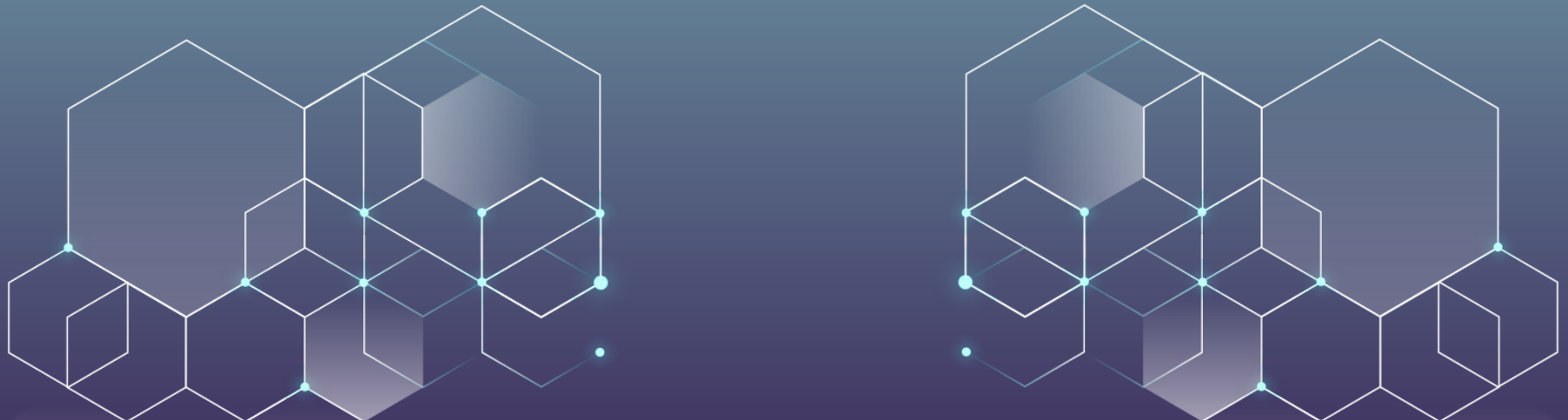
HOW PROGRAM TRACKING CONTRIBUTES TO IMPROVED QUALITY OF AST

Mary Joy M. Bonganay



Program Tracking

refers to the process of monitoring and analyzing various aspects of the testing process and the software under test.

Identifying Defects Early

Program tracking allows AST tools to monitor code changes and trigger relevant test cases automatically.

Test Prioritization

With program tracking, AST tools can prioritize test execution based on the impact of code changes.

Continuous Feedback Loop

Program tracking facilitates real-time feedback between development and testing activities.

Root Cause Analysis

In case of test failures, program tracking provides insights into the root causes of issues.

Adaptability to Change

Program tracking enables AST tools to adapt dynamically to changes in the development environment.

Incorporating Defect Prevention in AST Development

Reporter : Locsin, JB Rey S.

What is Defect Prevention?

Defect prevention involves taking proactive steps to minimize or eliminate software defects before they arise, rather than relying solely on detecting issues during testing or after deployment. In AST development, this becomes crucial because automated testing plays a central role in ensuring software quality.

Methods of Defect Prevention

1. Requirements Analysis and Validation

Initiate defect prevention by meticulously analyzing and validating requirements. Thorough understanding of software requirements and design is crucial. Inadequate comprehension by testers and developers escalates the risk of defects in later stages. Hence, meticulous requirement analysis is essential.

2. Review and Inspection

Reviews and inspections are crucial for early defect detection. They occur at various stages of defect prevention to address different needs. Self-review and peer-review are essential parts of software development, ensuring prompt defect identification and correction.

3. Defect Logging and Documentation

After analysis and review, maintaining defect records is crucial for understanding issues. Proper documentation aids in comprehending defects, enabling effective resolution. Logging and documenting defects ensures issues are addressed and prevented from progressing to later phases.

4. Root Cause Analysis

Root cause analysis is crucial for identifying the main cause of defects. By analyzing triggers, teams can prevent similar issues in the future. Understanding the root cause enables targeted preventive measures, reducing recurrence likelihood.

5. Static Code Analysis

Automate static code analysis to identify issues in the source code without running the program. These tools detect programming errors, standard violations, and other issues before compilation, enabling early defect detection and rectification.

6. Pair Programming

Implement pair programming, where two programmers share a workstation. One writes code while the other reviews each line in real-time. This fosters constant feedback and early mistake detection, reducing the chance of defects in later stages.

7. Test-Driven Development (TDD)

Adopt Test-Driven Development (TDD) practices, where automated tests are written prior to writing any code. This method helps identify errors early in the development process and ensures that the code complies with the requirements, thereby preventing defects from occurring.

8. Training and Skill Development

Invest in continuous training and skill development for team members. Skilled developers are better equipped to follow best practices and are less prone to making frequent mistakes, leading to improved defect prevention and overall software quality.

9. Checklists:

Utilize checklists to ensure that crucial actions are not overlooked during various stages of development. Developers and reviewers can quickly confirm that established standards are being followed, reducing the likelihood of defects due to missed steps or oversights.

CHARACTERISTICS OF GOOD AUTOMATED TESTING METRICS

DEZA, ARABELA GRACE A.

Relevance

Relevance refers to the degree to which something is connected or pertinent to a particular topic, situation, or context. It indicates how closely something aligns with the subject at hand or how much it contributes to understanding or solving a problem.

Relevance

- Alignment with business objectives
- Minimizing irrelevant results
- Aligning with project requirements

Actionability

Actionability refers to the quality of being actionable, meaning that something can be acted upon or used to take practical steps or make decisions. In other words, it assesses whether information, advice, or solutions can be implemented effectively to achieve desired outcomes.

Actionability

- Enabling incremental improvement
- Providing value to the business strategy
- Delivering actionable insights

Measurability

Measurability is the extent to which something can be measured or quantified. It involves determining whether an attribute, phenomenon, or outcome can be assessed or evaluated using specific criteria, metrics, or instruments to track its progress, performance, or impact.

Measurability

- Being objective and quantifiable

Interpretability

Interpretability refers to the ease with which something can be understood, explained, or interpreted. It involves assessing the clarity, coherence, and comprehensibility of information, data, models, or findings to enable meaningful analysis, communication, or decision-making.

Interpretability

- Ensuring comprehensive coverage

RECOMMEND, AND GIVE REASONS FOR YOUR SELECTION, A SET OF AST METRICS

CUSTODIO, MELVIN M.

What Is Automated Software Testing Metrics?

Automated Software Testing (AST) metrics are quantitative measures used to evaluate and assess the effectiveness, efficiency, and quality of automated testing processes within software development. These metrics provide insights into various aspects of automated testing, allowing teams to track progress, identify areas for improvement, and make data-driven decisions to enhance their testing practices.

Recommended AST Metrics

1. Defect Density:

- **Reason for Selection**: Defect density provides insight into the quality of the codebase by quantifying the number of defects relative to the size of the module or release. It helps in assessing the readiness of the software for release

$$\text{Defect Density} = \frac{\text{Defect Count}}{\text{Size of the Release/Module}}$$

1. **Identify Defect Count:** The total number of defects discovered in the "Customer Management System" module is 100.
2. **Determine Size of the Module:** The size of the module is measured in lines of code, which is 50,000 (50 KLOC).
3. **Apply Formula:** Plug the values into the formula to calculate Defect Density.

$$\text{Defect Density} = \frac{100}{50{,}000}$$

4. **Calculate:**

$$\text{Defect Density} = 0.002 \text{ defects per line of code}$$

Recommended AST Metrics

2. Defect Leakage:

- **Reason for Selection**: Defect leakage measures the effectiveness of the testing process before user acceptance testing (UAT). It highlights the number of defects missed by the testing team, indicating potential gaps in testing coverage

$$\text{Defect Leakage} = \left(\frac{\text{Total Number of Defects Found in UAT}}{\text{Total Number of Defects Found Before UAT}}\right) \times 100$$

Using the Formula:

$$\text{Defect Leakage} = \left(\frac{10}{50}\right) \times 100$$

Calculation:

$$\text{Defect Leakage} = \left(\frac{10}{50}\right) \times 100 = 20\%$$

Recommended AST Metrics

3. Defect Removal Efficiency (DRE):

- **Reason for Selection**: DRE evaluates the effectiveness of defect removal methods during the testing phases. It helps in understanding how efficiently defects are being identified and addressed before software release.

$$\text{DRE} = \frac{\text{Number of Defects Resolved by the Development Team}}{\text{Total Number of Defects at the Moment of Measurement}}$$

Using the Formula:

$$\text{DRE} = \frac{80}{100}$$

Calculation:

$$\text{DRE} = \frac{80}{100} = 0.8$$

Recommended AST Metrics

4. Defect Category:

- **Reason for Selection**: Defect category metrics provide a breakdown of defects based on various quality criteria such as usability, performance, functionality, etc. This breakdown helps in prioritizing quality improvements and understanding the distribution of defects.

$$\text{Defect Category} = \frac{\text{Defects Belonging to a Specific Category}}{\text{Total Number of Defects}}$$

For Usability Category:

$$\text{Defect Category (Usability)} = \frac{20}{100} = 0.20$$

For Functionality Category:

$$\text{Defect Category (Functionality)} = \frac{30}{100} = 0.30$$

For Performance Category:

$$\text{Defect Category (Performance)} = \frac{10}{100} = 0.10$$

Recommended AST Metrics

5. Defect Severity Index (DSI):

- **Reason for Selection**: DSI assesses the impact of defects on the software's quality and efficiency. It provides a weighted measure of defect severity, guiding prioritization efforts for defect resolution.

$$\text{DSI} = \frac{\sum(\text{Defect} \times \text{Severity Level})}{\text{Total Number of Defects}}$$

$$\text{DSI} = \frac{(10\times1)+(20\times2)+(30\times3)+(25\times4)+(15\times5)}{100}$$

$$\text{DSI} = \frac{(10)+(40)+(90)+(100)+(75)}{100}$$

$$\text{DSI} = \frac{315}{100}$$

$$\text{DSI} = 3.15$$

Recommended AST Metrics

6. Review Efficiency:

- **Reason for Selection**: Review efficiency helps in reducing pre-delivery defects by identifying and resolving errors during the review process. It validates the effectiveness of review activities and their contribution to overall defect prevention.

$$\text{RE} = \frac{\text{Total Number of Review Defects}}{\text{Total Number of Review Defects} + \text{Total Number of Testing Defects}} \times 100$$

$$\text{RE} = \frac{50}{50+100} \times 100$$

$$\text{RE} = \frac{50}{150} \times 100$$

$$\text{RE} = \frac{50}{150} \times 100$$

$$\text{RE} = 0.3333 \times 100$$

$$\text{RE} = 33.33\%$$

Recommended AST Metrics

7. Test Case Effectiveness:

- **Reason for Selection**: Test case effectiveness measures the efficiency of test cases in detecting defects. It helps in evaluating the quality of test cases and their ability to uncover potential issues.

$$\text{Test Case Effectiveness} = \left(\frac{\text{Number of Defects Detected}}{\text{Number of Test Cases Run}}\right) \times 100$$

$$\text{Test Case Effectiveness} = \left(\frac{20}{100}\right) \times 100$$

$$\text{Test Case Effectiveness} = 0.2 \times 100$$

$$\text{Test Case Effectiveness} = 20\%$$

Recommended AST Metrics

8. Test Case Productivity:

- **Reason for Selection**: Test case productivity measures the efficiency of test case preparation efforts. It provides insights into the productivity of the testing team and can aid in resource planning for future testing activities.

$$\text{Test Case Productivity} = \left(\frac{\text{Number of Test Cases}}{\text{Effort Spent for Test Case Preparation}}\right)$$

$$\text{Test Case Productivity} = \left(\frac{200}{100}\right)$$

$$\text{Test Case Productivity} = 2$$

Recommended AST Metrics

9. Test Coverage:

- **Reason for Selection**: Test coverage assesses the completeness of testing activities by quantifying the extent to which the software's functionality is covered. It helps in determining the thoroughness of testing and identifying areas that require ac

$$\text{Test Coverage} = \frac{\text{Number of Detected Faults}}{\text{Number of Predicted Defects}}$$

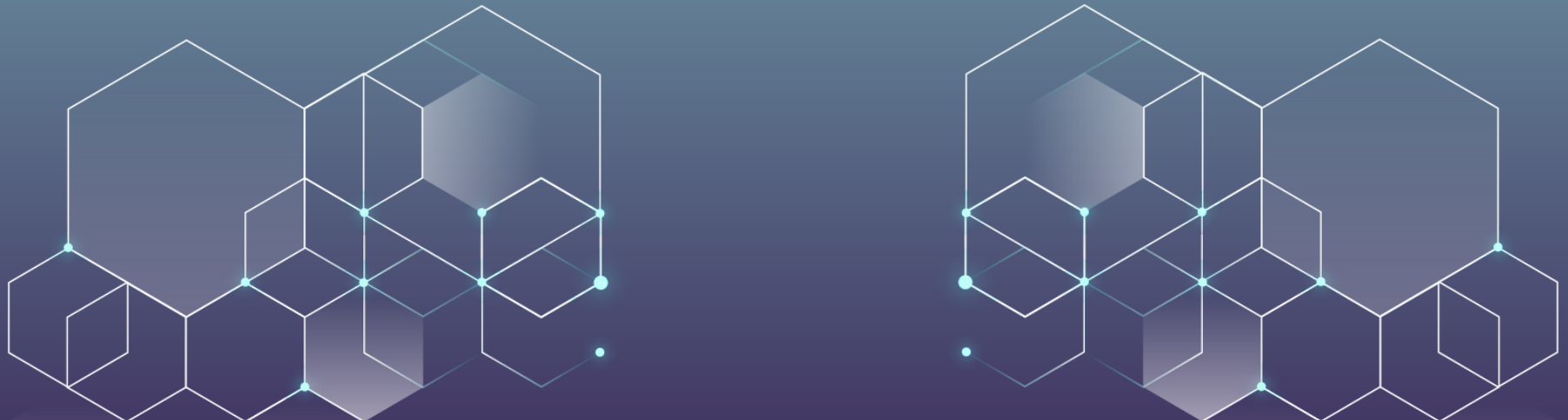
$$\text{Test Coverage} = \frac{50}{100} = 0.5$$

$$\text{Requirement Coverage} = \left(\frac{\text{Number of Requirements Covered}}{\text{Total Number of Requirements}}\right) \times 100$$

$$\text{Requirement Coverage} = \left(\frac{60}{80}\right) \times 100 = 75\%$$

USEFULNESS OF AST METRICS IN UNDERSTANDING THE VALUE OF AUTOMATION

Roger Malabanan



Test Coverage

AST metrics help measure the extent to which automated tests cover different parts of the software under test.

Defect Detection

AST metrics allow teams to track the effectiveness of automated tests in detecting defects.

Test Execution Time

AST metrics provide insights into the efficiency of automated test execution.

Resource Utilization

AST metrics help teams evaluate the resource utilization of their automated testing infrastructure.

Regression Testing

AST metrics assist in measuring the effectiveness of automated regression testing.

ROLE OF ROOT CAUSE ANALYSIS IN AST DEVELOPMENT AND EXECUTION

Roger Malabanan

Issue Resolution

RCA in AST helps pinpoint the underlying causes of test failures, anomalies, or performance bottlenecks.

Optimizing Test Efficiency

RCA allows teams to analyze the efficiency of their automated testing processes and identify areas for optimization.

Improving Test Reliability

RCA plays a crucial role in enhancing the reliability of automated tests and the accuracy of test results.

THANK YOU!