Validation of Calibration Software by the Calibration Facility

Patrick Kershaw, Associate Software Engineer
Fluke Corporation, P.O. Box 9090 Everett, WA 98206

Abstract

Today, in the United States, regulatory agencies such as the Food and Drug Administration (FDA), the Nuclear Regulatory Commission (NRC) and many others are moving toward tighter controls and verification on the instruments and standards used to calibrate the field instruments. To facilitate the increased traceability requirements, the various events and items which either directly or indirectly enable the comparison of a known standard to Unit Under Test (UUT) are divided into discreet steps. Prudent verification checks are then applied to each step.

Calibration software, such as MET/CAL, is a link in the calibration chain. Although there is extensive testing at the software developer's facility, it may be prudent to test the software after installation on the user's computer network. This would verify the installation was successful, the software is compatible with the user's hardware, and the software has the ability to deliver accurate measurement records taken during a calibration verification or adjustment event to file storage. This paper will discuss the testing which would be prudent, and will explore some of the expectations of some regulatory agencies.

1. Introduction

1.1. Objective

This paper focuses primarily on interpreting the guidance and draft guidance which has been promulgated by the U.S. Department of Health and Human Services, Food and Drug Administration (FDA) as it applies to a process automation software such as the Fluke Metrology Software. The principles presented fall into the category of good engineering practices and are easily applied to any situation where a high degree of confidence is required that software functionality meets internal process requirements and external auditability requirements of outside regulatory agencies. Meeting these audit requirements will also fulfill a major portion of ISO 9000 standards.

The published guidance is very flexible and thus somewhat vague. This paper is an interpretation intended to assist the calibration lab during the validation of software and it offers a tried method of developing and maintaining validation test plans. It is expected to be used in conjunction with the published guidance of the applicable regulatory agencies.

1.2. Background

On June 1, 1997, medical device quality system regulation became effective which included requirements for validation of software used in medical devices or the automation of development processes of medical devices. This encompasses off-the-shelf (OTS) software used to automate processes such as calibration and test device asset management. Fluke's MET/CAL calibration software and MET/TRACK asset management software are examples of many software packages which fall into this category.

The requirements to produce auditable records of software validation are published in the Current Good Manufacturing Practices (GMPs or cGMPs) issued in the United States by the FDA as the *minimum* requirements.[i] The cGMPs as they apply to software used in the development of medical devices draw their regulatory authority from 21 CFR Part 820. Most countries promulgate their own GMPs, but they are similar in nature.

The FDA, Center for Devices and Radiological Health is currently constructing a "Guidance for the Industry" paper titled *General Principles of Software Validation.* This guidance was distributed for comment purposes only on June 9, 1997, but it does reflect in text principles which have been apparent in the FDA's handling of OTS software.

What is evident in FDA guidance is acknowledgment of a fact which is not disputed by those in the software development industry**: Software made and tested without error on one hardware platform and configuration may fail to work identically on different hardware or with different configuration.** While uncommon, this may happen with two different pieces of hardware claiming to be compatible with the same standard, such as the case of PC clones. There is an *implied* compatibility, but no *guaranteed* compatibility with machines of different makes and configurations. The end result is that, although extensive testing on many different platforms is performed for most OTS software, there is the potential that software may not work with an untested combination of hardware. **Software validation by the end user is the only means of ensuring proper operation in the environment in which it is to be used.**

Finally, and what makes OTS software a challenge to validate, the software is very often not configured in an identical manner to that envisioned by the developer. This is especially true of software like the Fluke Metrology Software which allows a great deal of user customization. This makes the criticality of individual automatic functions as they apply to a given user impossible to evaluate by the developer, and thus **it is important that the end-user be involved in the development of the validation plan.**

1.3. Paper Overview

This paper contains a macro view of the Software Development Life-Cycle as it pertains to validating OTS software. It contains a micro view of the Analysis Phase where the Validation Plan is actually developed. Finally, it provides a practical exercise in developing a Validation Test Script on-site.

2. The Software Development Life-Cycle

There are numerous models of the software life-cycle which all differ subtly. The FDA advocates that whatever software life-cycle model is in use, that it be documented and utilized in the process of validating the software. For OTS software a typical cycle would probably include planning, procurement, analysis, validation and implementation.

2.1. Planning Phase

During the planning phase, the general and some specific requirements of the software are agreed upon. As indicated by the title general requirements, these should reflect the major purposes which the software must fulfill. The specific requirements are the individual functions which the software must perform to support the general requirements. Particular mention should be made of automatic functions.

For example, in the case of asset management software such as MET/TRACK, a general requirement might be that it provide a recall for assets which are due for calibration. Specific requirements to support this general requirement might be the automatic function of assigning the calibration due date based on the stored calibration period and the date of the last calibration. Another specific requirement might be the ability to generate a report of all assets which require calibration during the next week or month. There may be dozens of specific requirements for each general requirement.

General and specific requirements should be documented so the performance of the software versus its published requirements can be evaluated.

Information about the available products should be gathered at this time. This will help when determining what is reasonable to expect from software which is not custom-made for a specific site.

2.2. Procurement Phase

The Procurement Phase takes the place of the execution or development phase in user developed software models. While the function of the Procurement Phase may be intuitive, there are a few items to note:

Ensure the requirements which have been decided upon by the calibration lab are understood by the software vendor, and allow the vendor to verify that the software package will be able to meet the requirements. If the requirements cannot be met by OTS software, the vendor may be able to offer alternative methods used in similar situations.

Determine the status of the Quality Assurance program of the software developer as well as the ability of the software developer to support the calibration lab during the validation process.

2.3. Analysis Phase

In the Analysis Phase, a working model is used where general requirements are implemented and the remainder of the specific requirements of the software are determined. The Validation Test Plan is then developed.

2.3.1. The Working Model

The OTS software is installed on a test platform and the person or persons who will become the system experts will examine its capabilities and determine which capabilities will be used to meet the general requirements set forth during the planning stage. In the process, more specific requirements are defined which will enable the software to perform its general function. All of the specific requirements are documented in order to ensure that they be tested properly during the validation phase.

A working model of the desired system is created, and walk-throughs are conducted to determine if the functionality selected is adequate. If data migration or import is required it should be performed, and the validation plan annotated to require a verification of satisfactory data transfer.

Next, the specific requirements are evaluated for the level of risk their failure may pose to meeting general requirements. If the failure of a specific requirement may directly or indirectly cause a personal hazard, it should be specially noted. While all parts of the software should be validated, the FDA's guidance recognizes that those items which are not critical to the function of the software and which cannot cause personal hazard need not be tested as thoroughly as those which can.[ii]

2.3.2. Develop the Validation Test Plan

To simplify performance of the validation testing, discreet portions or modules of the software functionality are determined, and the test plan is divided into these modules to enable the validation to be conducted in stages. If the software operates as a number of separate executables as the Fluke Metrology Software does, there may be natural module boundaries.

The draft guidance paper under review by the FDA Center for Devices and Radiological Health prohibits partial validation of software.[iii] If there is no interaction across the executable boundaries, it may be possible to test a single module if only that executable or other programs it uses are changed. Any change to an executable or the programs it uses would require that the entire executable be re-validated. Proper selection of the module's boundaries can save significant validation time later, when changes to the software may be required.

As the validation plan is formalized and documented, one of the most difficult tasks is to determine the level of effort to be expended testing each function of the software. To assist with this the documented requirements should be consulted. OTS software packages probably contain a large number of features, and it is

likely, except in the case of very simple programs, that not all of the functionality will be needed or implemented to perform the requirements placed on the software. When configuring the software and creating the validation test, focusing on meeting and testing specific requirements will save time.

The validation should include:

- **A general functionality test of the software module. The magnitude of the this effort should be limited. This may be fulfilled by finishing the validation testing with a walk-through of the test asset path through the lab on the actual system.**
- **Tests for each of the documented specific requirements. These tests should be detailed. If the specific requirement has been identified as critical to providing the general requirements, it should be tested more thoroughly. If the specific requirement is noted to have the potential to directly or indirectly cause a personal hazard, even more effort should be expended in its testing. Note that there may be software capability which will not be tested in detail because it is not a specific requirement of the end user.**
- **Tests for each of the documented general requirements. These tests should be detailed, but should acknowledge the results of the above tests.**

Each of the above tests could be discreet, or they could be combined into a few test scripts which would encompass them all. What is important is that the specific and general requirements are traceable to a validation test.

2.4. Validation Phase

At this point, the software is installed on the final target system taking care to minimize the impact on the function of any existing software. In the case of upgrade software, a back-up should be performed prior to installing the new software, so that the older version can be restored in the event the upgrade fails validation. It is imperative that the validation test be performed on the actual hardware which will be used, because, assuming the developer performed appropriate validation testing, hardware is the main variable which is to be considered.

The test plan is now run on the system. Should the software fail validation, the tested functionality should be evaluated.

- **If the software behavior is acceptable, the test plan can be modified to document the behavior.**
- **If the software behavior is not acceptable, the software configuration may be modified, after which the validation plan for the entire module should be started again.**
- **If the software behavior cannot be made acceptable through user configuration, it should be reported to the software vendor. A determination can then be made whether to restore the older version of the software or to wait for a corrected version from the vendor. Most vendors will expend significant energy to correct software bugs which prevent their software from passing user validation.**

Once the validation plan has been completed successfully, it should be reviewed by a competent and objective party who is not a developer or user. This might be performed internally by the QA department.[iv]

2.5. Implementation

Commence using the software. Set up channels which technicians can use to provide feedback to the planners on ways to improve the system. These comments get discussed at the next planning meeting, and the software development life-cycle starts over again.

Remember that any significant changes to the configuration or version of the software may require retest for the entire module.

3. A Practical Exercise - Test Plan Development Using a Working Model

Now, the focus will be shifted to the details of creating a Software Validation Test Plan. This is not quite a "cook book" method, but will definitely speed the development of the validation plan. A working model of the software is important to implementing the methods used here.

3.1. Walk-Through

In this case, ***Fluke Metrology Software,*** which includes both asset management modules and calibration modules, will be used to demonstrate the validation test creation process.

At each point during the walk-through, evaluate:

- **Technician interface with the software.**
- **UUT interface with the software.**
- **Calibrator interface with the software.**
- **Automatic software functions.**
- **Data stored by the software.**
- **Action performed by specific command to the software.**
- **Future activities which may require additional validation.**

The walk-through itself might include:

⇒ **Prepare to receive the instrument**

- **Calibrate standards.**
- **Enter standards into the database.**
- **Configure standards.**
- **Record standards calibration record into the database.**

⇒ **Receive the instrument into custody of the calibration lab.**

- **Update to Location Record.**
- **Updated the instrument status in the Inventory Record.**

⇒ **Verify calibration of the instrument.**

- **Add an asset to the database.**
- **Perform a Calibration Verification Procedure.**
- **Update the Calibration Record.**
- **Print Calibration Certificate or Failed Calibration Report.**
- **Update status in the Inventory Record.**

⇒ **Conduct maintenance on the instrument.**

- **Update Maintenance Record.**
- **Perform a Calibration Adjustment Procedure.**
- **Update status in the Inventory Record.**

⇒ **Query the database for information.**

- **Print reports on specific queries.**
- **Print Calibration Certificates.**
- **Look for information based on a Query.**

⇒ **Manually correct information in the database.**

- **Add an asset to the database.**
- **Alter existing records.**
- **Security verification.**

⇒ **Return UUT.**

- **Update to Location Record.**
- **Updated the instrument status in the Inventory Record.**

3.2. Evaluating the General and Specific Requirements during the Walk-Through

Suppose that for this exercise the general requirements placed on the system were:

1. **Automate Calibration Verification.**
2. **Automate Instrument Adjustment.**
3. **Store Calibration Event Data.**
4. **Print Calibration Certificates.**
5. **Manage Calibration Assets for Recall.**

For this example, ***Manage Calibration Assets for Recall*** will be used, but the principles can be applied to each requirement. Many of the specific requirements to support this general requirement where documented during the Planning Phase. These included:

- **Automatically assign the calibration due date after a successful calibration.**
- **Access data at the personal computer (PC) workstation to determine what assets require calibration during the next recall period.**
- **Print a report based on the assets which will be due for calibration during the next week.**
- **Automatically assign the calibration due date beyond the year 2000.**

During the walk-through, the following specific requirements are noted and documented.

- **Automatically assign the due date as the day it failed calibration.**
- **Track the location of an asset so that it can be located when due for calibration.**
- **Store the calibration interval for each asset.**
- **Log the arrival and departure date from the calibration lab.**
- **Log the arrival and departure time from the calibration lab.**

Each of the specific requirements is then evaluated for criticality to the general requirement and hazard to persons. No potential hazards are detected.

Specific Requirement	Critical	Hazard
Automatically assign the calibration due date after a successful calibration.	Yes.	No.
Automatically assign the due date as the day it failed calibration.	Yes.	No.
Access data at the PC workstation to determine what assets require calibration during the next recall period.	No.	No.
Print a report based on the assets which will be due for calibration during the next week.	Yes.	No.
Track the location of an asset so that it can be located when due for calibration.	Yes.	No.
Store the calibration interval for each asset.	Yes.	No.
Log the arrival and departure date from the calibration lab.	Yes.	No.
Log the arrival and departure time from the calibration lab.	No.	No.
Automatically assign the calibration due date beyond the year 2000	Yes.	No.

3.3. Creating Modules to Simplify the Validation Task

The software chosen may provide natural barriers at which to declare modules. In the case of the Fluke Metrology Software, these barriers are created by separate executable files. Other software may be similar, or it may be that a form or view is most convenient. In this case the modules are declared as the 1) Query module, 2) Manual Entry module, and 3) Run-Time module. The natural modules may be broken down further as required for convenience. The modules are evaluated against the specific requirements to determine where each will be incorporated into the validation test plan.

1) Query module,
- **a) Database View test**
- **b) Report test**

2) Manual Entry module
- **a) Add Assets**
- **b) Inventory Record test**
- **c) Calibration Record test**
- **d) Maintenance Record test**
- **e) Location Record test**

3) Run-Time module
- **a) Calibrator Control test**
- **b) Procedure tests**
- **c) Data Storage test**
- **d) Add Assets test**

Specific Requirement	Critical	Hazard	Module	Step
Automatically assign the calibration due date after a successful calibration.	Yes.	No.	2c 3c	
Automatically assign the due date as the day it failed calibration.	Yes.	No.	2c	

			3c	
Access data at the PC workstation to determine what assets require calibration during the next recall period.	No.	No.	1a	
Print a report based on the assets which will be due for calibration during the next week.	Yes.	No.	1b	
Track the location of an asset so that it can be located when due for calibration.	Yes.	No.	1a 2e	
Store the calibration interval for each asset.	Yes.	No.	2b 3d	
Log the arrival and departure time from the calibration lab.	No.	No.	2e	
Log the arrival and departure date from the calibration lab.	Yes.	No.	2e	
Automatically assign the calibration due date beyond the year 2000	Yes.	Yes.	2c 3c	

The introduction to the test plan may include a cross-reference table similar to this which will prevent critical specific requirements from being errantly left out of the test plan. It also clearly demonstrates the validation plan was developed throughout the software development life-cycle and considers potential personal hazard and criticality to meeting documented requirements.

3.4. Creating the Test Scripts

Test scripts are step-by-step instructions to the person performing validation testing. The **test plan** is usually comprised of several **test scripts**. Once all of the documented specific requirements are listed in the cross-reference table, test scripts can be generated to verify the requirements. Construct the test scripts considering one specific requirement at a time. Then when considering the next specific requirement, it may be added to the existing script if convenient, or an entirely new script might be constructed. At the end of each script, it is appropriate to require a signature, date and comment area for the test-person.

Considering the specific requirement to **Automatically assign the calibration due date after a successful calibration,** the Manual Entry 2c module test will probably be a written step-by-step procedure. It can be as simple as the following:

1. **Start the Manual Entry tool and login.**
2. **Add a new asset with a periodicity of 90 days.**
3. **Manually enter a new successful calibration event for the asset.**
4. **When calibration due data and time fields are automatically updated, verify that the calibration due date is 90 days from the calibration date. (modify for any special macros to exclude holidays or weekends.)**
5. **Save the data and exit Manual Entry.**
6. **Start Query and login.**
7. **View the calibration record just entered to verify the calibration due date is 90 days from the calibration date.**
8. **Exit Query.**

It might be desirable to create a block at the top of the test script outlining the specific requirements to be tested. This will assist the test person in identifying errors.

Suppose that the specific requirement to test to **Automatically assign the calibration due date beyond the year 2000** is to be added to this test. It may then be revised as follows:

1. **Start the Manual Entry tool and login.**
2. **Add a new asset with a calibration interval of 90 days.**
3. **Add another new asset with a calibration interval of 24 months.**
4. **Manually enter a new successful calibration event for each asset.**
5. **When calibration due data and time fields are automatically updated, verify that the calibration due date is equivalent to the calibration interval plus the calibration date. (modify for any special macros to exclude holidays or weekends.)**
6. **Save the data and exit Manual Entry.**
7. **Start Query and login.**
8. **View the calibration records just entered to verify the calibration due dates are equivalent to the calibration interval plus the calibration date.**
9. **Exit Query.**

Later when preparing to create a test for the specific requirement to **Save a manually entered calibration event to the database**, it might be determined that this test already suffices.

By continuing to build additional tests in this fashion, the entire test plan can be created relatively quickly with little effort wasted testing non-critical or non-hazardous items.

After all of the specific requirements are covered in the test plan, a walk-through test on the final hardware platform should be created to test basic functionality not otherwise tested. This test script will satisfy overview testing.

3.5. Automating Portions of the Test Plan

The test plan need not be a series of manual entries documented step by step in a test script. The potential to use automated tests for a particular module or for a group of specific requirements should be noted during the walk-through. In many cases, automated tests can be created and run quickly. Their performance can then be documented in the test plan. When an automated test is created, the test should be frozen in it's form at the time of the successful test, and a copy of the automated test should be archived and maintained with the test plan.

This is not intended to imply that a company conducting a validation test on one piece of software should purchase and attempt to use automated testing software such as Microsoft Test. If the test is not one which requires multiple repetitions, it is likely that creating such a test would take longer than manually testing. What is intended, is that the calibration lab may be able to write test calibration procedures in the language of the calibration software whose successful accomplishment would be an indication that the software is operating properly.

3.5.1. Creating Automated Tests

It is relatively easy for the calibration lab to create automated tests. Taking each applicable specific requirement one at a time as was done when creating the manual tests, an automated test procedure can be constructed which may test specific requirements quickly.

In this instance, the specific requirement to **Automatically assign the calibration due date after a successful calibration** practically requires that a test case be constructed using a calibration test procedure. If only this one specific requirement was to be tested, the procedure could be as simple as a dialog box prompting the user to select if the test should be a pass or a fail. If pass is selected, the calibration passes, and the calibration due date should be assigned as today's date plus the length of the calibration interval. This would then be tested by printing a Certificate of Calibration or by viewing the database data using the Query functions.

As more and more specific requirements are evaluated to be appropriate for automated test, they could also be added to this or a separate test procedure. Maybe at the conclusion of the test, printing a Certificate of Calibration would serve to validate a specific requirement from a completely different general requirement branch. After that, perhaps the boundaries of a series of voltage tests on a UUT could be validated to insure they pass or fail at the expected tolerance. Remember that although they are tested within many different modules, all of the specific requirements can be cross-referenced to the validation plan using the cross-reference table.

3.5.2. Modifying Existing Automated Procedures

Some automated tests may be obtained from the vendor. For instance, Fluke provides a large number of calibration procedures for many UUTs on various calibrators. These could be modified to test the software packages control of calibrator functions.

3.5.3. Documenting the Performance of an Automated Test

A page instructing the validation test-person to run the automated test should be included in the validation test plan. This might be in the form of a table, but it is more often incorporated into the body of a test script combining both manual and automated activities. Once again, it is appropriate for the page to include a signature, date and comment area for the test person.

3.6. Running the Test Plan

Once the test plan is written, the software must be transferred to the hardware on which it will be run during actual operation of the calibration lab. This may temporarily impact work in the lab.

Be sure methods are in place to return the system to its previous state should the software fail validation for any reason.

Once the test plan is completed successfully, it should be reviewed by the lab supervisors and persons responsible for meeting FDA requirements. The entire validation process should be reviewed by an internal or external objective third party, then stored safely with disk copies of any automated tests utilized during its completion.

4. Items to Consider

The following is a list of items which should be considered during implementation of a validation plan for calibration software or asset management software. They have been compiled primarily from feedback from calibration labs to Fluke Corporation.

- **Test any automatic functions of the software such as:**
    1. **Assignment of calibration due date and time.**
    2. **Test failure at the test tolerance boundaries. (One procedure which utilizes the entire functionality of the attached calibrator might be run five times. First with perfect input, then at the upper tolerance pass limit, then at the upper tolerance fail limit, then at the lower tolerance pass and fail limits.)**
    3. **Assignment of pass or fail.**
    4. **Calculation of Test Uncertainty Ratio (TUR). (For MET/CAL, this has been tested by creating a procedure which utilizes every accuracy specification for the a given calibrator at a pre-determined ratio to verify the expected ratio is returned.)**
    5. **User defined automatic functions such as forwarding field information or actions executed by macros.**

- **Verify procedures meet the calibration requirements of the UUT manuals.**
- **Anticipate year 2000 compliance issues such as:**
    1. **Check manual and automatic assignment of calibration due dates across the turn of the century.**
    2. **Change the computer system date to beyond year 2000 and determine system response..**

- **Verify hazardous conditions are handled appropriately.**
    1. **Procedures operating at voltages which could cause grave personal injury should provide a warning message.**
    2. **Connections are only altered while the calibrator output is secured.**

- **Detailed test is only required for the functions utilized by the individual calibration lab.**

5. Conclusion

Validating OTS software continues to be one of the largest obstacles in the path of the calibration lab seeking to upgrade to the latest tools for automating their processes. There is heightened awareness of the software validation issue due to the need to upgrade to year 2000 compliant software, the ever advancing industry standard both in personal computer operating systems and in the complexity and flexibility of available OTS software to support the calibration lab. The

benefits of validating software are not doubted, but the choice to upgrade is tempered by the spectre of validating the new software which requires hardware and personnel resources.

Creating a proper validation test on-site is time consuming, but is not complicated. By following a method such as that outlined above, a large amount of time can be saved during the development of the validation test which will be easy to administer, and will provide a high level of confidence to the calibration lab and external auditing authorities in the software tested.

Biographical Sketch

Patrick Kershaw graduated with an Aerospace Engineering degree from the University of Southern California in 1984. He spent eight years in the U.S. Navy working as a Nuclear Engineer and Submarine Warfare Officer while pursuing post-graduate studies in Nuclear Engineering and Software Engineering. Since leaving the service in 1991, he has pursued a career in Software Engineering attaining certification in Software Test and C++ Programming. Patrick has worked at Fluke Corporation both testing and developing calibration software with the MET/CAL development group for the past two years.

References

1. *Federal Register, Current Good Manufacturing Practices - Medical Devices,* Vol. 61, Number 195, website www.fda.gov/cdrh/humfac/frqsr.html,Oct 7, 1996.
2. *Federal Register,*
   21 CFR 820.3, Definition
   21 CFR 820.30,Design Controls
   21 CFR 820.70, Production and Process Controls
   21 CFR 820.72, Inspection, Measuring, and Test Equipment
   21 CFR 820.75, Process Validation, Oct 7, 1996
   Each of these can be viewed at website www.fda.gov/cdrh/humfac/frqsr.html (way down at the bottom!)
3. *Guidance for Industry, General Principles of Software Validation,* Draft Guidance Version 1.1, .This can be viewed at website www.fda.gov/cdrh/comp/sweval.html, June 9, 1997.
4. Frank Scavo, CFPIM, CIRM, *Software Validation for Pharmaceutical and Medical Device Manufacturers,* paper delivered 1994 APICS International Conference in San Diego, CA, USA.
5. Lewis, Robert O., Independent Verification and Validation: A life Cycle Engineering Process for Quality Software. New York: John Wiley & Sons, Inc., 1992.

Footnotes:

---

[i] *Current Good Manufacturing Practices - Medical Devices*.

[ii] *21 CFR 820.30(g).*

[iii] *Guidance for Industry, General Principles of Software Validation,* Draft Guidance Version 1.1, Section IV.G.

[iv] *Software Validation for Pharmaceutical and Medical Device Manufacturers,* section 1.5.