



## Testing... Testing... 123

*By Susan North, Consultant, Epic Practice, Vitalize Consulting Solutions, Inc.  
Thomas Lanphear, Consultant, Epic Practice, Vitalize Consulting Solutions, Inc.  
Ann Mendlowitz, Practice Director, Vitalize Consulting Solutions, Inc*

You've designed and validated the workflows, built master files, workqueues, and created category lists. At this point, you feel you can breathe a sigh of relief, but the work is just beginning! The key to a successful implementation lies in developing and executing a comprehensive testing plan. Testing helps to identify problems with the build, ensures the system is configured properly for documented workflows, and validates that desired results are achieved. Along with testing that the system is working properly across the installed suite of applications, it is critical that the testing plan includes exhaustive interface testing. Normally, a well executed test plan will reveal operational and/or integration issues which were not addressed during the design phase. The completion of a successful testing plan not only ensures the system is ready to go live; it also serves to prepare the project team for system support.

### **The Project Plan**

While drafting the master project plan for an Epic implementation, it is imperative to allow adequate time for thorough testing of your applications. On average allow 8-10 weeks for the various phases of testing. This will allow time for major issues/defects to be corrected.

After your project plan has been finalized, a testing project plan is critical to keep track of each phase of testing.

During each phase of testing, it is imperative that all testers document their results on a daily basis. Whether it is using an Excel spreadsheet or use of a documentation repository, documenting by each tester is critical to a well executed test plan. The amount of time required to allow for testing can vary due to the number of applications. For example, if you are going live on all of Epic's Revenue Cycle suite (Cadence®, Prelude®/ADT, Resolute® Hospital and Professional Billing, allow for 2-3 months for perform a thorough testing of your development environment. This will allow for correction of any deficiencies that may crop up during the testing. If this amount of time is entered into the master project plan, your go live date should not be impacted.

### **The Testing Team**

An important component of any testing plan includes careful consideration with respect to the composition of the testing team. An experienced testing Project Manager with Epic knowledge is essential to keep all testing phases in check and on time. Due to the complexity of the system, it is critical that those involved in testing, especially in the initial phases, have a broad understanding of each application. The ideal candidate would be a certified or proficient person who is also a subject matter expert. It is important to include the Physician Champion whenever possible. The most successful testing will be achieved when the team is primarily comprised of people who are certified or proficient in the various Epic applications. It takes an understanding of the system to know the expected outcome of each executed test. Certified consultants bring added value to any testing team. It is important to draw on their experiences from previous implementations. It requires so much more than just being able to read a script and click buttons to successfully test the system. As the testing progresses into its final stages, there will be an opportunity to add super-users and others to the testing team.

## **Testing Scripts**

The testing scripts should come directly from the documented workflows. Write the testing scenarios to accurately reflect day-to-day operations. The unit testing scripts will focus on each application independently. Each team will be responsible for writing and executing their plan. For integrated testing, the scripts should be written using a team approach. Each application team will need to contribute their expertise to ensure the integrated scripts are complete and accurate. Take a test patient all of the way through an office visit, beginning with registration and scheduling, continuing to check in and then documenting the office visit, test orders, meds, and referrals. Validate order transmittal is working as designed. It's also important to include the entire revenue cycle in testing. Check billing to make sure charges drop as expected. Ensure that test scripts are written to test all of the different WQ types (charge review, insurance follow up, claim edit, credit, account, HPF routing). Make sure paper claim production is tested along with electronic file generation.

Write the scripts to accurately represent what the various office visits look like in a real life setting. Think of every common scenario, but don't forget the uncommon ones as well. The more realistic the scenarios are written, the more successful the testing and ultimately the deployment will be. Control the length and content of the scripts by focusing on only one or two variables in a single script. If the testing scenario gets too complicated or lengthy, it's easy to lose sight of what is being validated.

Include set up steps, testing instructions, and the expected results in the scripts, to assist the testing team. This is especially beneficial when trainers and super users are involved in testing. They may not know the application as well as the application coordinator (AC). Have the team document the actual outcomes as well as any comments they may have. Make sure the testing team is using mock users rather than their own log in. As a suggestion, include the mock user's log in information as part of the testing set up instructions. Consider using the testing scripts as a starting point when developing a training plan. No sense in reinventing the wheel!

### **First phase of testing is Unit Testing.**

The testing should be done in phases, and each phase has certain dependencies. Initial testing will most likely start within each application, as sections of the required build are completed. This is known as application workflow or unit testing. The testing scripts should be derived from the documented workflows for each application and executed by the appropriate team. For example, the Cadence team will test the workflows that involve registering and scheduling a patient, and the Ambulatory team will test the workflows that involve rooming a patient, documenting a chief complaint, entering vitals and simple orders, etc. The Revenue Cycle (PB and/or HB) team will test billing functions, such as charge entry, workqueues, payment posting, claims, etc. Functionality testing at this stage also begins the Master File and Category List validations. Be sure to include scenarios in the testing scripts to demonstrate what will happen if the workflow is not followed. This is called "negative testing", and is an important component of any testing plan. As much as one would hate to admit it, staff will deviate from established workflows! This is particularly true if things do not initially work in the manner the users are expecting. Testing and documenting how the system behaves when workflows are not followed better equips the team to handle those situations as they occur during go live. Identifying and resolving all issues in this first testing phase will set the stage for a more successful integrated test. At the end of any phase, it is important to report the results of the testing to the project sponsors and Epic company representatives. Normally, reports are generated on a daily or weekly basis to inform them of the progress of the testing. A wrap up meeting is necessary so everyone is well informed of the findings. If there are a high number of deficiencies, makes sure to have an action plan ready to address each deficiency. At this point, it is necessary to decide how to proceed with the implementation. Major defects need to be corrected before proceeding into the next phase of testing. Depending on the

### **The second phase of testing is Interface Testing.**

Interface testing will be done in phases as well, starting on a small scale to validate the interfaces have proper connectivity and that messages are filing correctly. This first phase is also considered unit

testing, or communication testing. The interface team will play a large role in this phase, working with the application teams to identify and correct issues quickly. It is important to start with a listing of all current interfaces and the expected result of the interface. For example, if you are using another EMR but sending all charges through Epic's charge router, it is critical that you test the path the charges flow through eGate or Bridges® into Epic.

The second phase of interface testing, known as procedure testing, will incorporate testing procedures on a much larger scale. Though it may not be practical to test each procedure, it is beneficial to test as many as possible. Start with more frequently ordered procedures and go from there. As a suggestion, start with the top 100 ordered procedures. Be sure to test each type of order possible. For example, for lab orders, include several from each section of the lab: hematology, chemistry, microbiology, anatomic pathology, send outs, etc. This will present the opportunity to identify build issues and validate that the procedures are mapped correctly. It is necessary to test results as well as orders. To accomplish this, enlist the help of the staff from each ancillary department to enter results for the various orders. Do not just test orders originating from Epic; include orders and results that are placed within the ancillary applications. This "round trip" testing validates the bi-directional functionality of the interfaces. Check Chart Review to ensure all procedures file as expected, under the appropriate tabs. Much of the build needs to be completed for this phase to be executed successfully. This includes completion of the EAP master file for interfaced procedures as well as any category lists that support EAP: specimen sources and order priorities, as examples. An order transmittal scheme needs to be in place, to send the orders to the interfaces. From the experience obtained through several implementations, this is one phase of testing that should not be shortchanged! Due to the sheer complexity of interfaces, this testing should be as robust as possible.

### **The third phase of testing is Integrated Testing.**

Once all of the unit testing is completed and deficiencies are resolved, it's time to move into full-blown integrated testing. Here's where system workflows and operational workflows comes together. All workflows need to be tested. The testing will encompass all of the applications and the functionality of all interfaces. Up to this point, the testing has been rather siloed. Now, the system will be tested as it will be deployed, as an integrated application. Validating the reliability of patient flow across applications will occur. Workflows will be tested from beginning to end. This is no small feat and will require a large team comprised of AC's, SME's, and perhaps trainers and super users. Be sure to include the ancillary staff that assisted in unit testing; they will need to enter results. Validate that result routing is working as expected by testing different scenarios involving inbasket messages. Whenever possible, schedule a room as well as the staff. Experience shows that greater success is achieved when the team is in one location and has a dedicated block of time to devote to testing. Deficiencies can be resolved much more quickly when the right people are available. It is necessary to assign a testing coordinator who will keep everyone on track, and ensure the appropriate people are involved in the testing sessions. The testing coordinator should also be responsible for documenting and tracking deficiencies, and facilitating timely resolution. If due diligence was done during unit testing, deficiencies should be at a minimum.

Once the testing team is confident deficiencies have been resolved and if time permits, perform one more round of testing. As a suggestion, have trainers or super users perform this "white glove" round of testing, rather than the AC's or other team members. This final check of the system will validate that the system is ready for a successful deployment.

### **The fourth phase of testing is Workstation Testing.**

Test each workstation and printer. Include at least one example of all order transmittal routing in the test plan for each workstation. Test the printing of orders, immunization reports, prescriptions, receipts and after visit summaries, as examples of the different types of printing from Epic. Workstation testing can begin in the test environment, but should also be done in the production environment prior to go live at each site. This will validate the workstation records have been set up and mapped correctly. It is crucial to test every workstation at every site. Workstation and printer issues at go live will hinder workflows if documents do not print when and where expected.

## **Documentation/Deficiency Management**

Deficiency tracking can quickly spiral out of control if not managed properly. Each deficiency should be reviewed by at least two people before being logged on the deficiency tracker. Deficiencies should also be ranked in order of priority. Such rankings could be Red, Yellow, Green or 1,2,3. Red and 1 would be ranked as major defects which need to be corrected. If a Red or 1 defect is not corrected, you cannot go live with your system. Yellow or 2 would be ranked as medium defects. Medium defects need to be analyzed to determine if you can go live with these type of defects. Green or 3 are low priority issues. They are not defects but items that need to be corrected in Epic. Such items could include a user not assigned to a certain workqueue.

All defects need to be review by certified staff such as an AC. This aids in separating true application deficiencies from user error. Multiple occurrences of the same item should be added to the first documented instance for that deficiency and not separately logged. Strive to log deficiencies as accurately as possible, and assign to the appropriate team member for quick resolution. As mentioned previously, management of the deficiency tracker should be one of the tasks owned by the testing coordinator.

Here are a few general tips to keep in mind:

- Testing should be done using mock users who represent the varying roles established by the organization. This will give an accurate assessment of how user roles, profiles and security will function during go live.
- Testing should always be performed in the same environment.
- As changes are made to the test environment, update the production environment.
- When writing the testing scripts, be sure to adhere to the workflows! Use scenarios that accurately represent the users, patients and providers.
- Include “negative” testing, to demonstrate and document outcomes when workflow is not followed.
- Remember to test all forms of printing; be sure to include order transmittal routing.
- Document and track approved changes. Remember six months from now someone is bound to ask why a process changed—now it can be answered.

In conclusion, a well planned testing project plan, which includes all of the above phases of testing, will provide the blue prints for your testing team. Remember to allow adequate time for testing all phases of your applications. After you have tested and retested your applications and defects have been corrected, your end user training and go live will be as successful as possible.

If you need more information or have any questions please contact VCS at (610) 444 1233 or [vcs@getvitalized.com](mailto:vcs@getvitalized.com).