Software Validation in Accredited Laboratories
A Practical Guide

Gregory D. Gogates
Fasor Inc.
2042 Hollis Road, Lansdale, Pennsylvania 19446-5721 USA
g.gogates@ieee.org   www.fasor.com

Abstract

There are three types of software products in computers and automated systems used for the acquisition, processing, recording, reporting, storage, or retrieval of accredited laboratory/test data.

- COTS – Commercial off-the-shelf
- MOTS – Modified off-the-shelf
- CUSTOM

These three types of software can reside on local hard drives, network hard drives, embedded on integrated circuits (IC, ROM, EPROM), or removable disks. This document explains each type of software and provides practical approaches to requirements, design, testing, installation, and configuration management practices. Utilizing this methodology will ensure evidence exists to consider the software suitably validated and adequate for use.

Software Classifications

Software can be classified into 3 categories to assist in determining the amount of work required ensuring validation. This simplifies in performing the validation and maintenance.

COTS software is code that is purchased without modification and either, cannot, or will not, be modified by the lab. An example of this would be Microsoft Word/Excel/or dedicated instrument interface. This software can be considered adequate by the "mass market" and used as is per the note in section 5.4.7 of [1]. The lab should have evidence that any math used in these tools and any formula strings written by the lab work as expected. Any loadable parameters should be considered a data transfer and be "checked". They should also ensure cells are locked and predetermined setups are controlled. Finally the lab should have a set of requirements that describe what they want the software to do. There should be testing to ensure that the software meets these requirements..

Embedded IC software inside of instruments or automated equipment can be assumed validated by the manufacturer and confirmed adequate during the calibration process. This assumes the calibration fully exercises the firmware functions. No further action is necessary.

MOTS software is code that is modified, or customized, for specific applications. Examples include Lab Windows, Lab Tech Notebook, Tile EMC, generic data acquisition software, etc. The purchased portion of the software can be considered COTS. The modified or customized portion is considered CUSTOM and should have evidence of validation. This can be achieved by documenting the functions and design of the modification. The code or blocks of the modification should be both documented and annotated showing each function. Finally testing should be performed on each block of the modification giving evidence that each piece works as designed and satisfies each function.

CUSTOM software is code that is lab written or subcontractor written. Examples of this includes MS Visual Basic, HP Basic, C++, SQL Database Design., etc. This software requires full validation evidence. This process is fully described in [2]. Minimally there should be evidence of Requirements, Design, Construction, Testing, and Installation.

Software Life Cycle

All software has an activity lifecycle. The whole process is controlled under a quality assurance process.
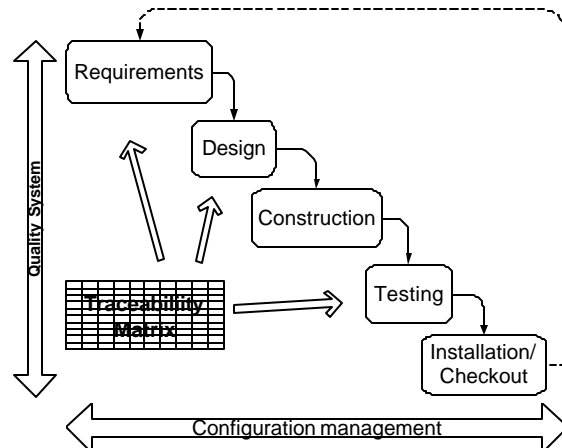

Figure 1

The lifecycle usually follows some variation of waterfall methodology. It states that activities are done in order with some iterative overlap. Following this lifecycle assures quality and provides validation evidence.

- **Requirements** – This is where the functions of the software are defined, in user terms, what the software needs to do. The output is a Functional Requirements document.
- **Design** – This is where the design of how the software is built, along with any diagramming showing the architecture is described. The output is a Software Design document.
- **Construction/Coding** – This is where the actual source code including header information and annotations is done. The output is code.
- **Testing** - This is where software testing occurs. It includes a test plan of what and how it will be tested, with individual test cases, along with successful completion, which satisfy the plan, and map directly back to the requirements. Important points to consider are verification of math, valid/invalid inputs, power failure, code walkthru's, structural and functional tests.
- **Installation/Checkout** – This is where the software is loaded onto lab hardware.
- **Traceability Matrix** – shows how requirements map to designs and to tests. It is usually not a one-to-one relationship. It gives assurance that each requirement has associated designs and tests.

The validation documentation need not be too cumbersome. One document with the following outline would suffice for simple software. More complex projects would separate these into separate documents.

I.	Requirements
	[Insert requirements here]
II.	Design
	[Insert design here]
III.	Source Code
	[paste code here]
IV.	Test Plan
	[insert plan here]
V.	Test Cases
	[insert cases here]
VI.	Traceability Matrix
	[insert table here]

These adequacy processes have to be performed, and documentation updated, each time there is a change to the software. Hence the dotted line, shown in figure 1. This is the reason for calling the process a software life "cycle".

Configuration Management

Configuration management ensures that all changes to software/hardware are controlled. It also ensures that all software installations are known, and have periodic checks. To achieve this, software used in accredited laboratories should be put into perspective of where it fits into the computing hardware. The term "Product" is used to describe each piece of uniquely configurable software as it sits on a shelf. The term "System" is used to describe the complete deployed software and hardware bundle. Therefore, a system can be comprised of one to many products. This concept is necessary because one software "Product" can be installed on many "Systems". Many labs have one version of software that is installed on many computers. It is important to separately consider each installation.
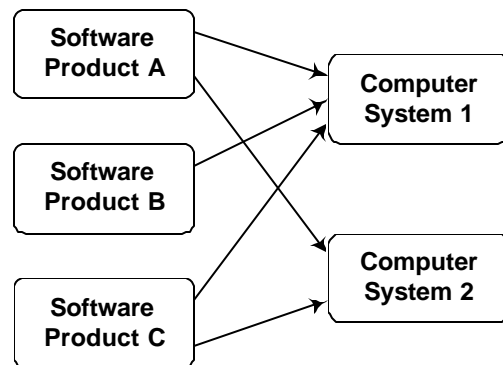

Figure 2

In all cases, the software should be under, software configuration management, version and access control. The labs should have records indicating what versions are current. They should also know which computers have what software installed. They should also control access to them so only authorized individuals have access. There should be a "check" that is performed periodically to ensure that the correct version is installed and no unauthorized modifications have occurred.

Good Practices

There are many good software practices to be considered. The following are some that have been found to assist in managing validation.

- Treat each software product as a piece of calibration equipment that has to be "re-calibrated" each time it is changed or modified.
- Place software product masters in a read only directory.
- Network computers that access a shared program on a server.
- Lock spreadsheet cells that contain math.
- Password protect configuration files or setup screens.
- Backup, back up, and backup off site!
- Plan for hardware/software disaster recovery.

A template has been created to assist the user in documenting validation. It is available from ftp://ftp.fasor.com/pub/iso25/validation/validation_template.dot
There is also guidance available to assist in managing Excel® spreadsheets. It is available from ftp://ftp.fasor.com/pub/iso25/validation/excel_locking.pdf

Software Checklist

The following is a good checklist to verify that adequate validation activities have occurred.

1) Has the Firmware been validated, via the calibration by a cal lab, who has the capability to thoroughly check the software (i.e. OEM or Authorized partner) (Note: for non-calibratable device Firmware, treat as Software)
2) Have all of the "used" firmware parameters been documented and confirmed that they are correct?
3) Does a Software Requirements document exist for the Software?
4) Does a Software Design document exist detailing either the full design or details of the configuration?
5) Does software testing documents exists describing completed, unique, test cases that exercise the design both +/- and confirms the requirements?
6) Is there a test log showing test failures and corresponding retests/dispositions?
7) Does evidence exist confirming correct software deployment at each target installation? [5.5.11]
8) Has configuration control been applied to all of their Software/Firmware to ensure that: [4.12.1.4]

a) The Software source code location is access controlled. [5.4.7.2.b]
b) Firmware/Software formulas & parameters are "locked" to prevent inadvertent changes. [5.5.12]
c) Equipment lists identify Software as a separate line item showing correct version and location. [5.5.5]
9) Does evidence exist showing that personnel involved in Custom Software development have adequate training?
10) Do Databases and spreadsheets include "audit trails" to not allow previously data to be obscured? [4.12.2.3]
11) Do adequate instructions exist for the operation & maintenance of the Software? [5.4.1]
12) Does the accuracy of the Firmware/Software meet or exceed the accuracy required by the test method? [5.5.2]
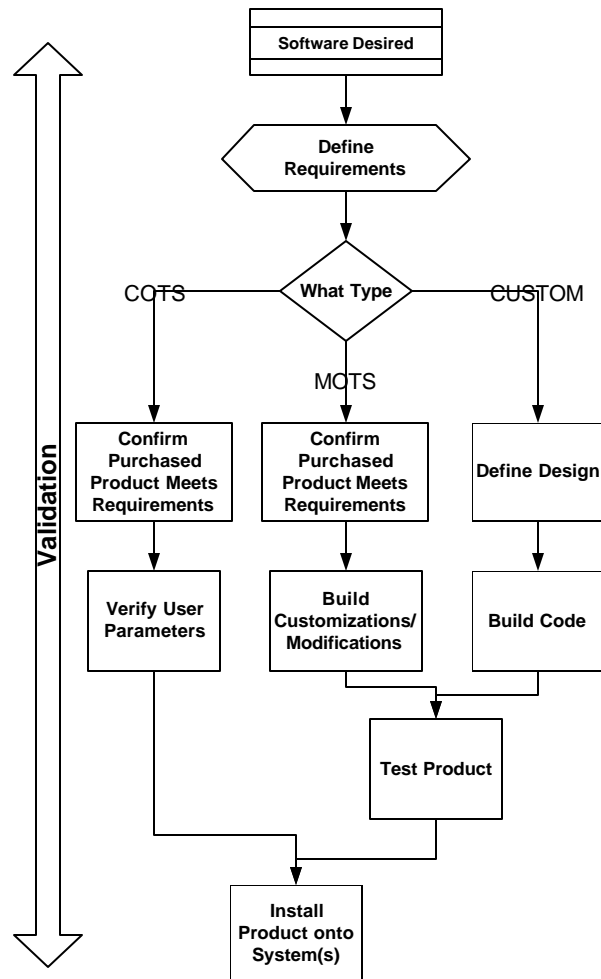
Figure 3

Summary

The processes described herein have been simplified from more rigorous processes. They do not take into consideration more stringent regulatory requirements. The philosophy is to ask for what is achievable within the confines of an accredited laboratory. These processes are summarized in figure 3.

It should be clear from this discussion that VALIDATION is not a state but a continuing conformance to a software lifecycle and quality system.

17025 Implications to Software

The following clauses imply the use of software or software controls that "could" require validation.

§4.1.5.c Procedures exist to protect client's information.

§4.3.1 Procedures to control software

§4.3.2.1 Quality system reviewed and approved by authorized personnel (electronic signatures)

§4.3.2.2 Authorized editions of appropriate documents all locations. (Intranet, NT file Share)

§4.3.3.2 Altered or new text shall be identified (electronic document)

§4.3.3.4 Procedures shall describe how changes in documents, including software are controlled.

§4.12.1.2 Records (electronic media) shall be stored and maintained so that they are retrievable.

§4.12.1.4 Procedures to protect and back-up electronic records.

§4.12.2.1 Retain records for the retention period (old versions of software also)

§4.12.2.2 Observations shall be recorded at the time they are made. (electronic).

§4.12.2.3 Electronic records shall avoid loss to original data (audit trails)

§5.4.1 Lab shall have instructions on the use and operation of equipment (and software).

§5.4.7.1 Calculations (spreadsheet) and data transfers (tables) shall be subject to checks.

§5.4.7.2.a Software shall be validated

§5.4.7.2.a Laboratory configurations of COTS software shall be validated.

§5.4.7.2.b Procedures are established to protect data.

§5.4.7.2.c Computer and automated equipment are maintained.

§5.5.2 Equipment & Software shall comply with specifications.

§5.5.4 Each item of equipment & software shall be uniquely identified.

§5.5.5 Records shall be maintained of equipment & software.

§5.5.11 When correction factors are used, procedures shall ensure software is updated.

§5.5.12 Software shall be safeguarded form adjustments.

§5.10.1 Reports may be issued electronically.

§5.10.2.j Reports may contain electronic signatures.

§5.10.7 Reports may be transmitted electronically.

References

[1] ISO/IEC 17025 General Requirements for the Competence of Testing and Calibration Laboratories, 1999

[2] ISO/IEC/IEA/IEEE 12207 – 1995 "Information Technology – Software Lifecycle Processes"

[3] ISO/IEC 12119 – 1994 "Information Technology – Software Packages – Quality Requirements and Testing"

[4] ISO9000-3 "Guidelines for the application of ISO9001 to the development, supply, and maintenance of software"

[5] ASTM E919-96 "Standard Specification for Software Documentation for a Computerized System"

[6] ASTM E 1579 - 93: "Standard Guide for Ensuring Data Integrity in Highly Computerized Laboratory Operations"

[7] ANSI/IEEE Std 1012-1998 "IEEE Standard for Software Verification and Validation Plans"

[8] ANSI/IEEE Std 1059-1993 "IEEE Guide for Software Verification and Validation Plans"

[9] EPA 2185 – 1995 "Good Automated Laboratory Practices

[10] FDA 21CFR§11, "Electronic Records, Electronic Signatures, August 1997

[11] FDA Guidance for Industry "Computerized System used in Clinical Trials", April 1999.

[12] FDA Guidance for Industry, "General Principals of Software Validation", June 1997

[13] Draft FDA Guidance for Industry "21CFR Part 11; Electronic Records; Electronic Signatures Validation, August 2001

[14] Software Engineering Body of Knowledge, http://www.swebok.org

[15] OCDE/GD(95)115, "The Application of the Principles of GLP to Computerized Systems" Monograph No. 116

[16] ACDM/PSI, "Computer Systems Validation in Clinical Research, A Practical Guide", 2001, http://ww.acdm.org.uk

[17] GAMP Guide for "Validation of Automated Systems in Pharmaceutical Manufacture" Version 3.0 March 1998. http://www.gamp.org

[18] EA/GA(98)95 Guideline "The Use of Computers and Computer Systems in Accredited Laboratories" Version 4, March 1998.

[19] MIL-STD-498, Software Development and Documentation