

Key Issues and Challenges of Testing



Healthcare Performance Group, Inc.

In this quarter's whitepaper, we address the key issues and challenges of effectively testing your system in order to validate its expected results. Whether you are testing an enterprise-wide system, a departmental application or an upgrade—testing is never a place to take a “short cut”.

WHY TEST? Can you say, “Job security”?

- ❖ To Provide the Best End User Experience Possible
 - ◆ First impressions...you only have one opportunity to create a positive first impression for your end user. It is well known that adopting technology change is difficult, and you want to avoid giving your end user reasons to resist change. By completing a thorough testing process, the end user will see fewer “bugs” and issues. This will build credibility for the IT team—and as we all know, it takes years to build it up and one slip to lose it.
 - ◆ Clinicians are particularly tough end users due to the demands on their time. Testing with them in mind will reduce their “system fatigue”, improve adoption and provide the foundation for “meaningful use” of the system.
 - ◆ Testing helps facilitate the change process, as many times this is where you “test drive” a new feature. Sometimes a new application or feature sounds like a good idea until you try to use it in a real script—better to stop and rework here before it gets to the end user. On the other hand, a good new application feature can really show its value in testing and help you communicate the benefits to the end users.
 - ◆ An effective testing process will result in fewer “roll outs” for the users to deal with and enables you to get it right the first time. Users will have better attention, better compliance and are more likely to use new features when they are delivered “glitch free”.
- ❖ Validating the Build (especially when using remote build services like the Solution Center)
 - ◆ Builds are frequently done independent of actual workflow, so testing is the perfect place to see the compatibility between workflows and the new functionality. This is particularly true in upgrades, as one should never assume “like for like” functionality. Upgrades sometimes change previous functionality, causing disruption of existing processes, and can actually create a downgraded view of the system by the end user.

- ◆ Validating will give you the internal confidence needed to know that what you are getting is what you expected.
- ◆ Catch the build problems early before you even get to user testing and you will save a lot of frustration. By stress testing the build, you can find the “break” points and correct them before the issues get downstream.
- ❖ A Quality Testing Process = “On Time and No Surprises” Implementations with Minimal Rework Post-Go-Live and Lower Costs
 - ◆ A commitment to quality will not result in delays but will actually increase project speed, success and end user satisfaction. If you shorten testing time, you will certainly increase go-live issues and lengthen implementation.
 - ◆ Quality-based testing will reduce post go-live rework and prevent the downstream “ripple effect” of problems that should have been found in testing.
 - ◆ The better the quality of testing, the less time it will take the end users to learn, helping avoid unnecessary overtime costs.

What to Test and How?

- ❖ The goal of any quality application implementation should be to provide the best possible experience for the end user. Effective testing can insure that the design and build will meet the user’s expectations; however, several levels of testing are normally required in any successful testing strategy.
 - ◆ Unit testing is typically the first level of testing. This is normally done by the build team as a quality assurance process throughout the build. Many times this level of testing is poorly documented, and, as such, can miss many simple common build mistakes. It’s important to take unit testing seriously, documenting test results and reviewing build audits thoroughly to minimize issues that could negatively impact future testing phases.
 - ◆ A key foundation to successful integration testing is to identify issues ahead of time. Look for common build mistakes. Using CCL based audits to find common build issues before Integration Testing begins can not only improve end users’ confidence in the system, it can also reduce testing time and lower costs. For example, in the lab, a common build error with charging can occur when the “Charge on Collection Event” is chosen. Many tests in the lab are actually built as “CareSets” and builders may set the charge point of the CareSet to “Collection”. Since CareSets do not get “Collected,” these charges will never drop or be submitted to the billing system, causing lost revenue. If this type of error makes it through the system to integration testing, each time a CareSet is tested, the orders in that CareSet will fail to charge. If an error like this is not corrected in the testing process, it will result in the creation of numerous incident reports, documentation of errors and extensive time

troubleshooting, correcting and retesting, not to mention the end user dissatisfaction caused by being exposed to these simple types of build errors.

- ◆ Integration testing usually follows completion of the build (and ideally build audits as referenced above). This level of testing includes integrated systems and devices. Integration testing can sometimes require a small army. One strategy to address the need for more resources is to engage end users during integration testing. Many times, a select group of clinicians are engaged; typically these users will be the designated “super users” for supporting other end users during the conversion. This is a successful strategy that has potential big benefits down the road, but it is not without risk. A clear understanding of those risks is needed to avoid the pitfalls of this strategy:
 - ◆ Benefits: The primary benefit of engaging end users to staff integration testing is to build their expertise about how the application functions and how the various features work together. Later, as super users, their testing experience and resultant expertise will enable them to more effectively support their end users because they’ll have confidence in their knowledge of the system and in the system itself...they’ll become believers.
 - ◆ Risks: It is important that the selection of the end users (future super users) be deliberate and with an eye on their relationships with their end user peers, as you want them to introduce confidence and not introduce doubt. Before bringing the selected end users on to test, you should be very confident in your design and build. You don’t want to bring users into a system that has not had sufficient quality assessment in the form of build audits and unit testing. If the quality is not there, bringing end users in to test can actually undermine confidence in the system. If the quality has not been confirmed, don’t risk exposing the end users. Further risk to the project can include failed integration testing rounds that result in extending project timeframes to allow for additional testing.
- ◆ A regression test occurs when an application stops working as intended. The most common time this type of incident occurs is after a code upgrade. Regression testing, therefore, is addressed differently than build and integration testing. The most common method used to address this is to create a standard set of test scenarios that focus exclusively on functionality; specifically, any functionality in the application that is currently in use. (Remember: the key to success is to focus these scripts on functionality rather than the build.) Sometimes though, the code upgrade can also include new build. So it’s important to be aware of any new build so that it can be tested appropriately while the new code is regression tested. One approach to documenting basic functionality is to use the application help file. Typically all of the functions are listed in the help file. Create a list of these functions for each application used and ask your users to document the specific functions they

use. Once you have the documentation, you can create test scripts that will give you a good measure of the function of the applications. The goal of any good regression test is to find any functional changes in the application that will affect the end users. Some of these can be positive, as in the case of new desired functionality, but unwanted changes in functionality are normally adverse. It's important to find them and manage them before moving them to production.

A solid regression test will allow identification and management of functional issues and increase user confidence in the IT staff and the application.

Strategies and Best Practices

- ❖ Iterative Builds/Prototyping – Be Collaborative Throughout the Process and Catch Issues Early
 - ◆ One of the most popular and successful methodologies today is AGILE. Agile methods promote a disciplined project management process that encourages frequent inspection and adaptation; a leadership philosophy that encourages teamwork, self organization and accountability; a set of engineering best practices that allow for rapid delivery of high quality software; and a business approach that aligns development with end-user needs and health system goals (Wikipedia).
 - ◆ Using this type of best practice keeps users engaged, and flaws in design and workflow are found early, resulting in a more user-friendly end product and less rework. If you are not familiar with Agile techniques, we suggest reading “Crystal Clear, A Human-Powered Methodology for Small Teams”, by Alistair Cockburn, published by Addison Wesley, and “Agile Project Management, How to Succeed in the Face of Changing Project Requirements”, by Gary Chin, published by Amacom.
- ❖ Independent Testing – A CIO’s Best Friend
 - ◆ Given tight timeframes for projects and many times a shortage of skilled resources, testing is frequently performed by the same people who designed or built the functionality. While this is normal for unit testing, it is a serious flaw beyond the unit test phase.
 - ◆ Best practice says to use an independent expert to do integrated testing who is not “too close to the design or build” to see potential errors. This independent expert may come from another area within the IT department, from an end user or from a 3rd party. Whatever the source, objectivity is an ally in seeing/identifying issues and creating a great foundation for your go-live. Finding and facing the problems early and objectively are the keys to quality and to saving frustration and costly rework.

- ◆ Don't get your feelings hurt, though – the truth of good independent testing may hurt in the short term but the value will far outweigh those hurt feelings.
- ❖ High Quality Test Scripts – Workflow-Based
 - ◆ Start early and don't wait till the last minute. These are time-consuming and hard to do. Make sure that test scripts are based in the “real workflow” of the health system as validated and approved by the end users.
- ❖ Use Automated Tools to Support Labor Intensive Tasks
 - ◆ Once a testing process has been determined, it is time to start the often monotonous repetition of detailed testing. Many test items go through the process to ensure that not only is the process correct but that it will work for all items. Due to the potential impact on patient care the ideal approach is one where everything can be tested.
 - ◆ This necessary testing process is time-consuming and a breeding ground for mistakes. Most humans are not built for long hours of repeating the same process. Our minds wander, our fingers cramp and we need physical and mental breaks. This opens the door for mistakes in testing such as items being skipped or processes not being repeated exactly for each item.
 - ◆ Many times a process is created and at some point in the testing something is identified to be incorrect or incomplete with the process, resulting in a rebuild or correction being applied to the system. It is at this point, that testing needs to start over from scratch, causing irritation among the testing resources. Retraining is then needed to get the testers to follow the new process.
 - ◆ An automated approach to these labor-intensive testing tasks can help solve this problem and free the human resources to work on more thought-provoking problems. Automated test scripts can be built to repeat around most processes as defined. They can identify when items are not following the process either due to an error in the system or unique steps for a particular item that were not identified in the process. Modifications can be made quickly to the automated process by modifying and retraining the automated scripts for multiple runs of the testing phase...and the tools won't complain or get tired or make mistakes and they will work 24/7.
 - ◆ The automated test scripts have the ability to work around the clock, saving the amount of time needed to test. They efficiently follow direction and can be made to identify and report unexpected situations. Summary report cards can be created to identify error groups to quickly assess the results of testing. Onsite resources can work the issues that were identified rather than spending time looking for the issues.

The following are three case studies where the combination of expert knowledge of Cerner Millennium[®], health care and advanced testing tools really made a difference to our customers:

Care Mobile[®] Pharmacy Build at Large Multi-Entity Health System

This customer came to HPG with the issue of needing to unit test their CareMobile[®] pharmacy build. The steps required to complete this test were:

- ❖ Identify the medications that needed to be tested.
- ❖ Determine medication types and ordering details required to place each medication order.
- ❖ Register a sufficient number of patients to insure adequate volumes and effective test results.
- ❖ Identify at a detailed level the patient/medication associations.
- ❖ Develop the test scripts and placed each of these orders in the Cerner Millennium[®] system using the details from step 2.
- ❖ Manually scan the barcode for each medication using the CareMobile[®] hand units.
- ❖ Check back into Cerner Millennium[®] to make sure the barcode process was successful.

HPG coordinated the testing activities, working with customer resources, HPG expert consultants, project manager, and HPG's automated testing tools and test scripts (ETS). By integrating each resource's expertise, we were able to create a plan that removed errors and reduced the time needed to complete the Unit Test of the customer's CareMobile[®] project.

- ❖ Careful onsite planning was used to sort the medication list in a way that would reduce the effort of manual bar coding. Onsite HPG resources worked with the pharmacy in the drug closets to observe how the drugs were being stored and formulated a plan to order the meds in Cerner Millennium[®] to reduce search time when bar coding would begin.
- ❖ Custom automated test scripts were created to register patients and place orders using the sorted medication list. These automated scripts were run over the weekend, eliminating the need for extra work days on the project. 3111 medications were ordered on 31 unique patients. These patients were given appropriate height, weight and allergies so that they would work with the customer's active Discern[®] Rules. In order to keep a manageable number of orders on each patient, we limited the medications to 100 per patient. This also reduced ordering time as we had a limited number of drug to drug interactions pop in the Cerner Millennium[®] system during the ordering process. Only 23 medications failed order initiation and all 23 were identified as orders that had been inactivated after the formulary was extracted for testing purposes. The ETS was able to identify these orders as not found and continued testing the remaining orders. A report of all 23 medications

was provided to the pharmacy analyst who verified that they had been correctly inactivated.

I think there needs to be a space between paragraphs here on the webpage.

- ❖ HPG then turned over the documentation of what orders were placed and patient demographic information to the customer. Using this information, the manual process of bar coding was completed. During this step, 279 display and general build errors were identified. These errors were then corrected and retested.

By working with the customer, creating a plan, and utilizing automated testing tools and scripts (ETS), we were able to reduce the testing time from an original estimate of 25 work days to 3 days and saved the customer 22 business days of work effort, delivering substantial cost savings.

Pharmacy Charges at a Community-Based Health System

This health system needed to test each medication in its catalog to ensure that accurate charges were being passed, thereby enabling ProFit[®] to generate bills correctly when each order is placed. The customer supplied HPG with the current formulary of medications. The formulary identified each order and the medication type of that medication.

Prior to creation of the ETS automated scripts to order the meds, several CCL audits were run to identify any major build issues related to pharmacy orders. These audits revealed that about 90% of the orders did not have the needed REV codes built. By identifying this issue prior to initiating the medication ordering scripts, we were able to save the time needed to register the patient and order the 800 medications (35 hours).

Once the bill code upload was completed and the issue was resolved, we were able to resume the testing. There were three separate scripts created to handle the three medication types. The scripts identified an issue within Cerner Millennium[®] Pharmmedmgr tool. The search option would not work on the primary mnemonic. The ETS script was also able to capture screen shots of the search option issue. A quick adjustment to the ETS scripts allowed us to work off of NDC instead without further delay to the testing. The scripts were executed and saved the customer 4 days of testing time based on the estimates in the project plan. All the medication orders were placed in 1 day.

Training Database at Large Academic Medical Center

This customer needed to set up a large and accurate training database. They were running training sessions every week and refreshing the database so that each training class had the same setup as the prior weeks. They wanted to register patients for each room and bed in the health system. This would require 1200 unique registration conversations using the Cerner Millennium[®] tool PMLaunch.

The ETS process identified the following needs to complete the registration process:

- ❖ Patient Demographic Spreadsheet – All fields that were to be filled out for each registration were identified. Then unique information for all 1200 patients was created. The information included patient name, Social Security number, address, insurance information, etc.
- ❖ Pediatric Aging – As part of the original list 200 newborns needed to be included in the population. However, newborns created a unique problem; if the newborns were registered once and left in the system for each refresh, then they would keep the same date of birth. Over time, these newborns would be too old to realistically be in a newborn room. The answer to this was to create a second list of newborns and to run the script each weekend after the refresh. These newborns would have the date of birth equal to the current day and would never get older than 1 week. (We found a way to stop aging 😊.)

After the preparation was complete, a custom ETS script was created to match the registration process used at the health system. The same script would be used to run the initial 1000 patients (non-pediatric) and reused each week for the remaining 200 (pediatric). ETS registered the initial 1000 patients in 16 hours. It then took an additional 4 hours to run each week to register the pediatric patients. This saved the customer an estimated 60 hours of manual registration for the initial registration. For the pediatrics it saved the customer from having to bring two resources in each Sunday to register 200 patients. The pediatric script was run each weekend for 6 months, saving the customer an additional 300 hours of overtime.



Healthcare Performance Group, Inc.

Corporate Headquarters

Healthcare Performance Group, Inc.

23419 West 215th Street

Spring Hill, KS 66083

Ph: 888-681-9501

Fax: 316-462-5900

info@HPGresources.com

HPGresources.com

Cerner Millennium, ProFit, Discern and CareMobile are all registered trademarks of Cerner Corporation and/or its subsidiaries.

HPG is not sponsored by and/or affiliated with Cerner Corporation®.

