

WMS Software: The Importance of Testing

**By John Seidl
Partner, Tompkins Associates**

Tell someone to plan on testing a new WMS installation, and you'll probably get the same response that school children reserve for adults who deliver painfully obvious information: "Well, DUH!" Kids oversimplify things, however, and the person you were talking to might be thinking too quickly and too shallowly when it comes to taking new software for a test spin.

You can smooth your system implementation a lot if you and your consultant devote time and thought to coming up with a rigorous testing plan to make sure the bugs are out of the application and all its interfaces before you trust it with your business. What's more, if you invest time and effort to design and carry out a thorough, planned test of the software application in which you invested your money, you stand a good chance of building teams and finding talent that will continue to pay big benefits long after the implementation project is history.

Testing isn't specific to a brand of software or type of application. It is not limited to making sure software works, either. Part of the process is trying to make it break. These guidelines apply whether you're putting in a new WMS, moving transportation decisions to a TMS, or integrating your supply chain partners with an inter-enterprise application integration suite. They apply equally to code on your own servers as well as connections to an application service provider.

Start Testing

Start with an overview of the system and what it is expected to do for the business. Figure out the approach. It sets the scope of system testing, the overall strategy to be adopted, the activities to be completed, the general resources required and the methods and processes to be used. Once the overview is complete, move to a detailed design effort. During this effort, you define exactly what functions the system will use and those you won't. Additionally, you will define the modifications required, including functional specifications, and all interfaces to external systems. This effort will allow you to then define the amount of effort required to complete testing.

Then, move to the test planning stage, which details the activities, dependencies and effort that will be required to conduct the appropriate test, or tests. The next level is called test conditions or cases, which document the tests to be applied, the data to be processed, the automated testing coverage and the expected results.

Sound like a lot of work? It is, but it's worth it.

Here is an outline of a sample system test plan:

- I. Introduction
 - A. Overview of this new system
 - B. Formal review points
 1. Design documentation
 2. Testing approach
 3. Unit test plans
 4. Unit test conditions and results
 5. System test conditions
 6. Post-system test review
 - C. Objectives of system test
 - D. Software quality assurance involvement
- II. Scope and objectives
 - A. Scope of test approach – system functions
 1. Inclusions
 2. Exclusions
 - B. Testing scope

- 1. Functional testing
 - 2. Interface testing
 - 3. Acceptance testing
 - 4. Final acceptance testing
 - C. Testing process
 - D. System test entrance/exit criteria
- III. Test phases and cycles
 - A. Project integration test
 - B. Operations acceptance test
- IV. System test schedule
- V. Resources
 - A. Human
 - B. Hardware
 - C. Software
 - D. Test environments
 - E. Error measurement system
- VI. Roles and responsibilities
 - A. Management team
 - B. Testing team
 - C. Business team
 - D. Testing support team
 - E. External support team
- VII. Error Management
- VIII. Reviewing & status reporting
- IX. Issues/Risk/Assumptions
- X. Signoff

Check your assumptions before getting into the work itself. Get your process flow mapping down on paper, numbering each section and detailing procedural as well as system actions. You are starting the documentation package for the entire implementation. Everything you develop – test scripts, standard operating procedures, training materials, job aides and anything else you create or are given – should reference the proper section number on your flow mapping so you can verify all details in the test plan support the desired system configuration. The flow documentation becomes the reference map for changes or upgrades later.

Make sure you and your vendors get on the same page at the outset. Make it clear what you expect from the vendors and what you plan to do. On their side, software will be delivered on time, with modifications if required, and fully tested, with proper version control and all required quality. “Show-stopper” bugs are to get immediate attention from each vendor’s development team.

For your part, you’ll provide all required resources. No one will make incremental changes that add up to big differences in the project. Management has to approve all changes in scope.

People Are Important

Now, glance back Step VI in the outline. Circle it in red. Blow it up on your copier and put it on your wall. It is about people, people, people. Get a team together – in practice, not in name only – and do whatever is necessary to make sure everyone on it understands that this is their top priority. Arrange for the team meet weekly and make it clear that no one is to miss any meetings. Ongoing communication leads to clarification about terminology and thinking.

Assign a project leader for testing and give that person full support and the authority to accomplish the mission. Integrate the testing schedule into the master schedule for integrating the system. Identify critical path items and work out solutions early.

The testing project manager should have one eye on deliverables and tasks that are behind schedule. Set up weekly vendor conference calls with the full team. Discuss progress made during the preceding week and review what’s coming up over the next two weeks and problem areas you can foresee. Make the people who need to resolve those

issues schedule side meetings. The weekly meetings and conference calls breed familiarity, get people used to one another's styles and allow all groups to communicate openly and be part of the team.

Establish the testing team as early as possible. That gives the vendor time to train your people who will do the tests, and thinking about testing will give everyone a better understanding of the integration requirements with legacy systems, other systems you are implementing, and hardware. During heavy testing, meet daily to ensure that all parties are resolving the problems. This process is where you plant seeds to harvest after the implementation is over.

Expect to find people who can be "champions" in your organization, bright people who see the big picture intuitively, who can think cross-functionally. Let them shine in planning and carrying out the testing process. Look, too, for people who show aptitudes that can make them leaders in specific areas of your operation after the new system is in place. These people also should be considered as trainers for other end-users and resources for developing standard operating procedures, job aides and other collateral materials.

Facilities preparation can cause a lot of issues if not managed and can even cause the overall project to be late. IT staff has to be on the testing team. Don't surprise them with setting up and configuring hardware peripherals, arranging system access, or deciding what menu options should be available. IT has to buy in early and help with decisions about establishing the operating environment (testing, training, and production) and with trouble-shooting in general.

Testing takes resources. Establish a conference room as your command post for all testing, with all necessary equipment and supplies. Most software vendors require data lines for remote processing so their programmers and support personnel can access the system for debugging and downloading software patches and upgrades. This is faster and it's cheaper for you because they don't have to be on site. Make it happen very early in the process.

Establishing the Process

Right at the get-go, create definitions for the processes in your testing plan. People can put different meanings on even the simplest terms or use different words for the same idea, and you need to eliminate confusion, even about terms that seem simple. What, for example, do you bring home your groceries in, a bag or a sack? It depends on where you live. Make sure everyone's speaking the same language about the new system, about your operation, and about the testing.

Here are some examples of terms to pin down.

Functional Testing – The objective of this test is to ensure that each element of the application meets the functional requirements of the business as outlined in the design documentation.

Integration Testing – This test proves that all areas of the system interface with each other correctly and there are no gaps in the data flow. The final integration test proves that the system works as an integrated whole when all the fixes are complete.

Acceptance Test – This test ensures that the system operates in the manner expected and that any supporting material, such as procedures or forms, are accurate and

suitable for the purpose intended. It is high-level testing, ensuring that there are no gaps in the functionality.

Regression Testing – A regression test will be performed after the release of each object code. It means going back and doing exactly what you did before that detected a problem, watching to see that the problem has been fixed. If, for example, you enter five digits of a ZIP code during a test of a customer database and the software populates that field with only four digits, do it after the repair and see if you get five this time.

The level of detail in testing is a judgment call, balancing what you would do in an ideal situation with the expertise of the staff and time available. At any level, tie testing back to the process flow, ensuring that all processes were tested.

The amount of data you need for a thorough test is huge, and it will require a lot of thought to assemble it and resources to enter it. When you're collecting old data or inventing test data, make it as easy as you can to check how the system performs. If, for example, you're entering data about product size and a WMS is supposed to tell you how it cubed out, use numbers that make it easy for you to tell almost at a glance if the system got it right. Don't devise tests that require you to spend a day with your calculator checking the system's answers. That's not testing, that's torture.

When the data is ready, do a backup. That will allow for regression testing or volume testing. An error management system is a must and is the yardstick evaluating the software. Set up a spreadsheet to track errors and create categories from serious to minor. You will also want to track the status of each error, such as whether the vendor is correcting, it's ready for regression testing, or it's confirmed as fixed.

Meeting the challenges of software testing is a huge task and shouldn't be underestimated. Remind the team that failing a test is not bad. It shows good work on their part. Remember that most vendors don't warranty the modifications they make to customize your system, and it is your responsibility to find problems and have them corrected before you sign off on project completion.

And the last piece of advice is to keep your sense of humor. It will help get everyone through the process.

###

About Tompkins Associates

Tompkins Associates is a global leader in Total Operations consulting, information technology implementation and material handling integration. Tompkins combines its hands-on approach with over three decades of industry experience to provide warehousing, logistics, manufacturing, supply chain and organizational excellence solutions.

Tompkins Associates' corporate headquarters is in Raleigh, NC. Tompkins has offices throughout the US and in the UK, continental Europe, Canada, Mexico and Australia. Worldwide, Tompkins helps clients plan, design, execute and measure supply chain solutions focused on bottom-line results.

Visit www.tompkinsinc.com for more information.

IDII Thanks Tompkins Associates for use of this white paper.

For additional white papers, see www.idii.com/wp/index.htm and subscribe to our FREE Software Newsletter at www.idii.com/esn/index.htm. Our website at IDII contains many educational items, including useful sites, research, and books focus on supply chain software. See the IDII website at www.IDII.com