 550 Number test cases executed = 500 Test Coverage = $\frac{500}{550} * 100 = 90.1\%$	End of every milestone / Project

9/22/2007

11

Project Level Metrics – Contd.

Metrics	Formula	Quality Objective	Description	Frequency
% Reopened Issues	$\frac{\text{Number bugs reopened} / \text{Number of bugs fixed}}{100} * 100$	Should not be more than 5%	This represents the % reopened bugs against the total fixed bugs. Number of bugs fixed = 125 Number of bugs reopened in 125 = 10 % reopened bugs = $[\frac{10}{125}] * 100 = 8\%$	End of every milestone / Project
% of unit defects found during QA testing	$\frac{\text{Number of Unit defects found by QA} / \text{Total number of bugs reported by QA}}{100} * 100$	Should not be more than 2%	Defects should be identified at early stage of life cycle. The unit defects are the ones, which should have been reported by DEV and should have got fixed before released to QA. This gives a picture about the unit test process followed in the project. Total Number of bugs reported by QA = 100 # of unit bugs reported by QA = 7 % unit bugs reported by QA = $[\frac{7}{100}] * 100 = 7\%$	End of every milestone / Project
Test Effectiveness Metrics Or Post-Release Defect Metrics	$\frac{\text{Total number of defects reported by customer} / \text{total number of defects reported}}{100} * 100$	Should not be more than 5%	This measures the number of defects not found / not fixed before release and found by customers. Project A Total number of defects reported by QA = 100 Total number of defects reported by Customer (post release) = 45 Test Effectiveness = $(45 / 145) * 100 = 31\%$ Project B Total number of defects reported by QA = 100 Total number of defects reported by Customer (post release) = 4 Test Effectiveness = $(4 / 104) * 100 = 3.8\%$	n-months after release of the project






9/22/2007 Protect what you value.

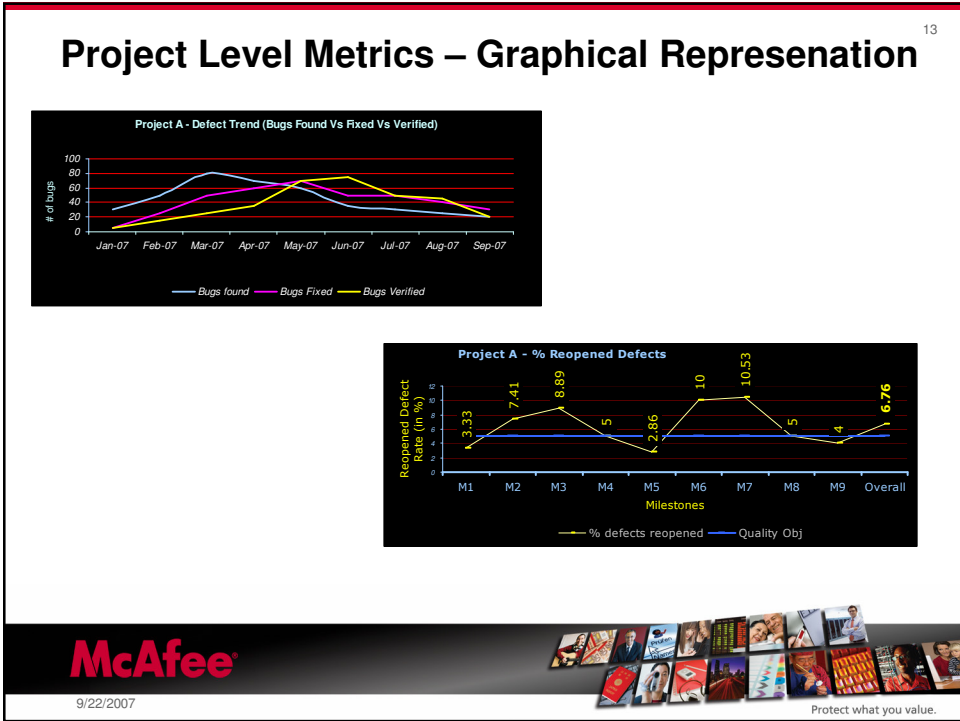
12

Project Level Metrics – Contd.

Metrics	Formula	Quality Objective	Description	Frequency
Defect Trend – Found Vs Fixed Vs Verified (Trend over a period)	Trend of bugs reported / fixed / verified over a period	Trend should be decreasing over a period	The defect trend shows the no. of bugs reported / fixed over a period. Ideally, the number of bugs found should come down to 0 as the project progresses close to the final release.	Through out the project
Test Productivity (Test Execution Productivity in the past Vs. Current)	Time taken to complete execution of one cycle (in the current cycle) / the time taken to complete the same in the past (specific time-period)	20% improvement over a period (specify the period)	This metrics measures the productivity of the team over a period of time. Test Execution of one functional cycle (in 2006) = 15 pd Test Execution of one functional cycle (in 2007) = 9 pd Productivity Increase = $[\frac{15-9}{15}] * 100 = 40\%$ This metrics can be implemented only by product teams, which executes the tests repeatedly over a period of time. Some of the reasons for higher productivity can be – 1. Automation coverage for testing 2. Deeper understanding of the product / domain 3. Strong knowledge on testing methodology / practices etc Note: This is applicable only for long-term projects / product testing.	End of every release of the product

9/22/2007 Protect what you value.



Org Level Metrics



Metrics	Formula	Quality Objective	Description	Frequency
Test Effort Variance Metrics (in %)	$\frac{[(\text{Total Actual Effort} - \text{Total planned Effort}) / \text{Total planned effort on Testing}] * 100}{100}$	Should not exceed more than 5%	This represents the effort overrun/under run for each project with respect to testing. <u>Project A</u> Total planned Effort = 70 Total Actual Effort = 85 Effort Overrun = $[(85-70) / 70] * 100 = 21.42\%$ <u>Project B</u> Total planned Effort = 115 Total Actual Effort = 120 Effort Overrun = $[(120-115) / 115] * 100 = 4.35\%$ <u>Organizational Effort Variance:</u> Total Planned Effort = 70 + 115 = 185 Total Actual Effort = 85 + 120 = 205 Effort Variance = $[(205-185) / 185] * 100 = 10.82\%$	End of every milestone / Project
Schedule Variance Metrics (in calendar days)	The number of days delayed for each deliverable	Should not exceed more than 1 day	Schedule Variance Metrics represents the total number of days delayed for each deliverable. <u>Project A</u> Planned Delivery Date of Milestone 1 = 10-Aug-2007 Actual Delivery Date of Milestone 1 = 14-Aug-2007 Delay = 4 days <u>Project B</u> Planned Delivery Date of Milestone 1 = 14-Aug-2007 Actual Delivery Date of Milestone 1 = 14-Aug-2007 Delay = 0 days	End of every milestone / Project

9/22/2007 Protect what you value.

15

Org Level Metrics – Contd.

Metrics	Formula	Quality Objective	Description	Frequency
Cost per bug (in \$value)	[Salary of Employee / Total number of Fixed Defects reported by employee]	Should not be more than \$7	<p>Cost of the bug represents the \$ value spent as salary to each employee to find each bug.</p> <p>This metrics gives a fair idea about the money spent for each bug. Effectiveness of each resource will be reflected directly.</p> <p>E.g.:</p> <p>Cost per bug (for each employee)</p> <p>A = 5,000 / 600 = \$8.3</p> <p>B = 7,500 / 1000 = \$7.5</p> <p>...</p> <p>X = 5,000 / 500 = \$10</p> <p>Y = 4,000 / 600 = \$6.7</p> <p>Z = 3,000/500 = \$6</p> <p>Cost of the bug at Organizational level = \$90,000 / 12000 = \$7.5</p> <p>Note: This is the simplest form of finding the cost of bug. This doesn't include any other cost for the resources except salary.</p>	Monthly / Quarterly
% Reopened Issues	[Number bugs reopened / Number of bugs fixed] * 100	Should not be more than 5%	<p>This represents the organization-wide % reopened bugs against the total fixed bugs.</p> <p>Number of bugs fixed = 125 Number of bugs reopened in 125 = 10 % reopened bugs = [10 / 125] * 100 = 8%</p>	Monthly / Quarterly






9/22/2007 Protect what you value.

16

Org Level Metrics – Contd.

Metrics	Formula	Quality Objective	Description	Frequency
% of unit defects found during QA testing	Number of Unit defects found by QA / Total number of bugs reported by QA] * 100	Should not be more than 2%	<p>Defects should be identified at early stage of life cycle. The unit defects are the ones, which should have been reported by DEV and should have got fixed before released to QA.</p> <p>This metrics gives a picture about the unit test process followed in the project.</p> <p><u>Project A</u> Total Number of bugs reported by QA = 100 # of unit bugs reported by QA = 7 % unit bugs reported by QA = [7/100] * 100 = 7%</p> <p><u>Project B</u> Total Number of bugs reported by QA = 55 # of unit bugs reported by QA = 4 % unit bugs reported by QA = [4/55] * 100 = 7.3%</p> <p><u>Project C</u> Total Number of bugs reported by QA = 130 # of unit bugs reported by QA = 2 % unit bugs reported by QA = [3/130] * 100 = 1.54%</p> <p><u>Organizational</u> Total Number of bugs reported by QA = 100+55+130 = 285 # of unit bugs reported by QA = 7+4+2 = 13 % unit bugs reported by QA = [13/285] * 100 = 4.56%</p>	End of every milestone / Project






9/22/2007 Protect what you value.

17

Org Level Metrics – Contd.

Metrics	Formula	Quality Objective	Description	Frequency
Test Coverage Metrics – Test Cases Planned Vs Executed (in %)	The total number of test cases planned Vs Executed for each milestone	More than 95% of test cases to be executed	<p>This metrics gives a picture as to what is the total coverage of testing across the organization. Deviation in this may lead to more bugs released to customer.</p> <p>Project A Number test cases planned for execution = 750 Number test cases executed = 725 Test Coverage = $725 / 750 \times 100 = 96.7\%$</p> <p>Project B Number test cases planned for execution = 550 Number test cases executed = 500 Test Coverage = $500 / 550 \times 100 = 90.1\%$</p> <p>Project C Number test cases planned for execution = 550 Number test cases executed = 520 Test Coverage = $525 / 550 \times 100 = 95.5\%$</p> <p>Project D Number test cases planned for execution = 550 Number test cases executed = 500 Test Coverage = $500 / 550 \times 100 = 90.1\%$</p> <p>Organizational Number of test cases planned for execution = $[750+550 +550+550] = 2400$ No. of test cases executed = $[725+500+525+500] = 2250$ Test Coverage = $[2250 / 2400] \times 100 = 93.75\%$</p>	Quarterly / Half-yearly / Annual

9/22/2007 Protect what you value.



18

Conclusion

Metrics collection and analysis is very critical to test organization to prove the quality of product / efficiency rate / effectiveness of the team.

Please follow the below listed guidelines while defining / tracking metrics.

- > Identify **critical & relevant** metrics for the project / organization. Collecting many metrics may confuse the audience and may not be effective. **Poor metrics can mislead management** or drive a wedge between development and test teams.
- > Make sure that the metrics defined are **directly / indirectly associated to your project / organizational goal**.
- > **Make it simple and straight** so everyone can understand.
- > **Share the metrics to all the interested parties** (whichever applicable) so the awareness and drive towards achieving common goal is increased.
- > **Work on continual improvement** based on metrics data.
- > **Review the Quality Objectives** over a period based on the metrics collected.


9/22/2007 Protect what you value.

19

Questions Please??

Thank You !!

jayaprakash_prabhakar@mcafee.com



9/22/2007

Protect what you value.