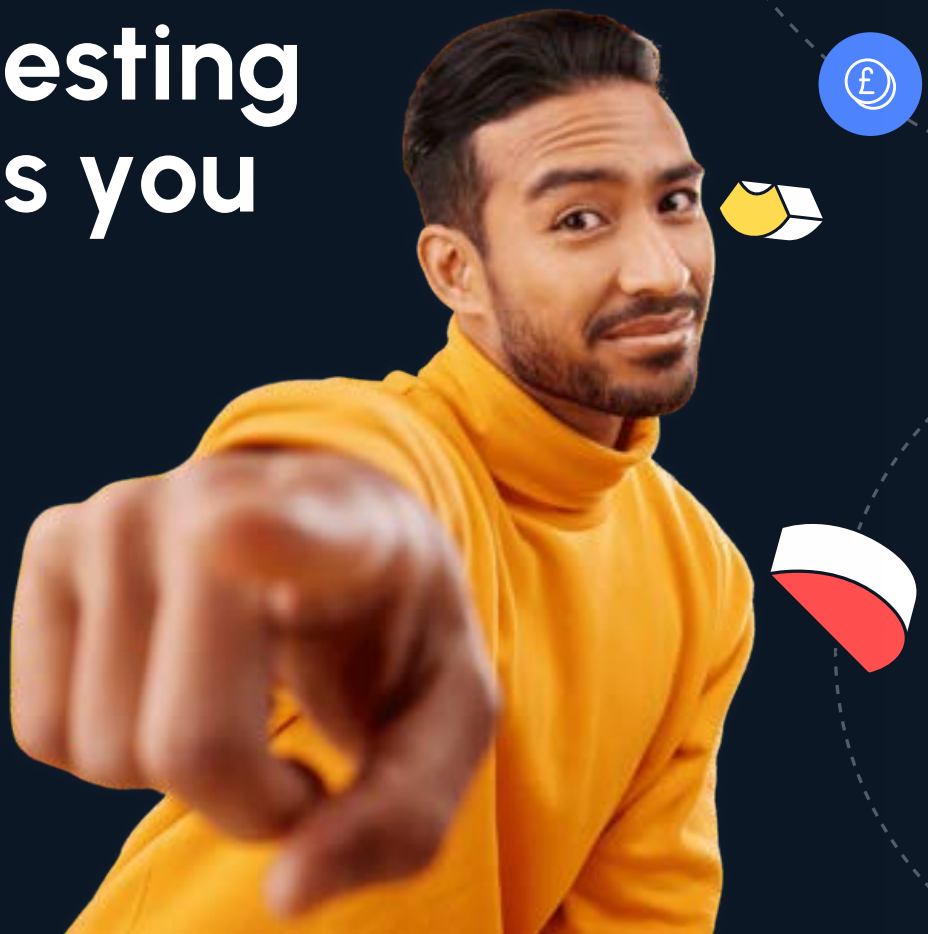
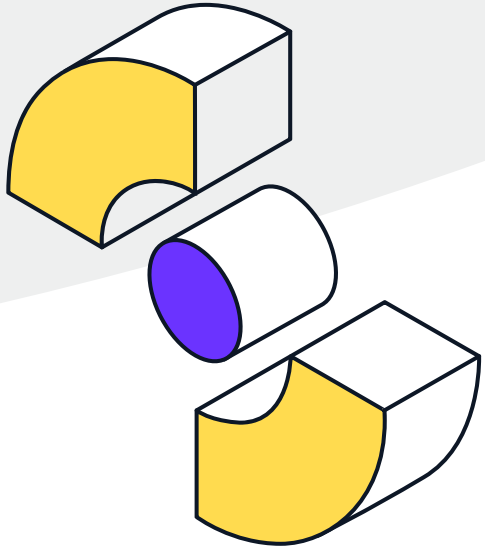
Original Software

10 software testing fundamentals you should know



eBook

Contents



01 Don't think about software. Think about business processes

02 Make sure you've completely captured all your business processes

03 Create and maintain test cases

04 Consider whether you need test data

05 Make tests easy for users to complete

06 Have a plan for when things get delayed

07 Think about how to work with your developers

08 Remember that 'bugs' can take many forms

09 You need an audit trail

10 What can you automate next time?

01

Don't think about software. Think about business processes

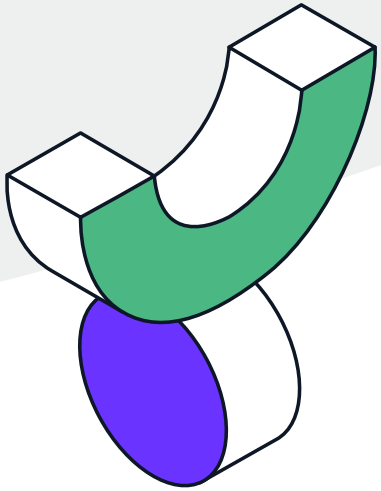
Though most of the time it's software updates you're testing, it's vital to remember that the software in your organization exists to support the various business processes you have. So your testing process isn't about asking "does this software work?" It's about asking "do our business processes still work with this new software?"

Remember: any software update released to you from a third party can generally be relied on to be functionally stable. You're not looking for bugs in the software as much as you're looking for bugs in your business processes as a result of the update.

The global testing market is projected to grow 50% by 2027

02

Make sure you've completely captured all your business processes



Can you list all the processes that your business has? If so, how sure are you that you've got all of them?

The point is that if you accidentally miss a business process that a software update affects, you run the risk of letting a business-breaking bug through into production. There's no real way around this; you need to spend time with every part of your organization, documenting the various processes they have – and the iterations of them.

For instance, your shipping process might follow different steps if the product is being shipped internationally instead of domestically – you need to check both versions of that process.

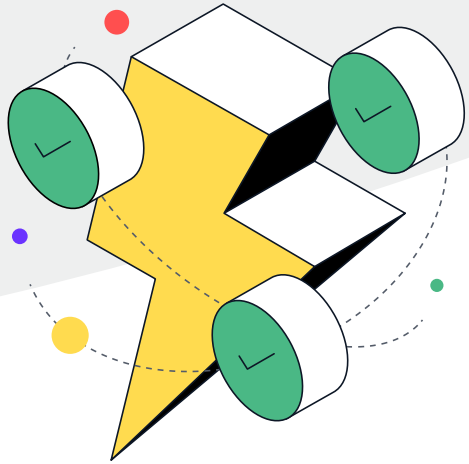
88%
of respondents saying UAT is key to achieving quality objectives*

Less than 50%
of people reported being provided with any tools to support UAT*

*Source: Original Software User Acceptance Testing Survey Report 2016

03

Create and maintain test cases



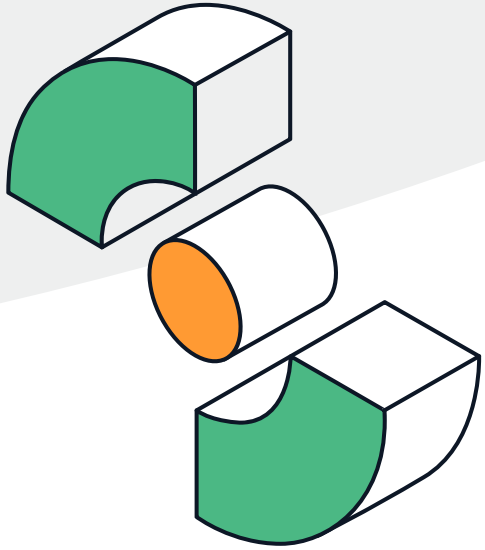
A test case is a set of instructions that you can give to a manual tester that enable them to properly test a business process. They drive most modern testing efforts, letting you know if you've got the right level of test coverage, and how your testing is progressing.

How detailed a test case needs to be is a topic of some debate. It's not uncommon to see long spreadsheets with individual steps saying "click here", "type this into that field," and so on – with check boxes to validate every single step. That's comprehensive, but it's painful to build, even more painful to update if the process changes – and chances are that the business users who are following the test case already know the process in detail anyway.

Modern testing tools often enable you to capture business processes through screen recordings and screenshots, which can then be easily converted into test cases that don't go overkill on the detail, but still give your tester enough specificity to do a good job.

04

Consider whether you need test data

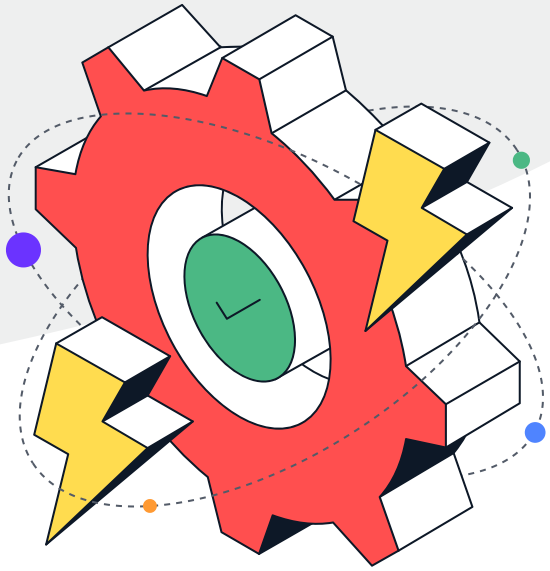


Often, as well as testing the interfaces that business users are working with, you also need to test that data entered is processed properly. In that case, you'll likely want to create specific test data for your testing.

The advantage of using test data that you've created is that you'll be able to describe exactly what should happen to it (discounts should be applied, orders should be sent to X, Y and Z applications, and so on). This makes it easier for testers to validate that everything has happened as it should.

05

Make tests easy for users to complete



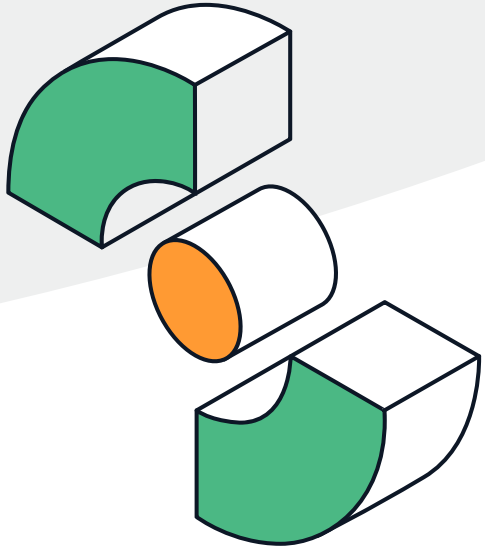
Sounds obvious, but it often gets forgotten in the drive to have comprehensive test cases and detailed tracking. Especially if you're working in spreadsheets.

The key here is to look for alternatives to spreadsheets. Our platform, for instance, passively monitors what users are doing as they run through a process, recording every step unobtrusively until the user wants to leave feedback – at which point all they have to do is click on the issue and explain it.

The easier testing is for users to do, the faster and better it will get done.

06

Have a plan for when things get delayed

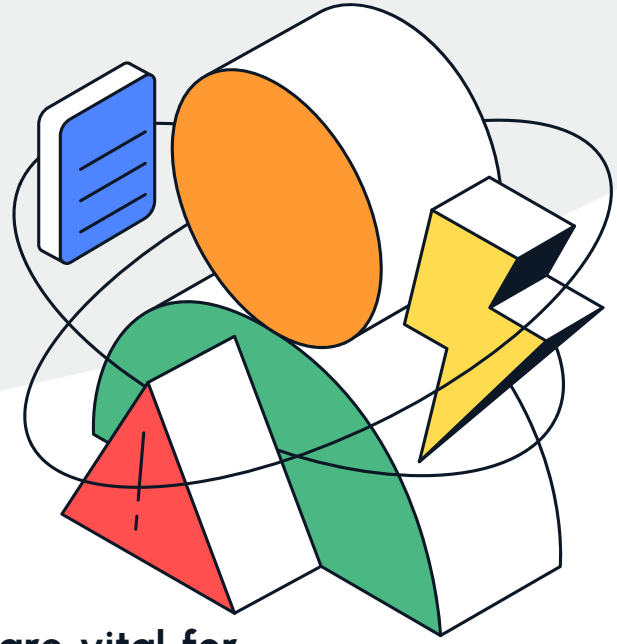


Things will get delayed. Someone will be ill; an urgent project will come in that's more urgent than your testing (outrageous!) No matter how well you plan, things like this will crop up. Knowing what to do about them will help you ride those waves more easily.

For instance, just having a priority list of tests will help you make sure that, when delays happen, the vital stuff doesn't slip.

07

Think about how to work with your developers



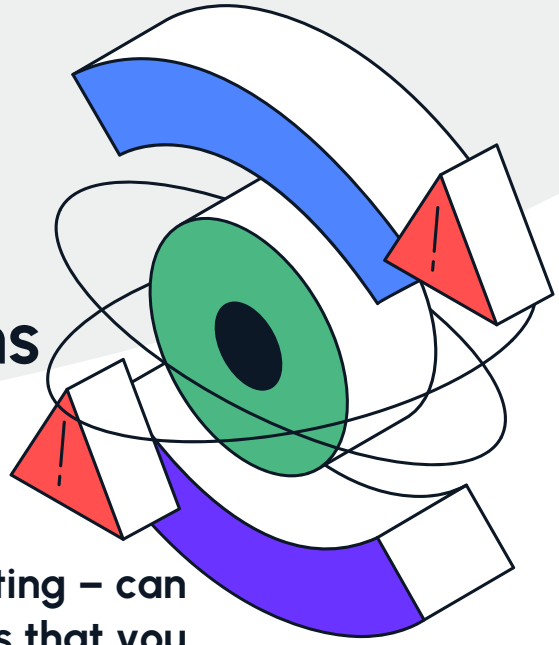
Technically, software development isn't testing – but they are vital for completing your software testing on time. After all, your developers are going to fix any bugs your users discover during testing.

The key here is giving testers enough context that they can replicate an issue themselves. A huge source of delay in testing is developers having to go back to testers to get them to replicate the issue- especially if you're using offshore developer resources working in different time zones to your business users. Screenshots and detailed comments will help here.

Is there a way you can integrate your testing processes with your developers' tools – automatically raising issues in their environment, for instance?

08

Remember that 'bugs' can take many forms



Functional issues are obviously one of the main foci for your testing – can users do their jobs? But those aren't the only issues. Other things that you could look out for include:

- Performance-related issues. Does the software work, but very slowly? If so, that could affect how quickly your organization can work.
- Cosmetic issues such as spelling errors, incorrect imagery, and so on. Though they might not stop your business working, they can make the process painful for users – everyone deserves great software.

You need to decide how you're going to identify those issues (if at all) and pass them to your developers just like functional issues. It's another area where specialist testing tools can help – especially for analyzing system performance. Also consider whether you will triage your issues before passing them to developers. Triage adds to your workload, but does allow you to prioritize the order in which your developers deal with bugs.

09

You need an audit trail

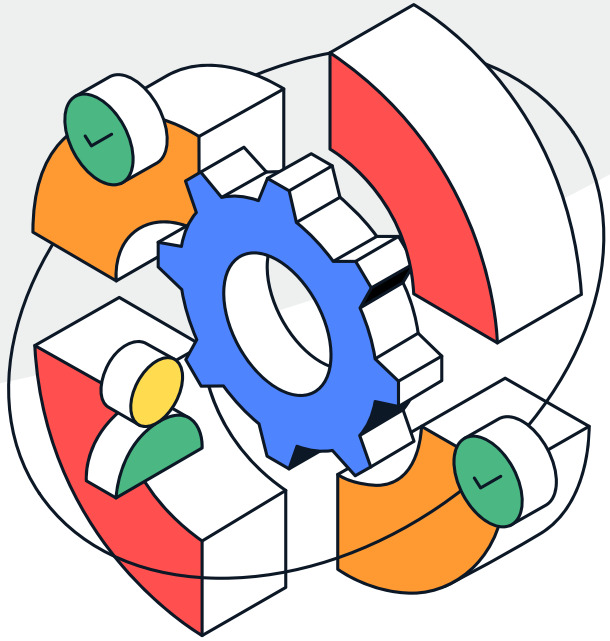
Even if you're not in a regulated industry like food and beverages, defense and banking, an audit trail is vital for a few reasons.

First, it allows you to perform root cause analysis if any bugs get through into production. Understanding where things went wrong will help you improve your processes, test cases, or testing techniques to prevent it happening again.

Secondly, an audit trail gives you a way to demonstrate to other areas of the business that you're doing good work. You can show them exactly what tests have been done, when, and what happened. If your organization is new to testing, this could well help stakeholders see the value of testing.

10

What can you automate next time?

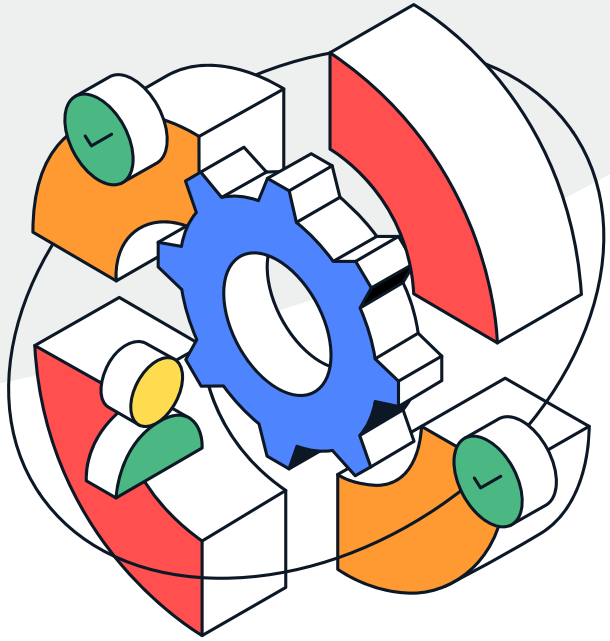


Test automation might be considered quite an advanced skill by some, but with the right tools automation can be done with no technical skills at all. If you already use automation, then you'll have been thinking about this from the very start of your testing. We've put it at the end of the list though, as many people will have to do a round of manual testing first before they can identify which elements of testing can be automated.

- Performance-related issues. Does the software work, but very slowly? If so, that could affect how quickly your organization can work.
- Cosmetic issues such as spelling errors, incorrect imagery, and so on. Though they might not stop your business working, they can make the process painful for users – everyone deserves great software.

10

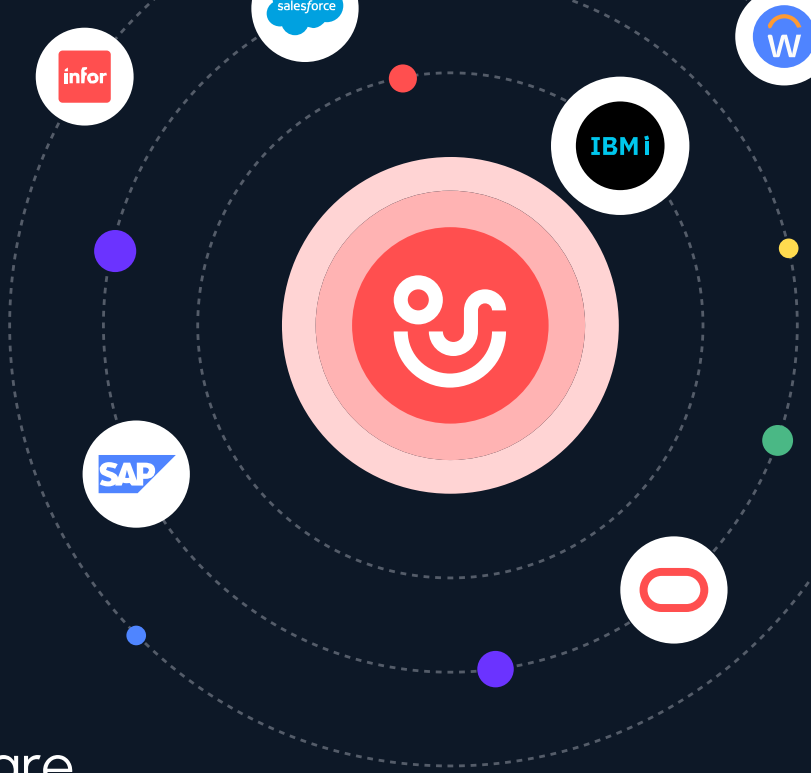
What can you automate next time?



You need to decide how you're going to identify those issues (if at all) and pass them to your developers just like functional issues. It's another area where specialist testing tools can help – especially for analyzing system performance.

Also consider whether you will triage your issues before passing them to developers. Triage adds to your workload, but does allow you to prioritize the order in which your developers deal with bugs.

About Original Software



Founded in 1996, our truly code-free software products cover all aspects of end-to-end application testing for ERP systems and beyond.

originalsoftware.com

Original Software