Panaya
Making ERP Easy

25 SAP Testing Secrets: Do's, Don'ts, and Customer Lessons

SAP customers and experts reveal their practical tips for testing

Whether you're embarking on a major upgrade, implementing an Enhancement Pack or simply performing routine maintenance, you need to get involved with an SAP chore that pretty much nobody looks forward to: testing.

In many ways testing is a thankless task, and often stands in the way of a go-live date that the business is likely looking forward to. But when companies shirk on testing or try to take short cuts, the result can be disastrous, causing delayed rollouts and potentially costly errors in production applications. So, SAP experts advise, it's best to learn to embrace testing and do a thorough job of it.

"What you're really trying to do is prevent code from going into production before it's ready," says Dennis Lake, a director with Lake Consulting Services who is currently working full-time as a project manager with the Johnson & Johnson Consumer Products Division. With the help of Lake and other experts, we offer these 25 tips for ensuring none of your SAP code goes into production before its time.

1. Conduct resource planning

The first step in test planning is figuring out what resources you need, says Nik Roy, an independent consultant who until recently was SAP Automation Test Architect for Wal-Mart Stores, Inc. These include IT resources, in terms of both infrastructure and personnel, as well as SAP functional experts from the end user community.

2. Identify stakeholders

Having stakeholders on the business side that you can routinely communicate with through the testing process is critical, Roy says. Stakeholders can help deal with the inevitable change requests in a timely fashion, ensure appropriate end user involvement, and generally act as a go-to person for any issues that may crop up requiring input from the business.

3. Communicate on requirements

Before you can design a test you have to establish what needs to be tested. That is likely more complicated than it may seem, given a change in one functional area can affect other areas. This is where IT has to work with stakeholders in the business, who in turn must communicate with various business managers and end users to determine how various changes may affect them.

4. Understand end-to-end business flows.

Many SAP transactions touch a number of different SAP functional modules and flow among several different business departments. A crucial element to proper testing is understanding the business flows involved in each business process, Roy says. Consider the task of issuing an expense reimbursement

check. Employee master data is in HR, payroll data comes from Finance and expense data likely comes from the employee's business unit. To properly test that process, you need to understand how the data flows from one area to another, where it starts and ends, and what data is required along the way.

5. Identify all use cases

Dozens if not hundreds of SAP use cases, like the reimbursement check example, likely exist in any given company. That means IT must work with business units to identify them all and come up with a test for each, says Matthew Stultz, an independent SAP consultant who until early this year was Vice President of Global IT at Newell Rubbermaid. It's not enough to test specific processes, such as an order process or procurement order. Think about how to test use cases such as when a major retailer places an order that qualifies for a volume rebate. Pay special attention to use cases that carry a potentially costly tab if something goes wrong, Stultz says, such as if you're a wholesaler and a retailer can fine you for a missed deadline.

6. Perform a reality check

Tests on end-to-end processes can get complicated, so Johnson & Johnson has checks and balances in place to ensure its tests are well designed, Lake says. "If I write a test, I'll run it past a key person, maybe a super user, to ensure it describes what's required to happen," he says. "And someone else has to execute the test and get business folks to sign off on it."

7. Hold back on specifics

To limit the sheer number of tests he has to develop, Lake says he'll often design a test that leaves out certain variables, so the same generic test can be used with different data to address different use cases. For example, a test may be designed to test the process of issuing a purchase order along with a receipt and to display inventory. A global company may want to process the same test using different international currencies, and perhaps different units of measurement, such as pallets vs. cases. All those variables can be entered into a data sheet and plugged in to the same basic test as needed.

8. Don't test what you're not using

Many companies have developed lots of custom code over the years and figure they'll have to test all of it whenever they make significant changes. Not so fast, Stultz says. "You may think you have 100 things to worry about, but 75 of them aren't being used," he says. At Newell Rubbermaid, Stultz employed a service offering from Panaya that examined all of the company's SAP code and determined that much of it had been dormant for months. "They came back and said we didn't need to test 90% of what we thought we had to because nobody's using it and haven't used it in 9 months," Stultz says. "If you turned it off today, nobody's going to know the difference." At the same time, the Panaya service made it clear which code the company did have to test, thus saving weeks or months of testing time.

9. Get good test data

One key to running accurate tests is having good test data to work with, Roy says, and enough of it. This is a stage that requires attention early in the process, to establish with the business what data will provide an accurate representation of the production environment. It's not enough to simply use the same data employed in the development environment, which is typically a handful of records, Lake says. And at Johnson & Johnson, the goal is to use test data that's less than 9 months old, so it's an accurate representation of the current environment.

10. Use a testing tool

Various sorts of testing tools are available to help with everything from test design, requirements, planning, managing errors and documentation. Some, like Panaya's are offered on a software-as-a-service basis, making them easy to employ. Others, such as HP Quality Center and HP LoadRunner, run on the customer premises but integrate well with SAP Solution Manager, Roy says.

11. Don't skimp on integrated testing

In a typical development environment, companies conduct discreet testing of an individual process, whether it's order processing or purchasing. In the test environment, the focus should be on integrated testing, involving various processes, Lake says. Maybe the sales team creates a sales order that gets passed to manufacturing and into the supply chain and ultimately to the finance team. "You want to see that everything works together," he says. "You'll find sometimes things that were changed create conflicts with another process."

12. Get end users involved

Experts agree that end users should not only be involved in testing, but on a dedicated basis. "The business has to dedicate their best people to the project and they have to be pulled away from their normal jobs," Stultz says. Part-time testing doesn't work, because the employees aren't focused enough and will be worried about taking care of their day-to-day work. The business may well push back on losing their best people for the 3 or 4 weeks required for testing, but think about how much is being invested in the SAP implementation; it's a lot more than it would cost to back-fill those employees temporarily, Stultz says.

13. Keep a cohesive team

Many companies try to dedicate resources just half time to testing, but Lake says that's a mistake. "The folks involved in development need to carry the project all the way through to go-live, and do troubleshooting for a month after that," he says. "Without that kind of consistently, you increase your risk. As someone goes off to another assignment, people will start making assumptions that are wrong."

14. Put it under stress

Part of the end-to-end testing cycle, Roy says, is ensuring the SAP applications can handle the load they will be under in the production environment. That means stress testing for both the hardware on which the applications run and the applications themselves, to ensure they execute within acceptable timeframes. "You need a separate environment for load testing, a scalable environment," he says, along with sizing tools such as SAP Quick Sizer, which can help you determine hardware requirements.

15. Make changes only in dev

The testing process will inevitably uncover issues that will require some re-writing of code, but don't be tempted to make such changes in the test environment. "Configuration changes need to be made in development and then transported across to the test environment to ensure you're testing things that are exactly equal," Lake says. Otherwise, the code you ultimately transport to production will be out of sync with the code that exists in development. And after you make a change, before testing again you need to re-authenticate the test, to ensure it's still valid with the changed code, he says.

16. Be prepared to re-test

Whenever you need to make a change as a result of a failed test, be mindful that it may have far-reaching repercussions. Some form of change control board should monitor the changes and determine whether the change will impact other tests currently underway as well as those that have already completed. "You may need to retest things that were already approved," Lake says.

17. Don't change the test

Once a test has been authorized and approved, resist the temptation to make changes to it midstream, he says. It's important that the test be run the same way for any applications to which it applies. Any changes you make may inadvertently skew results.

18. Budget lots of time

Estimating how much time you'll need for testing requires experience but Stultz's rule of thumb is to budget at least as much time for testing as for development. "People who think they're going to get away with four weeks of integration testing after three months of developed are doomed," he says. "You're not going to have time to react when something comes up in testing." That's especially true because it takes about three times longer to correct issues identified in testing as compared to original development time, he says.

19. Leave time for transport

Companies routinely budget a week or 10 days to move code from testing to production just prior to go-live, but they often fail to leave time to move code from development to testing, Lake warns. "You need to have a system available and ensure it matches what's in dev, and all the transports have to

occur, you need all the test cases written and have user IDs in place," he says. "There's a lot more to getting the test environment ready than meets the eye."

20. Start testing early

You don't necessarily have to wait until all development is complete to begin testing, however. "You can start testing baseline code before development is technically over," Stultz says. That's a strategy that Nike successfully employed when he worked there, and he's used it ever since. "That was a kind of secret sauce. I took that to Home Depot and to Newell and never had a bad go-live because of it."

21. Don't surprise users

When dealing with an upgrade, keep in mind that if it involves major user interface changes that your testing plan should include lots of end user involvement and training, Stultz says. "The whole key when you do an upgrade is, don't surprise your users," he says.

22. Use tools to document

Most companies want to document the testing process, to keep track of what's been tested and what remains. While it's possible to keep track using spreadsheets, larger environments will likely want to invest in a testing tool like HP Quality Center that automates the process. "It's very well integrated with SAP tools," says Roy, who has been using the tool for 7 years. It keeps track of every keystroke and screen shot. "It really helps us to cut testing time and we can figure out how SAP is behaving," he says. Another option is Panaya's Test Management software-as-a-service offering, which helps customers run test scripts far faster than they could manually – up to 150 scripts per day - and automatically log the results. The service also integrates with HP Quality Center, enabling the import of text scripts from Panaya to help build test libraries and the export of scripts from Quality Center to Panaya, to determine test script coverage.

23. Prove results and compliance

By taking snapshots of test results, tools like HP Quality Center and Panaya Test Management can help companies prove they are in compliance with regulations such as Sarbanes Oxley. "You can prove you are following the rules and did the testing you needed to do," Lake says. Similarly, such tools can prove to the business side that their testing requirements have been met.

24. Turn testing documents into training

While lots of testing documents just gather dust, Stultz says they can be valuable if you can turn them into training documents. "If you can put together coherent, good, concise information that an end user can use at some point in his career, then you've done a good job," he says. For example, one manufacturing application he worked with showed in tests that about 8% of pre-production materials wound up as scrap. "That type of information should be built into the training documentation, so when someone runs the job and find they have 8% scrap, they'll know it makes sense."

25. Don't forget reporting requirements

For new SAP implementations, especially, you need to pay attention to the new reporting capabilities. To get the full value out of them you'll need to educate users on the SAP data model throughout the testing process. "When you migrate to SAP your whole data model changes. What you call a product changes, a distribution center, a division, a sales organization, it all changes," Stultz says. "If you can wave a magic wand and get them to understand why they need to spend time and investment on analytics up front and understand the whole data model, it would be an enormous win."

About Panaya

Panaya's software-as-a-service helps companies that use SAP or Oracle reduce 80% of their upgrade and testing risk and effort. Utilizing a cloud-based supercomputer, Panaya simulates the upcoming upgrade, automatically pinpointing which custom programs will break as a result of the upgrade and automatically fixing most of these problems.

Panaya's testing solutions dramatically expedite ERP testing and eliminate the need for manual test script maintenance. Seamlessly capturing business knowledge in the background, as users work with the ERP applications, Panaya automatically generates plain-English test scripts that are rapidly executed and continually self-adjust based on test results.

To learn more, or apply for a **free trial** visit http://www.panayainc.com/Request-a-Trial.html.