DQRI Software Validation Group
Chapel Hill
North Carolina

July 8, 2004

Dockets Management Branch (HFA-305)
Food and Drug Administration
5630 Fishers Lane, rm. 1061
Rockville, MD 20852

Re: Docket No. 2004N-0133

To Whom It May Concern:

The Data Quality Research Institute (DQRI) is dedicated to the research and development of robust methodologies and approaches for assessing and ensuring data quality at all stages of clinical research and development. In the summer of 2003 the Data Quality Research Institute (DQRI) formed a working group to investigate risk-based approaches to validating software for clinical trials. This effort was in part a response to the FDA's current good manufacturing practice (CGMP) initiative, which places risk assessment at the center of product quality regulation. Following the agency's lead, the working group set out to clarify what a risk-based approach would look like in the domain of clinical trials software. In taking on this task, the working group also hoped to address some of the confusions and controversies that have surrounded the topic of software validation.

Our comments are organized around several specific questions included in the announcement of the public meeting on 21 CFR part 11.

Question A2 – "We are interested in comments on whether revisions to definitions in part 11 would help clarify a narrow approach and suggestions for any such revisions."

We are concerned that the language employed in part 11 has unintentionally fostered an overly simplistic and rigid conception of software development and software validation. Part 11 calls for "validation of systems." This simple phrase implies that validation is a single thing with a well-defined boundary. Unfortunately (from a regulatory point of view) that is not the case for software systems. There are numerous procedures that can be applied to safeguard software quality, and there are dozens of named methodologies currently in active use, each of which brings together a different collection of concepts, procedures, and deliverables to produce high-quality software. Each methodology embodies a strategy for safeguarding software quality. All of these methodologies have been shown to produce high-quality software in specific cases, but no consensus has emerged about which methodologies work best under what circumstances.

Many of these methodologies are effectively unusable within the regulated industries due to perceived regulatory constraints. The attempt (in guidance documents) to clarify what "validation" means was helpful in moving the topic forward, but it has created a de facto standard language for software development in the regulated industries. This occurred in large part because software validation cannot be separated from specific software methodologies. The deliverables itemized in guidance documents – plans, requirement specifications, structural and functional tests, code inspections, change control procedures, risk assessments, and so forth – are not just validation artifacts, they are deliverables that outline a development methodology.

In addition to presenting validation in an overly simplistic way, the current language conceals the variation in software systems and the implications this has for designing appropriate development and validation procedures. "Software" encompasses an enormous range of systems, including large scale applications, operating systems, third-party components, configuration files, software development tools, one-time statistical programs, vendor supplied software, electronic spreadsheets, and electronic documents that make use of macro programming. It is difficult, if not impossible, to make useful generalizations about validation procedures across this broad range of systems and development contexts.

Finally, the current language obscures the fact that in some contexts there are effective ways of controlling software-related risks that are outside of the software development process. In particular, data validation may be a viable strategy for addressing system risks. If software is used to produce a locked database that can be validated by comparisons to source documents (such as medical records), then validating the data directly can be an excellent strategy for controlling software risks. Part 11 makes no allowance for this strategy as an alternative to controls implemented during development of the system.

We believe that the across-the-board requirement for systems to be validated should be removed from Part 11, for the following reasons:

- Software validation cannot be separated from software development methodologies.
- Software validation has no meaning that can be usefully generalized across diverse development methodologies.
- Software development methodologies are still emerging; there is no consensus about which methodologies work best under what circumstances.
- Software validation is not the only strategy available for controlling software-related risks.

We respectfully recommend that, instead of requiring validation, part 11 should require risk assessment to identify system-related risks and to justify the measures adopted to control those risks.

Question D3 – "In what ways can part 11 discourage innovation?"

By including an across-the-board requirement that covered software must be validated, and publishing guidance that defined software validation in terms of specific deliverables, the FDA established a de facto standard software development methodology for covered systems, regardless of context and regardless of risk profile. The vocabulary used in the guidance documents became the working language for software compliance auditors and it framed their expectations of what a compliant system should look like. Innovation was discouraged by the simple fact that there was no agreed-upon language for justifying anything less or anything different than what was called for in the guidance documents.

Ironically, while the "de facto standard" methodology was being established in the regulated industries, many leading software providers were abandoning the very same procedures. These providers were adopting new "agile" methodologies, in part to address the failures and inefficiencies of the older techniques. And while the FDA has publicly called for industry to embrace new technologies and accelerate timelines, the regulatory entrenchment of a rigid, dated development methodology has created substantial barriers to doing just that.

Prevailing regulatory expectations for software validation have discouraged innovation in several ways:

1. Regulated organizations are reluctant to adopt new software development methodologies because of the substantial risk that the resulting software will fail to pass a compliance audit. Many of the newer methodologies bear little resemblance to the de facto standard that governs the expectations of software compliance auditors.

2. The barrier to using new methodologies means that regulated organizations cannot benefit from improvements in cost and efficiency obtainable through these techniques. As a result, software development in regulated organizations is more expensive than in other organizations.

3. Regulated organizations are reluctant to purchase new software tools from vendors because many of these tools have been developed using new methodologies – which again raises the risk of failing a software compliance audit.

What the industry needs now is (a) regulation and guidance that promotes validation at the appropriate level for the system, and (b) agreement between the agency and industry about how to assess software-related risks and how to justify efforts to control those risks (through software validation, data validation, or any other valid procedure).

Question D5 – "What risk-based approaches would help to ensure that electronic records have the appropriate levels of integrity and authenticity elements and that electronic signatures are legally binding and authentic?"

Risk assessment can provide a means of justifying diverse approaches to software development and validation. Industry and the agency must agree on a common framework for performing risk assessment. We need a standard vocabulary for doing risk assessment, particularly in the area of software validation.

It is important to recognize that the FDA began recommending risk assessment years ago. In fact, the agency has stated in a variety of contexts over the years that it is necessary to consider system-related risks when determining the appropriate extent of system validation. There are some important reasons why risk-based evaluation has not been adopted to the extent it should be.

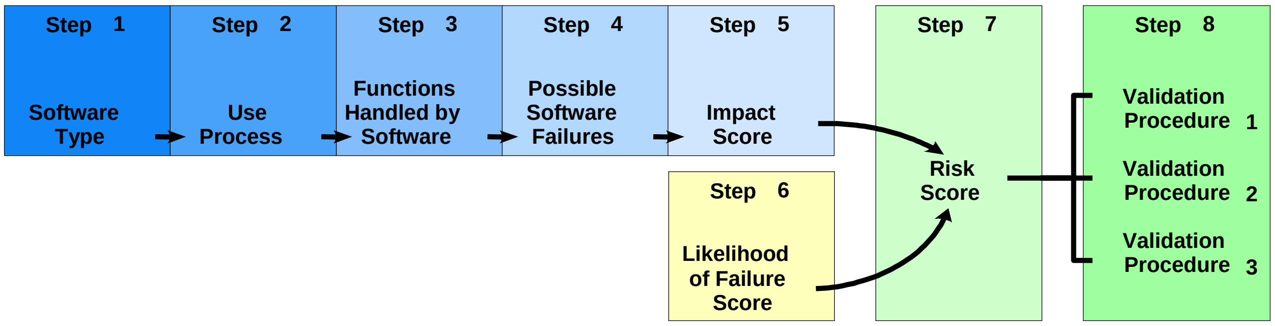
First of all, in previous discussions risk assessment has typically been discussed in the context of other procedures as if it were a potential element of a validation strategy, but not necessarily a means of determining and justifying a validation strategy.

Second, and more importantly, there has been no adequate vocabulary available to describe how to assess risks and justify validation decisions. As we have already noted, the guidance documents provided an extensive vocabulary for describing validation procedures. Unfortunately they provided no comparably rich vocabulary for justifying decisions about validation procedures. As a result, the only "safe" decision was to adopt the agency's validation language in its entirety, applying every concept, procedure, and deliverable to every covered system. There was no agreed upon language for justifying anything less or anything different.

If risk assessment is going to provide a solution to this problem, we must do two things. We must reposition risk assessment as preceding and driving decisions about software validation. And we must develop a common language for talking about the relationships between assessed risks and strategies for safeguarding software quality.

In other words, it is not enough to tell people that they should perform risk assessment. We must provide a common vocabulary for doing so. The area where we should be developing a preferred vocabulary is not software development, but risk assessment. This language gap must be filled before we can expect conversations about software validation to become more productive.

A standard risk model is needed. The diagram below shows a current draft of the DQRI Risk Assessment Model, which is under development.



DQRI is developing this model as a contribution to the ongoing industry discussions about risk-based approaches to software validation. Regardless of the model that is ultimately adopted, we believe that industry consensus is needed in four critical areas.

1. What are the common types of software used in clinical trials, and what are the risks typically associated with them? (Step 1 in the DQRI model.)

2. What outcomes should be considered in assessing the impact of possible software failures? (Step 5)

   In step five the list of possible failures is assessed and assigned an impact score. In the context of clinical trials, what outcomes should we consider when deciding how serious a problem we have if a system fails? We know that injury to patients is a central consideration, but what about other outcomes, like delays in completing studies, or violations of confidentiality? How should such outcomes be weighed when making decisions about software validation? We need to arrive at some kind of consensus to these questions if we are going to have any hope that our risk assessments are accepted as valid outside of our own organizations.

3. What factors contribute to the likelihood of system failures? (Step 6)

   Software development projects vary tremendously on many dimensions, including the experience and expertise of the development team, the size of the problem they are addressing, the conditions under which the developers work, the resources available to them, the technologies they employ, and so on. Many of these factors have a bearing on the likelihood of system failures. One example: an inexperienced developer is more likely to introduce design flaws than an experienced developer. Another example: teams that are geographically dispersed are more likely to have problems associated with weak communication channels than teams that work together in the same space. In the context of clinical trials, what attributes of a development project affect the likelihood of a system failure? We need to reach a consensus on a basic list of factors to consider when making this assessment.

4. What are the dimensions on which validation procedures may vary depending on assessed risk scores (Step 8)

   In step eight the risk score is used to select a validation strategy from among several possible strategies that have been outlined by the organization. The idea is that higher risk scores will lead to more rigorous validation efforts. This step raises some critical questions for evaluation: What kinds of variation are permissible when tailoring software validation efforts to different levels of risk? Is there a minimum level of validation that all systems must undergo, or is it permissible for extremely low risk systems to forego validation entirely? If there is a minimum level of validation, what is it? In order for decisions about level of effort to be accepted across organizational boundaries, we need to achieve consensus on these and other questions about acceptable levels of validation.

We respectfully recommend that the agency work with industry to develop guidance on risk assessment for software used in connection with electronic records. The guidance should address:

- A risk assessment model
- Risks associated with common types of software used in clinical trials
- Outcomes that should be considered in assessing the impact of possible software failures
- Factors that contribute to the likelihood of system failures
- Dimensions on which validation procedures may vary as assessments of risk increase or decrease.

Sincerely,

DQRI Software Validation Group
Chapel Hill
North Carolina
http://www.dqri.org