Validation of Spectrometry Software

Part V — Performance Qualification

R.D. McDowall

R.D. McDowall is principal of McDowall Consulting (Bromley, United Kingdom), and "Questions of Quality" column editor for *LCGC Europe*, *Spectroscopy*'s sister magazine. Address correspondence to him at 73 Murray Avenue, Bromley, Kent, BR1 3DJ, United Kingdom.

In a previous installment of this column, we looked at the initial phases of qualifying the spectrometer and its software (1). We covered the installation qualification (IQ) and the operational qualification (OQ). We'll now turn to the performance qualification (PQ) of the software; this can also be termed the "end user testing" or "user acceptance testing" of the software. The purpose of the PQ is to show that the software meets the documented requirements in your system requirements or user requirements specification (URS), which you will have written at the start of this validation journey. You did write a requirements specification — right?

What Do the Regulators Want to See?

There's no point charging into testing, so let's start with the basics and ask the question, "What do the regulators want to see when they inspect?" We'll start with one of the guidances that the U.S. Food and Drug Administration (FDA) has produced. In September 2001, the FDA published a draft guidance document on *21 CFR Part 11* validation (2). Although it was a draft and stamped "not for implementation," a number of points are worth reading and understanding before we look at what we, as end users, will undertake regarding testing of the spectrometry software.

Two sections in the document cover testing. The first, under "General Considerations," in Section 5.4, discusses dynamic testing; the second, under "Commercial Software" in Section 6.1.3, presents functional software testing. I'll quote each section verbatim and then discuss the principles and implications that spectroscopists will need to consider in each spectrometer performance qualification.

5.4.1 Key Testing Considerations

- Test conditions: Test conditions should include not only "normal" or "expected" values, but also stress conditions (such as a high number of users accessing a network at the same time). Test conditions should extend to boundary values, unexpected data entries, error conditions, reasonableness challenges (e.g., empty fields, and date outliers), branches, data flow, and combinations of inputs.

- Simulation tests: Some testing may be performed using simulators, usually conducted off-line outside of the actual user's computing environment.

- Live, user-site tests: These tests are performed in the end user's computing environment under actual operating conditions. Testing should cover continuous operations for a sufficient time to allow the system to encounter a wide spectrum of conditions and events in an effort to detect any latent faults that are not apparent during normal activities.

6.1.3 Functional Testing of Software

End users should conduct functional testing of software that covers all functions of the program that the end user will use. Testing considerations discussed above should be applied.

When the end user cannot directly review the program source code or development documentation (e.g., for most commercial off-the-shelf software, and for some contracted software), more extensive functional testing might be warranted than when such documentation is available to the user. More extensive functional testing might also be warranted where general experience with a program is limited, or the

software performance is highly significant to data/record integrity and authenticity.

Note, however, we do not believe that functional testing alone is sufficient to establish software adequacy.

Derived Principles for PQ Testing

The following principles for software testing can be derived from the sections of the draft guidance presented:

- End users are responsible for testing the system, not the vendor. End users may subcontract the PQ to a third party, but typically this third party is not the vendor because the testing has to be independent and the users can use a system differently from the vendor's design.
- At least some PQ testing must be conducted using the operational system and computing environment. If a networked spectrometer is used, the testing should cover a representative portion of the overall system. If another test environment is used for the remainder of the testing, this must be documented and any differences between this system and the operational system must be noted and explained.
- Test the system under all expected uses and operating ranges, and then try some worst-case scenarios, especially if you have a multiuser networked system, or your system is used in a potentially hostile environment, such as a warehouse. (But you'll have defined this situation in your specification long before, right?)
- Testing alone is not sufficient to validate a system. (Validation is a process that covers specification; quality software design, coding, and release; and installation in the operating environment.)

You cannot just accept a vendor's "validation package," run it, and assume the spectrometer is validated. This is naive and stupid from the vendor's perspective, because the end user will usually use the system outside of the tested parameters in the vendor's validation package. It is also naive and stupid from the user's perspective if they believe that is all that should be done to validate a spectrometer and its software.

What Does This Mean for PQ Testing?

Going back to my point at the start of this column, we can't just rush in and start testing. We have to look at our requirements and plan our testing approach. "Wait a minute," you may be thinking, "we have to sit down, think, and plan what testing to do?" Yes!

Also, don't assume that if you validate an analytical method that the underlying spectrometer and software are implicitly validated. If you take this approach, I suggest you start drafting the wording of the warning letter before the inspector arrives in your laboratory, because it will save you time later.

Test Approach: White Box or Black Box Testing?

Because we need to plan our testing, the first issue that we face is, what are our testing limitations as end users? Two main approaches to dynamic testing are conventionally known as white box and black box testing (see Figure 1).

White box testing. This testing requires the full knowledge of what the program unit or module does, including the complete specification of the inputs, outputs, and processing algorithms within each module of the software application. The design specification is used to devise tests to prove that the functions described work as designed. In essence, you need to have a programming background to execute white box testing. Normal users cannot undertake technical testing, either because they do not have the full technical specification of the system or they do not possess the technical skills to undertake this type of testing — usually both.

Black box testing. In contrast, in black box testing the tester only knows the overall function of the module or software with input limits. No programming knowledge is needed, just training in how to use the application. Therefore, users will undertake black box testing, where known inputs will be entered and the outputs compared with the anticipated results.

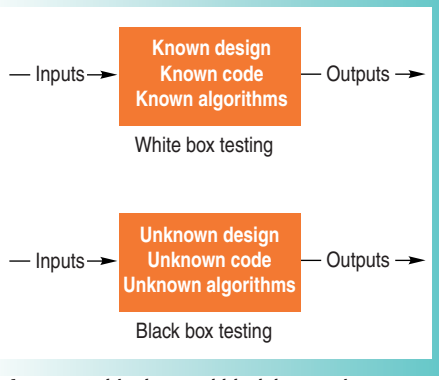

Figure 1. White box and black box testing.

So we now know how we will approach the testing of our spectrometer software.

Planning What to Test

The URS should give an explicit understanding of how the software will be used; however, a key requirement in the planning of the PQ testing is training and understanding of how the software actually operates. This process will take some time and can be rate limiting in planning the PQ testing; factor this time in when ordering the system so that key personnel can be trained to understand the software as the system is installed on site. Understanding what the software will actually do and how you will use it are crucial to your overall validation effort.

I know that some bright individuals will think that a short cut will be to use the user manual supplied by the vendor. *Please* don't think this way (I am groveling on the floor as I type this), because the manual and the software take different paths through the life cycle, and if the two match at the end, it is more by luck than planning. For instance, a version of mass spectrometry software that I once validated had a chapter on export using tab-separated values and comma-separated values, but these functions did not make it into the final release of the software. There was therefore a disconnect between the manual and the software. Been there, done that, unfortunately.

Outline of a test plan (based on IEEE Standard 829).

1. Test plan identifier
2. Introduction
3. Test system/item
4. Features to be tested
5. Features not to be tested
6. Approach to be adopted
7. Pass/fail acceptance criteria for all features to be tested
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing and training needs
14. Schedule (test order)
15. Risks and contingencies
16. Approvals.

Outline of a test script (adapted from IEEE Standard 829).

1. Test script identifier
2. Purpose of the test script
3. Special requirements
4. Test procedure steps
5. Test log identifier
6. Activity and event entries
7. Expected results and acceptance criteria
8. Actual results
9. Anomalies
10. Testers
11. Reviewers
12. Summary.

New Features? Update the URS!

As you learn how the software works, you might find that features you did not look at closely before will be useful, or that the business you are supporting needs a different type of analytical support. New features and functions may therefore be used; this means that your user specification is now out of date and must be updated to reflect the new way that you are using the spectrometer and its software.

The URS is a living document and needs to reflect the current way you use the system.

Tracing User Requirements to Qualification Testing

When you wrote the URS, each requirement was uniquely numbered, which means that it can be traced to where in the qualification it is tested. Therefore, you'll need a means to trace where an individual requirement or group of requirements are tested in the PQ (or indeed, the OQ, if the vendor's testing matches your specific requirements; instrument control is a good example of this). We'll look at this in more detail in the next two sections, which cover the PQ plan and test scripts.

PQ Test Plan

There are a number of ways to document PQ testing; I'll explain the method that I use, which is based on Institute of Electronic and Electrical Engineers (IEEE) software engineering standards. The documentation is derived from IEEE standard 829, "Software Test Documentation" (3). However, there are other ways to approach the problem, and you can also use test protocols. The bottom line is that the testing is documented and covers how you will use the system.

IEEE standard 829 lists the main sections of a test plan from the document, shown in the sidebar (left); I use a slight modification of this list when I write PQ test plans.

Once the system to be tested is defined, we will consider three key sections in more detail:

- Features to test: Identifying the test scripts, the features tested in each test script, and the requirements tested, traced from the URS.
- Features that will not be tested: Identify the parts of the system and the software that will not be tested and the rationale for this. For example, release notes for the application document the known features or errors of the system. Tests carried out in any PQ should not be designed to confirm the existence of known errors but to test how the system is used daily by the users. If these or other errors were found by the testing, then the test scripts have space to record the fact and what steps were applied to resolve the problem.
- Approach to be adopted, specifically section 6.3: The written notes of the assumptions, exclusions, and limitations to the testing undertaken. Because we cannot test everything, how can we concentrate on the most critical items from both a scientific and a regulatory perspective? When this is done what are the assumptions we have to make? What are the limitations and exclusions to this test approach? This section also provides the contemporaneous notes of testing that can be very useful when inspected because you can refer to it easily to refresh you memory as to why a specific approach was taken.

We will look at the interaction of these three areas and more detailed test script design in the next installment in this series.

PQ Test Scripts

Linked closely to the PQ test plan are the test scripts, which are the heart of any PQ testing effort and will take some time and effort to draft and get correct. The concept is that the test script will form the instruction set, the testing log, and the archive for the actual testing, and all experimental data and output will be recorded here. The structure, based on the same IEEE standard as the test plan discussed above, is shown in the sidebar, "Outline of a test script."

Features to Test in Any Spectrometer System

From the URS requirements, there are three main areas that must be considered for testing any spectrometry system:

1. Scientific and instrument functions
2. *21 CFR Part 11* technical controls and functions
3. Backup and preservation of electronic records.

Yes, I know that backup and preservation of electronic records is a *21 CFR Part 11* issue, but because many people forget to do it when considering a

standalone spectrometer, it is given its own specific section because there is more than meets the eye at first glance.

Okay, so that's the high-level list — what about the details?

Some of the scientific and instrument functions are typically:

- Data acquisition
- Calibration methods and analyte calculation
- Reporting results
- Custom calculations: If mathematical functions are available within the system to perform calculations, these need to be tested
- Library functions if used
- Capacity tests, such as analyzing the largest expected number of samples in a single run, especially if the system has an autosampler. This step is a specific response to the requirement in the U.S. good manufacturing practice regulations that the system should have "adequate size" (4).

Furthermore, tests were designed to demonstrate the handling of common problems and out-of-range entries that were designed to fail to show the predictability of the system in these areas.

Some of the *21 CFR Part 11* features to test are

- System security and access control
- Data File Integrity
- Audit trail
- Ability to discern invalid and altered records.

The backup and preservation of electronic records testing are

- Backup and restore
- Archive and retrieve
- Data migration from older versions of the software or from a different system.

Summary

This column is intended to give you an overview of the performance qualification (PQ) phase of validating a spectrometer and its software. In the next installment, we'll describe how to design tests, write test scripts, and link this with the PQ test plan.

Note

On February 20, 2003, the FDA issued a draft guidance for industry on *Part 11* Scope and Applicability. In this draft, validation guidance referred to in this article (2) was withdrawn. Notwithstanding, the validation approach outlined in this article remains unchanged.

References

1. R.D. McDowall, *Spectroscopy* **18**(2) 64–69 (2002).
2. "FDA Draft Guidance for Industry, *21 CFR 11* Electronic Records and Electronic Signatures Validation," Food and Drug Administration, Washington, DC, 2001.
3. IEEE Standard 829-1998 Software Test Documentation, Institute of Electronic and Electrical Engineers, New York, NY.
4. Current Good Manufacturing Practice regulations, *21 CFR 211;* 211.63. ■