WHITE PAPER

Don’t Wing It; Test Plan It!

Best-Practice Tips for Implementing a Successful Testing Plan for Your Workforce Management Solution

KRONOS®

What is a Test Plan?

A test plan is a document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst other test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.[1]

When you implement a workforce management solution, you want it to operate smoothly and deliver an optimal end-user experience from day one. That's why system, integration, and user acceptance testing are such critical steps in the implementation process. Systematic validation of the software configuration, usability, and interoperability using realistic test cases will help to ensure that your system meets all specified requirements before it goes into live production. By employing rigorous testing practices for new installations, service packs, and upgrades, your organization will be better positioned to avoid unwanted surprises and costly delays that can derail your implementation.

What can you do to make sure your testing efforts minimize implementation risk and support a successful system rollout? It all starts with a comprehensive testing plan that outlines the strategy, resources, processes, tasks, and schedules involved in verifying that your workforce management system meets your established business requirements and acceptance criteria. This paper presents five best-practice tips for developing and executing a formal testing plan that will get your implementation off to a strong start and lay the foundation for early success.

1. Define the best testing approach based on your solution, depth of configuration, and available resources

When it comes to testing and certification of your workforce management solution, "winging it" is not a plan; it's a surefire way to put your implementation at risk. You need to take a structured approach to testing in order to pinpoint, prioritize, and fix issues in a timely and effective manner so your system implementation stays on track. There are a number of approaches you can take depending on your organization's resource availability, testing experience, and desired level of involvement. Common methodologies include:

Agile: Following the principles of agile software development, this methodology focuses on the iterative, incremental validation of software functionality and interoperability in which solutions to identified issues evolve through collaboration between self-organizing, cross-functional teams. This approach is better-suited to organizations with established testing processes and experienced resources that wish to be active participants in the software testing and certification process.

Waterfall: In a waterfall model, each testing phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if testing is on the right path and whether the project team is ready to continue. Because each phase has specific deliverables and a formal review process, the waterfall model is a highly structured and easy-to-understand approach well-suited to organizations with limited testing experience and few established processes.

Vendor-Specific: Based on industry best practices and real-world experience with customers, many solution providers have developed their own structured implementation methodologies that encompass testing and certification. Project managers and consultants will use these proven processes to guide customers through system, integration, and user acceptance testing to resolve all critical open issues and effectively prepare for solution deployment.

[1] International Software Testing Qualifications Board, "Standard Glossary of Terms Used in Software Testing, Version 2.3," March 28, 2014, 44, http://www.istqb.org/downloads/finish/20/145.html.

Testing trends evolve to meet the needs of the ever-changing software industry. While there's no right or wrong approach to testing, **it's important to adopt some kind of structured methodology** to help ensure a successful system rollout and high user adoption rates.

No matter which approach your organization chooses to adopt, the point is you need to follow some form of structured testing methodology to help ensure a smooth system rollout, a rewarding user experience, and maximum return on your solution investment.

2. Identify who will do the testing, when, and where

Your plan needs to identify who will do the actual testing, when it will start, and when it will end. Larger organizations may have a quality assurance (QA) department that can devote resources to hands-on testing. In other cases, it's critical that you identify the people — HR staff, managers, employees, and others — who will be using the system and involve them in the testing effort. For example, store managers should test a retailer's scheduling application. Frontline supervisors should test a manufacturer's time and attendance solution. HR managers or in-house recruiters should test a hiring solution. And employees should test data collection devices and self-service applications.

Make sure that your testers are familiar with your workforce management software before they start the testing and certification process. If they're not, sign them up for online or classroom training available through your solution provider, or register an internal resource for a train-the-trainer program. Your testers need to know how to use the software in order to determine if its key functionality meets established requirements.

It's also important that your testers are able to dedicate significant time to the testing effort. Pull them out of their day-to-day activities and put them in a lab setting — even if it's a makeshift one — in order to:

- Minimize distractions
- Reduce duplicate issues logged
- Encourage participants to learn from each other — during testing and once the system is live

Define daily testing hours to keep the project on schedule. Most people can focus intently on testing efforts for no more than five or six hours per day. If you push them beyond that, they are likely to lose focus and overlook key issues. Be sure to leave adequate time for issue fixes and tracking before the next testing day begins, to avoid process bottlenecks.

Furthermore, you'll need to set up a limited number of test environments — a collection of servers that closely mimic the production system — for running your test cases. Define the migration process for making configuration changes or issue fixes across these environments and add this to your testing plan.

According to the 2013-14 ISTQB® Effectiveness survey, in addition to Agile Testing, Performance Testing, Testing for Mobile Devices, and Usability Testing have emerged as the **most important topics** in the industry.[2]

"If you fail to plan, you are planning to fail."

Benjamin Franklin

3. Determine how you will track issues through to resolution

The purpose of testing is to identify and resolve any issues that will impact the quality of the user experience, the accuracy and efficiency of labor data processing, and the reliability of decision-making tools and reporting. To this end, you need to define Entry and Exit criteria and document these in your formal test plan. Definition of Entry and Exit criteria — whether at the project level or for each phase of testing — will help you determine whether you have performed enough testing to meet your established goals. Consider these examples:

Entry:

- Application is available and configured for test
- Test cases have been created and reviewed by the project team to ensure that they trace back to the requirements design
- Interfaces have been tested and data is available

Exit:

- All major logged issues have been resolved and retested
- If, for whatever reason, an issue cannot be fixed, the project team has put in place an agreed-upon plan to minimize impact
- All planned test cases have been executed
- Applications perform as designed

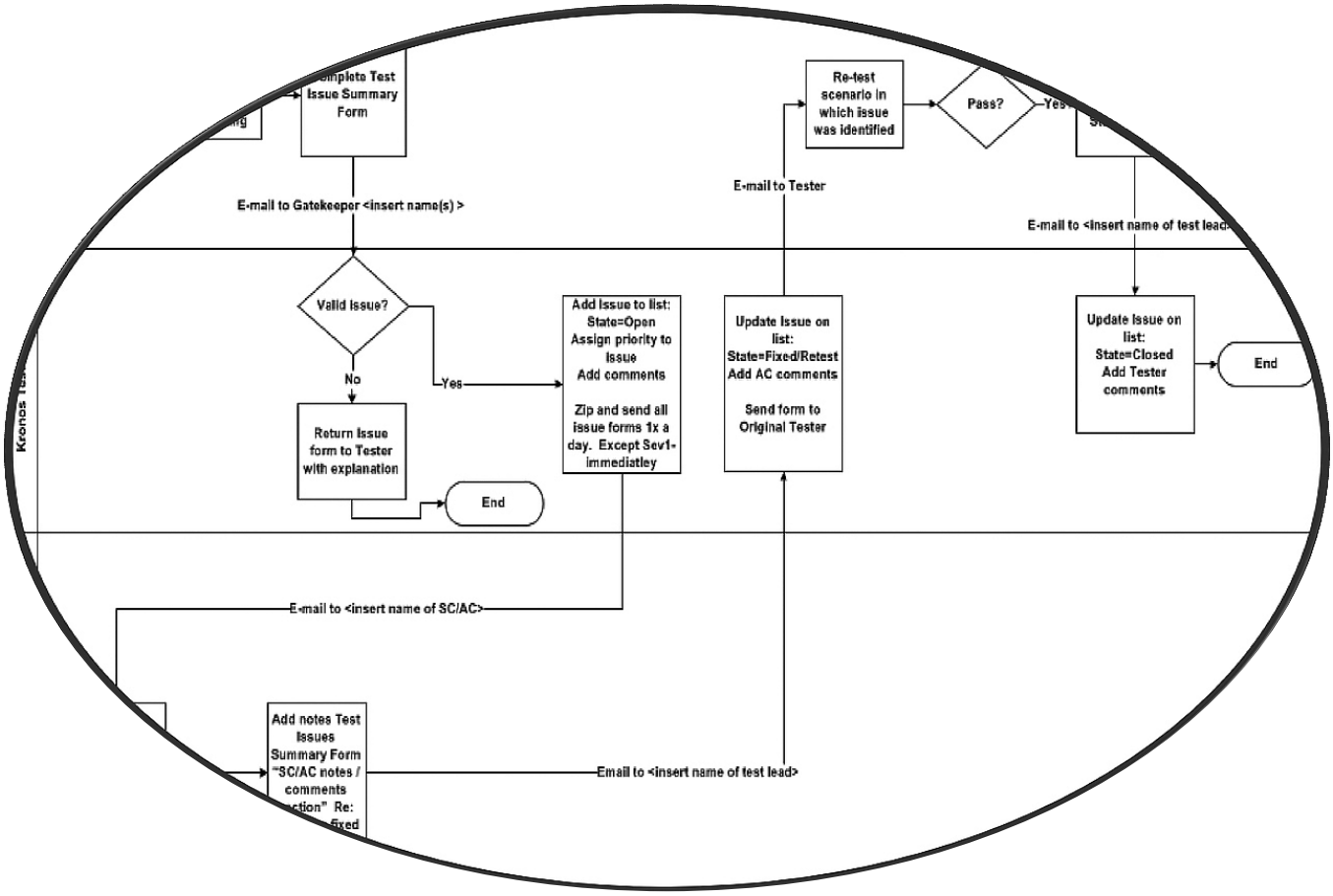
In addition, you need to consider how you will track any identified issues through to resolution. Some organizations use sophisticated test management software for project team communication, logging and assignment of issues, and scheduling of retesting once fixes have been made. Others rely on logging software or spreadsheets for issue tracking. Regardless of which tools you choose, it's important that you consider the following questions when defining your testing plan:

- How will you track different types of issues, including those associated with system configuration, data integration, and business processes?
- What happens when a tester finds a result that does not match the expected result? What process should he or she follow?
- Will testers log issues on a form or use your test management software?
- Once issues are logged, how will you track their status (new, pending, fixed, retest, etc.)?
- Who is responsible for gathering, reviewing, and prioritizing issues before assigning them to individuals to fix?
- What level of detail must testers provide for each logged issue in order to expedite the fix process? Will they be required to outline the steps needed to re-create the problem? Will they be asked to capture screenshots displaying error messages?
- Who will fix each type of issue, since testing best practices indicate that the same person should never test and fix issues? Who will give the tester the "all clear" to rerun the test cases associated with the issue fix?

[2] International Software Testing Qualifications Board, ISTQB Effectiveness Survey 2013-14 (ISTQB, 2014), 4.

- How often will you hold test review meetings to discuss issue status and remove roadblocks to resolution? Who is required to attend? Quick daily or weekly status review meetings — with testers, project managers, consultants, and other key team members in attendance — will help ensure that all test issues are resolved in a timely manner.

Bring it all together by diagramming your decisions



Issue prioritization is a key consideration because you want to focus your resources on those problems that 1) are blocking execution of multiple test cases; 2) will hamper performance of the production system; or 3) will negatively impact the end-user experience. A payroll calculation error, for example, will be assigned a higher priority than a typo on an employee self-service screen.

Knowledge of system requirements and business rules will enable you to better predict what the expected testing results will be. **A good test case checks that the system is meeting the established requirements and therefore is likely to uncover defects.**[3]

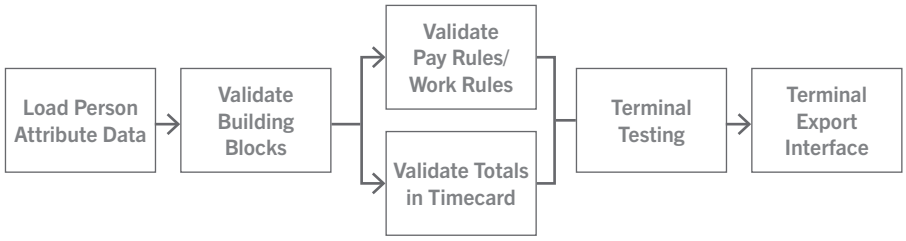
4. Identify the data needed to run your test cases and the optimal order of activities

Test case creation is an essential component of the system, integration, and user acceptance testing process. Test cases — sequences of steps designed to validate correct behavior of application features or functions — are developed based on your prioritized requirements and acceptance criteria. In addition to defining the sequence of steps to execute your test cases, you also need to identify and gather test data to use for each phase of testing. Your test plan should specify the data required for each test case, its source, and who is responsible for making that data available within your testing environment.

What types of data might your testers require? If a retailer is running a volume forecasting test case, for example, the testers will need point-of-sale (POS) data from the applicable store(s). If a manufacturer is testing its leave management functionality, testers will need punches (real or mocked up) to accrue over a period of time in order to validate that balances adjust the way they should. Similarly, if a healthcare organization is testing its payroll features, testers will need time and attendance data, along with employee benefit withholdings, tax information, and associated pay rules, to verify calculation of the perfect paycheck.

Because most workforce management solutions import data from other systems — ERP, POS, HR systems of record, and more — interface testing is equally important. You will need to run test cases to validate that all software interfaces are working properly. Integration testing helps to ensure each workforce management function receives the data it needs — in the right format and at the right frequency — for accurate processing and reporting once the system goes live.

If you are implementing or upgrading multiple workforce management applications, you'll need to identify dependencies across products and data types. For example, if you are implementing timekeeping, accrual, and scheduling solutions, you may decide to order your testing activities as follows:



[3] David Lile Brown, "The Five Essentials for Software Testing," accessed June 18, 2014, http://www.isixsigma.com/industries/software-it/five-essentials-software-testing.

Both your test strategy and testing plan are subject to change as the project evolves. **Modify your strategy first, if you need to, and then your testing plan.**[4]

5. Document your testing plan, review it with your project team, and make adjustments as needed over time

It's critical that you get your test plan down on paper and review it with the implementation project team so everyone understands his or her role in the testing process as well as associated processes and schedules. After all, the testing plan serves as your organization's roadmap throughout the testing and certification phase of implementation, detailing all activities that need to be completed for validation and acceptance. It also outlines the dependencies among the tasks to help you avoid missteps that may delay your implementation schedule.

Even the best test plans can be improved upon based on participant feedback and firsthand experience. Keep revisiting your plan throughout the testing and certification process and make adjustments as needed to improve efficiencies, drive quality and performance, and keep your implementation on the fast track to success.

Expert assistance is available if you need it

If your organization lacks the internal resources or QA expertise required to complete testing on your own, Kronos® Advanced Testing Services can help. Call **+1 800 225 1561** or visit **www.kronos.com/services** for more information.

[4] Ibid.

Sample Testing Checklist

Don't jump into system, integration, and user acceptance testing without proper planning. Take the time upfront to map out all tasks necessary for a successful testing and certification phase. Here's a list of items to build into your test planning and execution process to help ensure proper system functioning and a great end-user experience:

Test Team

- ☐ Create a test schedule
- ☐ Identify test team and span of dates during which resources will be required

Test Planning

- ☐ Create a Test Strategy and Test Plan
- ☐ Secure test environments
- ☐ Define the test approach
- ☐ Determine data needs and test groups (people, points of sale, census, etc.)
- ☐ Document issue tracking process, including problem prioritization, resolution, and retest
- ☐ Document change management process (scope items, environment process
- ☐ Document process for tracking daily tests to testing schedule
- ☐ Identify Entry and Exit criteria by phase or at project level
- ☐ Document roles and responsibilities
- ☐ Identify test team education needs and plan for training
- ☐ Identify location where testing will occur along with dates
- ☐ Document the order of testing activities and their dependencies
- ☐ Review Test Strategy and Test Plan with project team

Test Cases

- ☐ Obtain testing tools from software vendor
- ☐ Identify current business processes and procedures to be tested; include suite integration points as applicable
- ☐ Map Product Design documents to requirements and to the test cases that need to be developed
- ☐ Create test cases
- ☐ Schedule test case review meetings

Text Execution

- ☐ Assign test cases to testers according to order of activities and schedule
- ☐ Execute test cases and capture results via format agreed upon during test planning
- ☐ Document discrepancies as identified in the issue tracking process
- ☐ Conduct test issue review meetings at scheduled intervals
- ☐ Get formal sign-off on test cases per approval process
- ☐ Retest as necessary to ensure test cases execute properly