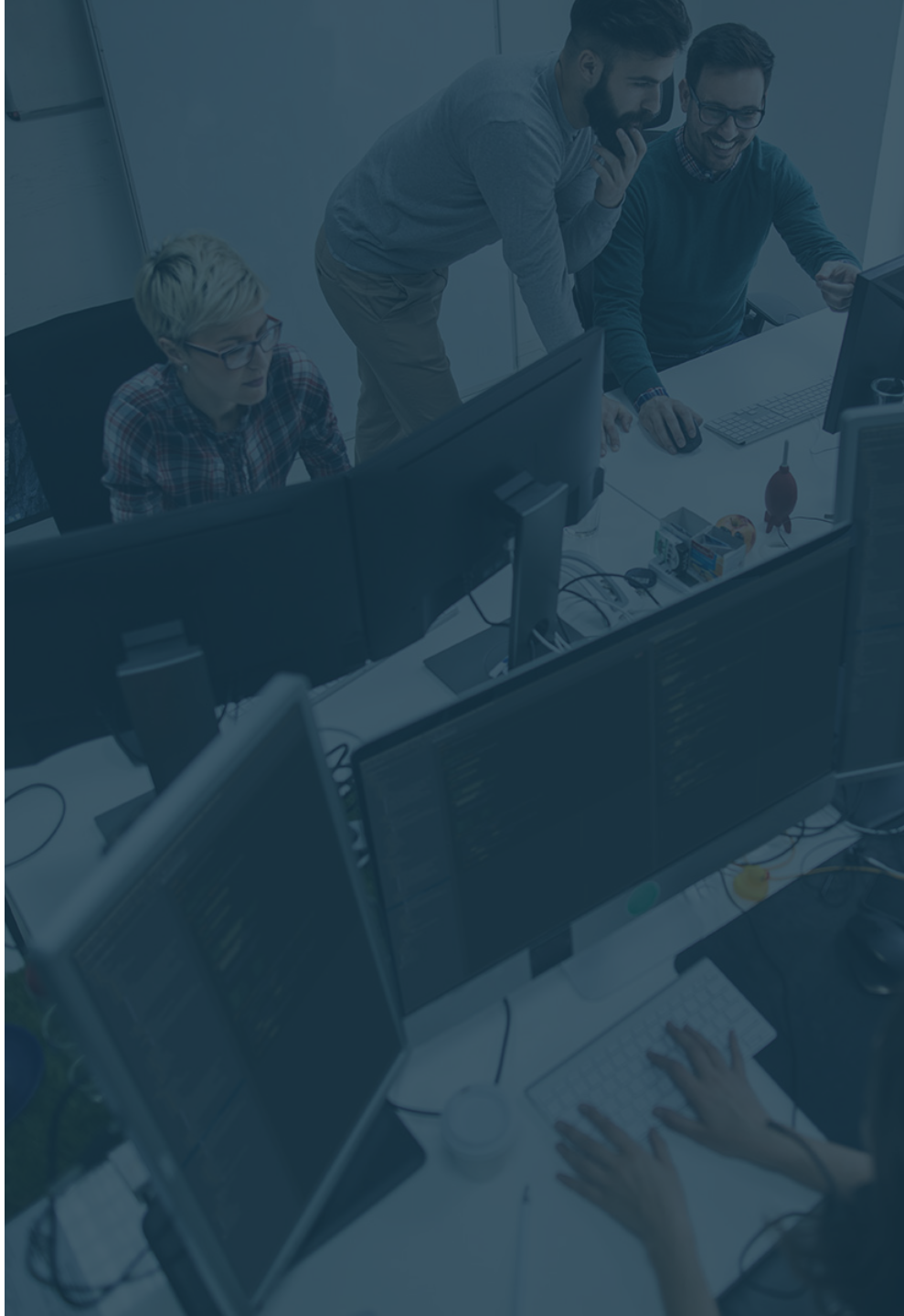
5 Fundamentals

to Building a Successful Test Management Strategy

SMARTBEAR

There are many facets of test management and we're going to look at just a few important aspects. Test management involves keeping track of your test strategy and planning, test data, test reporting, automation, and collaboration within the team. After reviewing these areas, you can look at ways to improve your current test management and look for ways to continuously improve going forward.



5 Fundamentals to Building a Successful Test Management Strategy

1 Testing Strategy and Planning 4

2 Test Data 6

3 Test Reporting 7

4 Automation 9

5 Collaboration 10

Evolution of your Test Management Process 11

Testing Strategy and Planning

Let's start with a test strategy. A test strategy is how you approach doing the actual testing for your systems in order to achieve a confidence that the product is ready for release. The reason you want to look at having a good test strategy is because you want to be able to find and fix risks in the product before releasing to your customers. The team will discuss how they want to track the testing being done as well as how to plan for the testing needed.

Test planning is where the team will determine the different types of testing they will do such as unit, functional, security, scalability, reliability, etc. During test planning, the team should also discuss how to track testing being done whether it's planned out test cases or ad-hoc sessions through exploratory testing and other methods.

Now your team may be using Jira, and if this is the case, you're likely aware of the way Jira tracks issues and work that the team is doing. If you install the Zephyr for Jira add-on, Jira becomes much more capable by giving you more detailed information about the testing work and tracking your testing progress. Zephyr provides a seamless integration that is native to Jira – enabling you to stay in the same environment you're already working in, and also eliminating the need for multiple tools to track testing and test cases. Consolidating everything into one native platform extends your already existing functionality and reduces the need for keeping track of multiple tools and testing activities.

Zephyr has features that allow you and your team to track bugs and their fixes. You can also use the multiple reports and dashboards available to track the amount of time that testing, especially the setup and amount of time spent on bugs, is taking in order to evaluate where automation could be helpful *(Fig. 1 & 2)*.

The test planning that your team decides to do may also depend on the development style that your development team is using. Say your development team is using a Scrum style and iterations are expected to be completed within two week sprints. Your test planning will need to take into account the testing that you can do within those two weeks that will also be appropriate for test coverage. However, if the development team is working in a more Kanban style, you'll need to look at how much testing you need to do in order to get that one story or feature completed. While most of your planning may be similar between these two styles of devel-

opment, there may be some things that need to be adjusted such as integration testing as well as when and how to do some performance testing. You'll also need to look at how much testing the development team is doing in order to supplement the testing that you are planning. The goal here is to try to avoid duplication of effort if possible.

It is important to note that your test strategy can change frequently. If you're testing a new piece of functionality in your system and discover a new behavior, you might change your focus on what you've learned from the application. This may change your initial test strategy or test plan, but it can also help you identify areas of risk that may need to be tested. From what you've tested in the past and what you've planned and experienced will impact the test strategies and test plans you create in the future. If your team is currently holding retrospectives at the end of each iteration/sprint, this may be a good time to review the test strategy and work with the team to discuss improvements that can be made.

Fig. 1

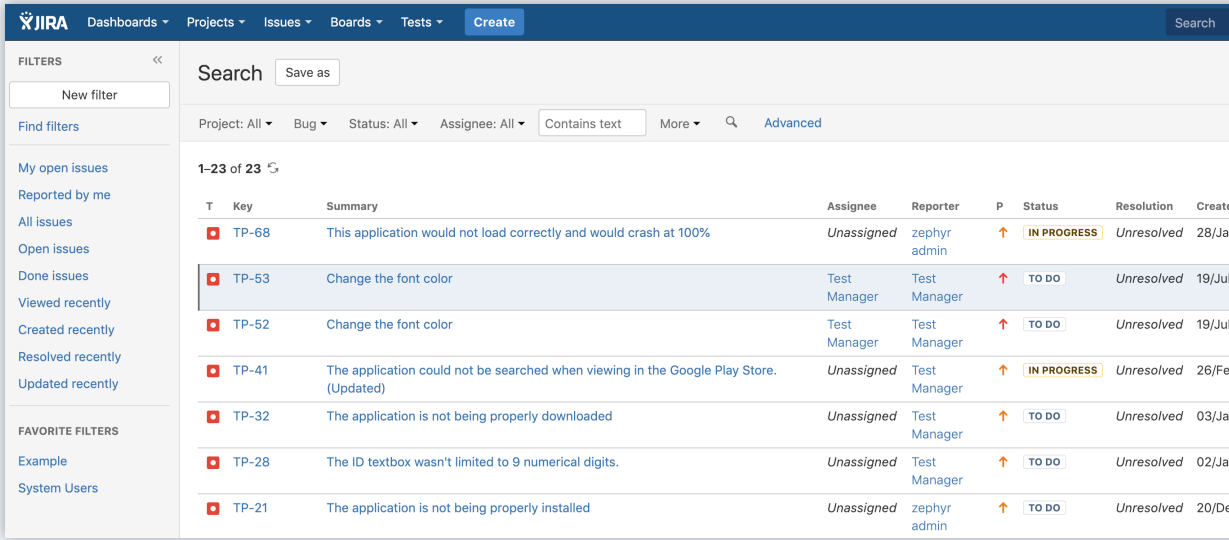
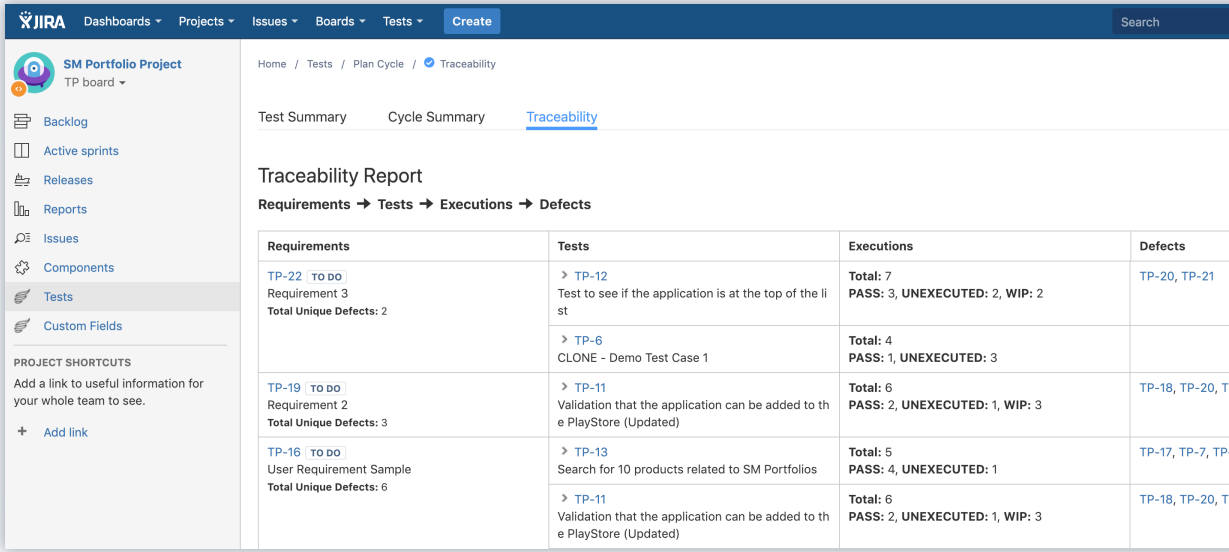


Fig. 2



Test Data

Test data management is crucial to effective testing. While developers generally have the luxury of creating whatever test data they want to use inside of their unit tests, the testing that is occurring in the test environment takes more care. When unit tests are written, they usually don't insert data into a database, but if they do, the developers have the ability to remove that data as part of their unit test cleanup. Developers can setup and remove data within their unit tests and therefore that data doesn't have to be cleaned up for different states within the system. Test data used in the testing environment generally needs to be thought of more carefully and managed appropriately.

Take for instance, creating one million customers within the system under test to see how the system performs, where that system generally has one or two customers inserted each day. This test data, that has now bloated the database, could cause issues for other tests that are done later. Although this could help identify a performance risk in the application, and should be tested to some degree, it's also using the system in a potentially unrealistic way. Sometimes, tests are created that have symbols or special characters to see how the system behaves, and that data then ends up affecting a downstream system and causes potential problems for testing those systems.

These tests are great for integration testing to make sure these risks are found before a customer finds them in production, however, this data might not represent the production data. When the data does not mimic the production environment as close as possible, as well as the hardware and network pieces, the results observed when testing with this data will differ from what can happen in production and could miss possible bugs.

Fixing some of these problems with test data can be handled in a wide range of different ways. Here are a few examples that tend to be helpful for managing test data:

1. Have jobs that run frequently throughout the day or once during the night to cleanup test data and reset IDs
2. Have backups that restore the data into a known state and potentially add a set of data that mimics the data in production and have masking or obfuscation applied where necessary

Test Reporting

Reporting on the testing that the team has done can help deliver meaningful feedback to the team and management about what has been covered by that testing. Reports have the ability to help answer some of the following questions:

- What is the current status of testing?
- Which tests are passing or failing?
- What is the team focused on testing currently?
- Who is available for which testing tasks?
- How much time is left for testing?

Are there any bottlenecks or impediments to doing the testing?

These reports will be the most valuable if they can answer the above questions in a clear and concise manner. They can help the team identify areas of the product that still need more testing as well as present a view of all the testing activities that have been completed so far. While these reports may seem valuable at first, you'll need to make sure you have customized these reports to provide you valuable insights into your testing activities and team.

Zephyr helps you customize your reports and also gives you example reports and dashboards to get started with such as the "Top Defects Impacting Testing." This report gives a metric that provides

Fig. 3

Top Defects Impacting Testing

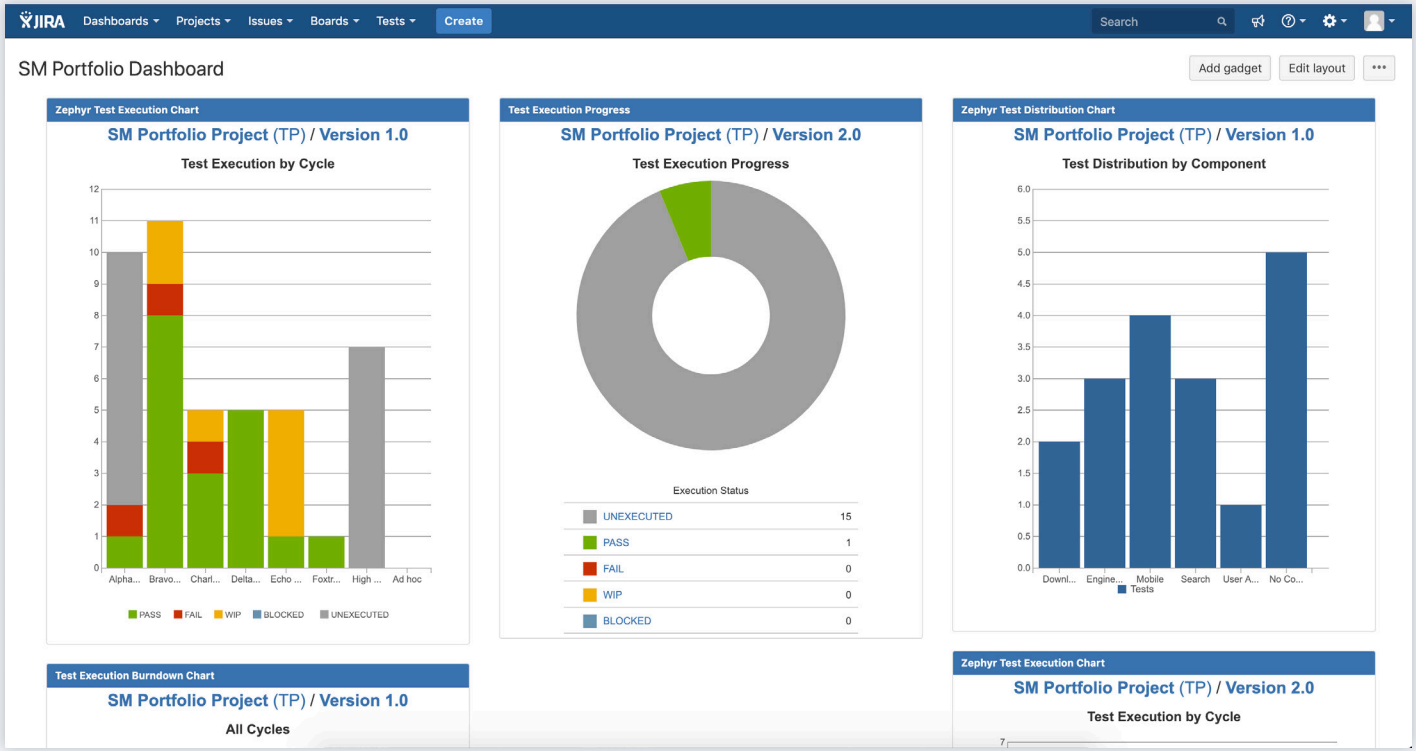
ZFJ Server (ZFJS) / **Release 4.0**

Top 5 Defects

Defect ID	Summary	Status	Tests Affected
ZFJS-13	Links are not working on IE 11 and Edge	To Do	2
ZFJS-14	Link Name should be Zephyr Custom Field and Not Custom Field	To Do	2
ZFJS-36	Execution failed due to missing parameters	To Do	1
ZFJS-48	Failed to Create Issue Link	To Do	1
ZFJS-94	Installation failed due to SQL error	To Do	1

information about which defects are currently holding up the maximum number of tests from passing *(Fig. 3)*. Essentially, tests that aren't able to be executed because those tests are blocked by a defect. The team can now focus on areas where defects are preventing more testing to be conducted. This is only one example of a test report that is available in Zephyr to help provide you and your team insights into your testing activities for making informed decisions moving forward. In order to answer some of the most important questions that were looked at above, a dashboard is a tool that can display these answers to the entire team. Zephyr includes dashboards that you can use and customize to show the important information quickly *(Fig. 4)*. In order to make a dashboard, you'll need to start with a gadget. Gadgets are pieces of information and statistics that are contained to be used across dashboards. Some useful gadgets are: project status, test case status, execution progress, open defects, and execution backlog. These gadgets are easy to customize and allow you to put the critical information into your dashboards that help you make the important decisions quickly.

Fig. 4



Automation

If you've been looking to scale your testing activities, automation may be something you're looking to or already have implemented. A common mistake however is when people focus on "What's the next thing to automate?", instead of taking a step back and looking at the big picture of automation. This includes being able to look at what automation you currently have as well as efficiency gains on writing automation where it is more helpful for the team.

Take, for instance, a system that sends emails out to customers. This system already has automation around checking the email content after it's been sent to a test account. The next few sprints of tasks for this system include being able to create communities that will get email notifications from this system. While the team may focus on how to automate managing these communities in a test environment, there may be more efficiency gained by creating automation that creates all of the email information for the tester so they can generate more of these email tests instead of having to craft them by hand.

While looking at making your automation more effective and efficient, it's important to be able to organize the manual test execution effort as well as the automation effort. Zephyr shows the manual tests along with the automation all in the same system. Being able to review these side-by-side can be helpful for identifying and removing manual execution that is taking more time than automation and needs to be run multiple times.

While we haven't covered all of the automation topics here, from a test management perspective, it is important to remember to prioritize the automation needed and look at where some of the biggest efficiencies can be gained. You might also want to take a look at the automation that you have running that is no longer providing value and removing it if possible.

Collaboration

A big part of the success of test management is how you and your team collaborate during the project. Though teams may succeed with little collaboration, it is generally easier to be more efficient with a team that collaborates together well. Collaboration is even part of the Agile Manifesto: "Individuals and interactions over processes and tools." This increases team morale and the ability to work better together.

HipTest is a product that offers a collaborative testing platform in the cloud that allows the team to create acceptance tests. The team can write and review scenarios using Behavior Driven Development (BDD). BDD is written in a syntax called Gherkin, which has a Given, When, Then structure. This structure allows the team to write requirements in a more natural language that is easy to learn by non-programmers, that can then be converted to automation to run a scenario against the system under test. The team run these scenarios inside the CI/CD pipeline as well.

Evolution of Your Test Management Process

As your test management process evolves, you'll need to review and reflect on changes that may be needed. A good time to do this may be during retrospectives where the team can give input on behaviors and/or tools need to change. Be sure to pick and choose tools that meet the needs of your team. Over time, your team will be able to experiment and learn about the test management process being used and make suggestions that will make the process better and more efficient going forward.

Software Testing **As Agile As *You* Need It to Be**

Spend less time testing and more time building. From agile development to predictive analytics, you'll achieve full Continuous Testing Agility.

Discover Zephyr


SMARTBEAR