

# Analytics solution for regression test optimization



# Abbreviations

DB	Database
SP	Stored procedure
AUT	Application under test
TPT	Test case prioritization technique
HFT	High functional test cases
LFT	Low functional test cases

## Introduction

Regression testing is pivotal in software testing cycles, ensuring that new development or fixes for existing ones do not disrupt the working functionality. In case the functionality changes, the number of test cases to run ensures that the code changes do not affect the other areas of the application. All existing and new test cases are required to be executed along with the new functionality test case set. However, executing every test case each time a change or addition is made in the Application Under Test (AUT) is impractical, as it will directly increase the time and effort required. Unstable test case identification and defects which are not mapped to test cases are very difficult to identify and retest. Introducing test case prioritization and defect analysis techniques can mitigate this challenge by ensuring that maximum faults are uncovered by the highly prioritized test cases.

The primary objective of this project is to optimize the regression testing process by strategically managing the test case suite. The optimization aims to:

- **Reduce test execution time:** Minimize the time required to perform regression testing by prioritizing high-impact test cases.
- **Maximize test coverage:** Ensure that the selected test cases provided encompass the critical functionalities and integration points of the software.
- **Resource efficiency:** Efficiently utilize available testing resources (e.g., time, manpower and hardware) without compromising the testing quality.
- **Early detection of defects:** Early identification of defects and issues in the development cycle minimizing the cost and effort required for bug fixes.
- **Documentation and reporting:** Maintain clear documentation of test case selection criteria and results, enabling effective communication between development and testing teams.

The below sections provide details about the industry challenges faced during the regression testing phase. The HCLTech solution was developed to prioritize test cases and defects for every regression cycle.

## Business challenges

In the software industry, the demand for regression testing with each build or release is significant, necessitating substantial resources and posing various challenges:

More resources required:

- Running regression tests after each build release is mandatory, but it consumes considerable time and resources, leading to difficulty in managing the process.
- Greater complexity: Over the span of development journey, regression tests become huge and complex, resulting in a huge amount of test cases to be executed.
- High cost: Testing extensive regression test packs requires significant resource investments with uncertain ROI, making it challenging to justify the commercial benefits to leadership teams.
- Schedule constraints: Organizations prioritize high-quality and quick delivery, imposing time constraints on regression testing. This often leads to risk-based testing to reduce the test case numbers and meet tight schedules.

## Problem statement

In the ever-evolving realm of software development, ensuring ongoing reliability and functionality amidst continuous updates and enhancements requires a comprehensive suite of test cases. However, efficiently managing and executing an ever-expanding set of test cases presents various challenges:

- **Resource constraints:** Limited resources, including time, hardware and manpower, make it difficult to execute the entire test suite for every software release. Testing teams are required to prioritize test cases based on their perceived criticality and importance, potentially overlooking corner cases or integration scenarios.
- **Increasing test case count:** Each software iteration introduces new test cases to verify new features or functionalities, leading to a substantial growth in the number of test cases within the test suite. This expansion can lead to prolonged testing cycles and significant resource utilization.

- **Regression test execution time:** Running the complete test suite for regression testing consumes a considerable amount of time, often causing release delays and discouraging frequent changes from developers due to the lengthy testing process.
- **Test coverage and quality assurance:** Ensuring adequate test coverage and quality assurance becomes an obstacle when dealing with an extensive test suite. Identifying test cases that are redundant, obsolete or providing insufficiently meaningful coverage becomes difficult, leading to inefficiencies in the testing process.

## Solution

In response to the challenges mentioned above, HCLTech has developed an analytics solution for a regression test optimization accelerator that provides a streamlined and optimized regression pack with every release. Furthermore, it includes features such as duplicate defect identification, preventing the logging of redundant defects and the orphan defect anomaly feature, ensuring adherence to process standards by mapping each test case to a defect. Additionally, defect clustering functionality identifies areas of high and low defect density areas, facilitating targeted quality improvement efforts.

The following diagram depicts the various features or functionalities.

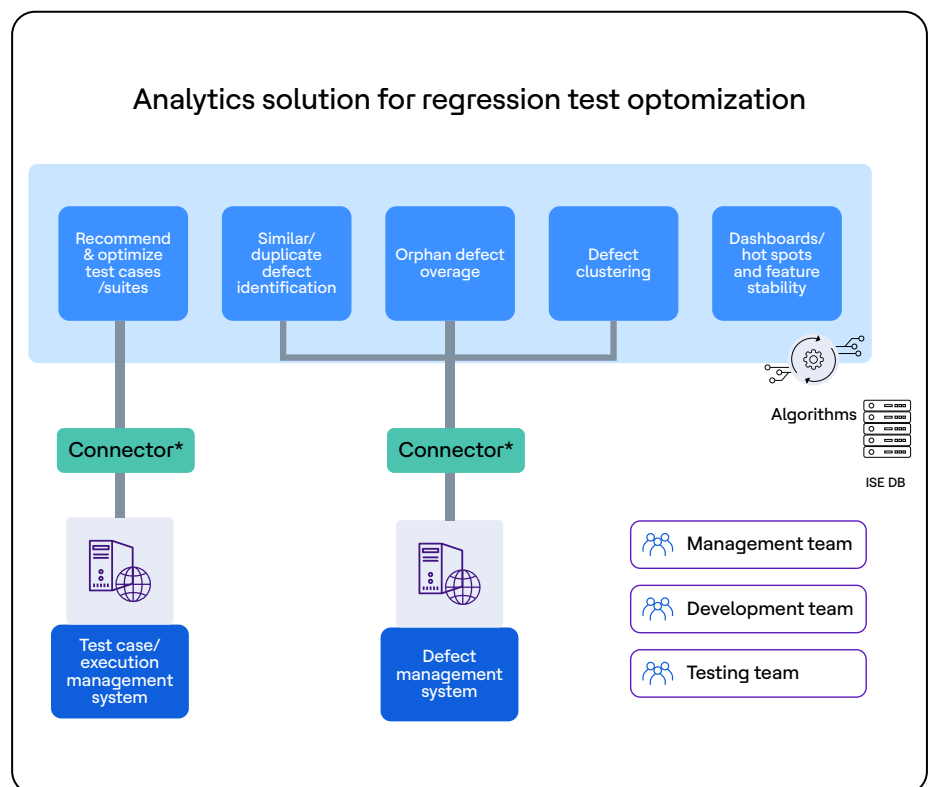


Figure 1. High-level architecture with features

The solution offers seamless integration with the third party tools like test management systems and defect management systems such as JIRA, Bugzilla, Clear Case and Quality Center, with the help of a connector that needs to be developed for unsupported third party tool. Its analytics solution provides the following functionalities:

### **Recommend optimized test cases or suites**

By leveraging the project's historical details like functional requirements, test cases and build execution history the analytics engine prioritizes test case for the analysis. Based on the requirements complexity corresponding to the test case, complexity and execution status of the test cases across the builds in the past, the solution prioritizes the test case inclusion or exclusion in the current regression pack execution.

The analytics engine analyzes the historical data of the project based on some rules like the ones given below:

- The user can associate high/medium/low priority with the test cases on the basis of the criticality of the corresponding functionality.
- Bug-related test cases are considered high-priority test cases.
- Test cases corresponding to features selected with the priority as 'HIGH' are displayed as High Feature Test cases (HFTs).
- Test cases corresponding to `testcase_priority_value` are also marked as HFT test cases.
- Test cases corresponding to features selected on low-with the priority are displayed as Low Feature Test (LFT) cases.
- Test cases corresponding to `lft_testcase_priority_value` are also marked as LFT.
- Test cases which are frequently failing are considered high priority test cases.
- Test cases whose execution status is volatile will be considered high-priority test cases.

### **Similar or duplicate defect identification**

The current and history of defects is the basis for this functionality. Users can input a keyword or sentence to identify similar defects in the defect database. The solution employs ML algorithms to analyze defect description, functional group, defect title, reproducible steps and defect log to search for similar or duplicate defects already present in the defects database. The system then provides a list of all the identified duplicate defects associated with the input keyword or sentence.

Following is some of the benefits provided by this feature:

- Identify the solution quickly, facilitating in routing the defect to the right engineer.

- Quickly identify all the similar issues.
- Quickly identify where and how similar issues are fixed.
- Reduce the number of duplicate defects reports.
- Reduce time spent by the development team on closing duplicate defects period.

### **Orphan defect coverage**

Orphan defects are not mapped to any test case demonstrating the incompleteness of the test bank. This feature relies on the test case document and defect bank. The feature also lists out all the test cases not mapped to any test case in test bank providing the following benefits:

- Identify feature areas where more test coverage is required as per field feedback (Complete Random Design for CRDs).
- Help the test engineer to identify the most relevant test case for orphan defects period.

### **Defect clustering**

The defect clustering feature enables the identification of non-uniform distribution of defects throughout the application and helps to provide the following benefits:

- Grouping similar nature of defects.
- Helps manager or lead in grouping similar sets for analysis and allocation.
- Eliminates duplicate defects to help managers period.

These features enable development teams to make them aware of volatile areas of failures, to assess the build stability and to inform them on frequent bug prone areas. It also enables testing teams to assist in identifying the regression pack, to pack the high and low stability feature test cases, to be aware of similar and duplicate defects and to ensure the testing team acts upon orphan defects. Finally, these features enable management teams to minimize the regression testing effort by optimizing the regression pack with a similar number of test cases, reduce the defect count by applying duplicate and similar defect rules to all open defects and to provide high-end visualization reports to provide detailed information on each aspect.

**Details of the solution usage and features are as specified:**

### **Input entry**

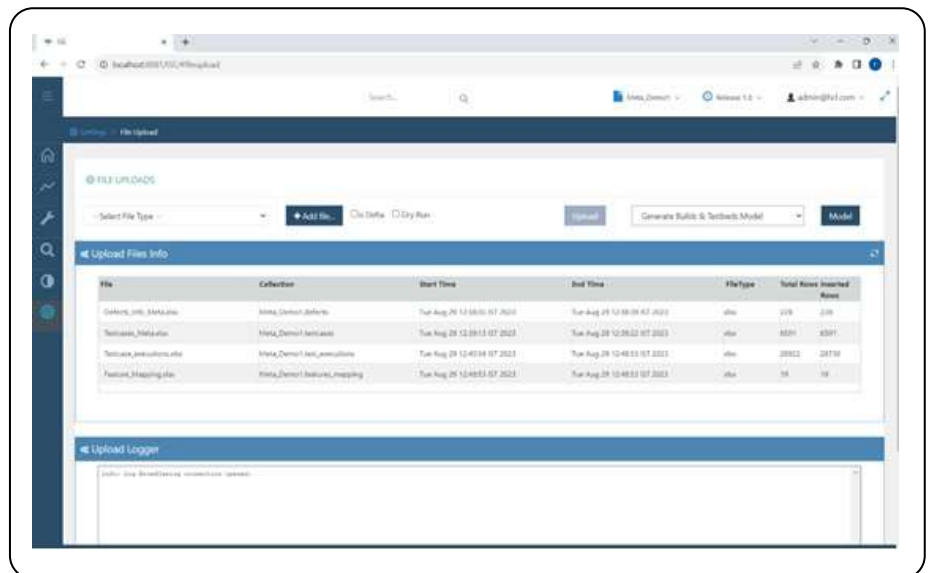
There are four kinds of input data required to perform regression optimization.

- Features list or requirements list.
- Test case details or information.
- Test case execution history details.
- Logged defect details.

The above details can be provided as an xls or CSV file, or the connectors are available to connect to different test management and defect management (like JIRA, Bugzilla, Clear Case, Quality Centre, etc.) systems to gather the required information.

In addition to that, we have connectors available for integration with various test cases or execution management and defect management tools to get the same or similar kind of information required to do analysis. Connectors to source code management systems are available to get build information.

The following image showcases how the input process exists in the system:



### Optimization analysis process

- Upload or gather all required information (features list or requirements list, test case details or information, test case execution history details, logged defect details) in File Upload and select Generate Builds and Test Beds Model and click on model.
- Based on that, all the test cases and defects gathered on test execution will be loaded into the database and all the required correlations will be mapped among the data.
- All the uploaded data is stored in DB as metadata.
- Apply ML algorithms to this metadata to get high and low features.

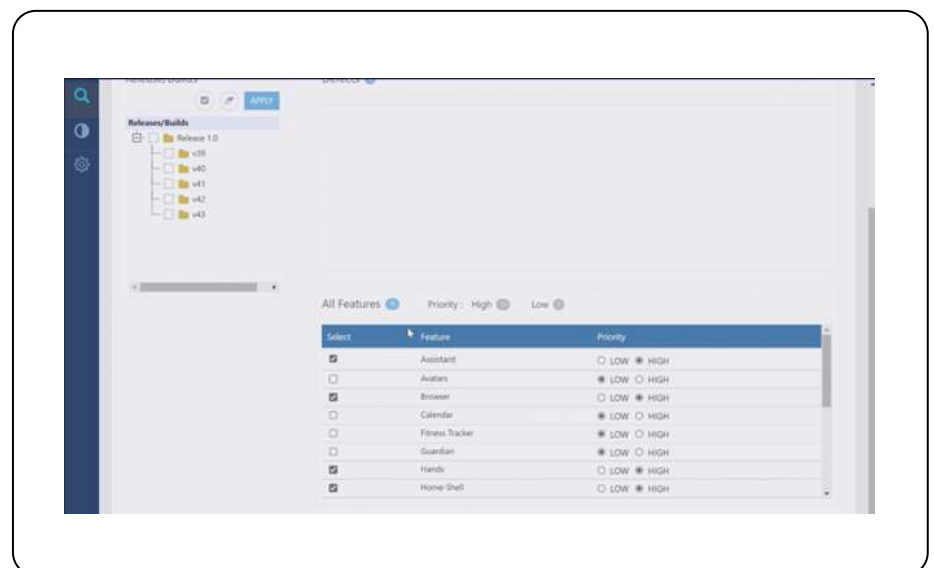
Based on the below norms, high and low feature functionality test cases are identified:

- HFT: Features selected with the priority "HIGH," then those high features of corresponding test cases are displayed as HFTs. (Feature recommendation rule for HFT) Test cases corresponding to testcase\_priority\_value are also marked as HFT (Test Case recommendation rule for HFT).
- LFT: Features selected with the priority as "LOW," then those low features of corresponding test cases were displayed as LFTs. (Feature recommendation rule for LFT) Test cases corresponding to lft\_testcase\_priority\_value are also marked as LFT (Test Case recommendation rule for LFT). Features List or requirements list, test case details or information, test case execution history details, logged defect details.

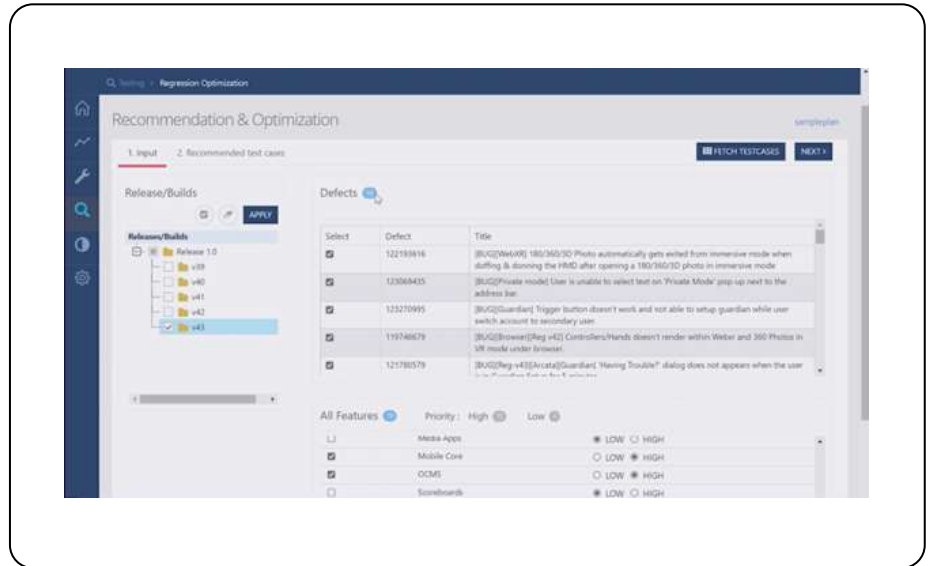
## Output

When you opt for regression optimization:

- Display the build information on the left-hand side
- It shows the features based on defects priority. High and low are classified based on the number of high and medium-priority defects logged period.

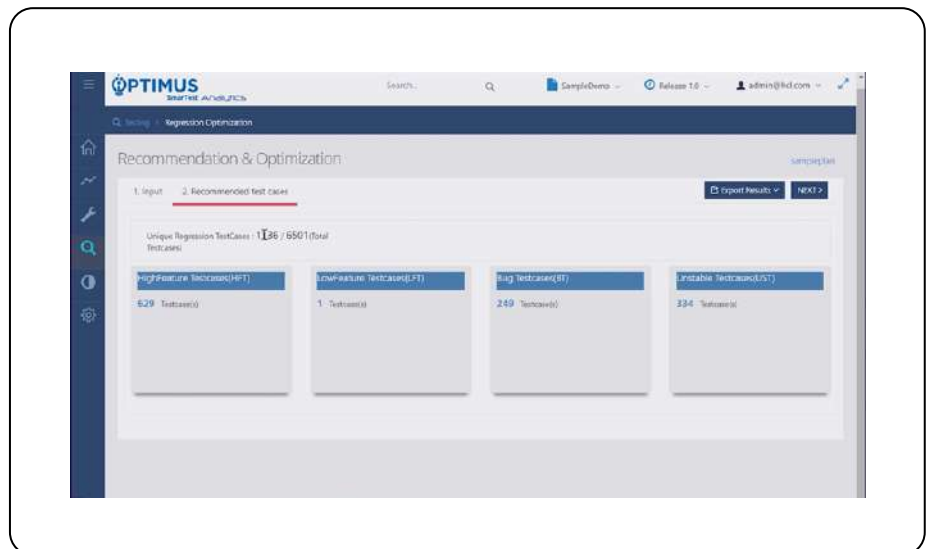


- Upon selection of the build for which defects are to be executed, click on apply period.
- Will list the defects to be evaluated as part of the selected build regression pack and high-priority defects period.



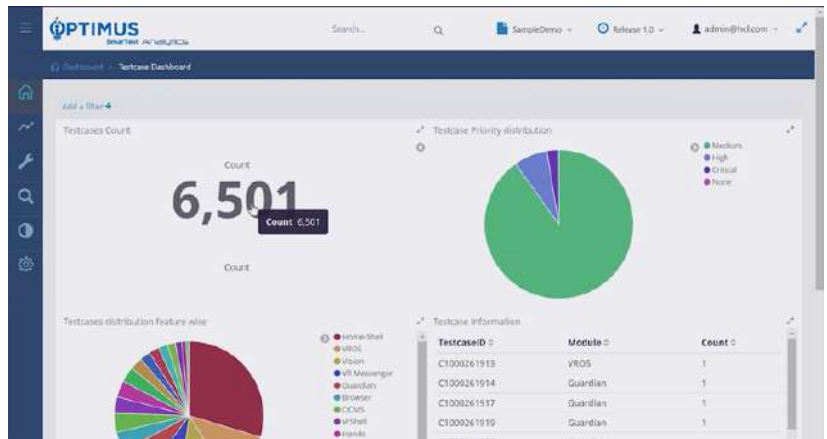
Recommended Test cases section gives the test case pack in different buckets.

- HFT – Test cases related to High Priority Features.
- LFT – Test cases related to Low Priority Features.
- Bug test cases – Test cases related to Current build bugs.
- Unstable test cases – Test cases whose execution result is not stable across the builds period.

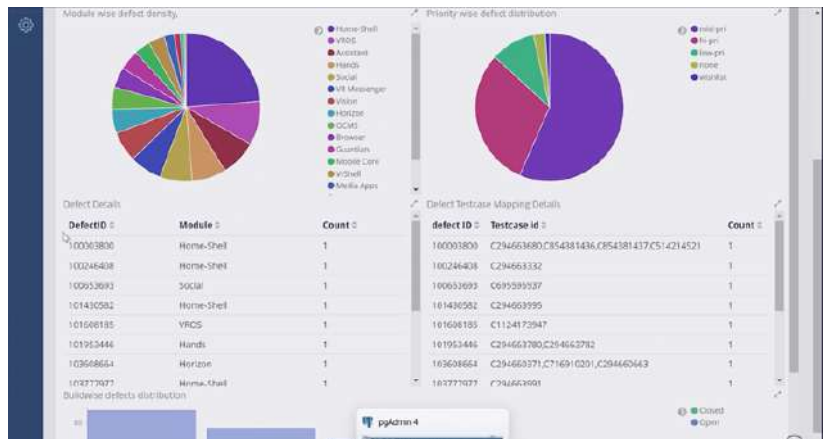


Dynamic visualization and reporting

## Dashboard – Test case



## Dashboard – Defect



## Similar or duplicate defects

Similar Search    Corbett Search    Advanced Search    File Search

Title:

Description:

Search Results of Corbett Search    Search with in results:     Show 15 entries    Columns:    

DefectID	Title	Similarity	Release	Req_Type
779279	Setback Web Video Downloader for Firefox is not working Firefox 11beta	37	20	DCR
781598	Firefox (3.0.) Freezes when in certain sites / switch tabs	83	15	DNFR
627574	Menu Bar is not restored after Firefox crashes	81	17	DCR
767984	Firefox Confidential does not work with Firefox 15.0.	81	17	DNFR
607452	Firefox is too slow	80	18	DNFR
700751	Problem on Firefox 15+ with event listener	79	16	DCR
704795	Firefox and CAPTCHA issue	77	16	DNFR
787979	Copy, Cut & Paste options (Cmd C, X & V, respectively) are not operating after keyboard command as Menu command when using Firefox on OS Mac OS	77	15	DNFR



**Continuous monitoring:** Establishing mechanisms for continuously monitoring the effectiveness of the optimization process and adapting it to changing software and business requirements.

- Applying regression optimization techniques continuously on every build.
- Integration with continuous integration tools to trigger the regression optimization technique on each build.
- Post the build trigger, apply the regression optimisation technique and then regression automation execution kicks off period.

## Benefits

The current solution benefits the project in the following ways:

- Provides the optimized regression pack based on historical data, which will reduce execution resources, cost and time variably.
- Reduce the time defect triaging as duplicate or similar defects and orphan defects will be identified by this solution.
- 50% reduction in duplicate defects.
- 50- 60% effort reduction in the initial triage phase.
- Reduces 20% of the effort which is being consumed in defect triaging.
- Identify feature areas where more test coverage is required as per field feedback.
- Helps management to understand current product status and trends.
- Identify areas of improvement for features or components period.

## Conclusion

Regression testing demands significant effort as test cases accumulated with each build become both resource and time-consuming to execute. Our solution offers an optimized pack of test cases for regression for each build, leveraging historical test execution and defect data. This approach gradually reduces resource usage and testing time while enhancing project benefits.

Furthermore, our defect grouping and clustering provides management with a concise summary of defects, categorized by various combinations of properties, facilitating in time reduction for defect triaging.

This helps the organization by reducing the effort taken for regression pack execution and providing the optimal way of selecting the test cases.

# References

- <https://www.sciencedirect.com/science/article/pii/S1877050916001514>
- <https://www.testquality.com/blog/tpost/vz2e6ifin1-regression-testing-challenges-and-best-p>
- <https://test.io/resources/blog/the-importance-of-regression-testing>
- <https://www.testrail.com/blog/test-case-prioritization/#:~:text=Keep%20customer%20requirements%20and%20preferences,of%20the%20software%20development%20project.>

# Author information

## Ravi Tanniru



Ravi is a Solution Architect working with PTS COE. He currently owns the design and development of PTS Solutions viz Falcon and LAMS. He has designed/implemented test solutions across multiple domains, such as banking, finance and insurance. He is actively engaged with the delivery teams in identifying test improvement opportunities and adopting test optimization practices through test automation solutions.

## Subramanian Dandapani



D Subramanian is working as Associate General Manager with HCLTech. He is an SME with expertise in the networking domain and has vast experience in Service Provider Networking testing and automation in large projects. His interest is in SDN/NFV, ML, understanding problems, solving them in an innovative manner and adopting suitable processes/tools/technologies.

# HCLTech | Supercharging Progress™

HCLTech is a global technology company, home to 222,000+ people across 60 countries, delivering industry-leading capabilities centered around Digital, Engineering and Cloud powered by a broad portfolio of technology services and software. The company generated consolidated revenues of \$12.3 billion over the 12 months ended December 2022. To learn how we can supercharge progress for you, visit [hcltech.com](https://hcltech.com).

[hcltech.com](https://hcltech.com)

