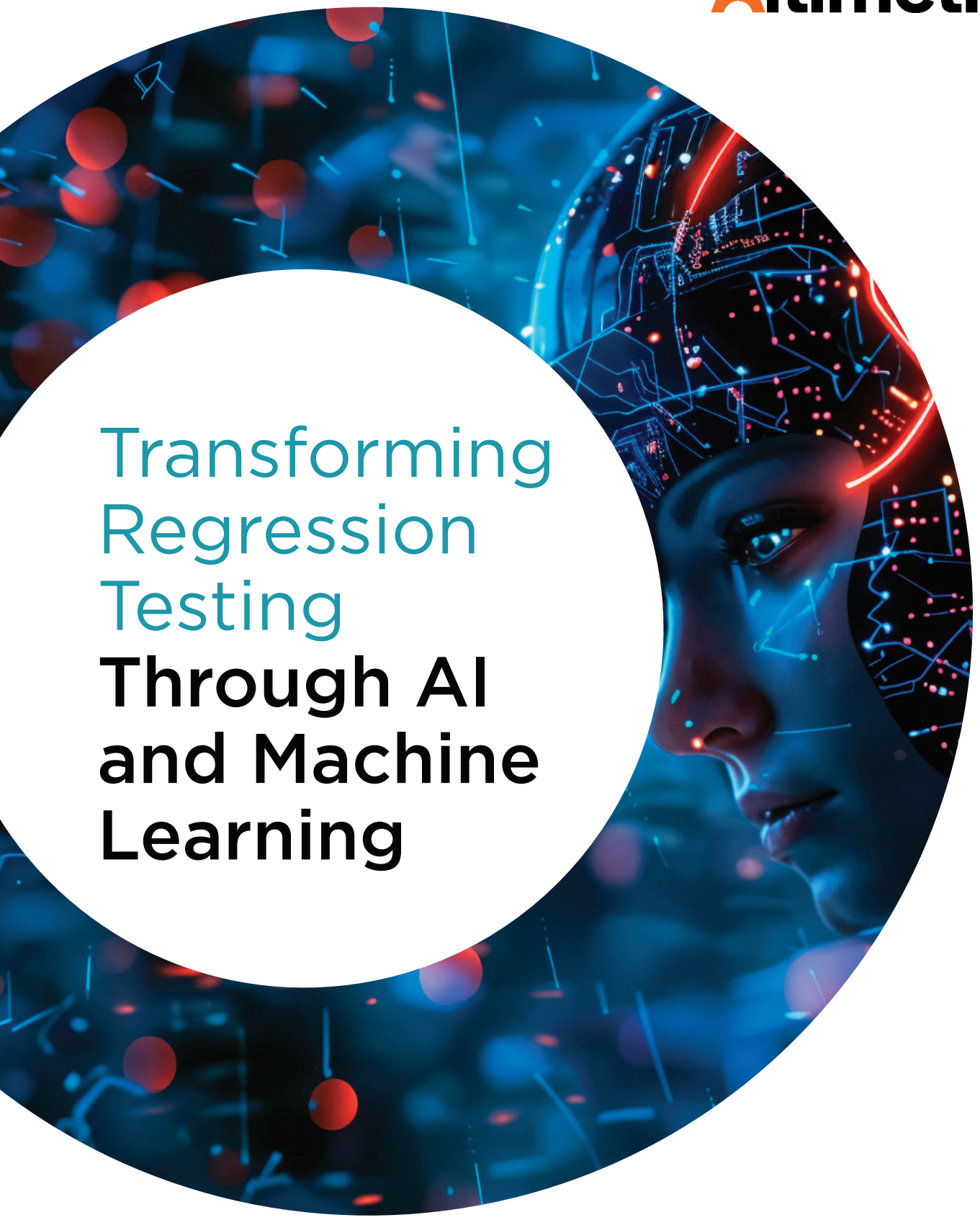
Altimetrik



Transforming Regression Testing Through AI and Machine Learning

Author/Architect

Rudra Thakur, Sr. Principal Architect
Kartik Pattar, Staff Engineer

White Paper

Index

Introduction

Regression testing forms the backbone of quality assurance in software development, encompassing a diverse range of tests accumulated over the lifecycle of a product. As software evolves, so does the challenge of maintaining an effective and efficient regression test suite. The advent of agile methodologies and continuous delivery models further compounds the complexity, necessitating a sophisticated approach to managing and executing these tests.

The Challenge of Scale in Regression Testing

The sheer volume of regression test cases, often reaching into the thousands, poses a significant challenge for quality assurance teams. Products in domains such as insurance and banking, where product lifecycles extend over years and incremental releases are frequent, experience an ever-growing regression test suite.

In such environments, test optimization i.e. identifying which tests to run based on code churn in the product and test prioritization are leveraged. Techniques like risk-based testing provide initial frameworks for test prioritization. However, these methods tend to depend excessively on the judgment of subject matter experts, which may inadvertently neglect vital scenarios that resonate with the end user's perspective. Despite their utility, traditional methods may not fully cater to the complex demands of current software development practices.

Available solutions

To tackle this issue, we can utilize risk-based prioritization and regular reviews of the regression suite. The primary focus is at Test to code mapping to understand the impact.

Approaches taken by other market tools

Approach 1: Uses JaCoCo for the execution tracker in which they have a wrapper class which will do the profiling and tracking of the execution. The JaCoCo gives the code coverage with respect to automation test cases. This process involves deploying the agents which will track the execution of the code and the test cases being tested, thereby making the T2C mapping.

Approach 2: It offers features such as code profiling and test-to-code mapping Test-to-code mapping is a crucial feature of tool that connects specific tests to the code they cover. This enables more targeted testing and ensures that all relevant parts of the code are tested adequately.

Both the approaches come with some limitations.

Let's understand this through an example:

Imagine a Software product company, that has developed a comprehensive platform for managing insurance policies, claims, and customer interactions. Over time, the platform has grown in complexity, resulting in an ever-increasing regression test suite.

Year	Version	New Features	Regression Tests	Releases per Year	Regression Cycle Time (Days)
2016	1.0	10	100	4	1.4
2017	2.0	30	500	4	7
2020	3.5	50	2000	6	28
2022	5.0	70	5000	6	70
2024 (H1)	8.0	40	7000	8	112

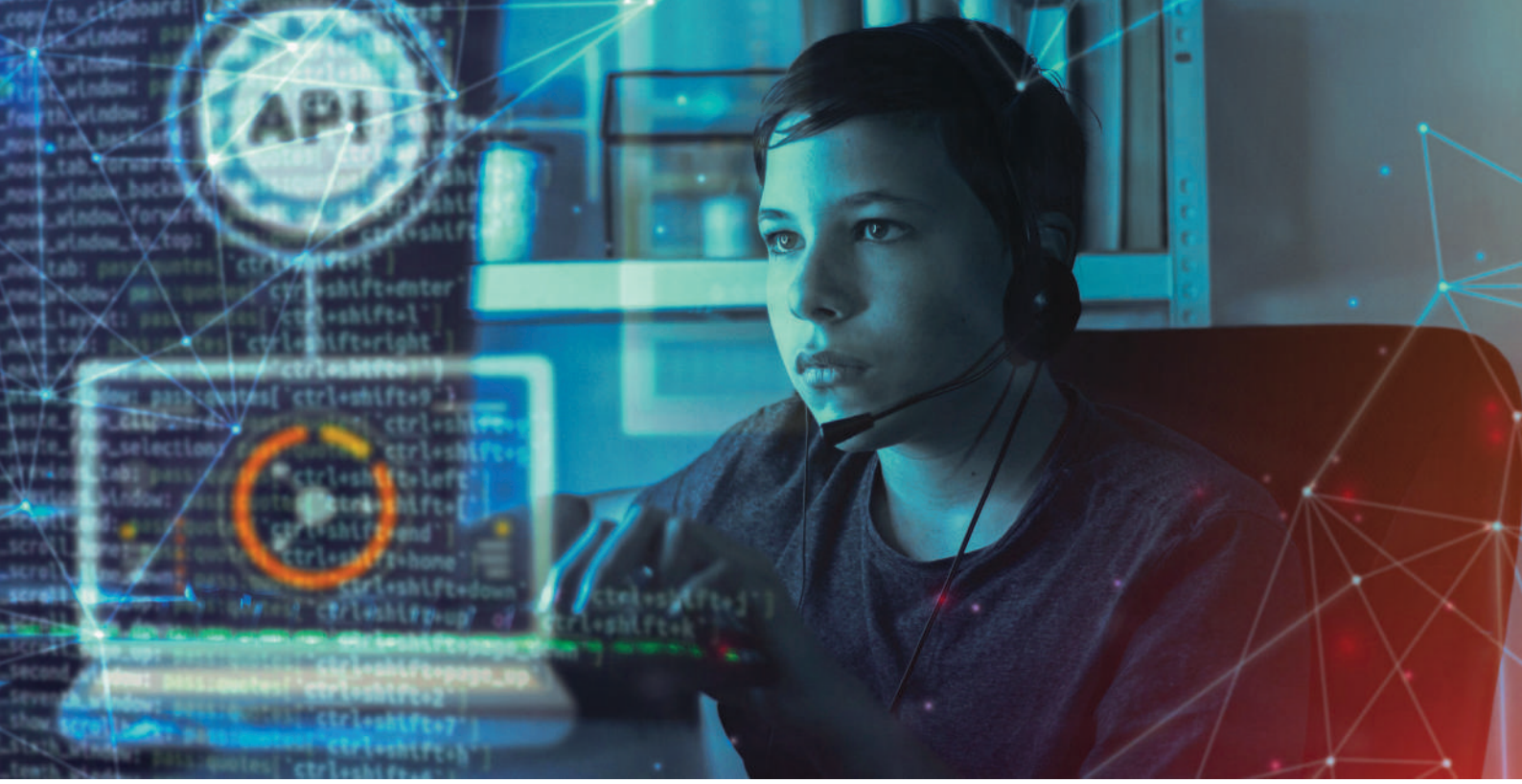
Regression Test Suite Growth

- Initial release with 100 regression tests was manageable
- By Version 2.0, the suite grew to 500 regression tests
- At Version 3.5, it expanded to 2000 regression tests
- By Version 5.0, the suite had 5000 regression tests
- For Version 8.0, it reached 7000 regression tests, requiring substantial time and resources for each release.

Drawbacks of above approaches

- **Complex logic for agents:** With 7000 regression tests and a regression cycle time of 112 days, the added overhead from instrumentation can slow down test execution further, increasing resource consumption and extending release cycles so as the difficulty in maintaining precise impact analysis

- **Cost:** Incurs high costs over years, which can be prohibitive for organizations with large and frequently updated test suites.



Altimetrik' s Approach to **Regression Test Optimization**

To address the above challenges which is common for our enterprise customers, Altimetrik used two distinct methodologies aimed at refining regression testing processes. Our approach is to combine White Box (Code to Test) & Blackbox (User stories to Tests) by building an agent with less configuration and independent of code coverage tool usage along with leveraging historical test execution data and advanced (GenAI + AI) models, to predict, optimize, and prioritize test cases for the current release, enhancing efficiency without compromising on coverage.

White Box

This approach establishes a comprehensive mapping between the regression tests and the application currently in development. Utilizing this mapping, it pinpoints Testing Gaps and identifies risks associated with untested code changes. Moreover, in instances of code churn within the product, our analytical engine evaluates the alterations and recommends an optimized set of tests. This strategic selection aims to maximize the efficiency of defect detection within constrained timelines.

Black Box

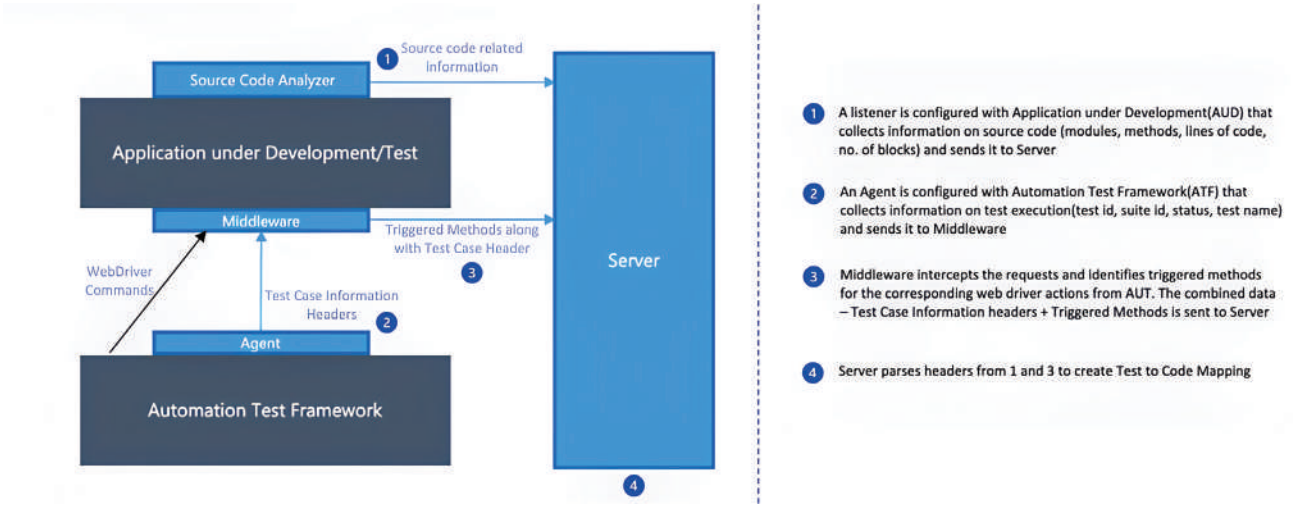
This perspective treats the software product as an opaque entity, focusing on requirements and their corresponding test cases. By leveraging historical data and advanced GenAI and AI models, we predict, optimize, and prioritize test cases for the current release, enhancing efficiency without compromising on coverage.

Let's discuss them in detail

White Box **Approach**

The White Box approach creates a detailed correlation between the regression tests and the ongoing application development. Through this correlation, it precisely identifies Testing Gaps, highlighting the risks linked to changes in the code that have not undergone testing.

Architecture



Workflow Explained

Step 1: Source Code Analysis

The initial step involves configuring the Application Under Development (AUD) to facilitate source code analysis. An algorithm is employed to extract detailed code information, including modules, applications, methods, classes, lines of code, and block counts. This extracted data is then relayed to the engine through a data dump API, laying the groundwork for subsequent analysis.

Step 2: Integration with Automation Test Framework (ATF)

The ATF is tasked with testing the AUD. Within this framework, listeners are implemented to tag each test with custom headers—such as test ID, suite ID, status, and test name. This information is captured by the ATF listeners and transmitted to engine Middleware for processing and insight extraction.

Step 3: Processing by Engine Middleware

Engine Middleware acts as an intermediary, capturing requests and pinpointing the methods in the application under development that correspond to actions initiated by the ATF. This Middleware amalgamates the test case information headers with the identified methods, forwarding the compiled data to the Engine for analysis and interpretation.

Step 4: Engine's Test to Code Mapping

In the final stage, the Engine leverages the collected Test Case Information headers and the processed data from the Middleware to generate a Test to Code Mapping. This crucial mapping forges a link between individual test cases and their respective code segments within the AUD.

The outcome is an enhanced framework that improves traceability, facilitates impactful analysis, and ensures a comprehensive assessment of test coverage.

Black Box Approach

The Black Box Approach treats the software product as a closed system, focused on matching requirements to their respective test cases. Utilizing historical data along with sophisticated Generative AI and AI models, this method forecasts, refines, and orders test cases for the imminent release. **The foundational premise of this approach is traceability, implying that requirements are systematically linked to the test cases developed for them.**

Story ID	Story Description	Points
ST-1	Create a Login Page	8
ST-2	Add Search Functionality	5
ST-3	Integrate Payment Gateway	8

Linked Testcases corresponding to created requirements

Test Case ID	Test Case Description	Priority
TC-1	Archived Users should get friendly access denied error	High
TC-2	Filter Options should be enabled	Medium
TC-3	Digital Wallet Payments should be allowed	Low

^Traceability between Requirements and Testcases

Designed to tackle the challenge of absence of traceability, the Engine leverages the White Box methodology to construct a traceability framework from scratch. This newly established traceability then supports the Black Box approach, enabling effective prediction, optimization, and prioritization of test cases. Through this symbiotic application of both methodologies, **the Engine ensures that products, irrespective of their initial state of traceability, can achieve comprehensive testing coverage.**

With this traceability in place, **additional parameters like Release, Build and Defect will also be tagged as metadata and will be collected and catalogued by Engine for analysis.**

Taking Product Release R1 as an illustrative example, the complete suite of test cases, denoted as T1, is subjected to a series of intermediate builds. The execution outcomes, whether successful or not, are diligently documented. This process generates a robust dataset capturing all stories within R1, along with the execution results of their corresponding test cases—forming a structured and traceable map.

The "**Story Details**" table meticulously records every story, enriched with metadata pertinent to Release R1. This is mirrored in the "**Test Case Details**" table, which archives exhaustive details of the test cases, ensuring a traceable link back to the respective stories.

Story ID	Story Description	Points
ST-1	Create a Login Page	8
ST-2	Add Search Functionality	5
ST-3	Integrate Payment Gateway	8

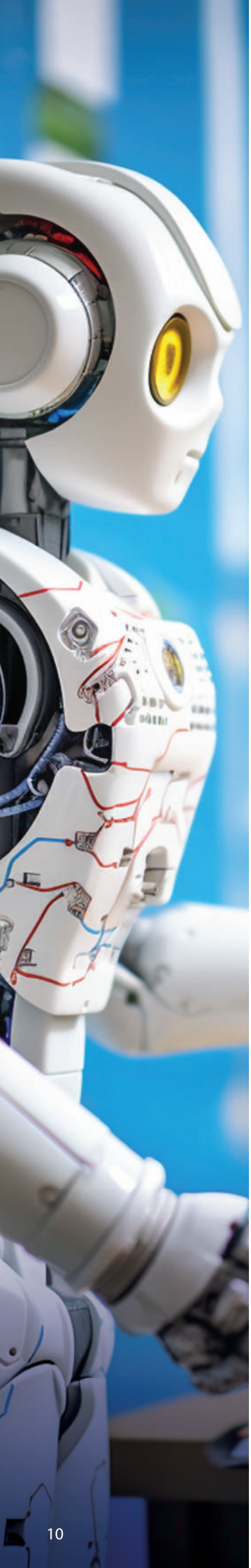
Linked Testcases corresponding to created requirements

Test Case ID	Test Case Description	Priority
TC-1	Archived Users should get friendly access denied error	High
TC-2	Filter Options should be enabled	Medium
TC-3	Digital Wallet Payments should be allowed	Low

Additional Metadata – Build, Test Set, Execution Result and Defect Details

Build 1				
Test Set 1				
Test Case ID	**Test Case Description**	**Priority**	**Result**	**Defects**
TC-1	Archived Users should get friendly access denied error	High	Pass	
TC-2	Filter Options should be enabled	Medium	Fail	PROD-3465
TC-3	Digital Wallet Payments should be allowed	Low	Fail	PROD-6762, PROD-987

^Additional Metadata of Build information,
Test Set, Execution Result, and Defect Details



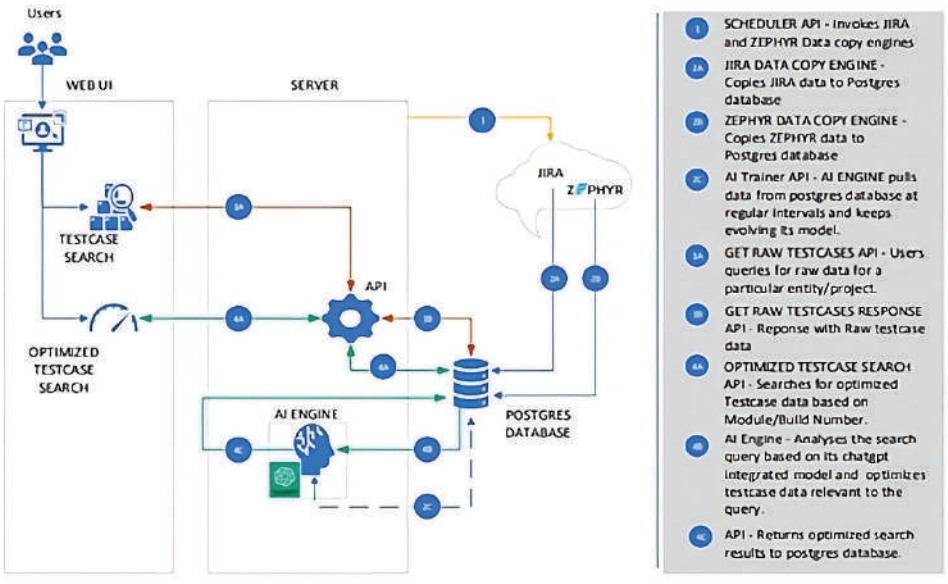
Consistent documentation is maintained for test sets across multiple builds (B1, B2, B3, etc.), within the scope of R1. This systematic approach is replicated for each subsequent product release, culminating in a historical archive that intertwines stories, test cases, and their outcomes along with defects across the product's lifecycle.

The large data repository becomes the keystone for the Model Engine, which initially harnesses this historical insight for training and continuously evolves with the infusion of data from each new release. In turn, this dynamic and ever-updating model feeds into the Gen AI Engine, which is Ready to use the historical dataset to predict the test cases required for impending releases, thereby optimizing future testing endeavours.

The architecture of Engine, tailored for the black box approach, delineates a flexible system compatible with a variety of test management and Agile tools.

At its core, Engine interfaces with a test cases repository—Zephyr being the primary example. However, Engine's design allows for seamless integration with any test management tool that provides REST API support, such as ALM, TestRail, or Xray.

Parallelly, Engine connects to a story repository to access the agile artifacts. While JIRA is the current repository in use, the architecture is agnostic to the choice of Agile tools, provided they support RESTful interactions. This allows for adaptability with other popular Agile tools.



The Engine is underscored by its robust training and data mapping protocols, as below:

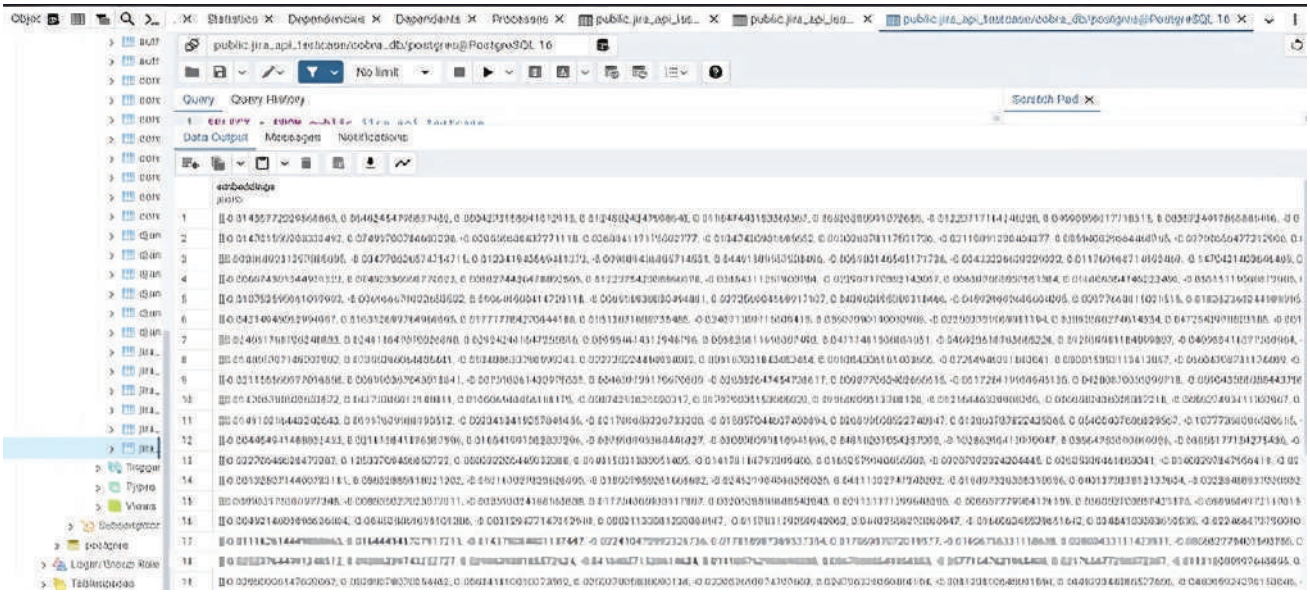
Trained Data

The engine's training regimen includes hundreds of story descriptions that have been processed through dual-model analytics.

One model is dedicated to keyword extraction, streamlining the search for relevant terms within the story text.

The other is tasked with generating embeddings, intricate numerical representations that capture the essence of a story's narrative, linked to specific epics.

Sample Embeddings
Output from the Framework



Code

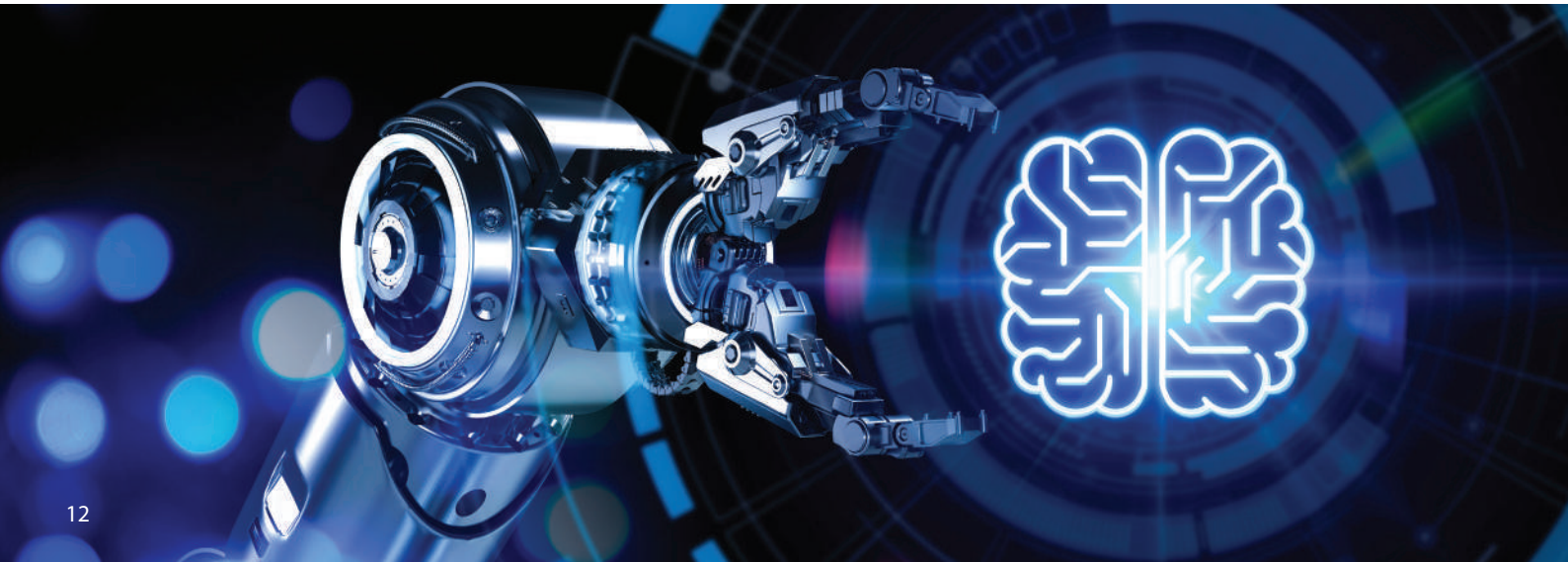
```python
def get_embeddings(texts):
    """Get embeddings for a list of texts."""
    logging.info("Getting the embeddings for texts.")
    try:
        # Convert a single string to a list
        if isinstance(texts, str):
            texts = [texts]
        # Use the model to encode the texts
        embeddings = model.encode(texts, show_progress_bar=True, normalize_embeddings=True)
        return embeddings
    except Exception as e:
        logging.error(f"Error in getting embeddings: {e}")
        return None

## Introducing custom cosine similarity
def cosine_similarity_vectorized(A, B):
    """Compute the cosine similarity between two 2D arrays."""
    dot_product = np.dot(A, B.T)
    norm_a = np.linalg.norm(A, axis=1)
    norm_b = np.linalg.norm(B, axis=1)
    return dot_product / np.outer(norm_a, norm_b)
```

The aggregate of this training data forms a comprehensive dataset encompassing story narratives, correlated keywords, generated embeddings, and their associated epics.

Mapped Data

Integral to the system's functionality, mapped data comprises a constellation of elements that add depth and context to the story descriptions. This data matrix includes but is not limited to, identifiers such as release_id and Story_id, the Build version, comprehensive story descriptors, and the associated testcase_id. These multifaceted data points converge to form a multidimensional view of each story and its journey through the development lifecycle.



Testcase Data

The Testcase Data subset is instrumental in cataloging the specifics of each test case. This includes a unique testcase_id, an elaborate testcase_desc that outlines the test's objectives, and the assigned priority level, which informs the test's execution precedence.

Upon the API's delivery of a new story description, Engine's integration with the GenAI model come into action, extracting pivotal keywords.

Concurrently, the text-embedding-ada-002 model embarks on creating new embeddings.

The synergy between these processes culminates in the calculation of cosine similarity scores, contrasting the new embeddings with the pre-established ones to determine the most relevant matches.

Predicated on these similarity metrics, the model then anticipates the test cases most likely to be impacted by the new story, streamlining the test selection process with unprecedented precision.

In conclusion, the daunting expanse of regression test suites is navigable through two distinct methodologies: the white box and black box approaches. By capitalizing on the wealth of historical test execution data and aligning it with the current state of the application under development, we forge a path toward streamlined test management. This data, when synthesized through Generative AI models, empowers us to forecast with greater accuracy, crafting a targeted and efficient regression testing strategy for future releases. Such predictive capabilities grant programs and test managers the acumen to judiciously distribute time and resources.

The result is not only a product of superior quality but also a significant reduction in infrastructure expenses by negating the need for repetitive, comprehensive regression tests.

Benefits

These are the benefits from the overall approach:

- **Accelerates cycle duration:** By reducing the time taken for each cycle, teams can achieve faster turnaround times, allowing for more iterations and quicker product development
- **Elevates overall quality:** Improved quality leads to enhanced customer satisfaction, reduced rework, and a stronger brand reputation, ultimately resulting in higher customer retention and loyalty
- **Expands test coverage:** With broader test coverage, teams can identify and address more potential issues, resulting in a more robust and reliable product that meets user expectations and requirements
- **Drives cost-effectiveness:** By optimizing resources and minimizing inefficiencies, organizations can achieve significant cost savings in development and maintenance, ultimately improving the bottom line
- **Empowers scalability:** Scalability enables organizations to easily adapt to changes in demand or growth, ensuring that the tool can accommodate increasing workload or user base without compromising performance or stability

References

Jackie Wiles, "Beyond ChatGPT: The Future of Generative AI for Enterprises," Gartner, January 26, 2023. GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used here with permission. All rights reserved.

About Altimetrik

Altimetrik is a pure-play data and digital engineering solutions company focused on delivering business outcomes with an agile, product-oriented approach. Our digital business methodology provides a blueprint to develop, scale, and launch new products to market faster. Our team of 5,000+ employees with software, data, cloud engineering skills help create a culture of innovation and agility that optimizes team performance, modernizes technology, and builds new business models. As a strategic partner and catalyst, Altimetrik quickly delivers results without disruption to the business.

Our unique Digital Business Methodology centers on business led ownership aligned to company goals. It is comprised of three pillars: experienced team of practitioners, an incremental approach, and an end-to-end self-service digital business platform. These combine to facilitate collaboration and agility between business and engineering teams to co-create products and solutions faster without disruption to the business. This is powered by a single source of truth and a culture of innovation that brings unlimited growth within reach.

We cater to companies of all sizes from Fortune 100 to digital disruptors and start-ups. We are a people-centric organization and talent is one of the central pillars of our business model and success. Employee engagement, diversity & inclusion, well-being and empowerment are central themes to our ethos. This project has been instrumental in taking our digital culture to the next level and brings our globally spread employee base on a unified platform.