International Journal of Multidisciplinary Research and Growth Evaluation.

Visualizing Test Failure Trends in Agile Qa Cycles Using Custom Reports from Jenkins + Maven Build Logs

Udayan Verma
Denver, USA

* Corresponding Author: **Udayan Verma**

Article Info

ISSN (Online): 2582-7138
Impact Factor (RSIF): 8.04
Volume: 06
Issue: 06
Nov-Dec 2025
Received: 08-10-2025
Accepted: 15-11-2025
Published: 22-12-2025
Page No: 1330-1333

Abstract
The paragraphs of this paper present the automated trend visualization of test failures in the Agile QA cycles that are founded on the execution data that is received in Jenkins pipelines and the Maven build logs. The old test automation systems usually show in a simple pass/fail format that does not reveal much information on how to make decisions in a rapid release cycle. The suggested structure is attentive to the visualization of the trends of defect clusters, flaky tests as well as long-term stability of builds. Dashboards are developed with the help of an extraction of structured information in Maven Surefire and Failsafe reports and Jenkins pipeline logs to describe the time-to-fix rates, environment-dependent anomalies and regression stability. These dashboards provide actionable information to enable QA teams to concentrate on automation maintenance and make sure that problem areas recurrent in the quality are dealt with and quality metrics are more reported to stakeholders. The approach is contrasted to delivering a report on the implementation of traditional tests as it is more oriented toward visualizing long-term trends, rather than brief snapshots. The result is more active quality assurance process which presupposes the theme of Agile of traditional checking and improvement.

DOI: https://doi.org/10.54660/.IJMRGE.2025.6.6.1330-1333

Keywords: Agile QA, Jenkins Pipeline, Maven Surefire, Test Trend Visualization, Flaky Tests

1. Introduction

The agile delivery models concentrate on agile releases and cycles, where the quality assurance must adapt itself to the shorter feedback loop and evolving architectures. In this case, failures and passages are not the only criteria that determine whether the testing is effective or not, but what can be learnt by the repetitive pattern and the failure tendencies. Interdependency between numerous services in large systems deployed in large enterprises such as Charter Communications should be automated as well as intelligent monitored.

Reporting on traditional test automation, useful as it is to the immediate detection of defects, merely lacks the ability to provide a longitudinal perspective of system wellness. They are barely able to document facts such as groups of chronic failures, environmental-specific abnormalities or intermittent tests that may change over time. Without these insights, the QA teams may rectify individual issues in an ad hoc fashion but not rectify systemic infirmities.

In order to overcome these weaknesses, a special visualization framework of test failures was developed. The framework is applied with Surefire and Failsafe reports, Jenkins pipelines to coordinate and Maven to manage dependencies. Such data sets lead to dashboards that report repeated areas of failures, time-to-fix, regression impacts. The result is the transition to active, data-driven quality assurance and QA is one of the Agile objectives of continuous improvement

2. Background and Rationale

Software testing has also become extremely complex, due to several microservice architectures and CI/CD pipelines. Nightly and even hourly test executions are needed in continuous deployments (which produce large amounts of execution information [1] to provide continuous coverage. Although these runs may be successfully automated by using Jenkins pipeline and Maven

build tools, the reporting is highly skewed towards pass / fail metrics.

There are two weaknesses of this strategy. First, it conceals those trends which had been failing as modules had not been solved in terms of dependencies or due to environmental factors. Second, it does not pay attention to the flaky tests which fail not all the time during the testing run and thus, generate noise that may confuse the developers and the QA engineers. Lastly, these issues destroy the confidence of automation, and devalue it over time.

It is in the light of these difficulties that visualization becomes necessary as a supplement of automation. Visual dashboards are used to notify the teams of recurrent anomalies, identify test suites that are unstable, and measure the impact of fixes. Organizations will also be enabled to make better decisions, have confidence in their QA and focus on remediation through the translation of logs into actionable insights. Thus, trend visualization, rather than enhances, is a requirement in Agile environments.

3. System/Framework Designs

The JenkinsMaven automation stack is easily compatible with the proposed system. It is primarily aimed at capturing, processing, as well as visualising the test execution data to determine any trends beyond binary outcomes. It is planned to be a modular system that has four significant components:

Information Retrieval - Jenkins pipelines provide the logs of the execution, and Maven Surefire and Failsafe provide the results of the unit, integration and regression tests in an organized format. They are either XML-exported or HTML-exported reports on which the dataset is based.

Information Processing- Raw logs are normally noisy because of irrelevant errors in the environment or not followed tests. This information is then analyzed with scripts that determine anomalies that are not real faults in the system. The result is a table of test cases versus modules, environments and times.

Visualization Layer - Custom dashboards, created based on open-source reporting applications (e.g., Grafana, Allure or an in-house Python script with Matplotlib) are used to display patterns of failures in tests [2]. These are time-series graphs of recurrent failures, pie charts of environment-specific defects, and heat maps of flaky test discoveries.

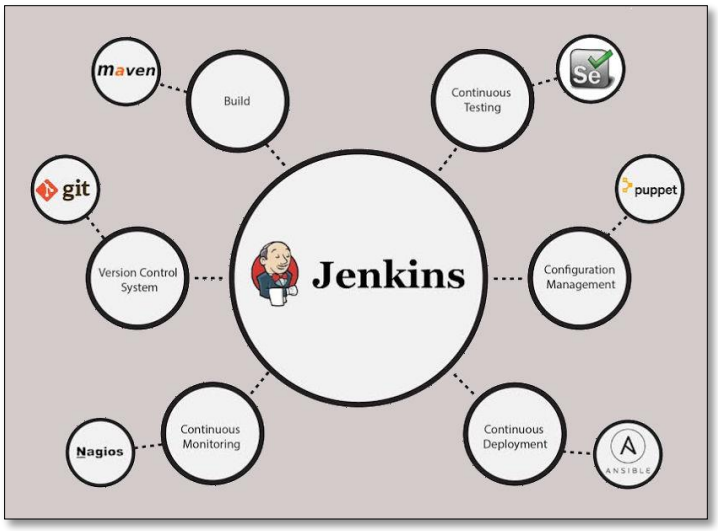

Fig1: How Continuous Integration works with Jenkins and Test Automation [7]

Insight Delivery - The dash boards are connected with Jenkins giving real time notification. Slack or email alerts identify anomalies, whereas trend reports provide an overview of the status to stakeholders during sprint reviews.

This modular design is scaled and layered. All of the components can be upgraded separately and, in the future, can be integrated with a cloud based testing platform or AI based failure prediction engine. With these dashboards integrated with the Jenkins environment, the QA teams will have instant access to the systemic issues and the feedback loop between detecting defects and resolving them will become quicker.

Table 1: Framework Methodology for Test Failure Trend Visualization

Step	Description
Information Retrieval	Collect Jenkins logs and Maven Surefire/Failsafe reports.
Information Processing	Clean logs, filter irrelevant errors, normalize datasets.
Visualization	Create dashboards (trends, clusters, flaky test detection).
Insight Delivery	Share reports/alerts with QA teams and stakeholders.

4. Implementation & Methodology

It is implemented using the Agile practice conforming methodology:

Collection of Data- Jenkins pipelines Auto logs will be collected at the conclusion of each build. Reports will be produced by the maven Surefire and Failsafe and will be stored in the form of artifacts. The module identifiers, the failure counts and the timestamps are located with custom parsers.

Data Transformation These artifacts are converted to CSV files or database tables using a preprocessing script. These include noise like infrastructure outage or non-followed through tests [3]. The flaky tests are characterised by a successive sequence of testing.

The visualization Construction Dashboards are built with the help of Allure or Grafana. Key visualizations include:
Trend over time (line graphs).
Flaws per module (bar charts).
Environment anomalies (cluster diagrams).
Heat maps.
Insight Delivery - Jenkins tasks are automatically monitored with reports and reported to stakeholders in Slack or email. This will ensure that the technical and non-technical departments are made aware of the trend of failures.

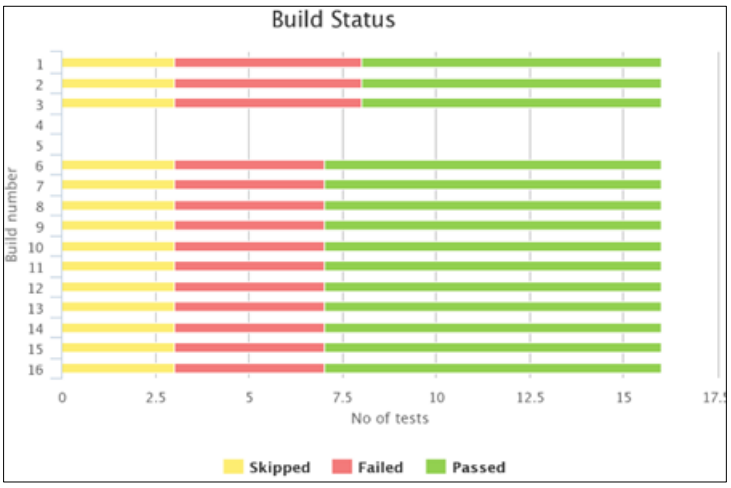

Fig 2: Bar Chart [8]

Maintenance - Modular scripts are used to make the framework flexible. With the change of services, the dataset and dashboards do not have to be adjusted to interfere with the Jenkins pipeline.
This approach transforms the test reporting to non-static pass/fail data snapshots to the production of continuous insight [4]. It improves the co-operation since it provides the developers, QA engineers and product owners with a common, graphic perspective of the quality of tests.

5. Reporting Resuls & Insights

Agile QA practices have produced actual outcomes by substitution of the traditional reports with dynamic visualization. Failure mode dashboards can assist teams to isolate problem areas occurring within modules and settings. An example is where a given microservice has history of being test unstable post-deployment, which is a candidate of prioritized refactoring. Flaky tests that traditional pass/fail output has not been preoccupied with are also reported by dashboards [5]. These tests can be tagged and isolated to enable the utilization of QA teams to enhance reliability and remediation can concentrate on significant defects. This allows a degree of trust in the automated test results which minimizes false alarms and wastes a significant amount of work. Real-time notification is also supported with Agility using Jenkins plug-in and inbuilt messaging systems. End-of -sprint-reviews are no longer available but the stakeholders can still find out the systemic problems as they arise; this has enabled the stakeholders to make decisions more quickly [6]. Also, the visual summaries make it possible to show the quality trends to the product managers and executives who might not be technical experts. The metrics such as the time-to-fix rates and regression-stability scores provide a high level of confidence that the system is ready to release. In short, the documented and graphicalized plan has made QA a proactive facilitator of Agile delivery rather than the responsive process.

6. Conclusion

Agile testing is not only automated testing, but actionable information to make decisions. The old forms of reporting that can only provide binary outputs are incapable of capturing the systemic trends that determine the long-term stability. The suggested framework will turn the raw data on execution into valuable dashboards with groups of failures, flaky tests, and regression patterns with Jenkins pipelines and Maven reports. The introduction of visualization has greatly enhanced communication between the developers, QAs teams and stakeholders. The teams are not considering failures at individual cases but are ready to experience failures again and, thus, the fixes are prioritized. This is free to QA and Agile ideas of incessant tracking and further improvement. It can be enhanced with alkyl-based forecasting algorithms that predict possible failure in the future and the prioritization of test suites should be enhanced. Distributed environments can also be increased with cloud-native scaling. Finally, pass/fail vs. visual trend analysis is one of the changes that are indispensable in the modern QA practices.

Reference

1. Brandt CE, Zaidman A, Amann S. LogChunks: A Dataset for Build Log Analysis. In: Proceedings of the 17th International Conference on Mining Software Repositories (MSR 2020). IEEE; 2020. p. 543–7.
2. De Paoli AJ. Automating Log Analysis and Management in Continuous Integration for Improved Efficiency [Master's thesis]. Espoo: Aalto University; 2024.
3. Fawzy AH, Abdelkader T, Ali A. A Framework for Automatic Detection of Anomalies in DevOps. J King Saud Univ Comput Inf Sci. 2023;35(6):101492.
4. Kolawole I, Fakokunde A. Improving Software Development with Continuous Integration and Deployment for Agile DevOps in Engineering Practices. Int J Comput Appl Technol Res. 2024;14(01):25–39.

5. Parry O, McMinn P, Harman M, Clark JA. A Survey of Flaky Tests. ACM Comput Surv. 2021;54(9):1–36. (Note: Some sources list this under ACM Trans Softw Eng Methodol with slight year/page variations; the provided details are retained.)
6. Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. A Survey of DevOps Concepts and Challenges. ACM Comput Surv. 2019;52(6):1–35.
7. How Continuous Integration works with Jenkins and Test Automation [Internet]. Novature Tech; [cited 2026 Apr 1]. Available from: https://novaturetech.com/how-continuous-integration-works-with-jenkins-and-test-automation/

How to Cite This Article
Udayan V. Visualizing test failure trends in agile QA cycles using custom reports from Jenkins + Maven build logs. Int J Multidiscip Res Growth Eval. 2025 Nov-Dec;6(6):1330–1333. doi:10.54660/.IJMRGE.2025.6.6.1330-1333.