Software Testing Metrics: Measure the software correctness

M Saeed Siddik, IIT University of Dhaka

Why Metrics in Software Testing?

- **How would you answer questions such as:**
  - **Project oriented questions**
    - **How long would it take to test?**
    - **How much will it cost to test?**
  - **Product oriented questions**
    - **How bad/good is the product ?**
    - **How many problems still remain in the software?**
  - **Test activities oriented questions**
    - **Will testing be completed on time?**
    - **Was the testing effective?**
    - **How much effort went into testing**
- **All these Questions require some type of measurements and record keeping in order to answer properly.**

Size of Test

- **Test size attribute may use different metrics:**
  - Amount of time to run test: **(bit convoluted ?)**
    - **Small size : less than or equal to 3 seconds**
    - **Medium size: between 3 seconds and 1 minute**
    - **Large size: 1 minute or above**
  - Number of lines of statements to document the test case:
    - **Small size: less than or equal to 3 statements**
    - **Medium size: between 4 and 7 statements**
    - **Large size: 8 or more statements**

Any suggestions - - - - ? Number of test cases? --- or
--- type of test such as unit test versus integration test ?----

Quality : # of Problems

- The attribute , Quality, is often measured with the metric of number of problems found; but number of problems alone does not tell the whole story - - - consider
  - Severity of problems
    - High
    - Medium
    - low
  - Type of problems
    - UI
    - Database
    - Network outage
    - Etc.

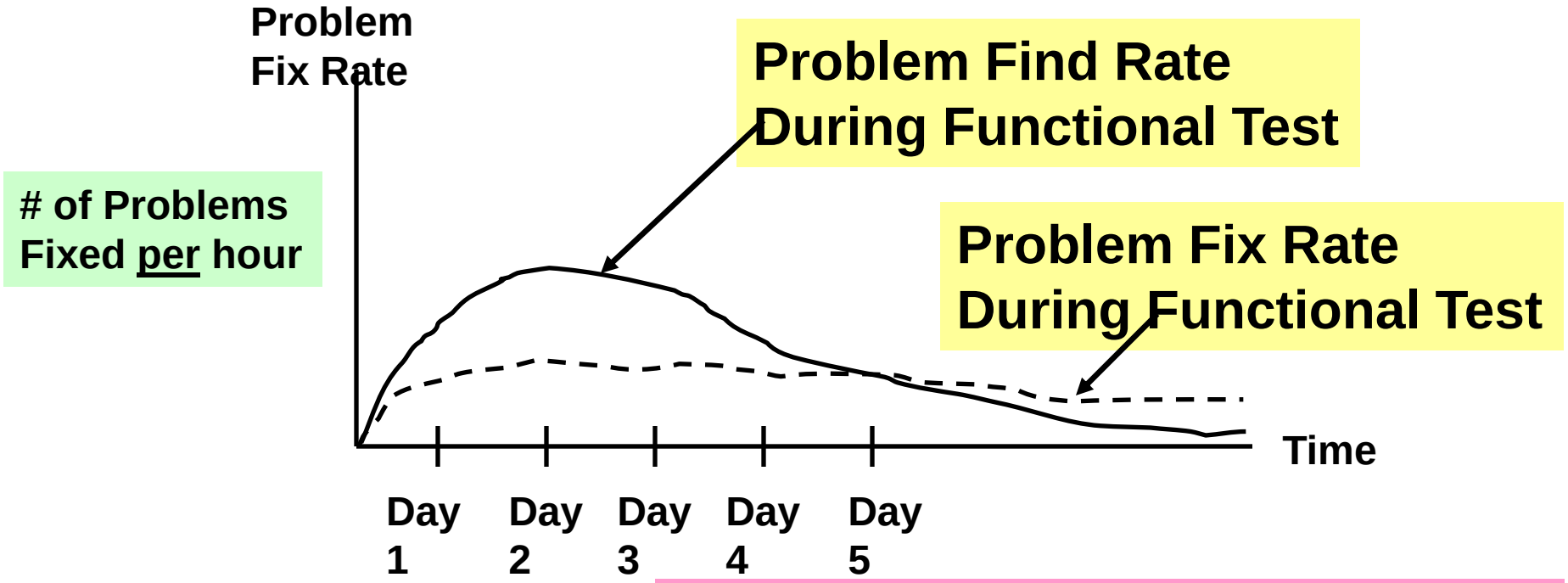
Quality (cont.)

- **Both Severity and Type are important**
  - # of problems found by severity
  - # of problems found by type
  - # of problems found when (when during development)
  - # of problems found when (months after release)
  - # of problems found where (UI,DB, Logic, Network, etc.)

- **Quality Information is relevant to both:**
  - Software providers
  - Customers/users

Why important to users? What would they do with it?

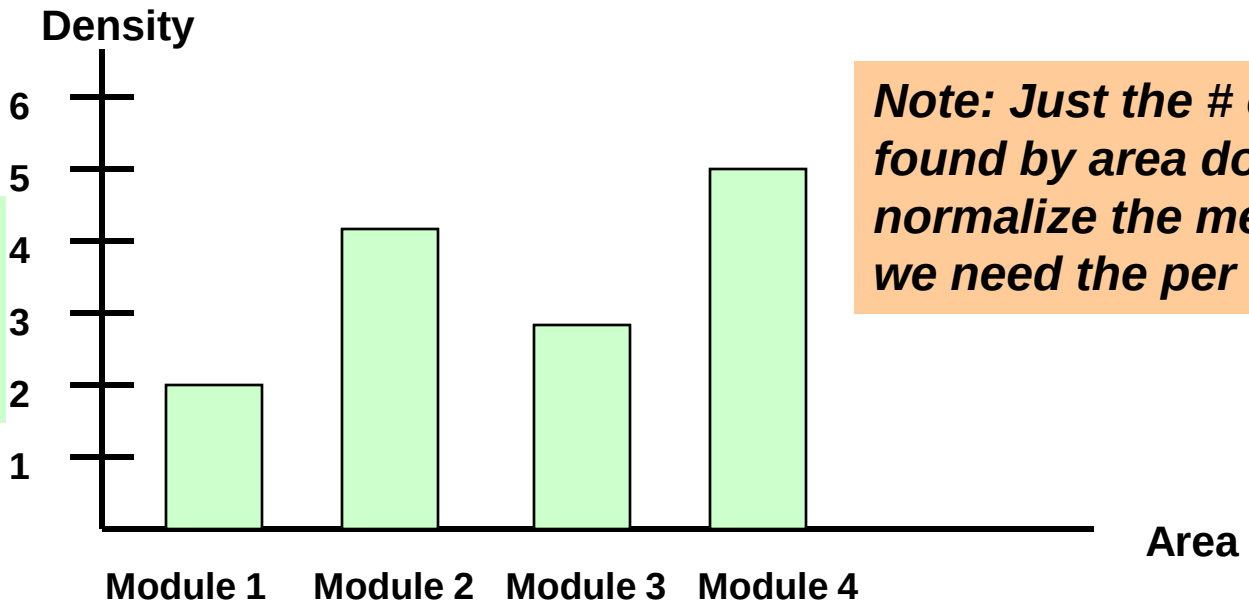
Problem Fix Rate



Would this fix rate present a problem ?
Would you also want to keep a backlog # by day ?

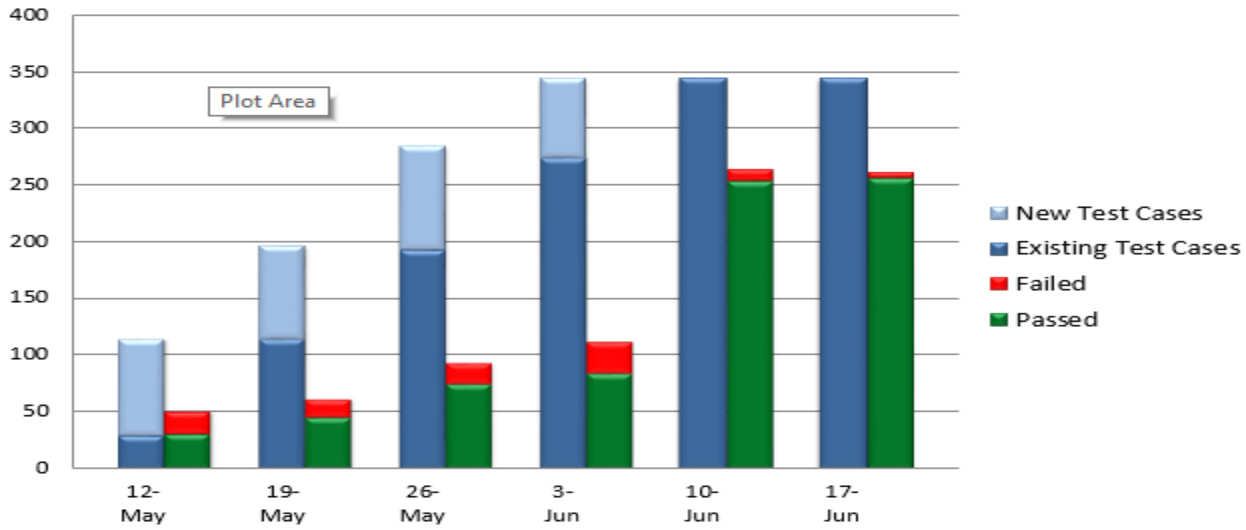
Problem Density

of problems found per KLOC



Note: Just the # of problems found by area does not normalize the measurement; we need the per KLOC.

Testing Execution Progress



Test Coverage Rate

- **Not all the planned test cases are actually run.**
  - ***# of test cases executed / # of test cases planned***
    - **By functional areas**
    - **By test phases**
  - ***# of source statements executed / total # of source statements***
    - **By functional areas**
    - **By modules**

Test Activity Effectiveness

- **Defect discovery and eradication activities occur at all phases of development. To see which is more effective one may use:**
  - **# of problems found / total # of problems found**
    - **By development phase (req. rev., design rev., func. test, system, etc.)**
  - **# of problems found / person-days of effort**
    - **By test activities (e.g. boundary value testing, branch testing, d-u testing, etc.)**

Fix Effectiveness

- Not all problem fixes resolve the problems.
  - # of fixes that worked / total # of fixes
    - The first time
  - # of fixes that required more than 1 fix / total number of fixes

Fix Cost

- **Fix cost is usually measured by amount of effort expended.**
  - **# of person-hours expended / fix**
    - **By severity**
    - **By areas**
    - **By phase type (including post-release)**

If the fix cost for post-release is higher than that of all of the pre-release phases, then that will be one reason for test and reviews.

Test Metrics

- **Defect Severity Index:** It is the degree of impact a defect has on the development of an operation or a component of a software application being tested. Defect severity index (DSI) offers an insight into the quality of the product under test and helps gauge the quality of the test team's efforts. Additionally, with the assistance of this metric, the team can evaluate the degree of negative impact on the quality as well as the performance of the software. Following formula is used to measure the defect severity index.
- **Defect Severity Index (DSI) = Sum of (Defect * Severity Level) / Total number of defects**

Test Metrics

- **Test Case Effectiveness:** The objective of this metric is to know the efficiency of test cases that are executed by the team of testers during every testing phase. It helps in determining the quality of the test cases.
- **Test Case Effectiveness = (Number of defects detected / Number of test cases run) x 100**
- **Test Case Productivity:** This metric is used to measure and calculate the number of test cases prepared by the team of testers and the efforts invested by them in the process. It is used to determine the test case design productivity and is used as an input for future measurement and estimation. This is usually measured with the assistance of the following formula:
- **Test Case Productivity = (Number of Test Cases / Efforts Spent for Test Case Preparation)**

Test Metrics

- **Test Effectiveness:** A contrast to test efficiency, test effectiveness measures and evaluates the bugs and defect ability as well as the quality of a test set. It finds defects and isolates them from the software product and its deliverables. Moreover, the test effectiveness metrics offer the percentage of the difference between the total number of defects found by the software testing and the number of defects found in the software. This is mainly calculated with the assistance of the following formula:
- **Test Effectiveness (TEF) = (Total number of defects injected + Total number of defects found / Total number of defect escaped) x 100**

Example: Defect Severity Index (DSI)

Formula: DSI = Sum of (Defect∗Severity Level) / Total number of defects

Imagine a software testing cycle where a team found 10 defects with the following severity levels:

Critical (Severity Level 4): 1 defect, **Major (Severity Level 3):** 3 defects,

Minor (Severity Level 2): 4 defects, **Trivial (Severity Level 1):** 2 defects

Calculation:

Sum of (Defect * Severity Level) = (1×4)+(3×3)+(4×2)+(2×1) = 4+9+8+2=23

Total number of defects = 1+3+4+2=10

DSI=1023=2.3

Interpretation: A DSI of 2.3 indicates that the average severity of defects found is between "Minor" and "Major." A higher DSI suggests the product has more severe bugs.

Example: Test Case Effectiveness

Formula: Test Case Effectiveness =
(Number of defects detected / Number of test cases run)×100

A testing team executes a total of 200 test cases on a module. And found 25 unique defects.

Calculation:

Number of defects detected = 25

Number of test cases run = 200

Test Case Effectiveness = 20025×100=12.5%

Interpretation: A Test Case Effectiveness of 12.5% suggests that for every 100 test cases run, the team found 12.5 defects. This metric helps evaluate the quality of the test cases; a higher percentage indicates more effective test cases.

Example: Test Case Productivity

Formula: Test Case Productivity =
Number of Test Cases / Efforts Spent for Test Case Preparation

Example:

A team of 3 testers works for 5 days to prepare test cases. Each tester works 8 hours per day. They successfully prepare a total of 150 test cases.

Calculation:

Number of Test Cases = 150

Efforts Spent = 3 testers×5 days×8 hours/day=120 person-hours

Test Case Productivity = 120150=1.25

Interpretation: This result means the team's productivity is 1.25 test cases per person-hour of effort. This metric is useful for project managers to estimate the time and effort needed for future test case preparation tasks.

Example: Test Effectiveness (TEF)

Formula: TEF=Total number of defects found / (Total number of defects found+Total number of defect escaped) ×100

Example:

During a testing phase, the team finds 125 defects. After the software is released to the market, customers report 25 additional defects that the testing team missed (these are the "escaped defects").

Calculation:

Total number of defects found by the testing team = 125

Total number of defects escaped to the field = 25

`TEF=125/(125+25)×100 = 125/150 × 100 = 83.33%`

Interpretation: A Test Effectiveness of 83.33% means that the testing team successfully found 83.33% of all the bugs that existed in the product before its release. A higher percentage indicates more effective testing.

Test Tracking & Efficiency:

- **Passed Test Cases Coverage:** It measures the percentage of passed test cases.
  (Number of passed tests / Total number of tests executed) x 100
- **Failed Test Case Coverage:** It measures the percentage of all the failed test cases.
  (Number of failed tests / Total number of test cases failed) x 100
- **Test Cases Blocked:** Determines the percentage of test cases blocked, during the software testing process.
  (Number of blocked tests / Total number of tests executed) x 100
- **Fixed Defects Percentage:** With the assistance of this metric, the team is able to identify the percentage of defects fixed.
  (Defect fixed / Total number of defects reported) x 100

Test Tracking & Efficiency:

- **Accepted Defects Percentage:** The focus here is to define the total number of defects accepted by the development team. These are also measured in percentage.
  (Defects accepted as valid / Total defect reported) x 100

- **Defects Rejected Percentage:** Another important metric considered under test track and efficiency is the percentage of defects rejected by the development team.
  (Number of defects rejected by the development team / total defects reported) x 100

- **Defects Deferred Percentage:** It determines the percentage of defects deferred by the team for future releases.
  (Defects deferred for future releases / Total defects reported) x 100

- **Critical Defects Percentage:** Measures the percentage of critical defects in the software.
  (Critical defects / Total defects reported) x 100

Example of Test Metric

- **Rework Effort Ratio =** (Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100
- **Requirement Creep =** (Total number of requirements added/No of initial requirements)X100
- **Schedule Variance =** (Actual efforts – estimated efforts) / Estimated Efforts) X 100
- **Cost of finding a defect in testing =** (Total effort spent on testing/ defects found in testing)
- **Schedule slippage =** (Actual end date – Estimated end date) / (Planned End Date – Planned Start Date) X 100
- **Passed Test Cases Percentage** = (Number of Passed Tests/Total number of tests executed) X 100

Example of Test Metric

- **Failed Test Cases Percentage** = (Number of Failed Tests/Total number of tests executed) X 100
- **Blocked Test Cases Percentage** = (Number of Blocked Tests/Total number of tests executed) X 100
- **Fixed Defects Percentage** = (Defects Fixed/Defects Reported) X 100
- **Accepted Defects Percentage** = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100
- **Defects Deferred Percentage** = (Defects deferred for future releases /Total Defects Reported) X 100

Example of Test Metric

- **Critical Defects Percentage** = (Critical Defects / Total Defects Reported) X 100
- **Average time for a development team to repair defects** = (Total time taken for bugfixes/Number of bugs)
- **Number of tests run per time period** = Number of tests run/Total time
- **Test design efficiency** = Number of tests designed /Total time
- **Test review efficiency** = Number of tests reviewed /Total time
- **Bug find rote or Number of defects per test hour** = Total number of defects/Total number of test hours

End of Test Metrics