Debre Markos University
Institute of Technology
School of Computing, Software Eng A/program

Software Quality Assurance and Testing (SEng4112)
Chapter Six
Software Testing Metrics and Tools

Objective

- **After successful completion of this chapter, students will be able to understand**
  - ✓ Software Testing Metrics
  - ✓ Software Testing Tools

Software Testing Metric

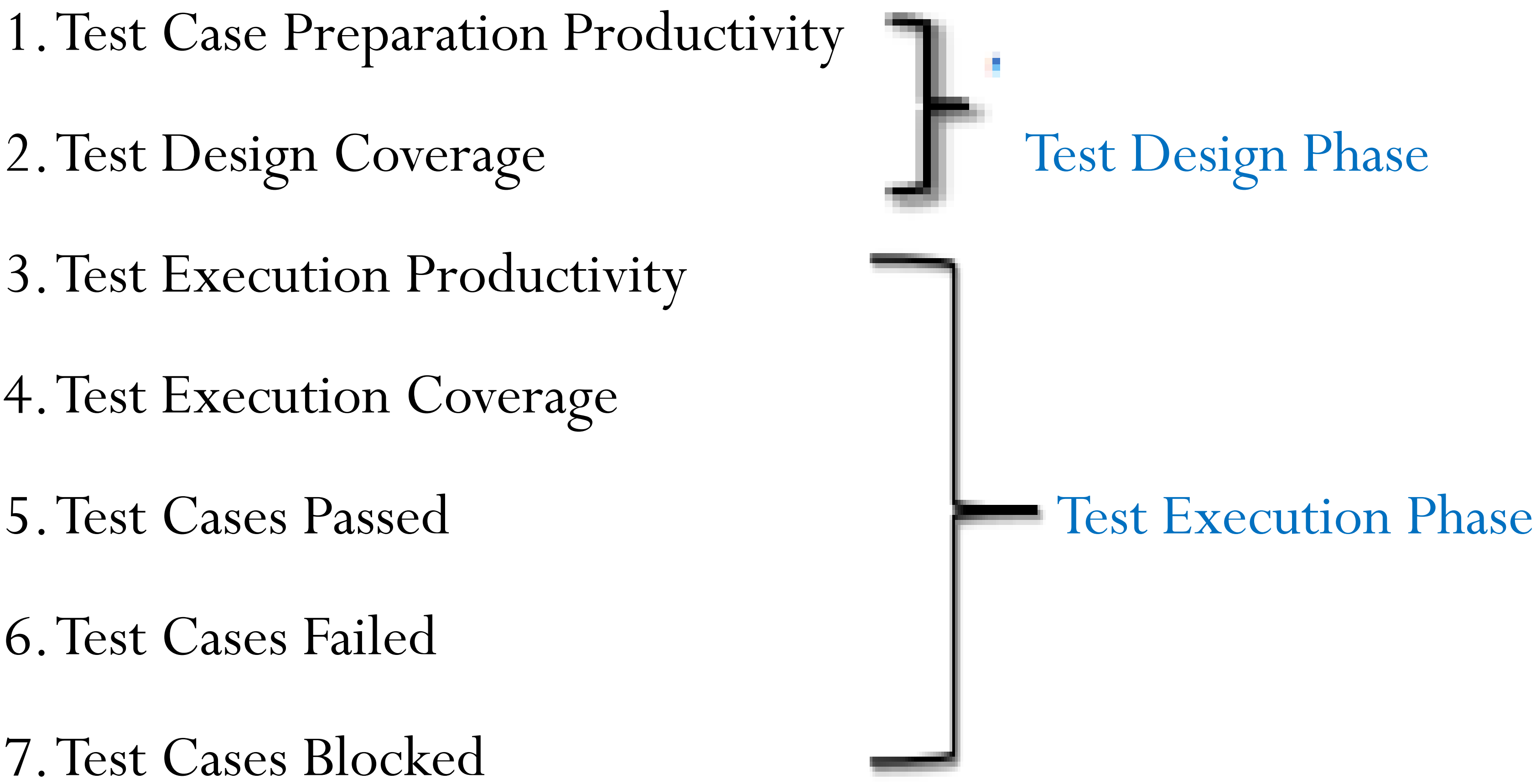
- In software testing, metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute.
- In other words, metrics helps estimating the progress and quality of a software testing effort.
- Software testing metrics improve the effectiveness and efficiency of a software testing process.
- Software test metrics is used for monitoring and controlling processes and products.
- It helps to drive the project towards our planned goals without deviation.

Why test metrics?

- Test metrics helps us to
  - Take decision for next phase of activities
  - Take evidence of the claim or prediction
  - Understand the type of improvement required
  - Take decision or process or technology change

Software Test Metrics

- Software test metrics used in the process of **test preparation** and **test execution** phases of the STLC include:



1. Test Case Preparation Productivity
2. Test Design Coverage
3. Test Execution Productivity
4. Test Execution Coverage
5. Test Cases Passed
6. Test Cases Failed
7. Test Cases Blocked

Test Design Phase

Test Execution Phase

1. Test Case Preparation Productivity

- It is used to calculate the number of test cases prepared and the effort spent for the preparation of test cases.
- **Formula:**

$$TCPP = \frac{\boldsymbol{No.of\ TCs}}{\boldsymbol{Effort\ Spent\ for\ TC\ preparation}}$$

TCPP: Test Case Preparation Productivity

TCs: Test Cases

- E.g.:-
- No. of Test cases = 240
- Effort spent for Test case preparation (in hours) = 80 hours
- Test Case preparation productivity = 240/80 = 30 test cases/hour
- Asap, higher test case productivity is recommended

2. Test Design Coverage

- It helps to measure the percentage of test case coverage against the number of requirements.
- **Formula:**

$$\mathrm{TDC} = \left(\frac{\textit{Total No.of requirements mapped to TCs}}{\textit{Total No.of requirements}}\right) * 100$$

TDC: Test Design Coverage

TCs: Test Cases

- E.g.:-
- Total number of requirements: 78
- Total number of requirements mapped to test cases: 69
- Test design coverage = (69/78)*100 = **88.5%**

3. Test Execution Productivity

- It determines the number of Test Cases that can be executed per hour
- **Formula:**

$$\text{TEP} = \frac{\boldsymbol{No.of\ TCs\ Executed}}{\boldsymbol{Effort\ Spend\ for\ Execution\ of\ TCs}}$$

TEP: Test Execution Productivity

TCs: Test Cases

- **E.g.:-**
- No. of Test cases executed = 180
- Effort spent for execution of test cases = 10 hours
- Test Execution Productivity = 180/10 = **18 test cases/hour**

4. Test Execution Coverage

- It is used to measure the number of test cases executed against the number of test cases planned to be executed.
- **Formula:**

$$\text{TEC} = \left(\frac{\textit{Total No.of TCs Executed}}{\textit{Total No. of TCs planned to be executed}}\right) * 100$$

TEC: Test Execution Coverage

TCs: Test Cases

- **E.g.:-**
- Total no. of test cases planned to be executed = 240
- Total no. of test cases executed = 180
- Test Execution Coverage = (180/240)*100 = **75%**

5. Test Cases Passed

- It measures the percentage of test cases passed during test execution. Also known as pass rate.
- **Formula:**

$$\mathrm{TCP} = \left(\frac{\boldsymbol{Total\ No.of\ TCs\ Passed}}{\boldsymbol{Total\ No.\ of\ TCs\ Executed}}\right) * \mathbf{100}$$

TCP: Test Cases Passed

TCs: Test Cases

- **E.g.:-**
- Total number of test cases passed = 80
- Total number of test cases executed = 90
- Test Cases Passed = (80/90)*100 = **88.8 ≈ 89%**

6. Test Cases Failed

- It measures the percentage of test cases failed the test.

Formula:

$$\mathrm{TCF} = \left(\frac{\textit{Total No.of TCs Failed}}{\textit{Total No. of TCs Executed}}\right) * 100$$

TCF: Test Cases Failed

TCs: Test Cases

E.g.:-

- Total no. of test cases failed = 10
- Total no. of test cases executed = 90
- Test Cases Failed= (10/90)*100 = **11.1 ≈ 11%**

7. Test Cases Blocked

- It measures the percentage of test cases blocked during the test.
- **Formula:**

$$\text{TCB} = \left(\frac{\textit{Total No.of TCs Blocked}}{\textit{Total No. of TCs Executed}}\right) * 100$$

TCB: Test Cases Blocked

TCs: Test Cases

- **E.g.:-**
- Total number of test cases blocked = 5
- Total number of test cases executed = 90
- Test Cases Blocked = (5/90)*100 = **5.5 ≈ 6%**

Defect Analysis Metrics

- Software test metrics used in the process of defect analysis phase of the STLC includes:

    1. Error Discovery Rate
    2. Defect Fix Rate
    3. Defect Density
    4. Defect Leakage
    5. Defect Removal Efficiency

1. Error Discovery Rate

- It determines the effectiveness of the test cases.
- **Formula:**

$$\text{EDR} = \left(\frac{\textit{Total No.of Defects Found}}{\textit{Total No. of TCs Executed}}\right) * 100$$

EDR: Error Discovery Rate

TCs: Test Cases

- **E.g.:-**
- Total number of defects found = 60
- Total number of test cases executed = 240
- Error Discovery Rate = (60/240)*100 = **25%**

2. Defect Fix Rate

- It helps to know the quality of a build in terms of defect fixing.
- **Formula:**

$$DFR = \left(\frac{\boldsymbol{Total\ No.of\ Defects\ reported\ as\ fixed - Totla\ No.\ Dffects\ Reopened}}{\boldsymbol{Total\ No.\ of\ defects\ reported\ as\ fixed + Total\ No.of\ new\ bug\ due\ to\ fix}}\right) * 100$$

DFR: Defect Fix Rate

- **E.g.:-**
- Total number of defects reported as fixed = 10
- Total number of defects reopened = 2
- Total number of new Bugs due to fix = 1
- Defect Fix Rate = ((10 – 2)/(10 + 1))*100 = (8/11)100 = **72.7 ≈ 73%**

3. Defect Density

- Defect density is the number of confirmed defects detected in the software or a component during a defined period of development or operation, divided by the size of the software.
- It is defined as the ratio of defects to requirements.
- Defect density determines the stability of the application.
- **Formula:**

$$\text{Defect Density} = \frac{\textbf{\textit{Total No.of Defects identified}}}{\textbf{\textit{Actual Size of Software Product}}}$$

Cont...

- E.g.: Suppose you have a software product which has been integrated from 4 modules and you found the following bugs in each of the modules.
  - Module 1 = 20 bugs
  - Module 2 = 30 bugs
  - Module 3 = 50 bugs
  - Module 4 = 60 bugs
- And the total line of code for each module is
  - Module 1 = 1200 LoC
  - Module 2 = 3023 LoC
  - Module 3 = 5034 LoC
  - Module 4 = 6032 LoC
- Then, we calculate defect density as
  - Total bugs = 20+30+50+60 = 160
  - Total Size= 15289 LOC
  - Defect Density = 160/15289

    = 0.01046504 defects/LoC

    = 10.5 defects/KLoC

4. Defect Leakage

- It is used to review the efficiency of the testing process before User Acceptance Testing (UAT).
- **Formula:**

$$\text{Defect Leakage} = \left(\frac{\textit{Total No.of Defects Found in UAT}}{\textit{Total No.of Defects Found Before UAT}}\right) * 100$$

- **E.g.:-**
- Number of defects found in UAT = 20
- Number of defects found before UAT = 120
- Defect Leakage = (20 / 120) * 100 = **16.6 ≈ 17%**

5. Defect Removal Efficiency

- It allows us to compare the overall defect removal efficiency.
- It includes defects found pre and post-delivery.
- **Formula:**

$$\text{DRE} = \left(\frac{\boldsymbol{Total\ No.of\ Defects\ found\ pre\_delivery}}{\boldsymbol{Total\ No.\ of\ defects\ found\ pre\_delivery + Total\ No.\ of\ defects\ found\ post\_delivery}}\right) * 100$$

DRE: Defect Removal Efficiency

- **E.g.:-**
- Total no. of defects found pre-delivery = 80
- Total no. of defects found post-delivery = 10
- Defect Removal Efficiency = ((80) / ((80) + (10)))*100

  = (80/90)*100 = **88.8 ≈ 89%**

Software Testing Tools

Manual Testing Vs Automated Testing

- Software testing is carried out using manual and automated software testing tools
- Tester should have the perspective of an end-user and to ensure all the features are working as mentioned in the requirement document.
- In this process, testers execute the test cases and generate the reports manually, without using any automation tools.

Manual Testing

Advantages:

- It can be done on all kinds of applications
- It is preferable for short life cycle products
- Newly designed test cases should be executed manually
- Application must be tested manually before it is automated
- It is preferred in the projects where the requirements change frequently and for the products where the GUI changes constantly
- It is cheaper in terms of initial investment compared to automation testing
- It requires less time and expense to begin productive manual testing
- There is no necessity to the tester to have knowledge on automation Tools

Cont…

Limitations:

- Manual testing is time-consuming mainly while doing regression testing.
- In the long run, expensive over automation testing

Automated Testing

- Automated testing is the process of testing a software using automated tools.
- Automated testing means running the software programs that carry out the execution of test cases automatically and produce the test results without any human intervention.
- In this process, executing the test scripts and generating the results are performed automatically by automated tools.

Cont...

Advantages:

- Automation testing is faster in execution
- It is cheaper compared to manual testing in a long run
- It is more reliable
- It is more powerful and versatile
- It is mostly used for regression testing
- It does not require human intervention.
- It helps to increase the test coverage

Cont…

Limitations:

- It is recommended only for stable products
- Automation testing is expensive initially
- Most of the automation tools are expensive
- It has some limitations such as handling captcha, fonts, color
- Huge maintenance in case of repeated changes in the requirements
- Not all the tools support all kinds of testing.

Testing Tool Acquisition

- Now days we can get lots of software testing tools in the market.
- Testing tool acquisition define how and when the tool will be used.
- Selection of tools is totally based on the project requirements
  - Proprietary/commercial tools or
  - Free tools/open source tools
- Off course, free testing tools may have some limitation in the features list of the product, so it’s totally based on what are you looking for & is that your requirement fulfill in free version or go for paid software testing tools.

How to select the right testing tools

- Understand your project requirements thoroughly.
  - Project type
  - Scope of project
  - Technical expertise
- Consider your existing test automation tool as benchmark
  - Understand pros and cons
  - Evaluate and determine the best tools
- Evaluate the selected tool on key criteria
  - Easy to use
  - OS compatibility
  - Multiple languages
  - Test reporting
  - Platform support
- Leverage a matrix technique for analysis

1. Selenium

- It is the most widely used automated testing tool among all web application testing tools.
- Selenium can be executed in multiple browsers and operating systems.
- It is compatible with several programming languages and automation testing frameworks.
- With selenium, you can come up with very powerful browser-centered automation test scripts which are scalable across different environments.
- You can also create scripts using Selenium that is of great help for prompt reproduction of bugs, regression testing, and exploratory testing.

2. Unified Functional Testing (UFT) QTP

- Unified Functional Testing (UFT) tool given by Hewlett-Packard Enterprise is one of the best automation testing software for functional testing.
- It was previously known as Quick Test Professional (QTP).
- It brings developers & testers coming together under one umbrella and provides high-quality automation testing solutions.
- It makes functional testing less complex and cost-friendly.

3. HP Quality Center

- It is basically an integrated IT quality management software.
- Automated testing is one of its key features which constantly allows you to test earlier and quicker.
- Asset sharing and reusability allows QC to deliver bug-free and reliable applications.
- HP-ALM helps us to manage project milestones, deliverables, and resources.

4. Testim.io

- It leverages machine learning for the authoring, execution, and maintenance of automated test cases.
- We use dynamic locators and learn with every execution.
- The outcome is super fast authoring and stable tests that learn, thus eliminating the need to continually maintain tests with every code change.

5.TestComplete

- It allows all level of users to quickly create powerful, reusable and time-saving GUI automation tests for the web, mobile, and desktop applications.
- It lets you combine the recorded scripts and tests into a single framework which reduces the training cost and testing time.
- The best part of the tool is TestComplete Visualizer, which is a screenshot based feature allows you to modify previously recorded tests.

6. Telerik Test Studio

- Telerik test studio is a comprehensive test automation solution.
- It is well suited for GUI, performance, and load.
- It allows you to test desktop, mobile and web applications.
- Its main features include Point-and-click test recorder, support for coding languages like C# and VB.NET, central object repository and continuous integration with source control.

7. Katalon Studio

- Katalon Studio is a powerful test automation solution for mobile, Web, and API testing. And it is completely FREE!
- It provides a comprehensive set of features for test automation, including recording actions, creating test cases, generating test scripts, executing tests, reporting results, and integrating with many other tools in the software development lifecycle.
- Katalon Studio runs on both Windows and MacOS, supporting automated testing of iOS and Android apps, web applications on all modern browsers, and API services.

Test Management Tools

Proprietary/Commercial Tools

- HP Quality Center/ALM
- QA Complete
- T-Plan Professional
- Automated Test Designer (ATD)
- Testuff
- SMARTS
- QAS.TCS (Test Case Studio)
- PractiTest
- Test Manager Adaptors
- SpiraTest
- TestLog
- ApTest Manager

Open Source Tools

- TET (Test Environment Toolkit)
- TETware
- Test Manager
- RTH

Functional Testing Tools

Open Source Tools

- Selenium
- Soapui
- Watir
- HTTP::Recorder
- WatiN
- Canoo WebTest
- Webcorder
- Solex
- Imprimatur
- SAMIE
- Swete
- ITP
- WET
- WebInject

Proprietary/Commercial Tools

- QuickTest Pro
- Rational Robot
- Sahi
- SoapTest
- Badboy
- Test Complete
- QA Wizard
- Netvantage Functional Tester
- PesterCat AppsWatch
- Squish
- actiWATE
- liSA
- vTest
- Internet Macros

Performance/Load Testing Tools

Proprietary/Commercial Tools

- WebLOAD Professional
- HP LoadRunner
- LoadStorm
- NeoLoad
- Loadtracer
- Forecast
- ANTS – Advanced .NET Testing System
- vPerformer
- Webserver Stress Tool
- preVue-ASCII

Open Source Tools

- ✓ Jmeter
- ✓ FunkLoad

THE END
THANK YOU!!!