STICKYMINDS™ A TECHWELL COMMUNITY Making Software Better Every Day

Published on *StickyMinds* (**https://www.stickyminds.com**)

How to Make 100 Releases Per Day with Only 6 Quality Engineers

By **Evgeny Tkachenko** - February 12, 2021

The online retail market is very competitive and demanding, so new features and functionality need to be delivered to customers as quickly as possible—every minute wasted can cost you thousands of dollars.

At Wayfair, we have dozens of different internal and external tools and applications and more than a hundred teams that support them. We release our code to production hundreds of times a day with confidence in the quality of the products. The Wayfair Catalog Engineering team has six Quality Engineers (QE) and 200+ Developers.

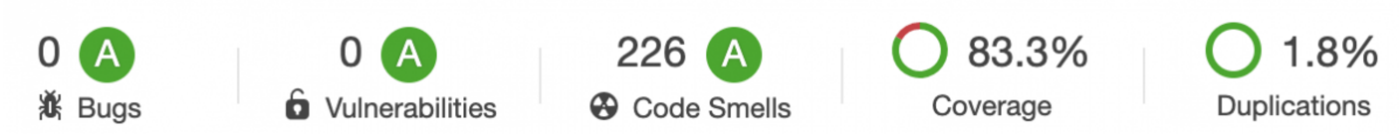
Let's start our journey from the very beginning of a feature life cycle. A Product Manager comes up with the idea for a new feature and shares it with the team, and then the testing begins.

We Analyze Requirements

The cost to fix bugs found during the testing phase could be fifteen times more than the cost of fixing those found during the requirements phase or design phase. We do not have business analysts within the teams, and most of the teams do not have dedicated QA/QE resources. We are focused on preventing bugs instead of finding them in the implementation. We train the teams on what to look for during requirements analysis and share the best practices to make sure everyone is on the same page (acceptance-test-driven development, behavior-driven development). Through conversations and collaborations between key stakeholders, we discover the value of the proposed functionality and build the right software.

QEs educate teams on how to write acceptance-criteria in BDD style where applicable and on how to analyze requirements for completeness, correctness, consistency, clearness, and testability.

We Use Static Code Analysis, Vulnerability Scanners, and Code Reviews



"Quality comes not from inspection, but from the improvement of the production process." — W. Edwards Deming

Quality is not an afterthought, and it must go beyond the product. We cannot add quality at release or after release or inspect it into the product. That is why, in our department, we heavily invest in the tools and processes that help us prevent defects or at least identify issues as early as possible in the process.

We actively use tools and platforms for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities. These approaches help us catch tricky bugs, prevent undefined behavior (impacting end-users), fix vulnerabilities that compromise our applications, and make sure our codebase is clean and maintainable with minimal tech debt. Our code review process helps developers learn the code base, as well as help them learn new technologies and techniques that grow their skill sets and improve the quality of the code they produce.

We Write Unit Tests

```
[Test]
public void The_nonpurchasable_optionCombination_key_provider_should_produce_keys_with_nonpurchasable_prefix()
{
    var nonPurchasableKeyProvider = new NonPurchasableOptionCombinationKeyProvider();
    var nonPurchasableKey = nonPurchasableKeyProvider.GetCacheKey(12345, BrandCatalogType.WayfairUS);
    StringAssert.StartsWith("NonPurchasable", nonPurchasableKey);
}
```

We spend ten times more time and effort reading code than writing it, because to write new code you should understand what the old code does.

Unit testing helps us to provide developers with a mechanism for producing self-documenting code, gives us a higher level of quality in our software, and uncovers problems early. We cultivate a culture of writing unit tests for new features within the sprint and making it a part of the Definition of Done. We set quality gates on the unit test coverage for newly added functionality to visualize our progress in expanding coverage.

We Automate Only What Matters Most

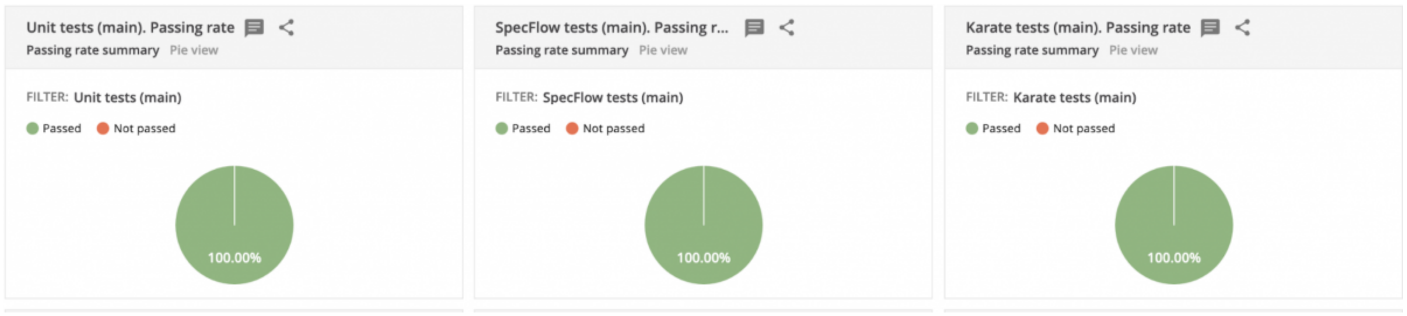
At Wayfair, we take care to differentiate automated tests and test automation. Automated tests are just scripts that help you avoid testing manually, while test automation is about how we write and apply tests as a key element of the software development life cycle (SDLC).

We focus on isolated unit tests and isolated component integration tests. GUI and integration tests are slow, and they do not put pressure on design. They are expensive to write and maintain, and they are also fragile.

We Provide Training and Coaching on What, Why, and How to Test or Automate

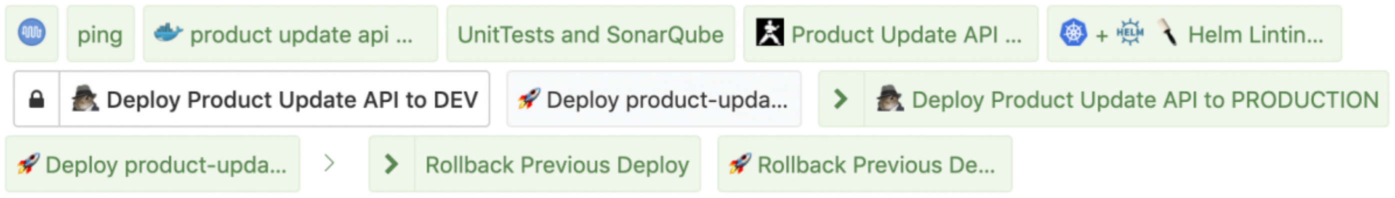
People are not good at testing their own code. A fresh set of eyes is needed to make sure the code works fine and to not miss a defect. For most of the teams in Catalog Quality Engineering, we have a process in place in which developers do a cross-check of features implemented by their teammates. You will never succeed in test automation if you are not good at “manual testing.”
If you automate "garbage," it will be nothing but automated "garbage." Teams must be trained or at least equipped with documentation about what to test, how to test, what to automate, and how to automate. The Catalog Quality Engineering team educates on test automation and testing in general, enabling development teams to own their own quality.

We Visualize the Health of the Product and Test Automation


Unit tests (main). Passing rate
Passing rate summary
Pie view
FILTER: Unit tests (main)
Passed
Not passed
100.00%
SpecFlow tests (main). Passing r...
Passing rate summary
Pie view
FILTER: SpecFlow tests (main)
Passed
Not passed
100.00%
Karate tests (main). Passing rate
Passing rate summary
Pie view
FILTER: Karate tests (main)
Passed
Not passed
100.00%

We acquire, aggregate, and analyze automated test results and metrics (code coverage, passing rate, performance metrics, etc.) from development and production environments to visualize the product's health. Based on that data, we create dashboards with fancy-looking charts and graphs and display them on TVs across our office to make everyone feel engaged in perfecting the quality of our products.

We Enable Continuous Delivery Through Deployment Pipelines



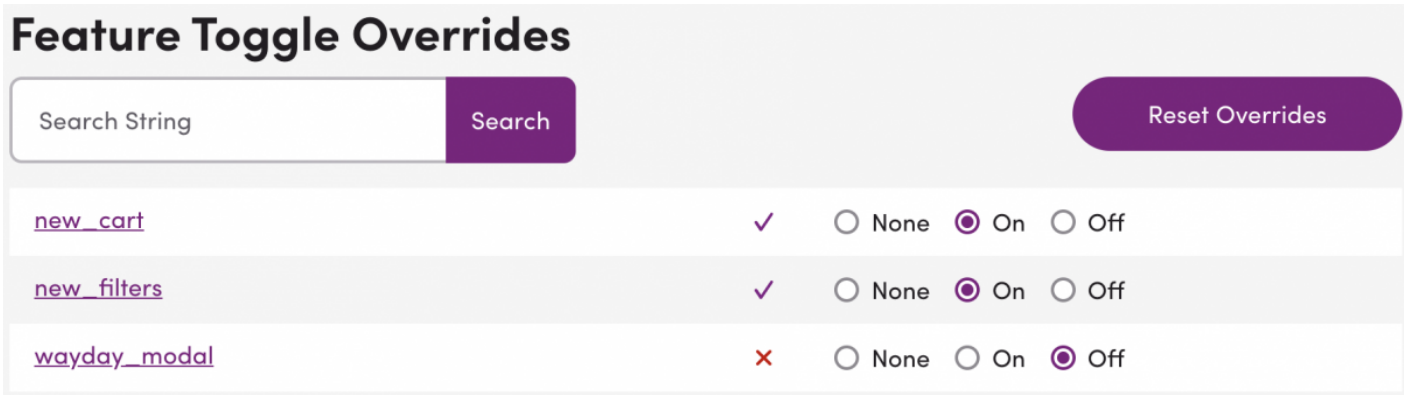
Automated tests are almost useless without proper integration into the delivery process. We design and build CI/CD pipelines for all of our tools to make it possible to deliver new features to our customers with a minimum of manual manipulation.

We have an infrastructure that allows us to deliver features one by one. We potentially can start developing functionality in the morning and release it the same day to production with confidence in quality.

Most of Our Tests Run in Isolated Environments

With so many teams and applications supported, it is almost impossible to have a stable development environment for comprehensive testing. We build tools that allow us to run tests locally within our own instances of shared services so we can manipulate data any way we want and be confident in the stability of the environment and tests. I can't emphasize this enough: tests are either trustworthy or useless.

We Use Incremental Rollout for "Big" Features

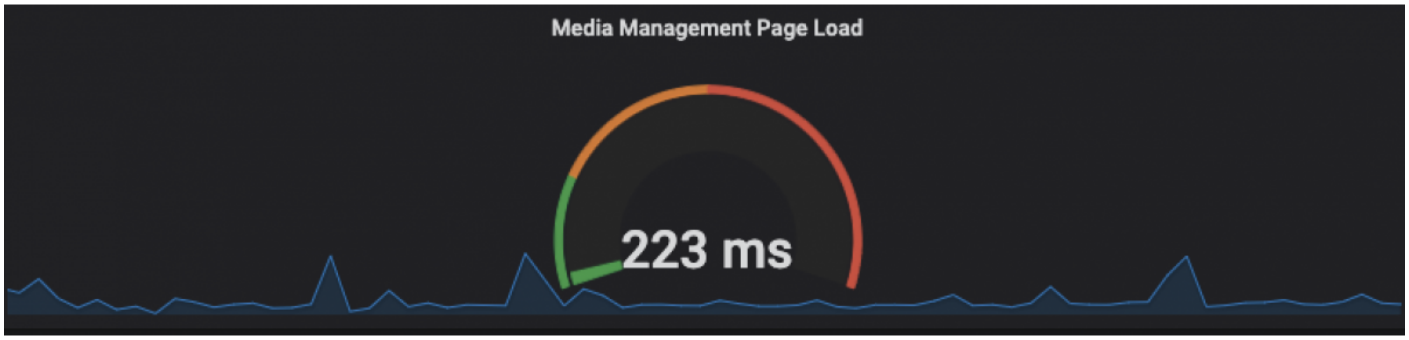


new_cart	✓	None [✓ On] Off
new_filters	✓	None [✓ On] Off
wayday_modal	✗	None On [✓ Off]

Rolling out a new tool or new features to millions of customers could be a risky venture. Fortunately, we have tools and techniques that help us to decrease the risk:

- We use Feature Toggles to easily turn features on or off in production or to make features available only on the Wayfair internal network.
- We use Supplier Toggles so we can turn a feature on for suppliers who want to contribute to making our tools even better. Our suppliers are eager to try new tools and features that allow them to boost their sales, so they are happy to participate in beta tests of some of the features and provide us with valuable feedback.
- Wayfair helps people around the world find the best items for their homes. We can take advantage of this geographical diversity by, for example, releasing a feature for the European audience first and then, based on the results, releasing it to the North American audience (or the other way around). We can do that by using Deploy-level Controls (servers).
- We conduct A/B Testing. We give one version of the feature to one group of our users and the other version to another group. Then we measure the difference in performance and collect feedback.

We Constantly Monitor Our Applications in Production



Slow and "buggy" website pages can be very costly and are a bad experience for customers. We always keep an eye on key metrics for our applications, such as usage, performance, system health, errors, etc. This is even more important if we release something “big” to our customers. We also communicate closely with a support team to gather feedback and complaints faster.

If Something Goes Wrong, We Rollback the Changes

We have infrastructure that allows us to easily rollback deployments or shut off code at a feature level, so we can easily remove or "hide" features if a large-impact-defect is identified after deployment. This can be achieved by using the Feature Toggles, rolling back changes, or delivering a hotfix in seconds.

Optimizing for a sense of urgency in getting code to our customers increases our risk of releasing the occasional bug. We accept this risk, balance it with the criticality of the system under development, and fix released faults in the next planned release.

A final note—we did not transition to our current team structure (without embedded Quality Engineers within the scrum teams) in one day. First, we had to get our team members test-infected (in a good way), inspired by good role models from adjacent teams, and engaged in the test automation process. Only then could we cultivate the culture in which the whole team is responsible for test automation and quality in general.

Now, we empower autonomous teams across Catalog Engineering to deliver value rapidly, repeatedly, and reliably through building tools for testing, defining standards and best practices, and training teams on test automation.

About the author

Evgeny Tkachenko

Evgeny Tkachenko has more than ten years of experience in test management, automation testing, and release management, mainly on complex projects in the telecommunication, financial, and online entertainment industries. Previously his experience as a QE Manager in Russia’s most important technology center gave Evgeny a unique perspective in test process development. Now at Wayfair, he applies his experience and expertise to do whatever is required to deliver the highest quality product possible. Evgeny is a frequent speaker at QA/Dev/Agile conferences in the US and Europe, and an author for international test and QA publications.