*Lebedynets V. O., Chornyi D. S.**
The National University of Pharmacy, Kharkiv, Ukraine
*** LLC "Business Center Pharmacy", Vyshgorod, Ukraine**

FDA guidance describes how certain provisions of the medical device Quality System regulation apply to software and software validation system. Planning, verification, testing, traceability, configuration management, and many other aspects of good software engineering discussed in this guidance are important activities that together help to support a final conclusion that software is validated. FDA guidance recommends an integration of software life cycle management and risk management activities.

Based on the intended use and the safety risk associated with the software to be developed, the software developer should determine the specific approach, the combination of techniques to be used, and the level of effort to be applied.

While FDA guidance does not recommend any specific life cycle model or any specific technique or method, it does recommend that software validation and verification activities be conducted throughout the entire software life cycle.

FDA guidance applies to:

- Software used as a component, part, or accessory of a medical device;
- Software that is itself a medical device (e.g., blood establishment software);
- Software used in the production of a device (e.g., programmable logic controllers in manufacturing equipment);
- Software used in implementation of the device manufacturer's quality system (e.g., software that records and maintains the device history record).

This guidance provides useful information and recommendations to the following individuals:

- Persons subject to the medical device Quality System regulation;
- Persons responsible for the design, development, production of medical device software;
- Persons responsible for the design, development, production, or procurement of automated tools used for the design, development, or manufacture of medical devices or software tools used to implement the quality system itself.

Any software used to automate any part of the device production process or any part of the quality system must be validated for its intended use, as required by 21 CFR §820.70(i). This requirement applies to any software used to automate device design, testing, component acceptance, manufacturing, labeling, packaging, distribution, complaint handling, or to automate any other aspect of the quality system.

In addition, computer systems used to create, modify, and maintain electronic records and to manage electronic signatures are also subject to the validation requirements (See 21 CFR §11.10(a)).

Such computer systems must be validated to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records.

Software may be developed in-house or under contract.

However, software is frequently purchased off-the-shelf for a particular intended use.

All production and/or quality system software, even if purchased off-the-shelf, should have documented requirements that fully define its intended use, and information against which testing results and other evidence can be compared, to show that the software is validated for its intended use.

The use of off-the-shelf software in automated medical devices and in automated manufacturing and quality system operations is increasing. Off-the-shelf software may have many capabilities, only a few of which are needed by the device manufacturer. Device manufacturers are responsible for the adequacy of the software used in their devices, and used to produce devices.

When device manufacturers purchase "off-the-shelf" software, they must ensure that it will perform as intended in their chosen application. In the table №1 lists the general principles that should be considered for the validation of software.

Table №1

REQUIREMENTS	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 (Ref: 21 CFR 820.3(z) and (aa) and 820.30(f) and (g)).
DEFECT PREVENTION	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 lifecycle.
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.
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 – 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 devices, 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.
INDEPENDENCE OF REVIEW	Validation activities should be conducted using the basic quality assurance precept of "independence of review". Self-validation is extremely difficult. When possible, an independent evaluation is always better, especially for higher risk applications. Some firms contract out for a third-party independent verification and validation, but this solution may not always be feasible. Another approach is to assign internal staff members that are not involved in a particular design or its implementation, but who have sufficient knowledge to evaluate the project and conduct the verification and validation activities. Smaller firms may need to be creative in how tasks are organized and assigned in order to maintain internal independence of review.
FLEXIBILITY AND RESPONSIBILITY	Specific implementation of these software validation principles may be quite different from one application to another. The device 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 devices with varying levels of risk. FDA regulated medical device applications include software that: • Is a component, part, or accessory of a medical device; • Is itself a medical device; • Is used in manufacturing, design and development, or other parts of the quality system. In each environment, software components from many sources may be used to create the application (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 device manufacturer or specification developer retains ultimate responsibility for ensuring that the software is validated.

In addition to the above validation requirement, computer systems that implement part of a device 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. (See 21 CFR Part 11.).

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 all aspects of medical device design, laboratory testing and analysis, product inspection and acceptance, production and process control, environmental controls, packaging, labeling, traceability, document control, complaint management, and many other aspects of the quality system.

Increasingly, automated plant floor operations can involve extensive use of embedded systems in programmable logic controllers, digital function controllers, statistical process control, supervisory control and data acquisition, robotics, human-machine interfaces, input/output devices and computer operating systems.

Software tools are frequently used to design, build, and test the software that goes into an automated medical device.

Many other commercial software applications, such as word processors, spreadsheets, databases, and flowcharting software are used to implement the quality system.

All of these applications are subject to the requirement for software validation, but the validation approach used for each application can vary widely.

Whether production or quality system software is developed in-house by the device manufacturer, developed by a contractor, or purchased off-the-shelf, it should be developed using the basic principles outlined elsewhere in this guidance.

The device manufacturer has latitude and flexibility in defining how validation of that software will be accomplished, but validation should be a key consideration in deciding how and by whom the software will be developed or from whom it will be purchased.

The software developer defines a life cycle model.

Validation is typically supported by:

- verifications of the outputs from each stage of that software development life cycle;
- checking for proper operation of the finished software in the device manufacturer's intended use environment.