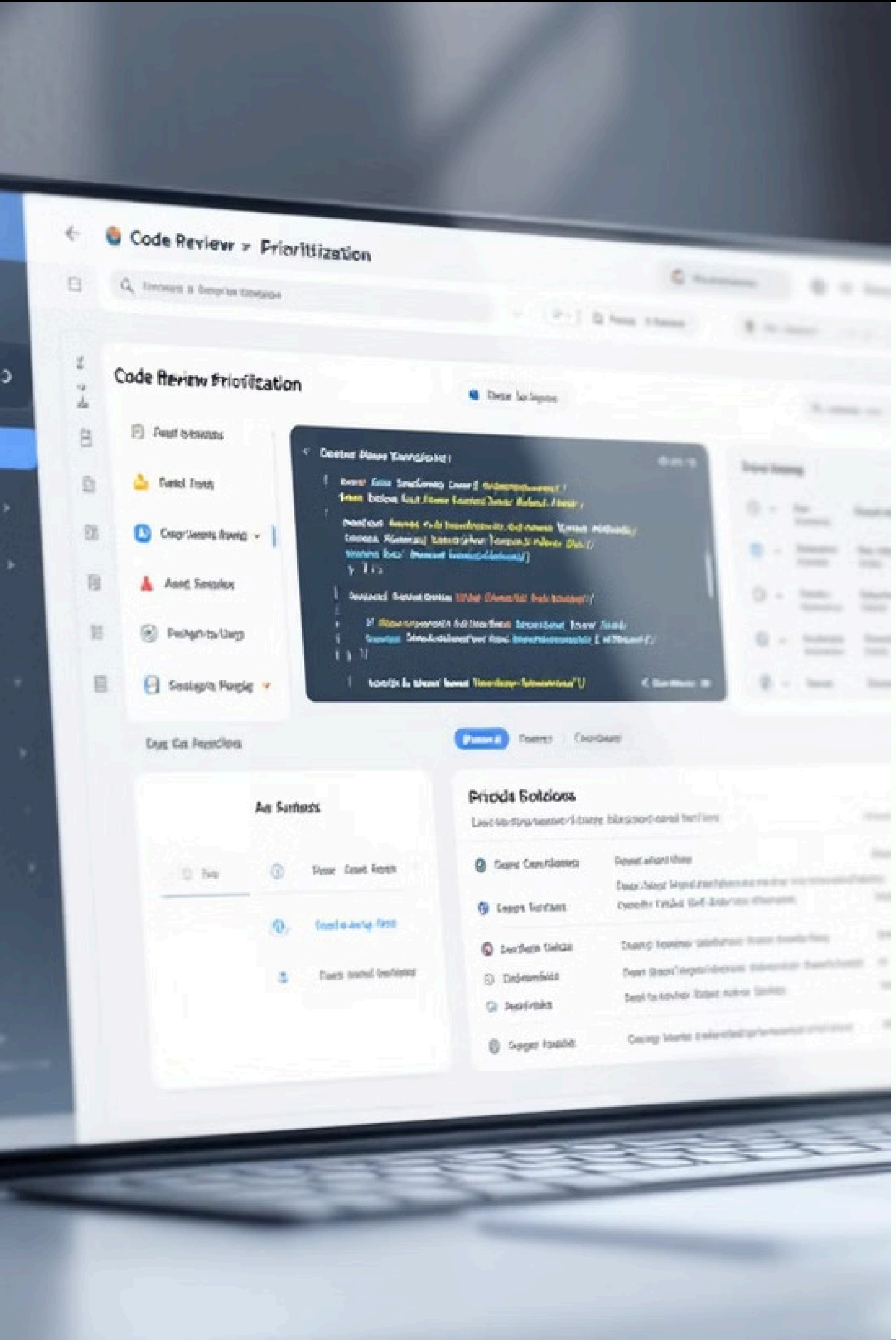


AASTU

Key Software Metrics

College of Engineering

Department of Software Engineering

Mahlet Hailu	ETS0794/13
Naod Ararsa	ETS0956/13
Naod Zekeria	ETS0961/13
Naol Yadete	ETS0968/13
Natnael Tafesse	ETS1026/13
Nebyat Bekele	ETS1052/13

Introduction

In modern software development, quality metrics are essential tools. They offer quantifiable data to evaluate various aspects of software quality, including reliability, performance, usability, and maintainability. This presentation explores key software metrics and their significance in building high-quality software.

Defect Density

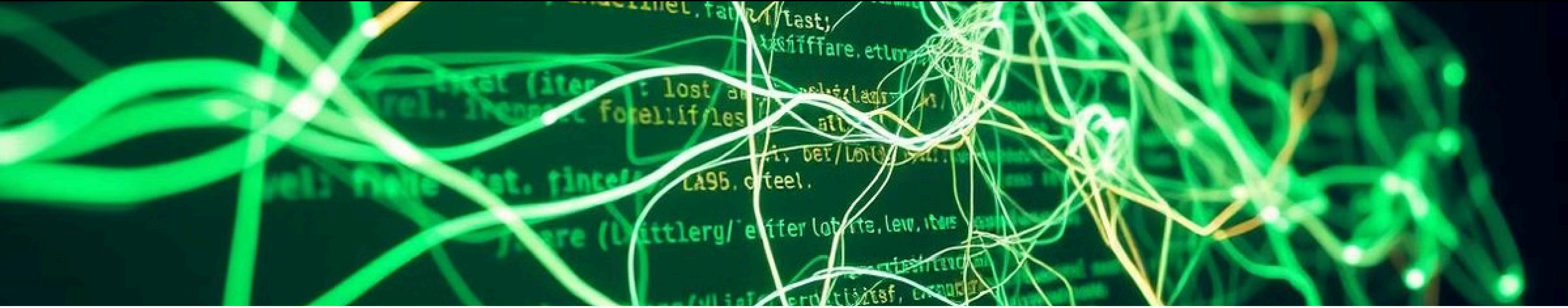
What is Defect Density?

Defect Density (DD) measures defects relative to software size.

Expressed as defects per thousand lines of code (KLOC).

Influencing Factors

Project size and number of developers are key factors.



Code Complexity

Intricacy Measure

Indicates difficulty in understanding and modifying code.

Quantifying Metrics

Cyclomatic Complexity (CC) and Lines of Code (LOC).

Negative Impacts

Increases defects and reduces software reliability.

Code Churn

Definition

Measures code added, modified, and deleted over time.

Assessment

Assesses code stability and developer productivity.

High Churn

Indicates instability or requirement issues.

Test Coverage

Thoroughness

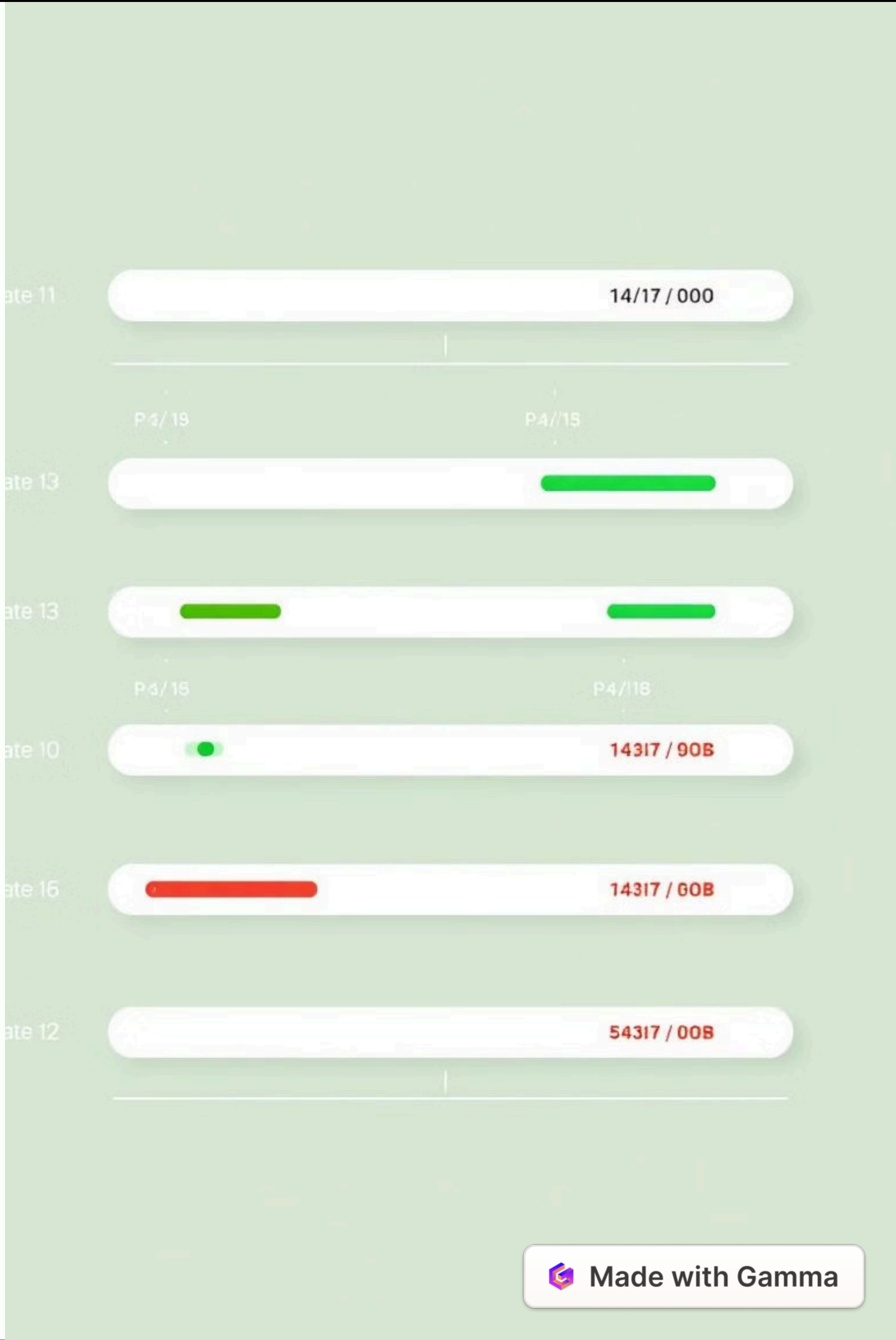
Determines how thoroughly software has been tested.

Extent

Expressed as a percentage of code executed by tests.

Effectiveness

Indicates completeness of the testing process.



Mean Time to Failure (MTTF)

1

Average Time

Measures average time an asset operates before failure.

2

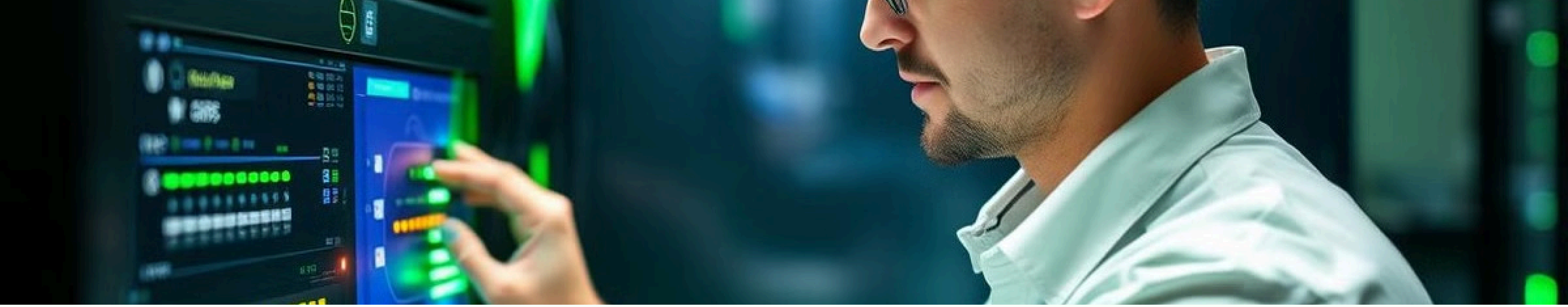
Reliability

Higher MTTF signifies a more reliable system.

3

Calculation

Operating time divided by number of assets.



Mean Time to Repair (MTTR)

1 **Average Repair Time**

Measures average time to repair a system after failure.

2 **Availability**

Used with MTBF to measure asset availability.

3 **Calculation**

Total maintenance time divided by number of repairs.

Performance Metrics

1

Efficiency

Assess the efficiency of a system.

2

Speed

Assess the speed of a system.

3

Reliability

Assess the reliability of a system.

Customer-Reported Incidents

1 **Incident Tracking**

2 **Root Cause Analysis**

3 **Response Time**

Effectively managing incidents means staying on top of important practices. Ignoring incidents can damage a product's reputation. Prompt responses help maintain user confidence.

Key Takeaways

Multifaceted Quality

Software quality includes reliability, performance, usability, and maintainability.

Insightful Metrics

Defect Density, Code Complexity, and Code Churn indicate code quality and stability.

Reliability and Availability

Test Coverage and MTTF/MTTR are crucial for assessing software reliability.

Effective Incident Management

Managing incidents maintains product reputation and user confidence.

Conclusion: Improving Software Quality

Monitor Key Metrics

Track defect density, code complexity, and code churn.

Optimize Workflows

Identify areas for enhancement in development cycles.

Ensure Customer Satisfaction

Track incidents and response times for reliability.

Data-driven decisions lead to higher quality software and efficient development.