WORLD JOURNAL OF PHARMACEUTICAL RESEARCH
SJIF Impact Factor 5.990
Volume 5, Issue 01, 1092-1134. Research Article ISSN 2277– 7105

COMPUTER VALIDATION AND ETHICAL SECURITY MEASURES FOR PHARMACEUTICAL DATA PROCESSING

Omprakash G. Bhusnure[1*], Kendre S.P., Kaudewar D.R., Bankar I. N[1], Sachin B. Gholve[1] and Giram P. S.[2]

[1]Channabasweshwar Pharmacy College (Degree), Department of Quality Assurance, Maharashtra, India.

[2]Channabasweshwar Pharmacy College (Degree), Department of Pharmaceutics, Maharashtra, India.

[3]Channabasweshwar Pharmacy College (Degree), Department of Pharmacology, Maharashtra, India.

Article Received on 04 Nov 2015,

Revised on 27 Nov 2015,
Accepted on 20 Dec 2015

***Correspondence for Author**
Dr. Omprakash G. Bhusnure
Channabasweshwar Pharmacy College (Degree), Department of Quality Assurance, Maharashtra, India.

ABSTRACT

A Computer System Validation is a set of activities that FDA Regulated companies must conduct for each of their GxP sensitive computer systems. The objective of these activities is to document evidence that each computer system will fulfill its intended purpose in a GxP production, laboratory, or research operation. The intention is to avoid software problems that could have serious impact. Dynamic testing of the software is an important part of the Computer System Validation. But Computer System Validation is more than just this type of testing. Computer System Validation requires a comprehensive set of equally important static testing activities that need to be conducted throughout the SDLC (Software Development Life Cycle). This includes a variety of analyses, audits, walkthroughs, reviews, and traceability exercises. Documentation must be accumulated that demonstrates that these activities have been performed effectively. Today, the term Computer System Validation refers specifically to the technical discipline used in the Life Sciences sector to help ensure that software systems meet their intended requirements. Through its regulations/guidance on Computer System Validation, the FDA has shaped IT testing and analysis processes to match the needs and requirements of the industries it governs. As a result, Computer System Validation has become an integral part of doing business in FDA regulated environments. It should be noted, however, that significant progress has been made in achieving consistency and harmonization between FDA regulations/guidance on Computer System Validation and relevant international IT standards and best practices. It is likely that the future will see convergence of Computer System Validation terminology and techniques as a common technical discipline across other industry sectors as well.

KEYWORDS: Computer System Validation, SDLC, FDA regulations, Data Management.

INTRODUCTION

The concept of validation was developed in the 1970s and is widely credited to Ted Byers who was then Associate Director of Compliance at the U.S. FDA. The concept was focused on: "Establishing documented evidence which provides a high degree of assurance that a specific process will consistently produce a product meeting its predetermined specifications and quality attributes".[1]

If you are a company that relies on electronic records for certain parts of your workflow, your systems may be subject to the requirements and limitations of US FDA 21 CFR Part 11, Electronic Records; Electronic Signatures, or other industry guidelines including GAMP5, ICH, EMEA regulations and 21 CFR regulations. For companies outside the pharmaceutical and biotech area, there could also be increasing ISO and clinical standards requirements requiring computer system validation to verify that electronic records are controlled and unadulterated.

Computer system validation (CSV) is the process of establishing documented evidence that a computerized system will consistently perform as intended in its operational environment. CSV is of particular importance to industries requiring high-integrity systems that maintain compliant with current regulations (EU, FDA) under all circumstances.

Throughout the 1980s, computer systems validation was debated primarily in the U.S.A. Ken Chapman published a paper covering this period during which the FDA gave advice on the following GMP issues.

- Input/output checking
- Batch records
- Applying GMP to hardware and software
- Supplier responsibility

- Application software inspection
- FDA investigation of computer systems
- Software development activities

Computer systems validation also became a high profile industry issue in Europe in 1991 when several European manufacturers and products were temporarily banned from the U.S.A. for computer systems noncompliance. The computer systems in question included autoclave PLCs and SCADA systems. The position of the FDA was clear; the manufacturer had failed to satisfy their “concerns” that computer systems should.

- Perform accurately and reliably
- Be secure from unauthorized or inadvertent changes
- Provide for adequate documentation of the process

Meanwhile two experienced GMP regulatory inspectors, Ron Tetzlaff and Tony Trill, published papers respectively presenting the FDA’s and U.K.’s MCA inspection practice for computer systems. These papers presented a comprehensive perspective on the current validation expectations of GMP regulatory authorities.

Topics covered included

- Life-cycle approach
- Quality management
- Procedures
- Training
- Validation protocols
- Qualification evidence\
- Change control
- Audit trail
- Ongoing evaluation

The Quality System regulation requires that “when computers or automated data processing systems are used as part of production or the quality system, the manufacturer shall validate computer software for its intended use according to an established protocol.” This has been a regulatory requirement for GMP since 1978. In addition to the above validation requirement, computer systems that implement part of a regulated manufacturer’s production processes or quality system (or that are used to create and maintain records required by any other FDA regulation) are subject to the Electronic Records, Electronic Signatures regulation. This regulation establishes additional security, data integrity, and validation requirements when records are created or maintained electronically. These additional Part 11 requirements should be carefully considered and included in system requirements and software requirements for any automated record keeping systems. System validation and software validation should demonstrate that all Part 11 requirements have been met. Computers and automated equipment are used extensively throughout Pharmaceutical, Biotech, Medical Device, and Medical Gas industries in areas such as design, laboratory testing and analysis, product inspection and acceptance, production and process control, environmental controls, packaging, labelling traceability, document control, complaint management, and many other aspects of the quality system. Increasingly, automated plant floor operations have involved extensive use of embedded systems in.

- PLCs
- digital function controllers
- statistical process control
- supervisory control and data acquisition
- robotics
- human–machine interfaces
- input/output devices
- computer operating systems

There are basically two types of computers, analog and digital. The analog computer does not compute directly with numbers. It accepts electrical signals of varying magnitude (analog signals) which in practical use are analogous to or represent some continuous physical magnitude such as pressure, temperature, etc. Analog computers are sometimes used for scientific, engineering and process-control purposes. In the majority of industry applications used today, analog values are converted to digital form by an analogto- digital converter and processed by digital computers. The digital computer is the general use computer used for manipulating symbolic information. In most applications the symbols manipulated are numbers and the operations performed on the symbols are the standard arithmetical operations. Complex problem solving is achieved by basic operations of addition, subtraction, multiplication and division. A digital computer is designed to accept and store instructions (program), accept information (data) and process the data as specified in the program and display the results of the processing in a selected manner. Instructions and data are in coded form the computer is designed to accept. The computer performs automatically and in sequence according to the program.[2]

For the EC Guide to Good Manufacturing Practice for Medicinal Products, Annex 11 identifies the following requirements that need to be addressed for computerized system application.

- GMP risk assessment
- Qualified/trained resource
- System life-cycle validation
- System environment
- Current specifications
- Software quality assurance
- Formal testing/acceptance
- Data entry authorization
- Data plausibility checks
- Communication diagnostics
- Access security
- Batch release authority
- Formal procedures/contracts
- Change control
- Electronic data hardcopy
- Secure data storage
- Contingency/recovery plans
- Maintenance plans/records
- Systems elements in computer validation which need to be considered are:
- Hardware (equipment)
- Software(procedures)
- People(users)

The purpose of computer system validation is to ensure an acceptable degree of evidence, confidence, intended use, accuracy, consistency and reliability.

WHAT IS COMPUTER SYSTEM VALIDATION AND WHY IS IT IMPORTANT?

What is Computer System Validation and Why is it Important? A key source document providing FDA guidance on the general topic of Validation is "General Principles of Validation, Food and Drug Administration" from the Center for Drug Evaluation and Research.

The definition of Validation in this document is: Establishing documented evidence which provides a high degree of assurance that a specific process will consistently produce a product meeting its predetermined specification and quality attributes. Validation, as described in this document, is aimed at manufacturers of pharmaceuticals and medical devices who must demonstrate that their processes produce consistent product quality. It applies to all processes that fall under FDA regulation, including, but not limited to, computer systems. For example, Validation applies to pharmaceutical manufacturing processes which include checking, cleaning, and documenting that all equipment used in manufacturing operates according to predetermined specifications. Computer System Validation (or Computerized System Validation as it sometimes called in the literature) is the result of applying the above definition to a Computer System.

Establishing documented evidence which provides a high degree of assurance that a Computer System will consistently produce results that meet its predetermined specification and quality attributes.

Note that a "Computer System" in the Life Sciences sector is more than computer hardware and software. It also includes the equipment and instruments linked to the system (if any) as well as the trained staff that operate the system and/or equipment using Standard Operating Procedures (SOPs) and manuals. As applied to computer systems, the FDA definition of Validation is an umbrella term that is broader than the way the term validation is commonly used in the IT industry. In the IT industry, validation usually refers to performing tests of software against its requirements.[3] A related term in the IT world is verification, which usually refers to Inspections, Walkthroughs, and other reviews and activities aimed at ensuring that the results of successive steps in the software development cycle correctly embrace the intentions of the previous step.[4-5] As we will see below, FDA Validation of computer systems includes all of these activities with a key focus on producing documented evidence that will be readily available for inspection by the FDA. So testing in the sense of executing the software is only one of multiple techniques used in Computer System Validation.

There are two key reasons why Computer System Validation is extremely important in the Life Science sector.

1. Systematic Computer System Validation helps prevent software problems from reaching production environments. As previously mentioned, a problem in a Life Science software application that affects the production environment can result in serious adverse consequences. Besides the obvious humanistic reasons that the Life Science sector strives to prevent such harm to people, the business consequences of a software failure affecting people adversely can include lawsuits, financial penalties and manufacturing facilities getting shut down. The ultimate result could be officers getting indicted, the company suffering economic instabilities, staff downsizing, and possibly eventual bankruptcy.
2. FDA regulations mandate the need to perform Computer System Validation and these regulations have the impact of law. Failing an FDA audit can result in FDA inspectional observations (“483s”) and warning letters. Failure to take corrective action in a timely manner can result in shutting down manufacturing facilities, consent decrees, and stiff financial penalties. Again, the ultimate result could be loss of jobs, indictment of responsible parties (usually the officers of a company), and companies suffering economic instabilities resulting in downsizing and possibly eventual bankruptcy.

A key point to be gleaned from 1 and 2 above is that not only do FDA regulated companies need to do Computer System Validation, but they need to do it right. Cutting corners on doing a Validation might save a little money in the short term but these savings will look minute and inconsequential when compared to the potential costs and impacts of not doing the Validation correctly.

RELATIONSHIP BETWEEN COMPUTER SYSTEM VALIDATION AND 21 CFR PART 11

Relationship Between Computer System Validation and 21 CFR Part 11 In 1997, the FDA added rule 21 CFR Part 11 to the Code of Federal Regulations. [6] This regulation introduces specific controls on the use of electronic records and includes strict administrative controls on electronic signatures. These controls deal with.

1. Making electronic records suitable for supplanting paper records.

2. Making an electronic signature as secure and legally binding as a handwritten signature. Regardless of whether or not a company uses electronic signatures, 21 CFR Part 11 impacts all companies that use computer systems that create records in electronic form associated with the GxP environment.[7] All computer systems in this category must have technical and administrative controls to ensure: 1. The ability to generate accurate and complete copies of records 2. The availability of time-stamped audit trails.

3. The protection of records to enable accurate and ready retrieval.

4. Appropriate system access and authority checks are enforced From the point of view of Computer System Validation, 21 CFR Part 11 has two key impacts. First, it affirms that the FDA expects all computerized systems with GxP electronic records to be validated (just in case this was not obvious before). Secondly, 21 CFR Part 11 says that when you do a Validation of a particular Computer System, items 1 through 4 above automatically become part of the requirements for the System. This means that every Computer System Validation must assess whether the system being validated satisfies requirements 1 through 4 above and must identify deviations, if any, and corrective actions. Since FDA regulated companies are anxious to avoid deviations in their Validations wherever possible, most companies in the Life Science sector are currently in a proactive mode of assessing all of their systems for 21 CFR Part 11 compliance and addressing deviations through procedural remediation, technical remediation (e.g. software upgrades), or replacement of non-compliant systems with 21 CFR Part 11 compliant systems.

RELATIONSHIP OF COMPUTER SYSTEM VALIDATION TO THE SOFTWARE DEVELOPMENT LIFE CYCLE

Computer System Validation is carried out through activities that occur throughout the entire Software Development Life Cycle (SDLC). The “V Diagram” (Figure 1) is widely used in the IT literature to emphasize the importance of testing and testing related activities at every step in the SDLC. The V-diagram is really a recasting of the oft-criticized “Waterfall” model of the SDLC. In fact the phases in the Waterfall Model are essentially the life cycle phases that appear on the left-hand side of the V-diagram. The V-diagram emphasizes the need for various forms of testing to be part of every step along the way. This avoids a “big-bang” testing effort at the very end of the process, which has been one of the main criticisms associated with the Waterfall model (or the way some have people have interpreted the Waterfall model). The activities represented in the V-Diagram (labeled V &V in Figure 1) include Static Testing as well as Dynamic Testing activities. Static Testing (sometimes called

Static Analysis) refers to inspections, walkthroughs, and other review/analysis activities that can be performed without actually executing the software. In Dynamic Testing, the software is actually executed and compared against expected results. While many IT people use the term “testing” to mean dynamic testing, both dynamic and static testing activities are used in Computer System Validation to help ensure that the results of successive steps in the SDLC correctly fulfill the intentions of the previous step.[8]

Different types of activities represented in the V-Diagram are sometimes distinguished by the terms Verification and Validation, words whose connotations in the IT world were discussed in the previous section. In some visualizations of the V-Diagram [see reference 3, for example], the term “Verification” is associated with the activities shown on the left hand side of the V and “Validation” associated with activities on the right hand side. At this point in time there are reasons why it may be preferable to avoid drawing this distinction. First, the IEEE definitions of these two terms have become so close[9] that it is hardly worth trying to articulate (or even remember) the difference. It is more productive to just call them V&V activities, in companies regulated by the FDA and other regulatory bodies throughout the world, the term Validation is often used interchangeably with Computer System Validation when discussing the activities required to demonstrate that a software system meets its intended purpose.

In a sense, Computer System Validation has actually extended the V-Model and put a more user driven spin on it. As shown pictorially in Figure 2, Computer System Validation has several important features.

- Computer System Validation is driven by the “User”. That is the organization choosing to apply the software to satisfy a business need is accountable for the Validation of that software. While the software supplier, the IT organization, the QA organization, and consultants can play important roles in a Computer System Validation, it is the User organization that is responsible for seeing that documented evidence supporting the Validation activities is accumulated.
- The User must write “User Requirements Specifications” (URS) to serve as the basis for a Computer System Validation. The URS provides the requirements the Computer System must fulfill for meeting business needs. A Computer System Validation cannot be done unless such a URS exists.

- The Supplier of the Computer System should provide Functional Specifications and Design Specifications, which satisfy the URS. Where such Specifications are not available for an existing system, they are sometimes developed by "reverse engineering" the functionality of the system.
- Users are involved in every step of the process (deeply involved for custom development, less for package based systems)
- A three level-structure is imposed on User Testing:
  - The Installation Qualification or IQ focuses on testing that the installation has been done correctly
  - The Operational Qualification or OQ focuses on testing of functionality in the system installed at the User site.
  - The Performance Qualification or PQ focuses on testing that users, administrators, and IT support people trained in the SOPs can accomplish business objectives in the production environment even under worst case conditions.

A TYPICAL COMPUTER SYSTEM VALIDATION[10-11]

As discussed in the previous section, Computer System Validation is definitely not a "one size fits all" procedure; the approach that an individual company may take to a specific Validation depends on the rules, guidance, best practices, and characteristics of the system being validated. On the other hand there are some strong similarities between the activities in most Computer System Validations and the type of documentation produced. In fact one way to get a good understanding of Computer System Validation is to take a look at the type of documents that would be accumulated. The following is a list of the documents that might result from the Validation of a Computer System application to be used in a GxP sensitive environment 2.

Document Name	**Function of Document in Validation**
User requirements Specification (URS)	Defines clearly and precisely what the User wants the system to do and states any constraints (e.g. regulatory) under which the system must operate.
Validation Plan	Defines the objectives of the Validation and the activities, procedures and responsibilities for accomplishing the objectives of the Validation. The Validation Plan should also deal with the approach for maintaining the Validation status This will generally involve referencing the organization's Quality Management System documentation that deals with such issues as Configuration Management, Change Control, and System

	Retirement.
Project plan	Details the tasks and time line for the project.
Documentation justifying Selection of System including Supplier Audit Report	Outlines the reasons for choosing the system including the results of auditing the supplier's quality management system
Functional Specifications	Detailed specifications showing the functions that the system performs
Design Specifications	Detailed specification showing how the system performs the functions documented in the Functional Specifications
Supplier Test Plans and Results	Documentation of Supplier Testing
Task Reports	Documentation of Design/Specification/Testing Reviews, Walkthroughs, and Inspections
Traceability Matrix	Analysis document that shows mapping between URS, Functional Specs, Design Specs and test cases in IQ, OQ, PQ (see below)
Risk Assessments	A Risk Assessment (sometimes called Failure Mode and Effects Analysis), is an analysis of failure scenarios associated with each of the functions and sub functions of a system. Each failure scenario is examined for potential business impact and likelihood of occurrence in order to determine the relative risks associated with each function and sub function of the system. Risk assessments may need to be performed at multiple strategic points in the SDLC.
Installation Qualification (IQ) Scripts and Results	Test cases for checking that System has been installed correctly in User environment Results of executing scripts • Deviations from expected results (if any)
Operational Qualification (OQ) Scripts and Results	Test cases for checking that System does what it is intended to do in User environment Results of executing scripts Deviations from expected results (if any)
SOPs (Standard Operating Procedures), Training Material, and Training Records	Documented procedures for users, system administrators, and IT related functions such as Backup & Restore and Archiving. Training records must be kept to show the appropriate people were trained in the correct operation of the system.
Performance Qualification (PQ) Scripts and Results	Test cases for checking that System does what it is intended to do with trained people following SOPs in the production environment even under worst case conditions Results of executing scripts Deviations from expected results (if any)
Validation Report	The Validation Report includes: A review of all activities and documents against the Validation Plan Evidence that deviations (if any) have been addressed and the system is validated The plan for ongoing activities to maintain validation status

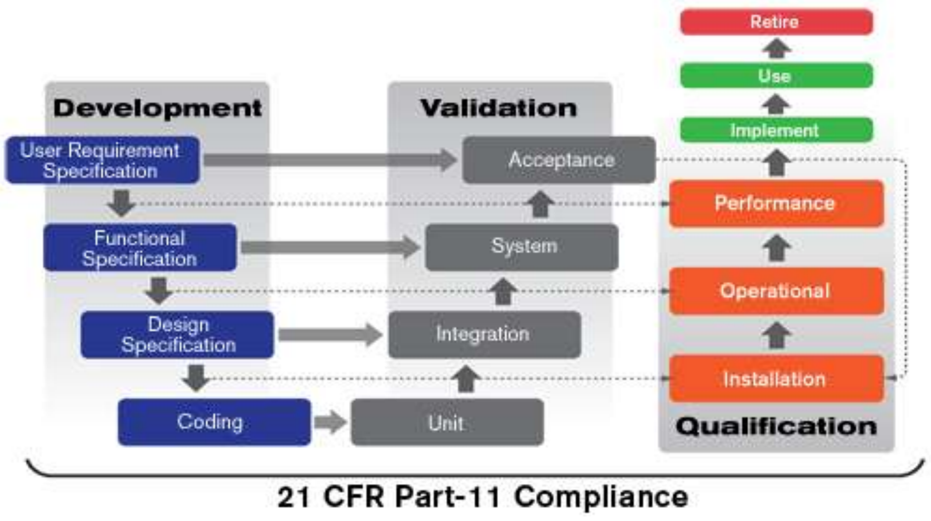
Table No1: List of the documents for the Validation of a Computer System application.

COMPUTER SYSTEM VALIDATION

Computer System Validation is not just testing or producing some paper as for qualification evidence, but an unique opportunity to verify the level of Compliance of outstanding process, provided that you have the appropriate competences and resources.[12-13]

The Computer System Validation includes:

- Definition of company policy and procedures
- Analysis of User Requirements
- Software selection and supplier qualification/audits
- GxP Risk Assessment of computerized systems and outstanding process
- Planning and Reporting (VMP, VP, VR)
- Review of supplier system specifications
- Definition of of Test Plans, Test Specifications (IQ, OQ, PQ) and support during test execution
- Definition of ongoing SOPs for computer systems
- Validation of Legacy Systems
- Network Validation
- Preparation of regulatory inspections



COMPUTERIZED SYSTEM VALIDATION QUALITY SYSTEM

The validation of a computerized control system to FDA requirements can be broken down into a number of phases which are interlinked with the overall project program. A typical validation program for a control system also includes the parallel design and development of control and monitoring instrumentation. The validation of a computer system involves four fundamental tasks. Defining and adhering to a validation plan to control the application andsystem operation, including GMP risk and validation rationale Documenting the validation life-cycle steps to provide evidence of system accuracy, reliability, repeatability, and data integrityConducting and reporting the qualification testing required to achieve validation statusUndertaking periodic reviews throughout the operational life of the system to demonstrate that validation status is maintained Other key considerations include the following.

Traceability and accountability of information to be maintained throughout validation life-cycle documents (particularly important in relating qualification tests to defined requirements). The mechanism (e.g., matrix) or establishing and maintaining requirements traceability should document where a user-specified requirement is met by more than one system function or covered by multiple tests All qualification activities must be performed in accordance with predefined protocols/test procedures that must generate sufficient approved documentation to meet the stated acceptance criteria. Provision of an incident log to record any test deviations during qualification and any system discrepancies, errors, or failures during operational use, and to manage the resolution of such issues A typical Quality System includes the following phases.

* Definition phase
* Commissioning and in-place qualification phase
* Ongoing maintenance phase

Definition Phase

Validation starts at the definition (conceptual design) phase because the FDA expects to see documentary evidence that the chosen system vendor and the software proposed meets the customer ’s predefined selection criteria. Vendor acceptance criteria, which must be defined by the customer, should typically include the following.

The Vendor’s Business Practices

· Vendor certification to an approved QA standard. Certification may be a consideration when selecting a systems vendor. Initiative which promotes the use of international standards to improve the quality management of software development shall be considered.
· Vendor Audit by the customer to ensure company standards and practices are known and are being followed.

- Vendor end user support agreements.
- Vendor financial stability.
- Biography for the vendor 's proposed project personnel (interviews also should be considered).
- Checking customer references and visiting their sites should be considered.

The Vendor's Software Practices

- Software development methodology.
- Vendor's experience in using the project software including: operating system software; application software; "off-the-shelf" and support software package (e.g., archiving, networking, batch software).
- Software performance and development history
- Software updates
- The vendor must make provision for source code to be accessible to the end user (e.g., have an escrow or similar agreement) and should provide a statement to this effect. Escrow is the name given to a legally binding agreement between a supplier and a customer which permits the customer access to source code, which is stored by a third party organization. The agreement also permits the customer access to the source code should the supplier become bankrupt. Vendor acceptance can be divided into these areas:
- Vendor prequalification (to select suitable vendors to receive the Tender enquiry package)
- Review of the returned Tenders
- Audit of the most suitable vendor(s)

Other documentation produced during the definition phase includes the URS, standard specifications and Tender support documentation. The Tender enquiry package must be reviewed by the customer prior to issue to selected vendors. This review, called SQ, is carried out to ensure that the customer 's technical and quality requirements are fully addressed.

- **System Development Phase**

The system development phase is the period from Tender award to delivery of the control system to site. It can be subdivided into four sub phases.

- Design agreement
- Design and development

- Development testing
- Predelivery or FAT

The design agreement phase comprises the development and approval of the system vendor's Functional Design Specification, its associated FAT, Specification and the Quality Plan for the project. These form the basis of the contractual agreement between the system vendor and the customer. The design and development phase involves the development and approval of the detailed system (hardware and software) design and testing specifications.

The software specifications comprise the Software Design Specification and its associated Software Module Coding. The hardware specifications comprise the Computer Hardware Design Specification and its associated Hardware Test Specification and Computer Hardware Production.

The development testing phase comprises the structured testing of the hardware and software against the detailed design specifications starting from the lowest level and working up to a fully integrated system. The systems vendor must follow a rigorous and fully documented testing regime to ensure that each item of hardware and software module developed or modified performs the function(s) required without degrading other modules or the systems as a whole.

The predelivery acceptance phase comprises the FAT, which is witnessed by the customer, and the DQ review by the customer to ensure the system design meets technical (system functionality and operability) and quality (auditable, structured documentation) objectives.
Throughout the system development phase, the systems vendor should be subject to a number of quality audits by the customer, or their nominated agents, to ensure that the Quality Plan for the project is being complied with and that all documentation is being completed correctly. In addition, the vendor should conduct internal audits, and the reports should be available for inspection by the customer. The systems vendor also must enforce a strict change control procedure to enable all mediations and changes to the system to be thoroughly designed, tested, and documented. Change control is a formal system by which qualified representatives of appropriate disciplines review proposed or actual changes that might affect a validated status. The intent is to determine the need for action that would ensure and document that the component or system is maintained in a validated state. The audit trail documentation introduced and maintained by the Quality Plan and the test documentation can

be used as evidence by the customer during the FDA's inspections that the system meets the functionality required. In particular, the test and change control documentation will demonstrate a positive, thorough, and professional approach to validation.

Commissioning and In-PlaceQualification Phase

The commissioning and qualification phase encompasses the System Commissioning on site, Site Acceptance Testing, IQ, OQ, and, where applicable, PQ activities for the project. The most important part of this phase must be identified as qualification based on system specification documentation. The system installation and operation must be confirmed against its documents. All system adjustments and changes occuring in this phase must result in updating of the corresponding specification document. It is an assurance when building a reliable system base document in support of a life cycle approach during a phase that most last minute changes are discovered. No benefit of any life cycle approach can be obtained when the system and its documentation do not match after completion of this phase.

Ongoing Maintenance Phase

The term maintenance does not mean the same when applied to hardware and software. The operational maintenance of hardware and software are different because their failure/error mechanisms are different. Hardware maintenance typically includes preventive hardware maintenance actions, component replacement, and corrective changes. Software maintenance includes corrective, perfective, and adaptive maintenance but does not include preventive maintenance actions or software component replacement.

Changes made to correct errors and faults in the software are corrective maintenance. Changes made to the software to improve the performance, maintainability, or other attributes of the software system are perfective maintenance. Software changes to make the software system usable in a changed environment are adaptive maintenance. When changes are made to a software system, sufficient regression analysis and testing should be conducted to demonstrate that portions of the software not involved in the change were not adversely impacted. This is in addition to testing that evaluates the correctness of the implemented change(s).

The specific validation effort necessary for each change is determined by the type of change, the development products affected, and the impact of those products on the operation of the system. All proposed modifications, enhancements, or additions to the system should be assessed to determine the effect each change would have on the entire system. This information should determine the extent to which verification and/or validation tasks need to be iterated. Documentation should be carefully reviewed to determine which documents have been impacted by a change. All approved documents (e.g., specifications, user manuals, drawings, etc.) that have been affected should be updated in accordance with the applicable site or corporate change management procedures. Specifications should be updated before any change is implanted.

Existing System Validation

For retrospective validation, emphasis is put on the assembly of appropriate historical records for system definition, controls, and testing. Existing systems that are not well documented and do not demonstrate change control and/or do not have approved test records cannot be considered as candidates for retrospective validation as defined by the regulatory authorities. Consequently, for a system that is in operational use and does not meet the criteria for retrospective validation, the approach should be to establish documented evidence that the system does what it purports to do. To do this, an initial assessment is required to determine the extent of documented records that exist. Existing documents should be collected, formally reviewed, and kept in a system "history file" for reference and to establish the baseline for the validation exercise. From the document gap analysis the level of redocumenting and retesting that is necessary can be identified and planned.

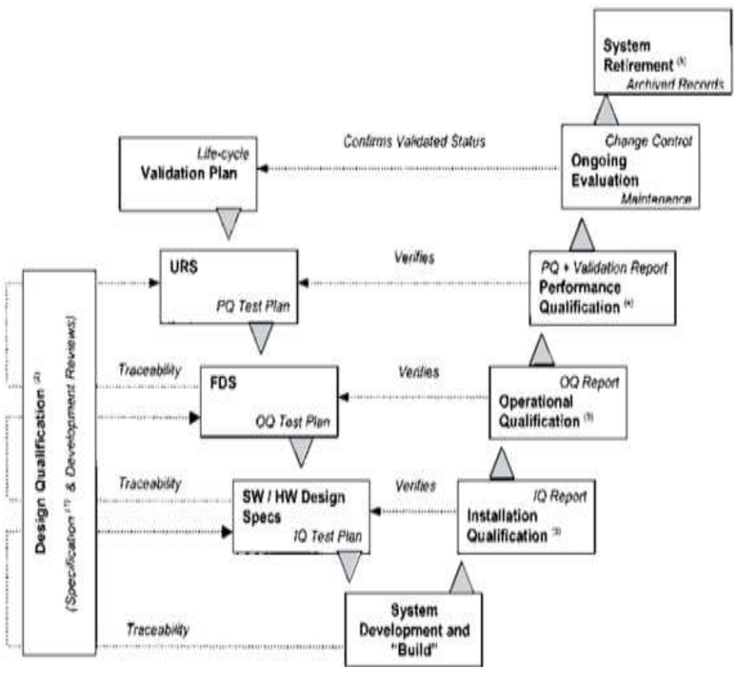


Figure1: Framework for system validation.

SOFTWARE VALIDATION

The Quality System regulation treats “verification” and “validation” as separate and distinct terms. On the other hand, many software engineering journal articles and textbooks use the terms verification and validation interchangeably, or in some cases refer to software “verification, validation, and testing (VV&T)” as if it is a single concept, with no distinction among the three terms.[14]

Software verification provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase. Software verification looks for consistency, completeness, and correctness of the software and its supporting documentation, as it is being developed, and provides support for a subsequent conclusion that software is validated.

Software testing is one of many verification activities intended to confirm that software development output meets its input requirements. Other verification activities include various static and dynamic analyses, code and document inspections, walkthroughs, and other techniques.

Software validation is a part of the design validation for the project, but is not separately defined in the Quality System regulation. FDA considers software validation to be “confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.” In practice, software validation activities may occur both during as well as at the end of the software development life cycle to ensure that all requirements have been fulfilled. Since software is usually part of a larger hardware system, the validation of software typically includes evidence that all software requirements have been implemented correctly and completely and are traceable to system requirements.

A conclusion that software is validated is highly dependent upon comprehensive software testing, inspections, analyses, and other verification tasks performed at each stage of the software development life cycle.

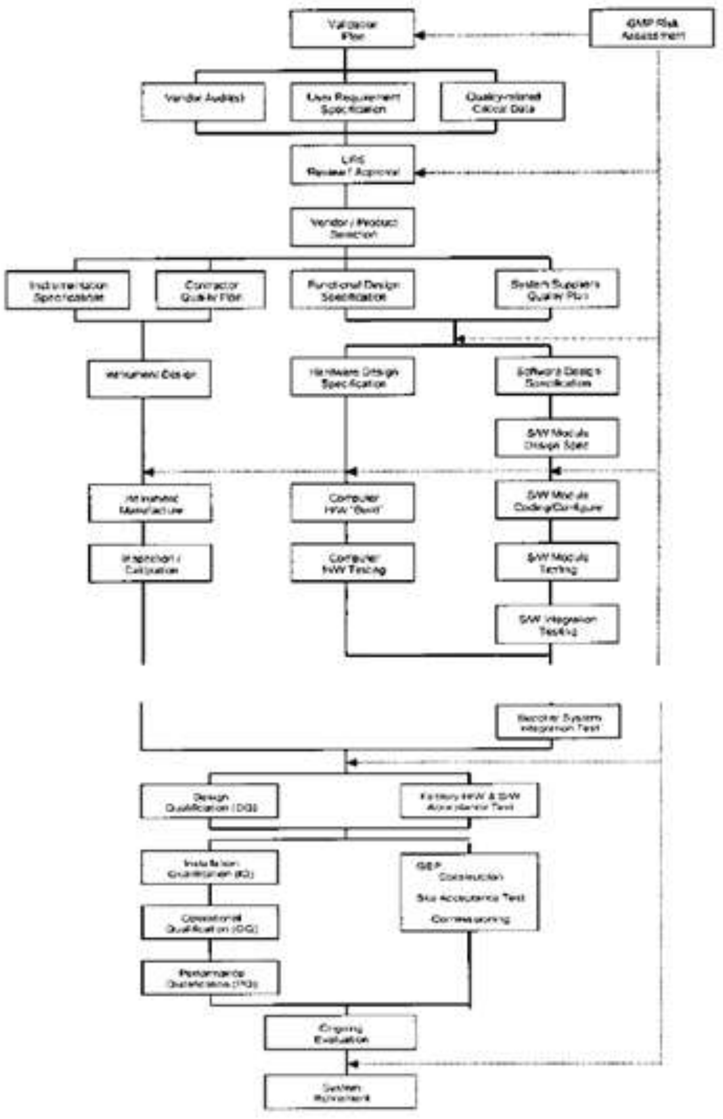

Validation Plan
GMP Risk Assessment
Vendor Audit(s)
User Requirement Specification
Quality-related Critical Data
URS Review / Approval
Vendor / Product Selection
Instrumentation Specification
Contractor Quality Plan
Functional Design Specification
System Suppliers Quality Plan
Instrument Design
Hardware Design Specification
Software Design Specification
S/W Module Design Spec
Instrument Manufacture
Computer H/W "Build"
S/W Module Coding/Configure
Inspection / Calibration
Computer H/W Testing
S/W Module Testing
S/W Integration Testing
Computer Systems Integration Test
Design Qualification (DQ)
Factory H/W & S/W Acceptance Test
Installation Qualification (IQ)
GEP Construction
Site Acceptance Test
Commissioning
Operational Qualification (OQ)
Performance Qualification (PQ)
Ongoing Evaluation
System Retirement

Figure 2: Computer system development and validation process

Software verification and validation are difficult in nature because a developer cannot test forever, and it is hard to know how much evidence is enough. In large measure, software validation is a matter of developing a "level of confidence" that the application meets all requirements and user expectations for the software automated functions. Measures such as defects found in specifications documents, estimates of defects remaining, testing coverage, and other techniques are all used to develop an acceptable level of confidence before shipping the product. The level of confidence, and therefore the level of software validation, verification, and testing effort needed, will vary depending upon the application. Many firms have asked for specific guidance on what the FDA expects them to do to ensure compliance with the Quality System regulation with regard to software validation. Validation of software has been conducted in many segments of the software industry for almost three decades. Due to the great variety of pharmaceuticals, medical devices, processes, and manufacturing facilities, it is not possible to state in one document all of the specific validation elements that are applicable. However, a general application of several broad concepts can be used successfully as guidance for software validation. These broad concepts provide an acceptable framework for building a comprehensive approach to software validation. Requirements SpecificationWhile the Quality System regulation states that design input requirements.

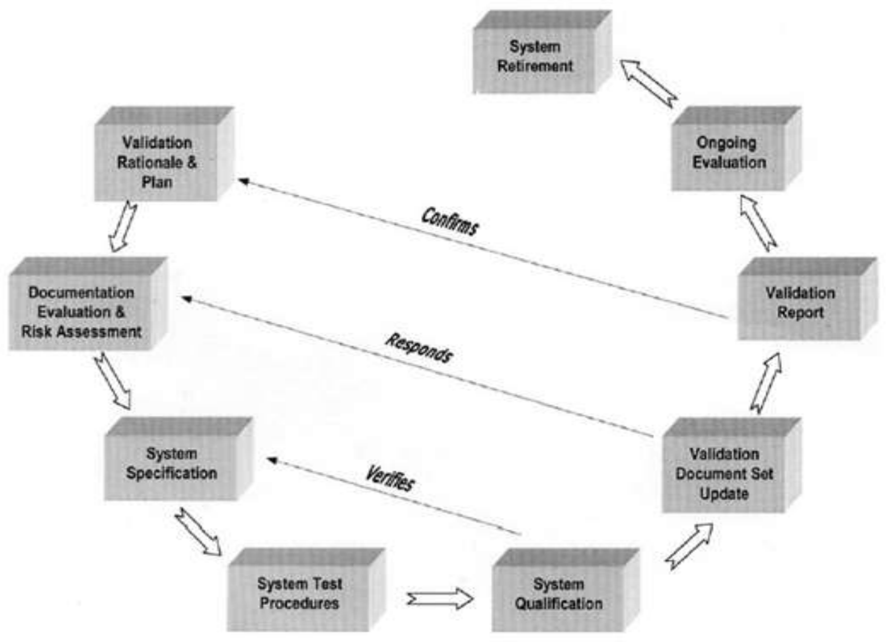


Figure 3: A validation cycle for existing systems.

must be documented, and that specified requirements must be verified, the regulation does not further clarify the distinction between the terms “requirement” and “specification.” A requirement can be any need or expectation for a system or for its software. Requirements reflect the stated or implied needs of the customer, and may be market-based, contractual, or statutory, as well as an organization’s internal requirements. There can be many different kinds of requirements (e.g., design, functional, implementation, interface, performance, or physical requirements). Software requirements are typically derived from the system requirements for those aspects of system functionality that have been allocated to software. Software requirements are typically stated in functional terms and are defined, refined, and updated as a development project progresses. Success in accurately and completely documenting software requirements is a crucial factor in successful validation of the resulting software.

A specification is defined as “a document that states requirements.” It may refer to or include drawings, patterns, or other relevant documents and usually indicates the means and the criteria whereby conformity with the requirement can be checked. There are many different kinds of written specifications, e.g., system requirements specification, software requirements specification, software design specification, software test specification, software integration specification, etc. All of these documents establish “specified requirements” and are design outputs for which various forms of verification are necessary.

A documented software requirements specification provides a baseline for both validation and verification. The software validation process cannot be completed without an established software requirements specification.

Software quality assurance needs to focus on preventing the introduction of defects into the software development process and not on trying to “test quality into” the software code after it is written. Software testing is very limited in its ability to surface all latent defects in software code. For example, the complexity of most software prevents it from being exhaustively tested. Software testing is a necessary activity. However, in most cases software testing by itself is not sufficient to establish confidence that the software is fit for its intended use. In order to establish that confidence, software developers should use a mixture of methods and techniques to prevent software errors and to detect software errors that do occur. The “best mix” of methods depends on many factors including the development environment, application, size of project, language, and risk.

Time and Effort

To build a case that the software is validated requires time and effort. Preparation for software validation should begin early, i.e., during design and development planning and design input. The final conclusion that the software is validated should be based on evidence collected from planned efforts conducted throughout the software life cycle.

Software Life Cycle

Software validation takes place within the environment of an established software life cycle. The software life cycle contains software engineering tasks and documentation necessary to support the software validation effort.

In addition, the software life cycle contains specific verification and validation tasks that are appropriate for the intended use of the software. No one life cycle model can be recommended for all software development and validation project, but an appropriate and practical software life cycle should be selected and used for a software development project.

Plans

The software validation process is defined and controlled through the use of a plan. The software validation plan defines "what" is to be accomplished through the software validation effort. Software validation plans are a significant quality system tool. Software validation plans specify areas such as scope, approach, resources, schedules and the types and extent of activities, tasks, and work items.

Procedures

The software validation process is executed through the use of procedures. These procedures establish "how" to conduct the software validation effort. The procedures should identify the specific actions or sequence of actions that must be taken to complete individual validation activities, tasks, and work items.

Software Validation After a Change

Due to the complexity of software, a seemingly small local change may have a significant global system impact. When any change (even a small change) is made to the software, the validation status of the software needs to be re-established. Whenever software is changed, a validation analysis should be conducted not just for validation of the individual change but also to determine the extent and impact of that change on the entire software system. Based on this analysis, the software developer should then conduct an appropriate level of software regression testing to show that unchanged but vulnerable portions of the system have not been adversely affected. Design controls and appropriate regression testing provide the confidence that the software is validated after a software change.

Validation Coverage

Validation coverage should be based on the software's complexity and safety risk and not on firm size or resource constraints. The selection of validation activities, tasks, and work items should be commensurate with the complexity of the software design and the risk associated with the use of the software for the specified intended use. For lower risk applications, only baseline validation activities may be conducted. As the risk increases, additional validation activities should be added to cover the additional risk. Validation documentation should be sufficient to demonstrate that all software validation plans and procedures have been completed successfully.

Flexibility and Responsibility

Specific implementation of these software validation principles may be quite different from one application to another. The manufacturer has flexibility in choosing how to apply these validation principles, but retains ultimate responsibility for demonstrating that the software has been validated. Software is designed, developed, validated, and regulated in a wide spectrum of environments, and for a wide variety of applications with varying levels of risk. In each environment, software components from many sources may be used to create the software (e.g., in-house developed software, off-the-shelf software, contract software,shareware). In addition, software components come in many different forms (e.g., application software, operating systems, compilers, debuggers, configuration management tools, and many more). The validation of software in these environments can be a complex undertaking; therefore, it is appropriate that all of these software validation principles be considered when designing the software validation process. The resultant software validation process should be commensurate with the safety risk associated with the system, device, or process.

Software validation activities and tasks may be dispersed, occurring at different locations and being conducted by different organizations. However, regardless of the distribution of tasks, contractual relations, source of components, or the development environment, the manufacturer retains ultimate responsibility for ensuring that the software is validated. Software validation is accomplished through a series of activities and tasks that are planned and executed at various stages of the software development life cycle. These tasks may be one-time occurrences or may be iterated many times, depending on the life cycle model used and the scope of changes made as the software project progresses.[1]

SOFTWARE LIFE CYCLE ACTIVITIES

Software developers should establish a software life cycle model that is appropriate for their product and organization. The software life cycle model that is selected should cover the software from its birth to its retirement. Activities in a typical software life cycle model include the following.[15-16]

- Quality Planning
- System Requirements Definition

- Detailed Software Requirements Specification
- Software Design Specification
- Construction or Coding
- Testing
- Installation
- Operation and Support
- Maintenance
- Retirement

Verification, testing and other tasks that support software validation occur during each of the above activities. A life cycle model organizes these software development activities in various ways and provides a framework for monitoring and controlling the software development project.

For each of the software life cycle activities, there are certain “typical” tasks that support conclusion that the software is validated. However, the specific tasks to be performed, their order of performance, and the iteration and timing of their performance will be dictated by the specific software life cycle model that is selected and the safety risk associated with the software application. For very low risk applications, certain tasks may not be needed at all. However, the software developer should at least consider each of these tasks and should define and document which tasks are or are not appropriate for their specific application.

Quality Planning

Design and development planning should culminate in a plan that identifies necessary tasks, procedures for anomaly reporting and resolution, necessary resources, and management review requirements, including formal design reviews. A software life cycle model and associated activities should be identified, as well as those tasks necessary for each software life cycle activity. The plan should include.

- The specific tasks for each life cycle activity.
- Enumeration of important quality factors.
- Methods and procedures for each task
- Task acceptance criteria
- Criteria for defining and documenting outputs in terms that will allow evaluation of their conformance to input requirements
- Inputs for each task

- Outputs from each task
- Roles, resources, and responsibilities for each task
- Risks and assumptions
- Documentation of user needs Management must identify and provide the appropriate software development environment and resources. Typically, each task requires personnel as well as physical resources.

The plan should identify the personnel, the facility and equipment resources for each task, and the role that risk (hazard) management will play. A configuration management plan should be developed that will guide and control multiple parallel development activities and ensure proper communications and documentation. Controls are necessary to ensure positive and correct correspondence among all approved versions of the specifications documents, source code, object code, and test suites that comprise a software system. The controls also should ensure accurate identification of, and access to, the currently approved versions.

Procedures should be created for reporting and resolving software anomalies found through validation or other activities. Management should identify the reports and specify the contents, format, and responsible organizational elements for each report. Procedures also are necessary for the review and approval of software development results, including the responsible organizational elements for such reviews and approvals.

Requirements

Requirement development includes the identification, analysis, and documentation of information about the application and its intended use. Areas of special importance include allocation of system functions to hardware/software, operating conditions, user characteristics, potential hazards, and anticipated tasks. In addition, the requirements should state clearly the intended use of the software. The software requirements specification document should contain a written definition of the software functions. It is not possible to validate software without predetermined and documented software requirements.

Typical software requirements specify the following.

- All software system inputs
- All software system outputs
- All functions that the software system will perform

- All performance requirements that the software will meet
- The definition of all external and user interfaces, as well as any internal software-to-system interfaces
- How users will interact with the system
- What constitutes an error and how errors should be handled
- Required response times
- The intended operating environment
- All ranges, limits, defaults, and specific values that the software will accept
- All safety related requirements, specifications, features, or functions that will be implemented in software.

Software safety requirements are derived from a technical risk management process that is closely integrated with the system requirements development process. Software requirement specifications should identify clearly the potential hazards that can result from a software failure in the system as well as any safety requirements to be implemented in software. The consequences of software failure should be evaluated, along with means of mitigating such failures (e.g., hardware mitigation, defensive programming, etc.). From this analysis, it should be possible to identify the most appropriate measures necessary to prevent harm. A software requirements traceability analysis should be conducted to trace software requirements to (and from) system requirements and to risk analysis results. In addition to any other analyses and documentation used to verify software requirements, a formal design review is recommended to confirm that requirements are fully specified and appropriate before extensive software design efforts begin. Requirements can be approved and released incrementally, but care should be taken that interactions and interfaces among software (and hardware) requirements are properly reviewed, analyzed, and controlled.

Design

The decision to implement system functionality using software is one that is typically made during system design. Software requirements are typically derived from the overall system requirements and design for those aspects in the system that are to be implemented using software. There are user needs and intended uses for a finished product, but users typically do not specify whether those requirements are to be met by hardware, software, or some combination of both. Therefore, software validation must be considered within the context of the overall design validation for the system. A documented requirements specification represents the user's needs and intended uses from which the product is developed. A primary goal of software validation is to then demonstrate that all completed software products comply with all documented software and system requirements. The correctness and completeness of both the system requirements and the software requirements should be addressed as part of the design validation process for that application. Software validation includes confirmation of conformance to all software specifications and confirmation that all software requirements are traceable to the system specifications.

Confirmation is an important part of the overall design validation to ensure that all aspects of the design conform to user needs and intended uses. In the design process, the software requirements specification is translated into a logical and physical representation of the software to be implemented. The software design specification is a description of what the software should do and how it should do it. Due to complexity of the project or to enable persons with varying levels of technical responsibilities to clearly understand design information, the design specification may contain both a high-level summary of the design and detailed design information. The completed software design specification constrains the programmer/coder to stay within the intent of the agreed upon requirements and design. A complete software design specification will relieve the programmer from the need to make ad hoc design decisions.

The software design needs to address human factors. Use error caused by designs that are either overly complex or contrary to users' intuitive expectations for operation is one of the most persistent and critical problems encountered by the FDA. Frequently, the design of the software is a factor in such use errors.

Human factor engineering should be woven into the entire design and development process, including the design requirements, analysis, and tests. Safety and usability issues should be considered when developing flow charts, state diagrams, prototyping tools, and test plans. Also, task and function analysis, risk analysis, prototype tests and reviews, and full usability tests should be performed. Participants from the user population should be included when applying these methodologies.

The software design specification should include.

· Software requirements specification, including predetermined criteria for acceptance of the software.

- Software risk analysis
- Development procedures and coding guidelines (or other programming procedures)
- Systems documentation (e.g., a narrative or a context diagram) that describes the systems context in which the program is intended to function, including the relationship of hardware, software, and the physical environment
- Hardware to be used
- Parameters to be measured or recorded
- Logical structure (including control logic) and logical processing steps (e.g., algorithms)
- Data structures and data flow diagrams
- Definitions of variables (control and data) and description of where they are used
- Error, alarm, and warning messages
- Supporting software (e.g., operating systems, drivers, other application software)
- Communication links (links among internal modules of the software, links with the supporting software, links with the hardware, and links with the user)
- Security measures (both physical and logical security)

The activities that occur during software design have several purposes. Software design evaluations are conducted to determine if the design is complete, correct, consistent, unambiguous, feasible, and maintainable. Appropriate consideration of software architecture (e.g., modular structure) during design can reduce the magnitude of future validation efforts when software changes are needed. Software design evaluations may include analysis of control flow, data flow, complexity, timing, sizing, memory allocation, criticality analysis, and many other aspects of the design. A traceability analysis should be conducted to verify that the software design implements all of the software requirements. As a technique for identifying where requirements are not sufficient, the traceability analysis should also verify that all aspects of the design are traceable to software requirements. An analysis of communication links should be conducted to evaluate the proposed design with respect to hardware, user, and related software requirements. The software risk analysis should be re-examined to determine whether any additional hazards have been identified and whether any new hazards have been introduced by the design. At the end of the software design activity, a Formal Design Review should be conducted to verify that the design is correct, consistent, complete, accurate, and testable before moving to implement the design. Portions of the design can be approved and released incrementally for implementation, but care should be taken that interactions and communication links among various elements are properly reviewed, analyzed, and controlled.

Most software development models will be iterative. This is likely to result in several versions of both the software requirements specification and the software design specification. All approved versions should be archived and controlled in accordance with established configuration management procedures.

Construction or Coding

Software may be constructed either by coding (i.e., programming) or by assembling together previously coded software components (e.g., from code libraries, off the- shelf software, etc.) for use in a new application.

Coding is the software activity where the detailed design specification is implemented as source code. Coding is the lowest level of abstraction for the software development process. It is the last stage in decomposition of the software requirements where module specifications are translated into a programming language. Coding usually involves the use of a high-level programming language, but may also entail the use of assembly language (or microcode) for time-critical operations. The source code may be either compiled or interpreted for use on a target hardware platform.

Decisions on the selection of programming languages and software build tools (assemblers, linkers, and compilers) should include consideration of the impact on subsequent quality evaluation tasks (e.g., availability of debugging and testing tools for the chosen language).
Some compilers offer optional levels and commands for error checking to assist in debugging the code. Different levels of error checking may be used throughout the coding process, and warnings or other messages from the compiler may or may not be recorded. However, at the end of the coding and debugging process, the most rigorous level of error checking is normally used to document what compilation errors still remain in the software. If the most rigorous level of error checking is not used for final translation of the source code, then justification for use of the less rigorous translation error checking should be documented. Also, for the final compilation, there should be documentation of the compilation process and its outcome, including any warnings or other messages from the compiler and their resolution, or justification for the decision to leave issues unresolved. Firms frequently adopt specific coding guidelines that establish quality policies and procedures related to the

software coding process. Source code should be evaluated to verify its compliance with specified coding guidelines. Such guidelines should include coding conventions regarding clarity, style, complexity management, and commenting. Code comments should provide useful and descriptive information for a module, including expected inputs and outputs, variables referenced, expected data types, and operations to be performed. Source code should also be evaluated to verify its compliance with the corresponding detailed design specification. Modules ready for integration and test should have documentation of compliance with coding guidelines and any other applicable quality policies and procedures.
Source code evaluations are often implemented as code inspections and code walkthroughs. Such static analyses provide a very effective means to detect errors before execution of the code. They allow for examination of each error in isolation and can also help in focusing later dynamic testing of the software. Firms may use manual (desk) checking with appropriate controls to ensure consistency and independence. Source code evaluations should be extended to verification of internal linkages between modules and layers (horizontal and vertical interfaces) and compliance with their design specifications. Documentation of the procedures used and the results of source code evaluations should be maintained as part of design verification.

Testing by the Software Developer

Software testing entails running software products under known conditions with defined inputs and documented outcomes that can be compared to their predefined expectations. It is a time-consuming, difficult, and imperfect activity. As such, it requires early planning in order to be effective and efficient. Test plans and test cases should be created as early in the software development process as feasible. They should identify the schedules, environments, resources (personnel, tools, etc.), methodologies, cases (inputs, procedures, outputs and expected results), documentation, and reporting criteria. The magnitude of effort to be applied throughout the testing process can be linked to complexity, criticality, reliability, and/or safety issues.

Software test plans should identify the particular tasks to be conducted at each stage of development and include justification of the level of effort represented by their corresponding completion criteria. An essential element of a software test case is the expected result. It is the key detail that permits objective evaluation of the actual test result. This necessary testing information is obtained from the corresponding predefined definition or specification. A software specification document must identify what, when, how, why, etc., is to be achieved with an engineering (i.e., measurable or objectively verifiable) level of detail in order for it to be confirmed through testing. The real effort of effective software testing lies in the definition of what is to be tested rather than in the performance of the test. Once the prerequisite tasks (e.g., code inspection) have been successfully completed, software testing begins. It starts with unit level testing and concludes with system level testing. There may be a distinct integration level of testing. A software product should be challenged with test cases based on its internal structure and with test cases based on its external specification.

These tests should provide a thorough and rigorous examination of the software product's compliance with its functional, performance, and interface definitions and requirements.

User Site Testing

Testing at the user site is an essential part of software validation. The Quality System regulation requires installation and inspection procedures (including testing where appropriate) as well as documentation of inspection and testing to demonstrate proper installation. Likewise, manufacturing equipment must meet specified requirements, and automated systems must be validated for their intended use. Terminology regarding user site testing can be confusing. Terms such as beta test, site validation, user acceptance test, installation verification, and installation testing have all been used to describe user site testing.

The term "user site testing" encompasses all of these and any other testing that takes place outside of the developer's controlled environment. This testing should take place at a user 's site with the actual hardware and software that will be part of the installed system configuration.

The testing is accomplished through either actual or simulated use of the software being tested within the context in which it is intended to function. User site testing should follow a predefined written plan with a formal summary of testing and a record of formal acceptance. Documented evidence of all testing procedures, test input data, and test results should be retained. There should be evidence that hardware and software are installed and configured as specified. Measures should ensure that all system components are exercised during the testing and that the versions of these components are those specified. The testing plan should specify testing throughout the full range of operating conditions and should specify continuation 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 more normal activities.

During user site testing, records should be maintained of both proper system performance and any system failures that are encountered. The revision of the system to compensate for faults detected during this user site testing should follow the same procedures and controls as for any other software change. The developers of the software may or may not be involved in the user site testing. If the developers are involved, they may seamlessly carry over to the user's site the last portions of design-level systems testing. If the developers are not involved, it is all the more important that the user have persons who understand the importance of careful test planning, the definition of expected test results, and the recording of all test outputs.

Maintenance and Software Changes In addition to software verification and validation tasks that are part of the standard software development process, the following additional maintenance tasks should be addressed.

Software Validation Plan Revision

For software that was previously validated, the existing software validation plan should be revised to support the validation of the revised software. If no previous software validation plan exists, such a plan should be established to support the validation of the revised software.[16]

Anomaly Evaluation

Software organizations frequently maintain documentation, such as software problem reports that describe software anomalies discovered and the specific corrective action taken to fix each anomaly. Too often, however, mistakes are repeated because software developers do not take the next step to determine the root causes of problems and make the process and procedural changes needed to avoid recurrence of the problem. Software anomalies should be evaluated in terms of their severity and their effects on system operation and safety, but they should also be treated as symptoms of process deficiencies in the quality system. A root-cause analysis of anomalies can identify specific quality system deficiencies.

Where trends are identified (e.g., recurrence of similar software anomalies), appropriate corrective and preventive actions must be implemented and documented to avoid further recurrence of similar quality problems.

Problem Identification and Resolution Tracking

All problems discovered during maintenance of the software should be documented. The resolution of each problem should be tracked to ensure it is fixed, for historical reference, and for trending.

Task Iteration

For approved software changes, all necessary verification and validation tasks should be performed to ensure that planned changes are implemented correctly, all documentation is complete and up to date, and no unacceptable changes have occurred in software performance.

VALIDATION AND SECURITY MEASURES FOR PHARMACEUTICAL DATA PROCESSING

BENEFITS OF QUALIFICATION

Software validation is a critical tool used to assure the quality of software and software automated operations. Software validation can increase the usability and reliability of the application, resulting in decreased failure rates, fewer recalls and corrective actions, less risk to patients and users, and reduced liability to manufacturers. Software validation can also reduce long-term costs by making it easier and less costly to reliably modify software and revalidate software changes. Software maintenance can represent a very large percentage of the total cost of software over its entire life cycle.

An established comprehensive software validation process helps to reduce the long-term cost ofsoftware by reducing the cost of validation for eachsubsequent release of the software. The level of validationeffort should be commensurate with the risk posed by theautomated operation. In addition to other risk factors,such as the complexity of the process software and thedegree to which the manufacturer is dependent upon thatautomated process to produce a safe and effectiveproduct, determine the nature and extent of testingneeded as part of the validation effort. Documentedrequirements and risk analysis of the automated processhelp to define the scope of the evidence needed to showthat the software is validated for its intended use.[17]

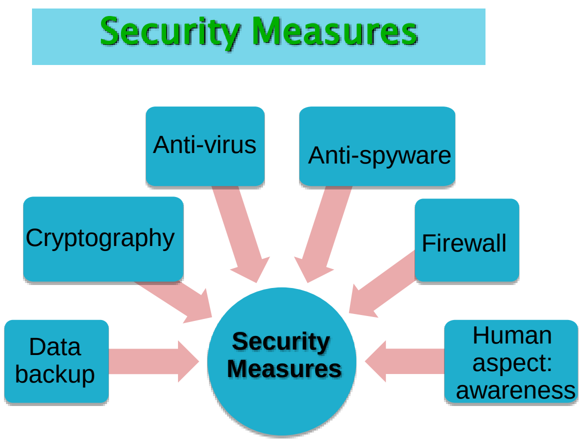


Figure 4: Diagram presenting Computer Security Measures

GENERAL TESTINGALL SYSTEMS

All computers or automated controllers that are used in or for the production of pharmaceuticals or medical devices require qualification prior to their use in the process. Computers need qualification just as any other system or component of the manufacturing process does. The main difference between general equipment qualification and CSV is that, as mentioned above, there are two stages for the completion of a computer or computer system. These include the software and hardware aspects of the system. The first part of any CSV program is the qualification referred to as structural; the second phase of the qualification is the functional aspect of the systems. The structural qualification and portion of the program is focused on the development of the software, while the functional qualification focuses on the actual operation or function of the system. Chapter 46 deals with the structural qualification aspect, this chapter will concern itself primarily with the functional aspects of the qualification program. As with software qualification, the hardware can be divided into various stages. Each stage requires a qualification phase in order to demonstrate that it is complete.[18]

These stages can be divided as follows

- Development—establishing system requirements
- Build—obtaining the correct components per specifications
- Implement (this is where the full qualification program is required)
- Operation (part of the full qualification program where a qualified state needs to be maintained)
- Retirement—decommissioning the system for replacement by another system

Functional qualification follows the same pattern as any other pharmaceutical equipment or systems qualification. Thus, in order to perform a functional qualification as described in chapter 9 of this book, an IQ and an OQ are necessary (Refer to chapter 9 for general IQ and OQ requirements). The IQ provides verification that the system is installed according to a written preapproved plan. The same is true for the OQ. All pharmaceutical systems should have the following.

- Vendor qualification via an audit
- User specification
- Design specifications However, in addition to the "usual" requirements for IQs and OQs the qualification of computer systems requires some additional items.
- Some of these are:
- Verification of system security
- Controlled access to the program
- Levels of access—e.g., an operator is allowed to input data but the supervisor is allowed to
- approve the data
- Protection of the system from outside interference (e.g., no access via phone lines or the internet) Note: Usually an intranet connection will be allowed.
- Ability to track all entries (audit trail) into the system—this includes the date, the person making the entry and why the entry is made or changed.1,4

Black box vs. White box testing

There are two methods of testing automated control systems. These are referred to as "white box" and "black box" testing. Both means of qualification are used for systems at or above Level 2 of the GAMP classification of computer systems. The difference between "white" and "black" box testing is in the level of testing of the software. Black box testing is primarily functional testing while white box testing includes a review of the source code (of the software program) as well as the means of code development. When doing black box testing the operation of each portion of the software is tested. In addition, the testing establishes that each function necessary for the correct operation of the unit(s). Typically, the black box testing grows exponentially with the amount of I/O while the white box testing grows linearly.[19]

GENERAL DOCUMENTATION

When beginning a CSV program, as with other qualification programs, certain documents need to be either prepared or collected. Since the qualification will involve components not usually seen and usually not accessible having the correct documents at the very beginning of the project will help assure its success. The list below covers the main documents to be prepared or collected

Prepare

CSQMP

User requirements

Functional specifications

Traceability matrix (Note: To be prepared AFTER all specifications and protocols have been collected and developed but BEFORE protocol execution).

SOPs (to include the "How to Prepare" SOPs)

a. System setup/installation

b. Data collection and handling

c. System maintenance

d. Backup

e. Recovery

i. Backup

ii. Crash

iii. Jam/freeze

f. Contingency plans (emergencies)

g. Security

h. Change control

i. Storage

6. Protocols

a. Commissioning

b. IQ

c. OQ

d. PQ (as necessary)

Collect

1. Ladder logic—As necessary for PLCs

2. Design or Vendor specifications for each component— part of the system (network interfacing, MMI)
3. Software version to be installed
4. Software source code (or 3rd party agreement)

After the documents are prepared and or collected, you are ready to begin the qualification program itself. (Note: this is assuming that the structural qualification has been completed and is acceptable). As with all qualification programs the commissioning phase usually is the first "field" effort undertaken. (Note: This follows the FAT and SAT portion of the program.) The commissioning portion of the qualification can be performed, at least in part, during the installation of the system. For example, while the lines are being run to the field instruments the loop checks can be performed.

A loop check is a check of continuity (and thereby function) of the connection between a field instrument and the controller. It is far simpler to perform and document the loop check as each loop is being installed rather than after the system is intact and ready to operate.

Other items that can be performed during the installation or as part of the commissioning phase are.

- Instruments adjusted/calibrated (loop checks)
- Ambient conditions
- Temperature
- Humidity
- Alarms and events (general testing—operational testing is left to the OQ phase of the qualification)
- Graphics
- Data base location
- Network configuration

The next phase of the qualification is the IQ. As pointed out in Chapter 9 this may be done at the same time or before the commissioning phase of the program. Either during or even before the IQ is started the structural phase of software testing is completed. Since the structural testing includes items such as the vendor audit, the code review, this part of the qualification must be completed prior to any functional or OQ testing as discussed below.

The general IQ consists of the following verifications.

Specific tests will be pointed out later for each of the types of automated systems.[20-21]

1. **List all components**

a. *Input devices—HMI and/or MMI*

i. Keyboard

ii. Mouse

iii. External devices

- Field instruments,
- External drives,
- Monitors, etc.

b. Output devices

i. Screen

ii. External data device—hard drive

iii. Printer

iv. Filed instruments

c. Data storage devices

i. Hard drives

ii. MP3

iii. Floppy drives

iv. Flash cards

v. Tape/CD/DVD (backup)

2. List type of hardware

a. Mother board—chip type

b. Controller cards

i. Video

ii. Sound

iii. I/O

c. Internal drives

i. Floppy

ii. CD

d. Output connections

i. USB

ii. Parallel

iii. Firewire

iv. Serial

v. S Video

vi. Other monitor connections

e. Network cards (discussed below)

3. Check for

a. Tight connections

b. Correct component type

c. Installed in the correct location (as applicable)

d. Model as per specifications

4. Power (source and distribution)

a. Volts

b. Current

c. Stability

d. Surge protection

5. Software (includes the structural testing—see below)

a. Version installed

b. Source code verification

i. Annotation

ii. Dead code

iii. Vendor testing verification (part of vendor audit)

c. Compliance to good software preparation The OQ follows the IQ. This set of testing cannot start until the IQ is complete or until the QU gives approval

In the case of automated systems, the completion of the IQ is necessary since the system will not function as specified without all components being installed correctly. While the system may seem to operate, some functions will be compromised if a component is lacking. This may not be immediately apparent but will, in the long term, compromise the final product. An example of this would be a missing printer. The controller would run, the machines would run, but the output data would not be able to be expressed or recorded. This may cause the system to shut down or to store the information that cannot be printed. It would be printed later (if possible). This may compromise the next lot of material being produced since it will get the incorrect label or printout.

It is during the OQ testing that the software undergoes its functional testing.
In general, the OQ will have the following general tests.

1**. Prepare test of each component listed in the IQ**
a. Meets design specifications
b. Meets functional specifications hardware
c. Power limits—may be included as part of the PQ (below)
i. Recovery after power loss
ii. Power line stability
d. Environmental stress
e. Alarms
f. All component functions over their full range
g. Software
i. version verification
ii. Ladder logic or source code review
h. Input limits (boundaries)
i. Functional testing
2. RFI—that is a radio frequency should not cause the controller to malfunction (allow incorrect data in or out)—e.g., a walkie-talkie (handheld radios).
3. EMI—a magnetic field should not interfere with the data integrity—e.g., an electric drill
4. I/O integrity
5. Calibration
6. Software
a. Compete structural testing
b. Functional testing
i. Restart after shutdown
ii. Restart after power loss
iii. All major operations function and results are appropriate If a PQ needs to be performed (as it most likely will), The following is a list of general tests that should be included.

1. **Power failure recovery**—computer and process equipment (as seen above this may be done as part of the OQ)

a. Recovery after power loss

2. Security—system accessibility

a. Password challenge

b. Security challenge

c. Biometric security

d. Levels of access

3. Archive/retrieve data in real time

4. Produce batch report

5. I/O Loops operation

6. Data lines transmission

7. General data integrity

8. Interference between programs/components

9. Software

a. Full operation of all functions in conjunction with the entire system

b. Stress the software boundaries

c. Non interference between modules or other programs 1,2,4,8

COMPUTER SYSTEM VALIDATION DELIVERABLES

- Validation plan
- System Impact Assessment
- Requirements specifications (URS, FRS, FS/TS)
- System configuration specifications
- Test plan
- Installation Qualification (IQ) testing and Reports
- Operation Qualification (OQ) testing and Reports
- Performance Qualification (PQ) testing and Reports
- Assessment for compliance with regulations pertaining to electronic records and signatures (e.g., 21 CFR Part 11)
- Traceability matrix
- Quality assurance review
- Validation summary report
- Draft/ Review of Standard Operating Procedures (SOPs)

- GMP, GDP Training

CONCLUSION

The Quality System regulation requires that "when computers or automated data processing systems are used as part of production or the quality system, the manufacturer shall validate computer software for its intended use according to an established protocol." This has been a regulatory requirement for GMP since 1978. In addition to the above validation requirement, computer systems that implement part of a regulated manufacturer's production processes or quality system (or that are used to create and maintain records required by any other FDA regulation) are subject to the Electronic Records, Electronic Signatures regulation. This regulation establishes additional security, data integrity, and validation requirements when records are created or maintained electronically. These additional Part 11 requirements should be carefully considered and included in system requirements and software requirements for any automated record keeping systems. The article is concluded about the e-documents. Now a days in pharmaceutical industry e-documents has superseded almost all type documents. The guideline told that every process should be properly validated. The e-documents are generated by mainly computer system and software. So, I tried to cover the validation of computer system and software in brief in the article. The another aspect is security of the e-documents. The article covered the above aspect in brief.

REFERENCES

1. Agalloco, J., Carleton, F.J. Validation of pharmaceutical processes. Third edition. New York. Informa healthcare., 607-627.
2. General Principles of Validation, Food and Drug Administration, Drug Evaluation and Research; Rockville, Maryland; 1987; (http://www.fda.gov/cder/guidance/pv.htm).
3. Kaner, Cem; Falk, Jack; Nguyen, Hung Quoc; Testing Computer Software; New York: John Wiley & Sons., 1999.
4. Perry, William E.; Rice, Randall W.; Surviving the Top Ten Challenges of Software Testing, New York: Dorset House., 1997
5. Deutsch, Michael S. ; Software Verification and Validation: Realistic Project Approaches, New Jersey: Prentice-Hall., 1982
6. 21 CFR Part 11 Electronic Records; Electronic Signatures; Final Rule; Department of Health and Human Services, Food and Drug Administration; Federal Register, March 20, 1997; 62(54).

7. Lande, Victoria V., An Update on Electronic Records and Compliance, Seminar presented by NuGenesis Technologies, 2001.
8. Deutsch, Michael S. ; Software Verification and Validation: Realistic Project Approaches, New Jersey: Prentice-Hall., 1982
9. IEEE Standard of Software Verification and Validation; Institute of Electrical and Electronics Engineers (IEEE), New York., 1998
10. Deutsch, Michael S. ; Software Verification and Validation: Realistic Project Approaches, New Jersey: Prentice-Hall., 1982
11. Petschenik, Nathan; System Testing with an Attitude; New York: Dorset House; expected to be published 2004.
12. IEEE Standard of Software Verification and Validation; Institute of Electrical and Electronics Engineers (IEEE), New York., 1998
13. GAMP 4 Guide for Validation of Automated Systems (2001), International Society of Pharmaceutical Engineers (ISPE), Tampa, FL and Brussels, Belgium
14. Agalloco, J., Carleton, F.J. Validation of pharmaceutical processes. Third edition. New York. Informa healthcare., 607-627.
15. Boylan, J.C. Encyclopedia of pharmaceutical technology. 2nd edition. Volume-3. Newyork. Marcel dekker, Inc., 2917-2932
16. Collefello, J.S. (1998) Introduction to software verification and validation. Available from URL: sei.cmu.edu/library/abstracts/reports/89cm013.cfm Accesed on 29th july, 2011.
17. Nash, R.A., Watcher, A.H. Pharmaceutical process validation. Third edition. Marcel Dekker Inc., 529-550, 560-599
18. Pharmaceutical validation of software available from URL pharmaceuticalvalidation.blogspot.com/2009/12/general-principles-of-software.html Accessed on 29th july, 2011.
19. Sharma, P.P. Validation in pharmaceutical industry. Delhi. Vandana publication pvt.ltd. Page no.: 383-396
20. Weinberg, S. Good laboratory practise regulations. Third edition. New York. Marcel Dekker Inc., 165-179.
21. Validation of computerized system available from URL scribd.com/doc/93410573/13243184-Validation-Part-8 Accessed on 20th sep, 2011.