Agile Development for Medical Products

Kevin Thompson, Ph.D.
www.kevinthompsonphd.com
April 20, 2018

Contents

1 Background

One of my most memorable encounters with Agile development of medical products occurred in 2011. The company's software product improved access to patient medical information by aggregating data from multiple sources. I read the engineering group's internal process documentation during my flight to their headquarters. The documents were so clear and beautifully formatted that I wondered what the company could possibly want from us.

Once I arrived, the answer emerged. In spite of the beautiful process documents, the company had been struggling to get a new product revision out the door. They had spent two years in a nightmare of analysis-paralysis and "death by scope creep" that had made it impossible for the company to finish and ship a product upgrade. They were desperate for help.

The president told me, "Just tell us what to do, and we'll do it."

I liked his attitude.

Their Agile transformation ended the logjam, and they began to ship products again.

In hindsight, one interesting aspect of the experience was that no one was terribly worried about compliance with US Food and Drug Administration regulations. They knew what the FDA wanted, and

devoted the effort required to meet the FDA requirements. This was just part of the cost of doing business.

Again, I liked their attitude. It was very matter of fact, and while they might rush to hit a deadline related to the FDA, they simply got on with it.

In the years since then, I've learned that many of our clients are not quite so blasé about regulatory issues. I've been told that many people live in constant terror of an FDA audit, even though few companies are actually shut down because of one.

The fear of audits can lead to some strange behaviors, bordering on superstition. For example, I recall many people at one client saying, "Our Quality system requires that we document everything we do in this fashion." This statement followed by a mind-numbing exposition on a hugely onerous process, in which the phrase "Quality System" appeared frequently. Much of what was said seemed more like idiosyncrasy than utility. I have a niggling suspicion that some medical company, somewhere, is terrified that re-arranging the potted plants in the lobby will bring the wrath of the FDA down upon them. I assume those plants are well cared for.

2 What the FDA Wants

The regulatory environments that I have encountered, FDA included, want to ensure that the company has a standard process, and that any conception of requirements or other important documentation is stored in some kind of document-management system, with signatures, audit trails, and change controls. Test cases must be defined and be traceable to requirements. Claims of medical effectiveness must be validated, with the evidence made readily available.

The details of what the FDA wants are spelled out in writing. (See "Title 21 Food and Drugs", often referred to 21 CFR Part 820, or as "Quality System Regulation.") The full documentation is available at www.ecfr.gov.

Much of the regulatory focus relates to *verification* and *validation*. Verification confirms that the implementation of the product aligns with its design and engineering specifications, while validation confirms that the functionality or benefits of the products are as intended.

For example, verification might confirm that the specified adhesive was used, while validation would confirm that the product adheres as intended under various environmental conditions.

One of the key elements in this CFR is the "Design history file" (DHF). This 'file' is a "compilation of records which describes the design history of a finished device." In other words, it is not literally a single document, but the collection or compilation of documents that describe the design history. This collection must be accessible at any time and must contain the complete history of information about the product's definition and development.

I am not going to address all details of how companies manage regulatory compliance, as many good articles have already been written on this topic. I am going to focus more narrowly on the relationship between the regulatory concepts of requirements and testing, and the world of Agile product development.

3 Impact of FDA Regulations on Agile Development

I am referring here to the aspects of the FDA regulations that relate to product development (not, for example, to the management of clinical trials). The regulations apply to any medical product, whether it be hardware, software, or some combination of the two.

Two key aspects of satisfying the FDA requirements are maintenance of the DHF, and the explicit connection between product requirements and test cases (often called the "traceability matrix"). Of these, the DHF is the easy part. The traceability matrix is more difficult to manage. I'll explain why.

Let's begin by looking at the Agile view of product definition.

3.1 Agile Specifications

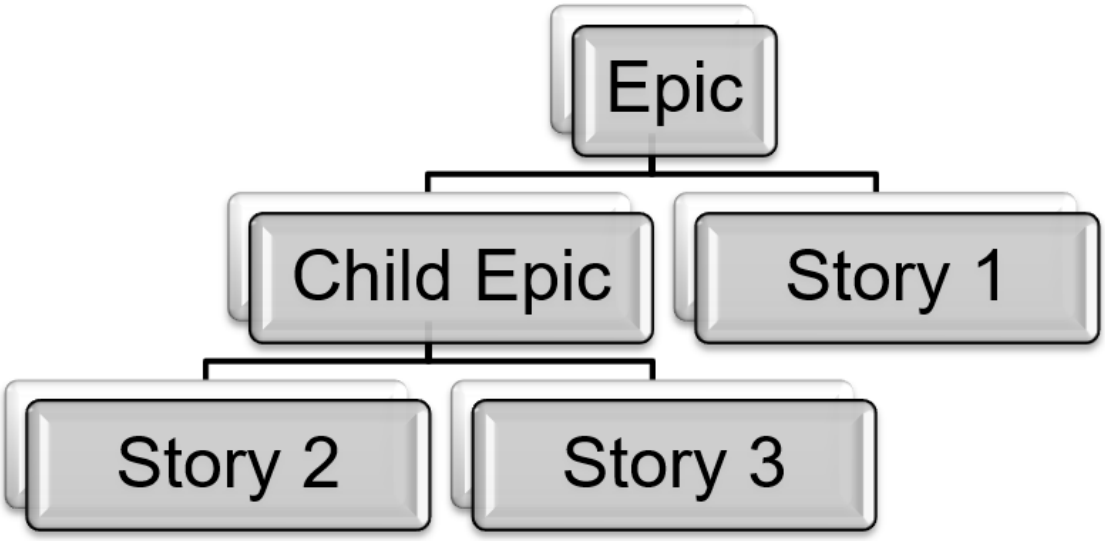
At a high level, product development is preceded by product definition. We must develop a concept for the product which incorporates basic notions of functionality, non-functional requirements, market, and so forth. This definition should contain enough information to provide direction for development work but not attempt to provide detailed specifications for all aspects of the product.

At some point, of course, we do have to provide the fine-grained specifications which are used directly in the development work. Without getting into too much detail, the specifications used in Agile processes, such as Scrum, are usually referred to as *Stories* and *Epics*.

- A Story is a specification for a deliverable that is small enough for one Scrum Team to develop and test in a few days' time. In the language I use in training, *User Stories* are specifications for deliverables that provide some kind of user experience, while *Technical Stories* are specifications for deliverables that no user of the product experiences directly. A typical Scrum Team will complete five to fifteen Stories in the course of a single two-week Sprint. By complete, I mean that the team not only develops the deliverables, but also tests it with a defined set of test cases, and fixes any defects that cause test cases to fail.
- An Epic is a specification for a deliverable that one Scrum Team cannot complete in a few days' time. An Epic is either too large to be a Story, or it defines scope that must be implemented by more than one Team.

Product Owners and Team members often write Epics to represent various aspects of the product, in order to define basic elements of the product. They later decompose those Epics into a larger number of Stories that can be implemented within Sprints by Scrum Teams.

The decomposition of Epics into Stories always creates a tree structure:



The example shows the hierarchical decomposition of one Epic into three Stories of appropriate size. In this example, we end up with an intermediate-sized Epic as well. The decomposition of an Epic into Stories very commonly creates smaller Epics along the way. There is no standard number of such levels or nodes in the tree. What is standard is that the leaf nodes are Stories of appropriate size, and all other nodes are, by definition, Epics.

(This concept of scope decomposition is similar to the Work Breakdown Structure of classic project management. Aside from terminology, the main difference is that a WBS breaks down scope for an entire *project*, while the Epic/Story decomposition breaks down various aspects of a *product*.)

To summarize, the specifications artifacts (Epics and Stories) used in Agile product development describe the scope of various deliverables, and each deliverable must satisfy a set of test cases in order to be considered complete. These fine-grained test cases are normally defined at roughly the same time as the development of the deliverable.

3.2 Product Requirements

The product requirements, in the sense relevant for regulatory purposes, are statements about the medical claims of the product that must be validated. Examples could include:

> The pump shall supply medication intravenously at rates over the desired range.
>
> The intravenous tubing shall not react chemically to any of the supported medications.
>
> The intravenous tubing shall not promote the formation of blood clots.

3.3 Engineering Requirements

Engineering requirements are statements about the design of the product. These must be documented and verified. Examples include:

> The pump shall supply medication at rates from 0.01 to 1000 ml / hour.
>
> The delivery rate shall be accurate to within 2% of the selected rate.

The device shall operate with nominal AC voltage of 120 or 240 V.

This concept of requirements contains statements that mix functional and non-functional requirements and often have no direct link to scope or deliverables. This is a fundamentally different perspective than we see in the Agile concept of specifications.

4 Reconciliation of FDA Requirements with Agile Concepts

Every Story specifies a deliverable that is tested with a set of test cases. Thus by the time product development is complete, every part of the product has been tested. The product as a whole has also been tested and confirmed to work. In other words, the deliverables of the product have been confirmed to work as intended, at all scales.

A naive conclusion is that the test cases needed to ensure that the FDA-style product requirements have been validated by doing the product-testing as described. However, the alignment of those requirements and the test cases defined for the product's deliverables is not easily defined or necessarily guaranteed.

The problem we have is that the medical requirements do not align directly with any concept of product deliverables of scope decomposition. Any one requirement may relate to many of the fine-grained deliverables, or to none in particular. Thus, the relationship between the requirements and the test cases developed for the creation of the product is a many-to-many relationship and may be incomplete.

Many-to-many relationships are difficult to manage. The tools commonly used to define Stories and associated test cases assume, at best, a purely hierarchical decomposition (one-to-many). Tools developed for support of regulatory issues make it easy to define requirement and matching test cases, but do not map readily to the day-by-day work of Agile development and Stories.

Complicating the picture further is that the medical claims relate to patient outcomes, as opposed to the physical or behavioral characteristics of the product as such. Validation of the medical claims often involves results from clinical trials, chemical analyses, and other kinds of information that likewise has no direct connection to any concept of product deliverables or their test cases.

I am not aware of any convenient ways to reconcile these two worlds. I am aware of inconvenient ways to reconcile them.

4.1 The Simple and Redundant Reconciliation

One approach is to allow the FDA requirements to drive a requirements-definition, validation, and testing solution with full traceability. In effect, we have two worlds co-existing: One is the world of FDA-regulated traceability, and another is the world of Stories and Story-level test cases.

This approach makes satisfaction of FDA traceability and validation requirements straightforward, which is critical. (Products such as Jama do this well.) The approach handles the regulatory aspects well, but risks redundancy with test cases developed at the Story level.

4.2 The Complex and Parsimonious Reconciliation

Another approach is to meet the regulatory needs by establishing the many-to-many relationships between the higher-level product requirements (medical and engineering) and the Story-level test cases. In this scenario, the Agile view of product scope decomposition is the driver, and the regulatory aspects are handled along the way by creating the linkage between requirement and Story-level test cases. Tools such as Jira and Confluence handle the Agile perspective well and can be used in this fashion. Regulatory-style requirements can be created in Confluence and then linked to deliverables and test cases in Jira.

This approach can reduce redundancy of test cases, by using at least some of the deliverables' test cases to satisfy some aspect of the validation of the medical claims.

There are down sides to this approach.

First, it tends to be onerous, requiring significant effort, such as the manual creation of hyperlinks.

Second, validation of the product's medical claims will generally require that information be provided that cannot be obtained from the test cases developed for product deliverables. That information will have to be identified, developed and supplied by whatever means makes sense.

4.3 The Ideal Reconciliation

In my opinion, the ideal reconciliation between the two worlds would enable easy definition of the product's medical requirements for regulatory purposes, and their many-to-many linkage with the Agile development of product deliverables. It would also have to support the necessity of managing requirements and validation that cannot be derived from testing the product's deliverables.

I am not aware of any solutions like this yet, but I believe that the growth of both Agile development and medical products fields will lead eventually to this result.

5 Conclusion

Effective response to the FDA regulatory requirements is no more difficult in an Agile world than in the world of classic project management. Defining product requirements, maintaining the DHF, validating the product, and providing traceability between requirements and test cases takes a certain amount of effort. The effort does not change significantly between the classic and Agile styles of managing work.

The differences mostly have to do with the tooling employed, and the path by which validation and traceability are managed. The classic style leads to more up-front work, while the Agile style is more incremental. Maintaining alignment with FDA regulations is no more difficult in the Agile world than in the classic world. Given the growing use of Agile methods for developing medical devices and software, that is a good thing.