OUTLOOK ASSOCIATES
A Division of Qualis Health

Tips for Success

EHR System Implementation Testing

By this stage of your EHR system implementation, you have fully grasped the significant flexibility of your EHR system, and have devoted significant hours to making major set-up and configuration decisions to tailor the system to your specific needs and clinical practices. Most likely, you have spent many long hours and late nights designing new workflow processes, building templates, configuring tables and dictionaries, setting system triggers, designing interfaces, and building screen flow logic to make the system operate effectively within your practice. Now, it is absolutely critical to thoroughly test your "tailoring" to be sure the system and screens work as intended before moving forward with "live" system use. It is far easier to make changes to the system configuration while there are no patients and physicians sitting in the exam rooms growing impatient and frustrated. Testing (and more testing!) will help you be sure all your planning and hard work thus far pays off in a successful EHR "Go-Live".

The following outlines an effective, proven testing process for assuring your EHR system works effectively and as planned before initial "Go-Live".

1. Assign and Dedicate the Right Testing Resources

Effectively testing a system requires significant dedicated time and attention, and staff that have been sufficiently involved with planning the EHR workflows and processes, to fully assess the set up of the features and functions to support its intended use. Testing also requires extreme patience and attention to detail that are not necessarily skills all members of your practice staff possess.

It is very important that the right individuals be appointed to system testing activities. This is the time to highlight the investigative skills of the auditors/analysts in your practice, and focus their attention on finding any holes in the templates, hardware, logic, interfaces, etc. while everything remains easy to correct.

2. Understand all Components of Testing

There are many components and features to the new system that will need to be tested in a methodical, organized manner and sequence, to assure each is effectively reviewed and approved. A standard testing approach consists of a number of testing phases that build upon each other for full system verification. These include:

- ***Unit Testing*** – verification that individual menus, templates, forms, etc., can be accessed/displayed and/or printed and individual hardware devices are operational.
- ***Component Testing*** – verification that a system module or subsystem (e.g., patient look-up, problem list updates, action triggers, prescription writing, etc.) logically flow and work as intended.
- ***System Testing*** – verification that all relationships between modules or subsystems are in place and functioning as intended (e.g., when a new prescription is ordered the patient's medication list is updated, when a referral is ordered the correct forms print). System testing confirms that all internal workflows, handoffs, outputs (e.g., forms, reports, correspondence) and triggers are fully operational and effective.
- ***Integration Testing*** – confirmation that all information flow and control processes between the EHR system and external systems are working accurately. This includes full evaluation of practice system interfaces (e.g., practice management and EHR interfaces for demographic updates, charge capture for billing, scheduling updates, etc.) and external vendor interfaces

(e.g., lab orders and results are sent and received, direct pharmacy orders are sent and confirmed).

- ***Full "Production" Testing*** – confirmation that the complete system performs as intended for all internal and external uses (i.e., combined system and integration testing) and that, under planned transaction volumes, the screen response times for access and update are sufficient.

3. Develop a Testing Plan and Tools

Just as you used demonstration scripts and scenarios during your vendor evaluation process, you will want to develop and use similar scripts and scenarios for system testing. It will be necessary to break down each scenario into individual test units and components to correspond with each testing phase. Test execution should focus on each individual unit test, then combine related units into a component test, then combine component test into a system test, etc., until the full scenario can be executed from beginning to end.

Documenting each test case (i.e., unit, component, scenario) prior to beginning testing, including listing the ordered test action to be executed (e.g., printing of a prescription with signature when electronic medication order is charted), sequencing of the test, and the expected result (e.g., RX form printed from the nursing station printer containing the following data: …) will help organize the testing effort and support tracking of results. A separate file folder should be created for each completed scenario, and a sheet should be prepared for each test case within the scenario with space for noting issues, unexpected results or other considerations.

It is critical that each function and feature to be used by the practice is included in the scenarios, and every visit type, message type, template, etc., is addressed in the testing process. Test cases should include expected instances as well as deliberate errors and problems to test error detection and system logic capabilities. The time spent developing these detailed test plans will be well worth the effort when your EHR system works accurately and as efficiently as planned. These test plans will also be invaluable in the future for assuring continued system accuracy and performance as you roll out new capabilities within your practice, or as new system enhancements and upgrades are added by the vendor.

4. Create an Effective Testing Environment

A testing environment consists of a full set of software and data files that is separate from your production environment and used for testing and verifying new modules and components as they are implemented. At the beginning of your testing, the production environment will be clean and the test environment will be loaded with the various tables, templates and logic to be tested. Testing data will be entered into the test environment to verify accurate processing. As successful testing progresses, proven tables, templates and logic will be moved to the production environment as staging for live system use.

A testing environment will need to be refreshed (i.e., old test data removed while retaining approved templates and tables) to support new scenarios and testing efforts. To initially test interfaces it will be important to have "live-like" data streams coming across interfaces. Ideally this data will be test data prepared by the testers allowing control of tests and results. As interface testing progresses, actual, live data should be processed for final verification of system accuracy.

Following "Go-Live", a testing environment should be maintained for use in testing all future system enhancements and upgrades. Ideally, this test environment will mirror your live production environment for maximum testing ability.

5. Execute Testing as Planned and Document Issues

All individual test scripts and each set of scenarios should be tested according to the test plan and each issue or discrepancy from expected results should be noted and evaluated. Some errors detected will be the result of incorrect system setup/configuration updates, template errors or other system parameters that will need to be corrected by the implementation team and retested. Others may be issues with the software or hardware that will take vendor intervention. All issues detected in testing that cannot be easily addressed by your implementation team need to be reviewed and discussed with your vendor.

6. Retest Until All Issues are Resolved

Errors should be expected as part of the testing process. You are learning your EHR system as you are implementing it, and learning by analyzing and correcting errors is an excellent method for gaining full competency with your new system. Addressing each error as it is detected will help you strengthen your understanding of system functions and controls, and will build your troubleshooting skills for addressing future testing scenarios.

It is critical to retest each correction made to any part of the system. These retests should begin with the unit testing specifically related to the error then progress up through the component and full scenario testing to assure that a "fix" for one aspect did not create errors in another, unintended area.

Test and retest until no errors are detected. Then fully test one more time to be sure!