Validation Views

Perspectives on Good Validation Practices

Len Grunbaum

Welcome to ***Validation Views: Perspectives on Good Validation Practices***. The purpose of this section of ***The Compliance Advisor*** is to promote ***"good validation practices"***; that is, practices that are technically sound, practical, cost-beneficial and compliant with industry standards.

We will analyze validation issues through the prism of the knowledge, insights and experiences of individuals who are involved with, responsible for and/or affected by computer systems validation.

Each issue of ***Validation Views*** will contain various perspectives on a particular validation topic. The topics will include general information, such as *"What do I have to do to comply with industry standards and regulations?"*, as well as the specific, such as *"What are my responsibilities regarding ownership of vendor source code?"*

We will present several perspectives on the topic, using input from Regulatory Affairs professionals, GLP/GCP compliance personnel, quality assurance specialists, and system developers and users. We will also solicit input from FDA personnel, as well as consultants and vendors to the industry.

I look forward to providing ***Validation Views*** as an effective means of promoting ***good validation practices***.

Len Grunbaum is the Director of Validation Consulting Services at META Solutions, Inc. He has over 25 years of experience in software development, computer auditing, security, and systems validation.

Having Validated and Unvalidated Software in a Production Environment

Len Grunbaum

A recent situation encountered by a client raises an interesting question. *Can validated and unvalidated software co-exist in the same environment?* The specific circumstance involved unused (and therefore, unvalidated) modules in a vendor-supplied package that required validation. Two scenarios warrant a more detailed analysis:

- Given a broad definition of *"environment"* (e.g., client server environment) and the impracticality of constructing an environment in which only validated software exists, how can you assure that the data quality and integrity associated with validated software are not compromised by the unvalidated software?
- What validation procedures should be employed if a vendor-supplied package contains functions or modules that are not used (or even required)?

In its broadest sense, *"environment"* refers to the entirety of the software, both local and distributed, that supports the respective data processing operation. Included are the following types of software:

- Application software that acquires, records, analyzes, stores and summarizes regulated data
- Systems software, such as operating systems and network software
- Configurable software used to develop applications, such as database management systems
- Utility software that perform non-application specific operations, such as security, back-up, etc.

See **Validated Software**, *page 8*

Validated Software

continued from page 7

From a regulatory standpoint, regulated application software and selected utility software (e.g., logical security, file backup, change control) should be validated. Additionally, the use of configurable software (such as the use of SAS to write analysis programs) should be validated. However, a typical environment may also include non-regulated systems, such as financial and human resources applications, which do not typically require validation. There are few organizations that have one data processing environment for software that requires validation and a separate environment for software that doesn't. *What issues should be investigated to assure that the quality and integrity of data are not compromised?*

I feel that two critical issues should be considered: 1) ***logical security*** and 2) ***recoverability***. The specific objective of logical security in this context is to ensure that the various files, databases, program libraries, etc., cannot be accessed either via the operating system or any of the unvalidated software in the environment. This emphasis on security must be reflected in the system design and in the structural and functional test plans.

The second critical issue is the ability to recover and restore the functionality of computer systems in a timely and effective manner. This is significant because unvalidated software may be unstable and therefore the risk of uncontrolled processing and loss of resources is increased. Thus, design and testing emphasis must be placed on the ability to re-create any aspect of previous processing (e.g., restore a data file or program, re-generate a report, re-establish an audit trail, etc.) following a system or processing failure.

The second scenario involves unused modules that exist in the *"environment"* of a vendor-supplied package. This situation is quite common. A vendor designs software to meet a general need so that it is unlikely that any given organization will use all of the functionality that is designed in the software. In the case of a regulated system, one of two options are available: 1) obtain a version of the software without the unnecessary code; or 2) identify the unused code and develop appropriate restrictions to assure that it cannot be accessed.

While the first option is *"cleaner"*, it may not be practical from the vendor's standpoint (the vendor now has to maintain and support another version of the software). The second option necessitates *"validating"* the unused code in the following manner: 1) implementing the software with the ability to *"turn off"* unused code in an appropriate fashion, 2) designing and implementing tests to confirm that the unused code cannot be accessed, and 3) including in the system and validation documentation an identification of the functionality that is not being used and detailed substantiation of the testing performed.

This option assumes that 1) the software can be designed to enable the user to *"turn off"* unneeded functionality, and 2) the code is developed in a manner that facilitates the identification of the software functionality. If either of these assumptions are not applicable, then serious consideration should be given to obtaining a package more suited to exercising control over unused code. However, if these assumptions do apply, then the case can be made that the software, including the unused code, can be validated for an intended purpose.

To summarize, the existence of validated and unvalidated software in one *"environment"* results in specific regulatory implications. However, the application of common sense and good business practices will help to decrease potential risks.

483 Monitor

continued from page 6

Final Note:

The inspection report contained a two page, single-spaced summary of the exit discussion. On most points, laboratory personnel did not comment on the presented observations. I raise this since I think it is important to be proactive during the exit discussion. Argue politely on issues in contention or indicate corrective action on issues of fact. It is to your benefit.