Liferay Quality **Assurance Program**

Overview

LIFERAY

Table of Contents

Introduction: Business Benefits of Liferay's Quality Assurance Program

At Liferay, we pride ourselves on the level of careful attention we put into creating excellent products for our customers. An essential part of any software development process is Quality Assurance, including the continuous and thorough testing of products. In order to ensure that we deliver the highest possible quality, Liferay's QA Program centers on one guiding philosophy: test the right things, at the right times, in the right ways.

Our QA Engineers take this charge very seriously. We test before, during and after development, utilizing different methods as appropriate for each situation, including the following:

- Functional End-to-End Testing
- Integration Testing
- Unit Testing
- Non-Functional Performance Testing
- Non-Functional Security Testing
- Exploratory Testing
- Smoke Testing
- Defect Validation Testing

We currently have 14,389 automated tests that run multiple times per day on many different release branches of our code. This breaks down to 3,869 unit tests, 8,661 integration tests and 1,861 functional tests. Altogether, Liferay is executing over 13 million tests per day against our code.

This is quite an extraordinary number of test executions occurring daily at Liferay, and a clear demonstration of Liferay's dedication to resolving any potential problems for our customers.

In this overview of Liferay's Quality Assurance Program, our goal is to describe the testing philosophy that we have refined through years of practice, as well as explain the extensive testing methods we employ. Our team values our customers' time, and it is for that reason that we continually test our products, searching out problems and fixing them before they become issues. It is our hope that this overview demonstrates our commitment to producing and empowering excellence for the thousands of Liferay deployments worldwide.

Liferay Testing Philosophy

Why do we test?

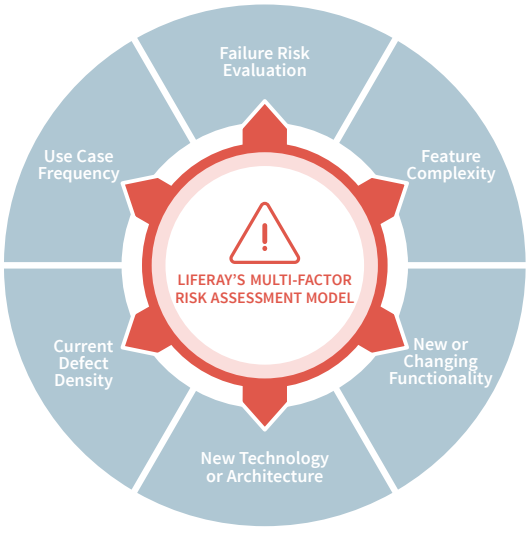
There are many different ways to gain knowledge about a software system. We could look at documentation, talk to developers, talk to software users and even look at the code, but out of all of those different paths to learning about a system, testing is the only avenue that explicitly validates that our system does what we want it to. Because of this, Liferay is committed to serving our customers with a robust testing strategy which ensures that we are effectively and efficiently producing products of high quality.

How do we determine what to test?

When testing large, complex systems, an organized, systematic testing method must be used. This is because it is impossible to test all combinations of input data. Likewise, it is impossible to test all possible execution paths through a program (there are most likely millions of paths that could be taken through the code).

Since we cannot test everything, we want to test the right things, at the right times, in the right ways. Testing is always a sample of possible execution paths, and Liferay takes care when selecting what input data to use and what paths to test.

Our first goal is to find the defects that must be fixed in order to successfully release the product, and to find as many such defects as possible. To do this, Liferay uses a multi-factor risk assessment model to determine what to test.



Liferay's Multi-Factor Risk Assessment Model

1. What is most important within the application?

A. **Use Case Frequency** - The Product Management Team at Liferay evaluates the use cases of different components of the application to determine where there is heavy usage of a component or feature, and prioritizes those over features that are infrequently used.

B. **Failure Risk Evaluation** - Would a failure in a certain portion of the application be damaging to our customers, either in terms of a data loss or system downtime? Or would a failure in a certain portion of the application be less damaging?

2. Where is there greater failure probability?

A. **Feature Complexity** - Where the functional areas are more complex, there will likely be a greater density of defects.

B. **New or Changing Functional Areas** - Where there are new or changing functions within an application, there will likely be a greater density of defects.

C. **New Technology or Architecture** - When new technology or a new architecture is introduced into the application, there will likely be a greater density of defects found.

D. **Current Defect Density** - If the defect density of a particular functional area is high, there is greater potential to find defects within that functional area, regardless of whether the area is actively changing or complex.

How do we determine when and how often to test?

"Test early, test often" is a phrase commonly used when discussing software testing. Liferay's testing strategy embraces that philosophy.

Test Early

Our Test Engineers are directly involved as part of our cross-functional product development teams when the software is being designed, which allows them to begin "testing" and identify potential problems before the code is even developed. Continued testing during the development process helps identify issues that arise as early as possible, resulting in more stable code throughout and at the

end of development. Our testing strategy implements both automated and manual testing frequently until release in order to continually enhance the quality of our software at every stage of the development lifecycle.

Test Often

Liferay employs a continuous integration (CI) architecture which allows us to run our suite of tests every time code is changed. These tests run hundreds of times per day as development teams fix defects, incorporate new features and refactor code on a continuous basis. Our CI infrastructure is described in detail in Liferay's Testing Technology section below.

What methods do we use to test?

There are myriad testing methods used in the software industry today. Liferay employs many different methods to achieve our goal of delivering high-quality products to our customers. Not every method is valuable at every stage of the project, nor is every method valuable for every type of project.

Here is a list of the testing methods Liferay regularly employs:

- Manual Functional Testing
- Automated Functional Testing
- Ad-Hoc Testing
- Compatibility Testing
- Non-Functional Testing (Load, Performance and Security Testing)
- End-to-End Testing
- Integration Testing
- Unit Testing
- Exploratory Testing
- Smoke Testing
- Defect Validation Testing
- Regression Testing
- Source Code Analysis
- Source Format Testing
- Semantic Versioning Automated Testing

The following sections describe when and how we employ some of these methods.

Testing Methodologies at Liferay

There are two primary types of testing methods used at Liferay that we will break down in further detail: automated testing and manual testing.

Automated Testing at Liferay

Automated testing uses tools, scripts and software to repetitively run tests against software that is continually changing. Automated testing allows for tests to be written once and run multiple times, even hundreds of times per day. Because automated tests are run by software tools, they can be continuously run without requiring the manpower that manual tests demand. Automated testing is also inherently more reliable because it is run by a computer and not a human. These tests allow Liferay to ensure that our software remains stable, even during rapid development cycles. Liferay has built a robust automated testing framework and analysis tools which allow our Test Engineers to quickly identify problems and work with the Software Engineers to efficiently fix those problems. Within our automated test suite we have four different types of automated tests: functional end-to-end tests, integration tests, unit tests and non-functional tests.

Functional End-to-End Testing

Liferay currently runs approximately 2,000 unique functional end-to-end tests on our software multiple times per day. A functional end-to-end test is a type of test that asserts that a user path through a system is behaving in the correct manner. That is, the entire system is started up and the test performs an action or actions that a user would take within the system, finally checking the given result. The test doesn't have to know or care what is happening internally to accomplish the result because a Functional Test checks the outermost point of input into the system, as well as the outermost result that is produced.

Functional end-to-end tests are uniquely suited for the role of asserting that an end user's product experience meets expectations, because the system under test is as close to "real" and "complete" as possible. All of the interactions between the various parts, or architecture, of the system and all of the complexity within each of those parts are being exercised during the execution of the test.

However, functional end-to-end tests tend to be slow and fragile. They are slow because they are navigating through all of the layers in the architecture of the system, and this takes time. They also tend to be more fragile than other types of tests because they rely on every single piece of the architecture behaving as when the test was written. Any change to the UI layout, the user

messaging within an application or the response time of the application can potentially break a functional test. Since functional tests mimic user interaction, the tests themselves also tend to be more difficult to maintain when changes to the tests are needed.

Functional end-to-end tests cannot ever fully test the entire scope of a large, complex application. The sheer number of components, as well as the substantial number of potential pathways through the system, prohibit functional tests from covering all the paths and making all the assertions needed to ensure a robust, well-tested application. This is why Liferay depends on more than functional end-to-end tests to guarantee the stability of our application.

Integration Testing

Integration testing is a key component of Liferay's testing suite. Liferay currently runs approximately 9,000 unique integration tests on our software multiple times per day. An integration test does just what its name implies; it tests the integration between two components of a system. Integration tests don't go through the user interface to simulate user interaction, but instead, start one layer down and test the underlying services.

The advantage of integration tests is that they don't have to deal with the fragility of the UI, and they retain the ability to test out the connectivity between two separate components to make sure everything is properly connected. These tests are a key component for ensuring the application is properly hooked up and communicating well both internally and externally.

On the other hand, integration tests are not appropriate to test every single interaction between components. In order to have a full picture of the system's behavior, a large number of integration tests would be needed. This would lead to costly maintenance that only gets costlier as more components are added to a system.

The other disadvantage to using only integration tests in software testing is their lack of precision. While integration tests can effectively tell you when something is wrong with the interactions between two components, they often cannot tell where exactly the software is broken. This requires even lower level testing. Liferay addresses this with unit testing, which provides the lower level information that is needed.

Unit Testing

Liferay currently runs approximately 4,000 unique unit tests on our software multiple times per day and relies on unit tests for their precision, speed and coverage. Unit tests are small, method-level tests written to prove that a function works. They don't go end-to-end through all layers of a software system. They tend to be very localized. Because of these properties, unit tests are fast, focused and easy to maintain. They are great at testing edge cases and boundary conditions in the code and are an essential tool in making sure that each method of code is covered by a test.

The only downside to all of this speed and precision in unit tests is that they cannot test integration within a system. Certain defects only appear when a system is hooked up and running together with all of its components. That is why, at Liferay, we use all three types of tests in our automated functional testing suite, which helps us achieve a clear, robust picture of the stability of our software at any point in time.

Non-Functional Testing

PERFORMANCE TESTING

Performance Testing is a type of testing that intentionally pushes the limits of software in order to identify its limits and bottlenecks. The Liferay engineering team performs intensive tuning and testing to monitor and improve the stability and scalability of Liferay Digital Experience Platform across an array of use cases, including: infrastructure portal, collaboration and content management.

The team uses a combination of industry standard performance testing and measurement tools, such as Grinder with distributed load injectors, JMeter and Dynatrace. Through the use of these tools, the team is able to eliminate performance bottlenecks and produce a product that can support millions of users and assets while under heavy concurrent load.

For more information on our performance testing methodology, please review our latest performance whitepaper.

Manual Testing at Liferay

At its most basic, manual testing is the act of having a person navigate through a product to assert the expected functionality of the product. Manual testing is the activity most frequently associated with a Quality Assurance team. Executed too frequently or at too great a scope, this type of testing activity can result in lower overall product quality because of the high level of effort and time required. At Liferay, we recognize manual testing as a valuable part of assuring the highest

possible quality in our products, despite its negative stigma. Our skilled and experienced testers have deep testing knowledge and solid integration with the product development teams. Because of these strengths, the testing team is able to judiciously employ various types of manual testing as a regular and vital part of our development and support processes.

Exploratory Testing

Exploratory testing is an unscripted type of manual testing that emphasizes individual tester freedom, testing skill and experience in the product under test. This type of testing is guided by integrated feedback cycles within the development and support contexts. Exploratory testing is a high skill form of testing that can often uncover new and critical product defects that no written test case could prepare for. Here at Liferay, we most frequently leverage this method of testing during our feature development and release cycles, to identify new and distinct quality defects that automated testing would not otherwise find.

With exploratory testing, the quality of the results can depend on the skill and engagement of the testers involved. Teams that employ exploratory testing without strong product knowledge or testing expertise can struggle to effectively employ this technique, which can lead to directionless and ineffective testing cycles. Testing teams at Liferay avoid these pitfalls by being tightly integrated with a cross functional development team that hones an expert knowledge of the product and fosters creativity, diligence and independence.

End-to-End Testing

Manual end-to-end testing is a type of testing that involves ensuring a product functions as expected in as real-world a scenario as possible. Here at Liferay, many of our automated functional tests would fall under this definition as well because they involve testing against real binaries that are connected to real databases, with real browsers, launching and executing real user interactions against the product. However, automation has its limits, and with a product library as varied and complex as Liferay's, there are still many points of integration that would be cost prohibitive to test in a fully automated way.

One example of how Liferay executes manual end-to-end testing for the Liferay Digital Experience Platform product would be as follows:

- Unzip the binary
- Configure the application to connect to a live database
- Launch and configure the product for connection with other independent systems, like our Liferay Cloud Services offering or Liferay Marketplace
- Validate that the configured integrations and related functionality work as expected

Manual end-to-end testing in this scenario enables us to fully vet that the product functions as we expect it to, from start to finish across multiple product integrations that have their own independent release and support cycles.

Manual end-to-end testing is an excellent manual testing tool for Liferay because it enables us to identify difficult-to-automate points of product integration.

Challenges and difficulties with end-to-end testing can arise because of the increased number of dependencies and integrations required to execute satisfactory testing. In Liferay's case, this type of testing can be particularly demanding for project managers to coordinate across product teams, particularly in the cases of integration with products that are also currently in production.

Smoke Testing

The term "smoke testing" comes from the testing of electronic hardware, specifically referring to turning on an electronic device and seeing if the device begins to smoke. Other related terms to smoke testing include "sanity testing" and "build verification." We periodically employ smoke testing to rapidly ensure that a given product build is both worth testing and genuinely appears to be in releasable condition. Like end-to-end testing, this particular methodology has crossover with automated functional tests.

The greatest benefit of smoke testing is that it is a low cost and extremely high value exercise that returns rapid results to the tester on whether or not a given product build is ready for release. We supplement our automated smoke tests with manual smoke testing to ensure that, prior to any release, a tester has reviewed and made their own personal assessment about the perceived quality level of the product in question.

Defect Validation Testing

Any person who has used software for a period of time knows that defects are an unfortunate part of software development. At Liferay, every user-facing defect that is fixed by a developer is then manually tested by a Quality Assurance team member to ensure that the defect fix meets our expectations.

Automated Testing vs. Manual Testing

At Liferay, we use both automated and manual testing because both of these methods have advantages that can complement each other when used in the appropriate situations.

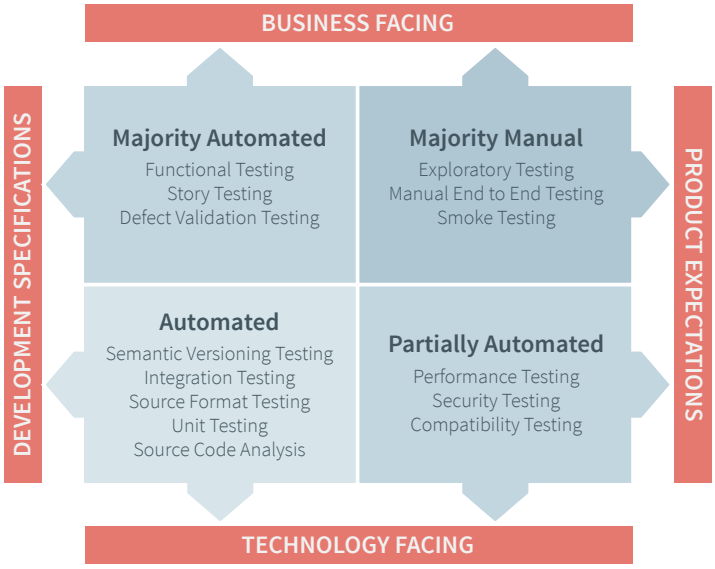
Liferay uses automated testing to achieve the following goals:

- To catch regression defects in the code
- Repetitive smoke testing to make sure the basic system is functioning correctly
- Data-driven testing which allows the testing of a single function with a lot of different data
- Performance testing
- Load testing
- Environment smoke testing

Liferay uses manual testing to achieve the following goals:

- Finding defects related to changing or new functionality
- Accurately testing highly complex functionality
- Successfully testing functionality that requires subjective feedback (usability, look-and-feel, etc.)

The various types of testing and their distinctions can be summarized in the following diagram:



Four Quadrants of Liferay Testing Methodology

Liferay's Testing Technology

Liferay's testing infrastructure works together across a variety of software, from standards oriented automated test technologies and continuous integration to custom test management and analysis solutions. Our testing stack executes thousands of tests, hundreds of times per day, providing detailed quality insights that drive product development, quality assurance and support decisions our teams can rapidly implement.

Automated Functional Tests

Liferay's automated functional tests are backed by industry standard Selenium-Webdriver technology that enables user-like interactions through live web browsers running against the full application stack. On top of Selenium-Webdriver is a Liferay-developed DSL that we use internally to develop and maintain our tests. These tests are designed following the industry standard Page Object design pattern which can significantly reduce test maintenance.

Integration Tests

Liferay integration tests use JUnit as the primary test runner and we categorize our integration tests into two different types:

- Integration tests for the product core (portal-impl)
- Integration tests for modules

Our Liferay integration tests use Spring to initialize the application context. By doing this, we are able to access a reduced set of product functionality with real and not-mocked services.

Our integration tests for modules require a live portal to have been started. If these tests are kicked off, they will start the portal on the local environment if one has not already been started by the user. Once the portal is up, we use Arquillian to connect to the running instance, in order to send the test execution to the correct instance and get the test results back to the test runner that launched the tests.

Unit Tests

We use JUnit to run our unit tests in combination with Mockito and PowerMock to mock objects as necessary.

Test Lab Infrastructure

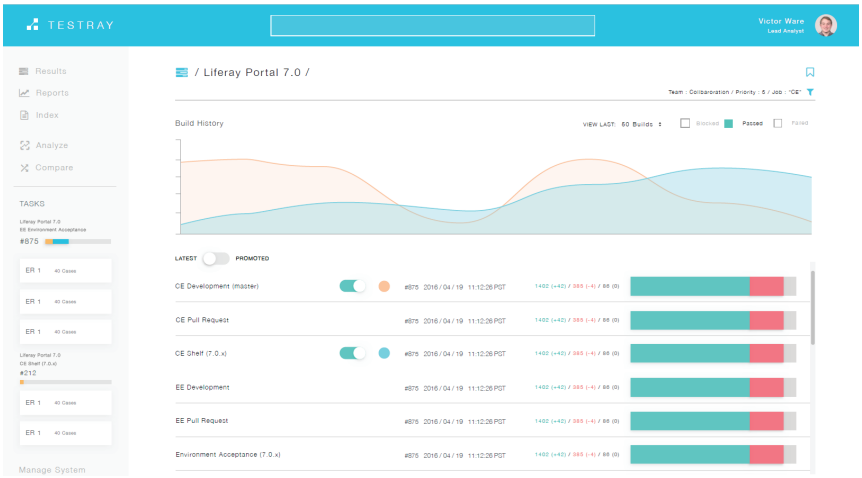
At Liferay HQ in Los Angeles, CA, our Test Lab hosts 700 servers running a combined 2,800 virtual machines that host our automated testing. Machine and server provisioning management are fully automated through a combination of custom and industry standard tools.

This environment executes all forms of automated testing across the entirety of the Liferay product catalog. Including the execution of tests against our code bases under active development, fix packs, service packs, hotfixes, corporate intranet and external facing sites. These machines host the full spectrum of environments that our Liferay products support. This includes selected permutations of operating systems, browsers, application servers, databases and JDK.

One major feature and responsibility of this infrastructure is the automated testing of ongoing development. Liferay executes the industry standard practice of continuous integration on a massive scale by not only regularly testing our product from the source code repository, but by testing every change against that code as it is sent in by the hundreds of Liferay developers. Liferay's Pull Request Tester executes a broad battery of tests that ultimately results in millions of test executions running multiple times per day against several products and across multiple environments.

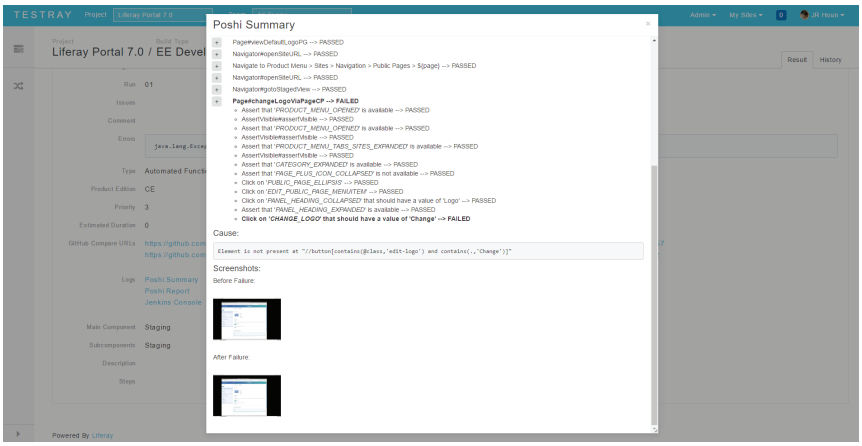
Test Management and Analysis

With so many test executions in a given day, Liferay generates gigabytes of test data on a regular basis. With the help of a special in-house Test Management solution, built on Liferay, we are able to turn what could be an overwhelming and ultimately useless amount of data into ongoing quality insights that drive our Support and Development efforts.



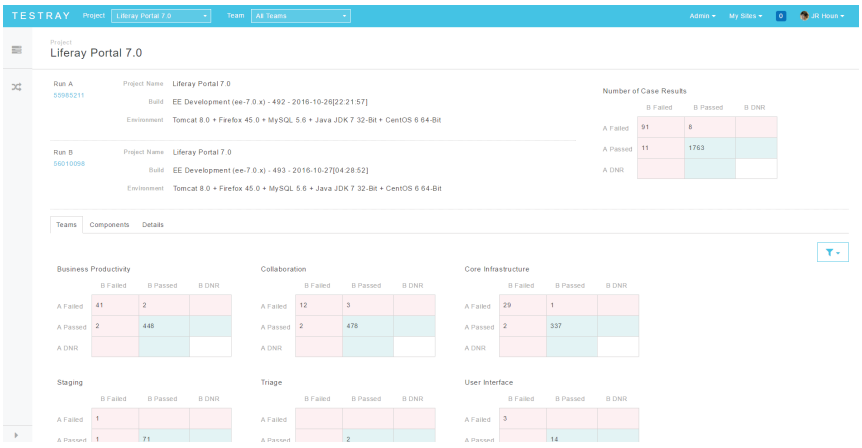
Testray Dashboard View

This Test Management solution, internally referred to as Testray, organizes and aggregates test results from a top level product view, and traces aggregated test result data all the way down to the individual test case results containing test case history, product debug information and test execution data. This information is indispensable to our quality and testing efforts.



Selenium-Webdriver Test Summary with Screenshots

For our front-end functional test developers and analysts, we are able to review and debug individual automated front-end functional test results by reviewing test reports and screenshots in a single unified location. For our Support and Release teams we are able to quickly execute comparative analyses to identify critical regressions in our product support releases.



Automation Test Comparative Analysis View

Conclusion

At Liferay, producing excellence and valuing people are core business values. We believe that providing consistently high quality software for our customers is an essential part of that. The testing philosophy that fuels our Quality Assurance Program enables us to create the best value, service and products possible.

We have built our business by becoming the best partner for providing enterprise-level support, quality assurance and management tools to global companies running Liferay in production. Companies such as Carrefour, Coach, Danone, Fujitsu, Lufthansa Aviation Training, Siemens, Société Générale, VMware and the United Nations use Liferay.

What's Next?

Learn how you can benefit from a Liferay Enterprise Subscription by contacting us at: sales@liferay.com.

Request a quote by visiting liferay.com/get-price-quote.

Liferay makes software that helps companies create digital experiences on web, mobile and connected devices. The Liferay platform is open source, which makes it more reliable, innovative and secure. Companies such as Carrefour, Coach, Danone, Fujitsu, Lufthansa Aviation Training, Siemens, Société Générale, VMware and the United Nations use Liferay. Learn more at liferay.com.

180131