PARASOFT®

CASE STUDY

Fitch Delivers High Code Coverage & Quality for Microservices Applications



OVERVIEW

Fitch Solutions is a world-leading provider of credit intelligence and the primary distributor of Fitch Ratings content. Today, 90% of the world's leading financial institutions, multinational companies, governmental bodies, and consulting firms based in more than 118 countries depend on Fitch content to inform their business decisions. They are a global leader in financial information services with operations in more than 30 countries.

Fitch Solutions follows a microservice architecture with over 200 microservices. The majority of those microservices are based on Java Spring Boot, while some use Vue.js and C++. The Fitch Solutions team responsible for these services includes more than 50 developers spread out across the world.

CODE COVERAGE	DOWNTIME	YOY PRODUCTIVITY
90%	0 days	+12%



THE CHALLENGES

Fitch had three main problems to solve:

- Unplanned downtime
- Lack of feedback for developers
- Rigid two week release cycles

UNPLANNED DOWNTIME



In 2019, Fitch had 18 incidents that caused their website to go down due to breakages in the production environment. These unplanned outages were related to either their own APIs or other products that they host. Eighteen incidents averaged to more than once a month. That's bad news for the service level agreements (SLA) and their clients.

LACK OF FEEDBACK FOR DEVELOPERS

The software development team at Fitch Solutions is fluid. Developers move from project to project so it's common for them to work on and become responsible for microservices unknown to them. Adding features to unfamiliar code was difficult because there was a low number of tests to offer results and help developers understand expected behavior. In addition, subject matter experts (SMEs) were often unavailable to consult.

All these factors combined—developers switching teams often, not having access to SMEs, and having low test coverage of approximately 20%—led to lack of feedback. The missing information hampered the team's ability to keep services working and add new functionality. That's why Fitch took action to reach a higher level of test coverage.

RIGID TWO WEEK RELEASE CYCLES

Two week release cycles are the standard development process at Fitch. Developers were not able to release on their own schedule. So if development of a new feature was completed before the end of the deployment cycle, it had to wait until the end of the sprint to move forward to QA. Ideally, they needed to be able to develop and deploy a feature whenever it was implemented and tested. Doing this would allow them to get new features released more often and accelerate their time to market.

THE APPROACH

Fitch Solutions considered these three major challenges and looked for a common denominator between all three. It became apparent that all three were related to lack of testing resulting in poor code coverage. They decided to increase code coverage to ensure they were fully testing their microservices and help their developers deliver better code at a faster pace.

Good code coverage on a microservice means that most, if not all, paths through the code are executed. Creating a test suite for each microservice that covers all the different functions means that untested code is not exposed to clients. For any future changes, automated regression tests give developers instant feedback as to whether something in the code broke.

This approach eventually reduces outages. It's more effective and efficient to test current and new functionality before it moves to production. This approach helped Fitch Solutions reduce the reliance on their QA group to stage and test new functionality and decouple feature development from the two week release cycle.

A TWO-PRONGED APPROACH TO INCREASE CODE COVERAGE

The plan that Fitch Solutions came up with was to achieve 90% unit test coverage (specifically line coverage). This was a big task as they have over 200 microservices! The team was up to the challenge and used a two-pronged approach to tackle it.

Create Many New Tests

The first prong of the approach was to retroactively increase the test coverage of all existing code. To reach the 90% code coverage goal, they needed to create a lot of new tests. They assembled a dedicated team to tackle this. The team was responsible for writing unit tests exclusively to build up the code coverage and fully test Fitch Solutions' existing microservices.

Follow Unit Testing Guidelines

The second prong to this approach was to make sure that developers writing new features are following the unit testing guidelines and achieving the code coverage goals as new code is written. These guidelines were applied across the organization for all developers to put into practice. The team introduced quality gates to ensure the quality of code merged into the main production branch.



ADOPTING AN INTEGRATED JAVA TESTING TOOL TO ACHIEVE GOALS

To achieve their testing, coverage, and quality goals, Fitch Solutions adopted Parasoft Jtest. They found it particularly useful to use both the code coverage capabilities and the unit test assistant in Parasoft Jtest. If the developers were writing new features or re-testing legacy services, they measured code coverage first to gauge how extensive the existing testing was. They used Jtest's code coverage reports at all different levels to discover where their testing was lacking, including investigating missing coverage at the file and line of code level.

The Unit Test Assistant helped Fitch Solutions fill in the gap of unit tests. Developers created new unit tests with a click of a button while hovering over a method in the code. The Unit Test Assistant creates a test file, harness, and stubs/mocks as needed. Developers just add assertions to verify behavior and adjust mocks, and a new unit test is ready.

The entire team accelerated the creation and execution of tests using Parasoft Jtest. It was especially helpful for their junior developers who needed some guidance on unit test creation.

SOLUTION BENEFITS

Fitch Solutions is still on their journey to improving their software testing. They're already experiencing benefits from their Parasoft solution.

A reliable way to measure and improve code coverage. It was critical for Fitch Solutions to improve their testing. They knew their current test coverage was low and needed a way to reliably measure coverage and tools to help them improve it.

Accelerated unit test creation and maintenance. Achieving their ambitious goal of 90% test coverage meant creating a lot of new tests. Parasoft Jtest Unit Test Assistant was key to helping the team create and maintain tests easily and efficiently.

Reduced downtime. Fitch Solutions' new testing approach and improved quality directly resulted in cutting downtime from more than monthly to zero downtime in a 100 day period.

Increased developer productivity. Despite a small hit in productivity when the improved testing initiative started, the team observed 12% more developer activity in the following months, as measured by their internal data analysis.

THE RESULTS

"The very exciting thing is that we've gone 100 days without any downtime in our production system. Something that was happening more than monthly."

By Q3 of 2020, Fitch Solutions improved their testing of 10% of microservices to their goal of 90% code coverage. Although they're still on their journey to reaching their coverage goals, they've gone 100 days without any downtime in their production system. It's their longest streak since adopting their new unit testing approach. Vince Recupito, senior software engineer at Fitch Solutions, said, "The very exciting thing is that we've gone 100 days without any downtime in our production system. Something that was happening more than monthly."

Fitch Solutions engaged with a third-party company that specializes in measuring developer productivity. The in-depth analysis uses direct access to code repositories to measure commit history and find trends and metrics based on developer activities.

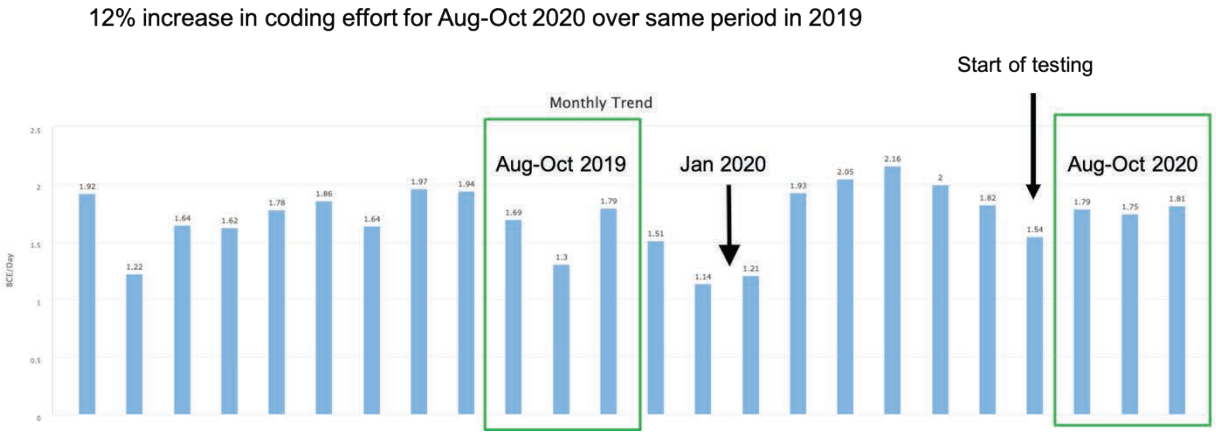


Figure 1:
A chart showing developer productivity over time based on source repository activity.

Figure 1 was created from the productivity data collected showing the monthly average amount of time that developers spend writing code per day.

The data shows Fitch's team was spending about two hours per day of just typing on a keyboard, writing code. The dip in January is expected to be due to holidays. The interesting part is the start of their new testing effort in July.



At this point, the team was going back through their legacy microservices, evaluating, and increasing code coverage. It's also the point where the team insisted that all new features be tested properly.

After this initial dip, the team was back into code development at a rate 12% higher than the same period in 2019. This was a good sign since developers were spending more time coding and less time in meetings. However, the root cause of this productivity increase was the significant decrease in production system downtime.

> *"One of the conclusions we're drawing is that our developers are spending less time fixing production issues and more of that time writing code. In addition to that, as developers are writing more tests, hopefully they're finding bugs sooner before they get into QA."*
>
> —Vince Recupito, senior software engineer at Fitch Solutions

Fitch Solutions not only saw a reduction in production system downtime, but also a reduction in developers "fire fighting" to solve production problems. In addition, the new testing regime had little impact on their developer productivity with an observed increase in productivity of 12%.

In addition to these gains, Fitch Solutions' developers are writing more tests and finding bugs sooner, before they get into QA. Finding bugs earlier also means less back and forth with QA and less time in the QA process overall.

"One of the conclusions we're drawing is that our developers are spending less time fixing production issues and more of that time writing code. In addition to that, as developers are writing more tests, hopefully they're finding bugs sooner before they get into QA," said Vince Recupito.

TAKE THE NEXT STEP

Save your development team time and empower them to significantly enhance product quality. Download a complete tutorial on how to set up, write, and run JUnit tests and scale unit testing with automated Java testing.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives — security, safety-critical, Agile, DevOps, and continuous testing.