**Four Reasons Medical Device Companies need agile development**

The waterfall style of development is so deeply engrained into the culture of most medical companies can’t imagine anything else being used to develop software that has power over human life. However I argue that precisely because of patient safety, medical device companies in particular need to adopt agile practices. I’ve seen too many bloated medical device project fail or limp across the finish line for causes that can be directly linked to the waterfall method. Specifically:

1. **Large Unprioritized Feature Sets.** The extended detailed specification development phase prior to a waterfall project’s start tends to lead towards large lists of unprioritized features that are worked on in whatever order desired by the engineers on the project. Agile development on the other hand requires features to be prioritized and then worked on in priority order. This ensures that at the end of the project if a push comes to ship, the outstanding items are the low priority bells and whistles and not something that may affect patient safety.

2. **Bloated Architecture.** The waterfall practice of trying to document and lock in a detailed design at the beginning of a project is often used by senior engineers to commit organizations to their favorite pet architecture projects. These architectures tend not to be the simplest most direct means to meet the project’s needs. Unlike the Big Up Front Design (BUFD) exercises favored by waterfall development, agile development favors lean emergent architecture concepts that recognize the fact that the best design is often not known upfront.

3. **Fragile in the Face of Inevitable Change.** Waterfall projects tend to drag on even more than the average software project (see Unprioritized Feature Sets and Bloated Architecture for some of the reasons). The long schedules mean high risk that the product developed may not meet changing market needs. When change comes to a waterfall project it’s treated as a broken promise and generates a lot of push back from engineers. Indeed the architecture built by engineering may have been built with the premise that the requirements would never change. Agile development expects changes in requirements and develops architectures and work strategies that can accommodate this change.

4. **Design Misses from Lack of Early Feedback.** In a waterfall project people tend to put their faith in the specification and trust that a year or two down the line the engineers will deliver the product they imagined. There are two problems with this strategy. One is that there are many subtleties and details that can never be fully captured in a spec (if you could you could just compile the spec). The second is that it is impossible for most people (especially marketing and customers) to envision a product until it’s built and in their hands. Both of these things can lead to significant design misses. The agile methodology addresses this problem by frequent delivery of working code.

The problem with the waterfall method of development is that it is built on the following false premises:

1. That all requirements are known up front and will never change.
2. That it is possible before a single line of code has been written to envision in detail the one true best design that will last you through the entire project.

The agile process is built on the exact antithesis of these assumptions. It believes change is inevitable and design requires multiple iterations. We all know that once something is built it’s easy to look back and say what was good and what was bad. The waterfall method works at the exact opposite side of that experience curve and tries to get you to lock in your design and requirements at the point of maximum ignorance.

Just to be clear, it’s not that the agile method is against doing as much up front design and requirements gathering as you can. It simply recognizes that it is unlikely that this exercise will be perfect and therefore a process is required to address this reality. Specifically it time boxes the upfront blue sky period and proceeds with development on the highest priority items that typically everyone agrees on. As development proceeds with tangible demos along the way, the true requirements and true correct design choices are much easier to discern.

**But what about the FDA?**

Many of us have had the unpleasant experience in waterfall oriented medical device projects of being stuck in the requirement definition phase for years as developers and marketing people struggle with the practically impossible task of trying to come up with the perfect final and unchanging detailed requirements specifications for a product that most can only dimly picture in their heads. Similarly many of us in engineering have wasted countless hours in pointless abstract philosophical technical debates that could have been resolved faster with some quick prototyping. These discussions can become quite heated because everyone sees these paper exercises as their one and only chance to affect the direction of a project that often may last years.

Companies repeatedly bang their head against this wall because they assume that it is a requirement of the FDA. However, as the FDA clearly states in the preamble to 21 CFR 820: “It is not the intent of the FDA to apply the design control requirements to the research phase. Instead the regulation requires the establishment of procedures to ensure that whatever design is ultimately transferred to production is, in fact, a design that will translate into a device that properly performs according to its intended use and user needs. To assist FDA in applying the regulation, manufacturers should document the flow of the design process so that it is clear to the FDA investigator where research is ending and development of design is beginning.”

I suggest that the FDA’s position here is entirely reasonable. I agree that the final product transferred to manufacturing needs to be well documented. However, the mistake I believe many companies make is locking themselves into a design without an adequate research phase. Most medical device companies will have an upfront specification development phase in their standard development processes. My proposal is that this phase would be more effective it were used as a specification development and prototyping phase in which agile iterations are used to zero in on the core design and features of the project before creating the traditional software requirement and design
specifications. To formalize the transition from the research/prototype phase to the formal design control phase I would have a phase review that bought off on the documentation and commented on the code developed during the phase. I would timebox this upfront phase to maybe 6-9 months for a 2 year project. The goal of this is twofold. One it serves the purpose of clarifying to FDA regulators that there will be a transition from the research phase to the formal design controls phase. Secondly it gives a timeframe as to when a phase review will be held that is similar in format to that expected by traditional waterfall advocates. This can blunt any criticism of those in the company that are skeptical of agile development.

My assumption is that at the end of the research phase almost all of the code would be transferred over for use in the production product, however even if the upfront agile development process went horribly off track and waterfall oriented reviewers in a phase review wanted to reject the entire code base, I still think there is huge benefit to this upfront exercise.

Specifically some of the benefits are:

- Make specification development discussions more productive through the agile practice of prioritized feature lists (a.k.a. the product backlog).
- The agile process of alternating between requirements gathering and code development helps keep participants more in touch with the reality of the trade off between features and schedule.
- The frequent delivery of working code can clarify discussions that often otherwise degenerate into unhelpful abstract debate.
- Development gets a head start because it goes on in parallel with other discussions.
- The agile iterative process provides a framework for marketing and engineering collaboration.

What about after formal design controls begin and design documents have been formally created? Do you go back to waterfall development? The funny thing about waterfall development is that, unlike agile development, it doesn’t actually give you guidance on how to do the actual development that goes on between phase reviews. Typically the next time the waterfall process weighs in is would be when code is handed off to the V&V group for test. So my expectation would be that at the end of the research phase iterations would continue unabated as if they phase never existed. If everything is working well, the development group will have settled into a rhythm where they’re working with marketing and a product manager to produce regular releases. I can’t imagine at this point that anyone involved will want to stop and go back to the old way where marketing and engineering only talk to each other at the beginning of the project and at the end once code is delivered.

Because in this phase you will have the traditional set of documents expected by waterfall development, the typical QA department will be satisfied. Although that documentation is at risk of needing updating, the amount of updating you can expect will be much less than that in a typical waterfall process because you have invested in an upfront iterative
development period where marketing, engineering, and the code have oriented themselves in the same direction.

**Conclusion**
When QA hears that you want to start coding before formal specifications have been generated, they’re going to think the engineers are planning to run wild and start hacking. A well run agile project couldn’t be further from this. It is a methodical process designed to addresses some of the realities of software development. Its goal is to make higher quality code that better meets the customer needs. This is as applicable to medical device projects as any other projects.

**About the author**
Mike Dobbles is the Senior Software Manager at Cardiac Science Corporation. He would love to exchange ideas about software development strategies with others in the medical device industry.