Agile in an **FDA Regulated Environment**

PATHFINDER

Table of Contents

The Increasing Importance of Software 4

The Problem with Current Development Methods 5

Waterfall Doesn't Match Reality 5

Why does Everything Change? 6

Addressing Waterfall's Shortcomings with Agile 8

Does Agile Work in Practice? 10

The FDA and Agile 11

FDA Does Not Prescribe Waterfall and in Fact Recognizes Agile as a Consensus Standard 11

Delivering on Safety, Effectiveness, and Quality with Agile 12

Summary 19

About the Author 22

About Pathfinder Software 23

Contact Pathfinder Software 23

Overview

The medical device and diagnostics industries are facing unprecedented challenges: regulatory and reimbursement model changes, dramatic technology changes, increasing technology interdependence, and rising expectations of physicians and patients.

Current "waterfall" product development methods are ill suited to dealing with the pace of change and uncertainty that product development organizations are facing.

This ebook explores how agile development methodology, prevalent in many other industries and recently recognized as a standard by the FDA, addresses waterfall development's shortcomings and can be adapted to meet the safety, reliability and regulatory needs of the medical device and diagnostics industry.

The Increasing Importance of Software

Driven by the increasing sophistication and complexity of medical devices and the environment in which they operate, an increasing share of device functionality and value depends on software. [2]

A passage from a US Food & Drug Administration report in October 2011 illustrates this increase in complexity:

"... an insulin pump in 2001 could be programmed to deliver varying amounts of insulin throughout the day. Now, a more compact pump communicates via radio frequency to a continuous glucose monitor and suggests insulin dosing using an algorithm. "[3]

Software has become a key differentiating success factor in the industry, helping companies drive more rapid innovation and creating opportunity due to the increased flexibility and agility of software-based products.

The importance of software as a driver of value creation is likely to be magnified by a series of interrelated, mutually amplifying trends:

- Reimbursement Model Changes
- Interoperability
- Wearable Sensors
- Mobile
- Consumerization
- Big Data/Healthcare Analytics

Key Takeaways:

These trends will affect different device and diagnostics firms to a different degree, but collectively, they amplify each other, and will have significant impact on how medical device and diagnostic firms create and capture value and how they compete and innovate:

1. Place a premium on human factors and good user experience: The more different types of users faced with an ever increasing set of interfaces you have, the less training is a viable option as a primary means for users to understand how to use your product. Systems and features that are hard to use or don't create immediate value for the user will be abandoned, reducing chances of upsell, renewal, and/or creating data that supports improved outcomes.
2. Increase the need to adapt more quickly to changes (in reimbursement, policy, technology, platforms, interdependence of systems) and to meet consumerized expectations. You've got less time to get it right, to create and capture value.

The Need for Speed

"Many companies believe that there is significant pressure to enter a device market early to maximize payoffs due to intense competition." [1]

Complexity in Medical Devices

"Thirty years ago, the medical device industry essentially made simple tools. Today new innovations are becoming increasingly complex, driven by the advent of new technologies." [1]

Understanding Barriers to Medical Device Quality

FDA's Center for Devices and Radiological Health (CDRH)

3. Increased Complexity and Increased Risk. The speed of change, interdependency and increased importance of software increase risk. Medical device and diagnostic companies need to deal with this risk and change while maintaining safety.

The impact of these trends have begun to make traditional methods of software development obsolete, and put those that hold on to traditional methods at a competitive disadvantage. They also create tremendous opportunities for growth, new products and new lines of business for those who can adopt new methods to take advantage of them.

The Problem with Current Development Methods

Medical Device and Diagnostics have typically had long product development cycles, often 3 to 5 years, and Pharma cycles have been even longer, 5 to 10 years. Software development cycles tend to be much shorter, and have been accelerating in many industries.

Waterfall Development

The Software development strategy for medical devices has traditionally been a "Waterfall" model. The Waterfall development model originates in the manufacturing and construction industries both highly structured physical environments and where after-the-fact changes are prohibitively costly, if not impossible.

In this strategy, each activity, from requirements analysis through systems testing is performed and completed before the next activity begins.

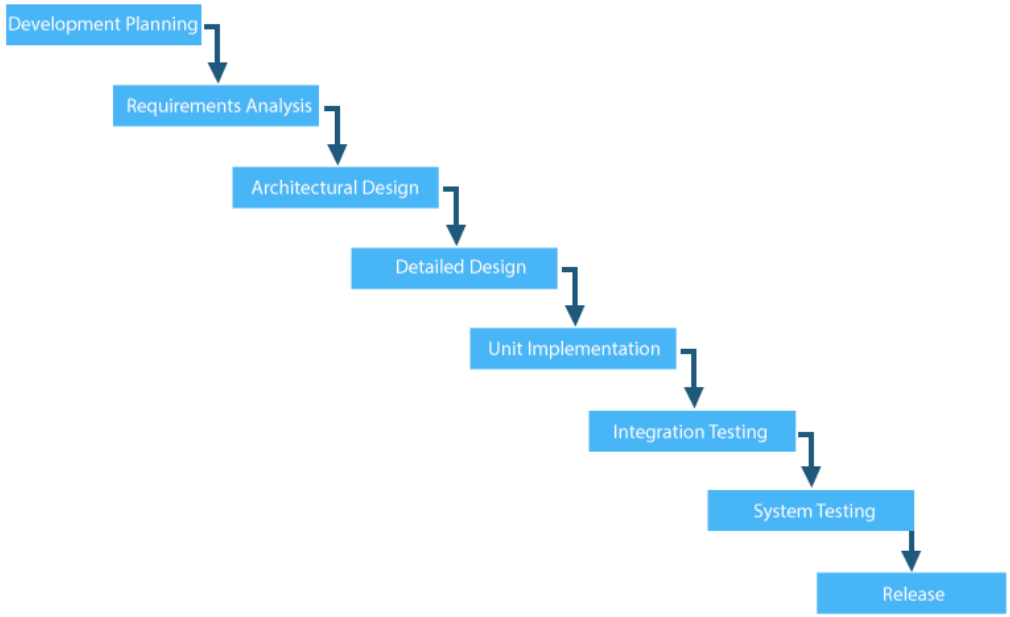


Figure 01: Waterfall Process

This process can work when all is understood and well defined, and there are no changes, hidden complexities, undiscovered requirements, or defects.

Waterfall Doesn't Match Reality

But the reality is that there are very few cases where requirements don't change throughout the course of a

project. This is true for medical devices just as much as it is for other industries.

The FDA acknowledges this in the Regulatory Information Guidance:

"Software requirements are typically stated in functional terms and are de!ned, re!ned, and updated as a development project progresses." [5]

Not only do requirements change, but in complex systems, changes discovered about one part of a system in one stage may well change earlier and later stages for other parts of the system.

Why does Everything Change?

The reality of developing complex systems with significant human interaction and interaction with other systems, is that change is inevitable. Change can come from many sources:

Outside Change

Outside changes, whether driven by technology, regulations, competition, or demographic forces is inevitable and accelerating.

Internal Change

Internally as companies grow or contract, acquire businesses, or get into new products, the legacy means of designing and developing products clash and become

burdensome. Often these internal changes create a heavy debt (people, process, technology) that is cumbersome to work within, let alone change to better capture innovation or to execute more quickly. The waterfall cost (and time) curve is part of this legacy debt and it places a high burden on those looking to keep up with or get ahead of these changes.

Requirements Change

For complex systems, it is rare if not impossible to be able to define requirements completely before starting design and development work. Additional requirements are typically discovered during design and development activities.

This is especially true for systems with significant user interface elements. One reason for this is that traditional requirements documents are a poor way of conveying and getting feedback from actual users, customers and other stakeholders. They are often not read, and if read, not fully understood. People need to visualize these requirements.

Design Change

As noted in the previous section, user experience design plays an increasingly important role, and users design expectations have been significantly raised through their experience as consumers. Designing user experiences is difficult, and requires feedback from users to get right.

A waterfall process locks the design into a set of requirements developed long before a solution is realized and a product gets into the hands of users. Limited end-

user engagement means increased risk of implementing a suboptimal solution.

As one senior director at a large medical device firm noted to me:

"In our current process, by the time people see it, it's done, and it's difficult to change."

Development Change

One of the fallacies of traditional waterfall development is to treat software development as a production activity. In software development we are creating the first instance of a product, not copying a piece of software that has been designed. In software development, development is actually a detailed design activity. The need to revisit requirements and design specifications during development is frequent. As a development manager at a major device firm noted to me:

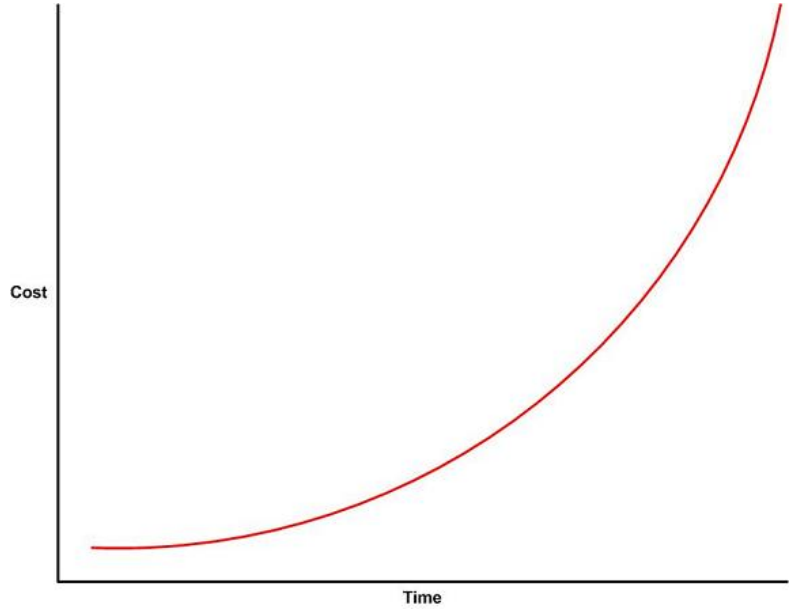
"Many of the [system's] details only become known to us as we progress in the [system's] implementation. Some of the things that we learn invalidate our design and we must backtrack."

Change Discovered in Testing

In a complex system, one bug may hide another. Your code base grows over time, increasing the chance that any change that does occur will touch more things later in the project. The longer the development process before testing, the greater the risk of multiple integration conflicts and failures, making it exponentially more difficult to find and fix problems and to maintain quality.

Why You Care -- The Cost of Change Is Unnecessarily High

Once you start adding change into the mix, you start running into waterfall development's cost of change curve:



The cost of change curve shows the relative cost of addressing a changed requirement, either because it was missed or misunderstood, at different stages of the product life cycle. The cost of fixing errors increases exponentially the later they are detected in the development lifecycle because the artifacts within a serial process build on each other, and each previous activity needs to be updated when an error or missing requirement is detected.

Waste in Waterfall

Missing or incorrect requirements are not the only consequence of the lack of early feedback.

Another consequence that has been noted in numerous studies is that features that would prove to be unnecessary or unused with more frequent feedback end up specified, developed, tested and deployed, creating little or no value, and increasing costs and risks.[6]

If change is inevitable, and waterfall is ill suited for change, is there a better way? That's where Agile comes in.

Addressing Waterfall's Shortcomings with Agile

The shortcomings of waterfall development are not new, and have been generally accepted in the software community for a long time. Agile methodologies, emerging as approaches to address these shortcomings in the early 1990s. They were harmonized in 2001 through the Agile Manifesto, and were widely adopted in large enterprises across many industries during the mid to late 2000s.

Agile does not attempt to lock down all requirements up front. It assumes that the requirements and design will evolve and change as more is learned and discovered during the process.

It works by breaking projects down into user stories - little bits of user functionality that create value for the user - prioritizing them, and then continuously delivering them in short one to three week cycles called iterations.

Requirements, design, development and testing are all part of the iterative process, as well as risk management

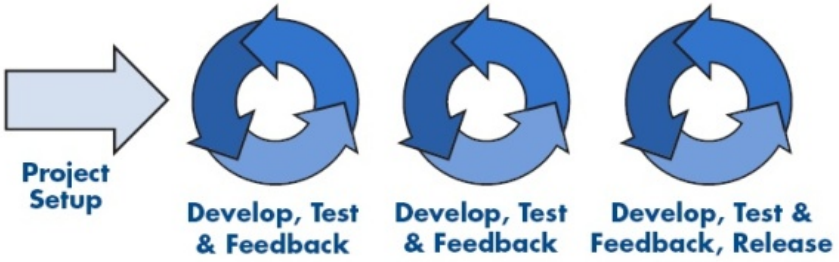


Figure 02: iterative Process

and customer feedback. Instead of treating each of these as fixed phases, agile explicitly incorporates feedback so that they can evolve and change

The emphasis of Agile is on adaptive planning, evolutionary development and delivery, encourages rapid and flexible response to change.

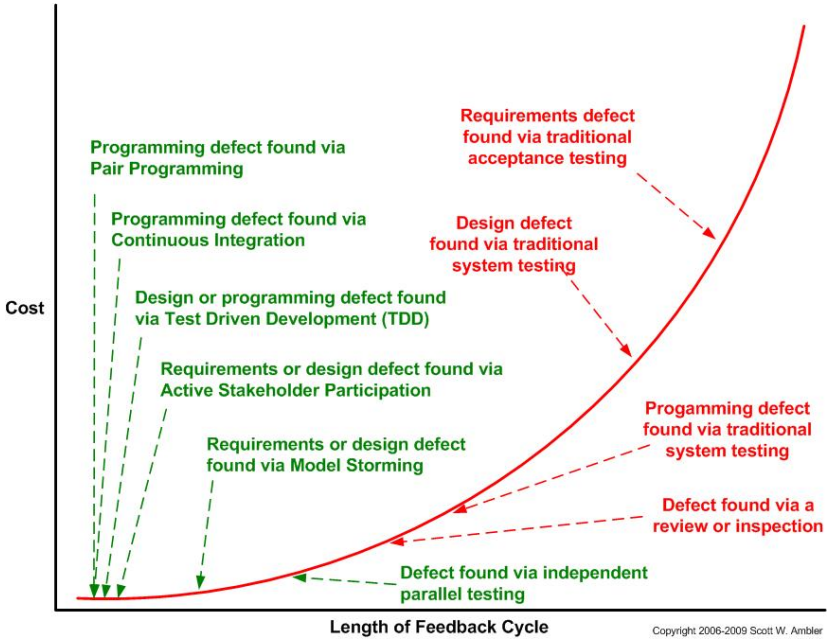
The Benefits Are Easy To Explain

- End users and business stakeholders get to see and experience the system as it unfolds and catch errors earlier in the development process.
- Course corrections become more apparent and easier to navigate.
- Quality improves because testing starts from day one.
- Risk is reduced because the risk process is incorporated early.

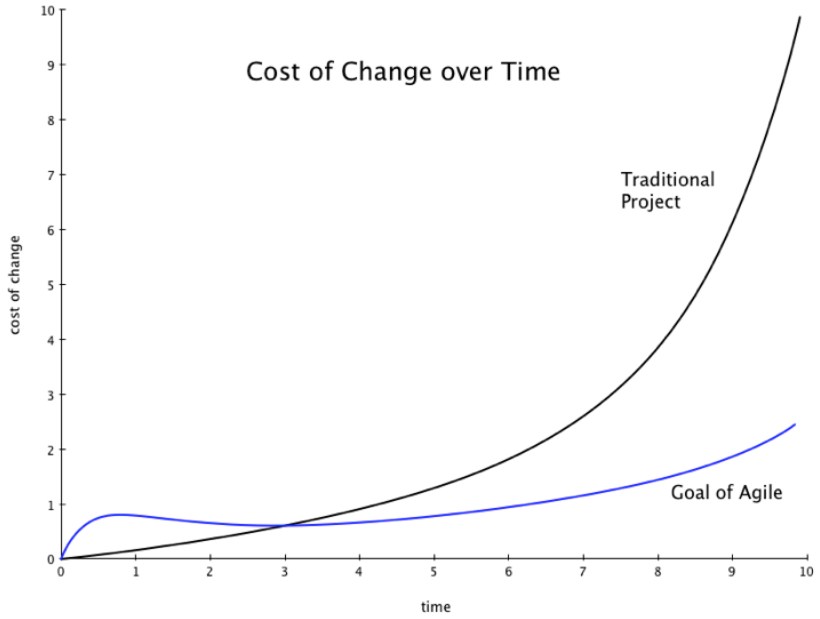
How Agile Bends the Cost of Change Curve

Traditionally, change has been shunned on software projects because of it's high perceived cost late in the game. Agile challenges this notion and contends the cost of change can be relatively flat if you reduce the feedback loop, the time between creating something and validating it.

Agile includes a number of practices, such as pair programming, test driven development and continuous integration which reduce the feedback loop.



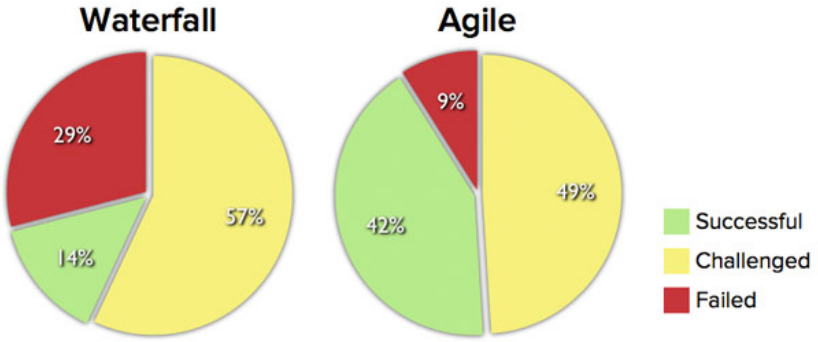
The following chart from Scott Ambler illustrates how agile practices reduce the feedback cycle and act earlier in the process, at the flat end of the curve.[7] If agile practices are consistently applied, the cost of change curve is flattened significantly:



Does Agile Work in Practice?

When agile methodology was first introduced, there was understandable resistance to an unproven new approach. As Agile began to be adopted by large organizations across many industries, many studies were done to analyze agile's effectiveness.

The Standish group CHAOS report, published annually since 1994, is one of the most respected studies of the state of software projects. It has consistently shown a high correlation between agile and success since enterprise adoption started taking hold in the mid 2000s.



Source: The CHAOS Manifesto, The Standish Group, 2012.

By 2012, the Standish Group used their strongest language yet to describe the benefits of agile:

"The agile process is the universal remedy for software development project failure. Software applications developed through the agile process have three times the success rate of the traditional waterfall method and a much lower percentage of time and cost overruns."[9]

Other studies, including some in the medical device industry, have shown consistent improvements in quality and project success.[8]

Is Agile Suitable for an FDA Regulated Environment?

Over the last 15 years, agile has been adopted and shown better results in many industries, including many with mission critical software requiring a high degrees of safety and reliability such as aerospace [9], defense [10] and energy.[11]

But if agile methods produce better results, why has the medical device and diagnostic industry been slow to adopt agile? Is the methodology just not suitable for medical products?

Objections to agile in the medical device industry have generally centered on three main concerns:

1. Medical devices need to be safe, effective and reliable, and cannot pose additional, new risks to the patient.

Agile and Lean Manufacturing

Agile methodology was originally inspired by Lean Manufacturing practices that arose out of the Toyota Production System (TPS), practices which have revolutionized manufacturing. Core principles of lean manufacturing that have parallels in agile include: Eliminating waste (Activities that don't create value for the customer) Just in Time Production Quality Built In (Zero Defects) Levelized Production Continuous Improvement.

Agile methodology does not have sufficient rigor and sophistication to be used in safety critical systems.

2. Agile methodology eschews planning, controlled processes and documentation, all of which are required by the FDA's Quality System Regulation (QSR).
3. Even if Agile had appropriate processes to meet FDA regulations, it wouldn't matter, because the FDA prescribes waterfall processes.

As we shall see, each of these concerns is based on a misconception. The reality is that agile methodology, appropriately applied, improves reliability, safety and effectiveness, is compatible with quality system regulations, and is in fact recognized as a consensus standard by the FDA.

Let's tackle these issues one at a time.

The FDA and Agile

FDA Does Not Prescribe Waterfall and in Fact Recognizes Agile as a Consensus Standard

While many industry professionals believe that FDA regulations require waterfall, neither the FDA's 21 CFR Part 820 Quality System Regulation nor other regulations derived from it prescribe a particular development methodology. The confusion arises because many of the standards such as IEC 62304 that the FDA does recognize are written in a way that suggests waterfall.[12]

In fact, the FDA explicitly cautions against using waterfall for complex devices:

"The Waterfall model's usefulness in practice is limited, for more complex devices. a concurrent engineering model is more representative."[13]

Despite the barriers to agile adoption, a number of firms in the industry, including Pathfinder, have recognized the value of agile, and have adapted the methodology to comply with both the letter and the spirit of applicable standards and guidelines, and have used it to develop and launched FDA cleared and approved products.

In order to provide clarity and guidance on aligning Agile at both the concept and practical level, the AAMI Medical Device Software Committee formed the Agile Software Task Group which produced ***AAMI TIR45: 2012, Guidance on the use of AGILE practices in the development of medical device software***[14]. AAMI TIR45 covers several key topics such as documentation, evolutionary design and architecture, traceability, verification and validation, managing changes and "done" criteria.

The US FDA recognized AAMI TIR45 in January of 2013.[15] This served to provide clarity and direction on mapping Agile practices to regulatory requirements, and assurance that the Agile methodology can be aligned and used to develop software for medical devices according to compliance requirements.

Delivering on Safety, Effectiveness, and Quality with Agile

Annex B of IEC 62304 provides a useful framework for standards associated with the safety of medical device software. On page 77 it states:

"There is no known method to guarantee 100% safety for any kind of software. There are three major principles which promote safety for medical device software:

- *Risk Management*
- *Quality Management*
- *Software Engineering"*

Agile addresses each of these principles in a way that is superior to traditional waterfall.

Comparing Risk Management with Agile

Similar to other aspects of a software development process, not all risks can be known at the beginning of a project, and we discover most hazards as a system evolves. This is acknowledged in ISO 14971, the international standard for risk management systems for medical devices:

"Risk can be introduced throughout the product lifecycle, and risks that become apparent at one point in the life-cycle can be managed by action taken at a completely different point in the life cycle."[16]

While AAMI TIR:45 does not specify how risk management activities should be integrated into the agile practices, our experience has been that this is best done by embedding risk management at each level of the lifecycle, with inputs and outputs from each activity within that level.

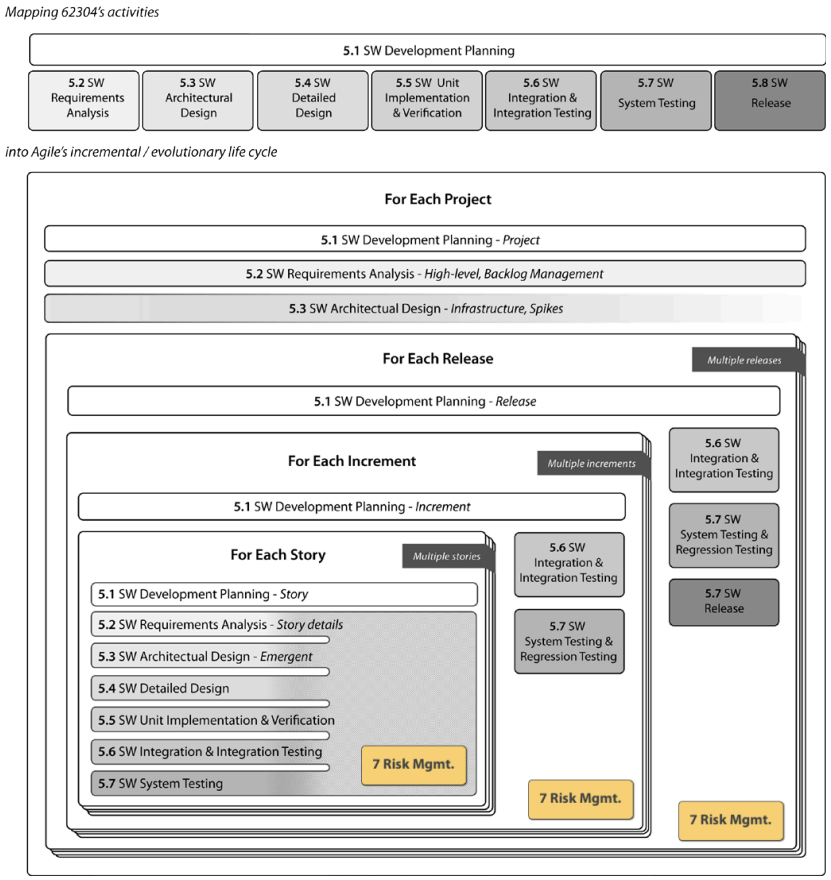
Risk analysis, evaluation, and risk control are performed in each sprint/iteration. You maintain a backlog of requirements with associated risk scores for each.

For any story, increment or release, risk assessment becomes part of your definition of done: If new hazards have been introduced or existing hazards have been mitigated, the DFMEA should be updated. User stories can be used to implement mitigations, and results can be reviewed during iteration demos and release reviews.

By embedding risk management early in the process, you can discover and eliminate or mitigate more risks, and do so much earlier - during the flat part of the cost of change curve.

This approach is illustrated in the modified version of figure 4 from AAMI TIR45: 2012

(modifications in yellow) [17]:



Early Discovery and Risk Prioritization

In the typical product development lifecycle, release dates are fixed. In waterfall, since requirements and hazards that are discovered late cost more and take more time to implement or mitigate, they either greatly extend the project or are inadequately dealt with and tested.

Hazard response is difficult, unless it's done early. Agile allows fine grained prioritization of features and reprioritization after every iteration. **If you use risk based prioritization of features and continuous risk assessment, you are much more likely to spend the right effort on the highest risk areas at a much lower cost.**

Incorporating Human Factors in Agile to Improve Effectiveness and Risk Mitigation

Human Factors in software is becoming more important. With complexity, greater access and availability and more user types, comes a "richness of opportunity for user error."

Many of the risks for complex medical devices are directly related to human factors, including permitted misuses, user complacency, user interface confusion and user interface designs that are not adapted to user workflows.

The FDA's expectations on user error are getting much more rigorous, and greater emphasis is being placed on comparative effectiveness as well as safety. Performing a few formative usability tests and dealing with most usability issues through labeling and training is no longer acceptable, neither is submitting "me too" products that are no better than the ones already in use.

Insufficient emphasis on user experience can also have major business consequences: Systems that are hard to use and don't provide what they users want, when they want it, in a way that is actionable, don't get used. And systems that don't get used don't result in future sales.

Human Factors Engineers and User Experience Designers bring expertise, experience, talent and methodologies to a product team that can result in dramatic improvements to project speed, product quality and user satisfaction.

Agile can be adapted to incorporate human factors directly into the development process. User experience designers or human factors engineers create storyboards, task flows and wireframes, get feedback from users, and then create detailed designs with acceptance criteria that make it very clear for developers what they need to develop. They can then get usability feedback from end users as increments are delivered by the software development team. Hazards related to human factors are surfaced earlier and mitigations can be designed and tested quickly.

Achieving Quality at Speed with Agile

Quality is one of the main aims of the agile methodology, and many of the core agile practices are designed to improve software quality while maintaining project velocity.

While Traditional software development teams often spend 1/3 to 1/2 their time on defect troubleshooting and rework, agile practices are designed to catch and fix defects early, when they are easiest and cheapest to fix.

Agile practices such as user stories, unit tests, test driven development, continuous integration and zero bug tolerance are designed to improve and maintain software quality. This allows teams to move much faster with fewer defects than in traditional waterfall.[19]

The agile methodology is supported by a number of practices that enable teams to improve quality, reduce risks and improve project velocity. These practices include:

- **Feedback Loops** are the core to evolutionary development and continuous improvement. Agile is designed to rapidly provide outputs and take inputs from different sources .
- **User Stories**, written based on user goals provide the 'who', 'what' and 'why' of a requirement in a simple, concise way. They can be easily grasped by end-users, focus designers and developers on the goals of the end-user, and have low overhead. User stories are formalized through acceptance procedures.
- **Unit Testing** - A unit test is an automated piece of code that invokes a unit of work in the system and then checks a single assumption about the behavior of that unit of work. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.
- **Test Driven Development** - writing the test first before adding new functionality to the system. First the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that

test, and finally refactors the new code to acceptable standards. Doing this defines success up front, provides traceability, ensures code coverage, and enables continuous integration.

- **Continuous Integration (CI)** implements continuous processes of applying quality control — small pieces of effort, applied frequently. Continuous integration aims to improve the quality of software, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control after completing all development. Combined with test driven development, unit testing and automated integration and functional testing, continuous integration allows defects to be discovered when they are introduced (i.e., when the code is checked into the version control repository) instead of during late-cycle testing.

Meeting FDA Quality Objectives with Agile

In it's essence, the FDA Quality System Regulation requires you to have a repeatable process, follow it, and demonstrate that the process will produce a quality product.

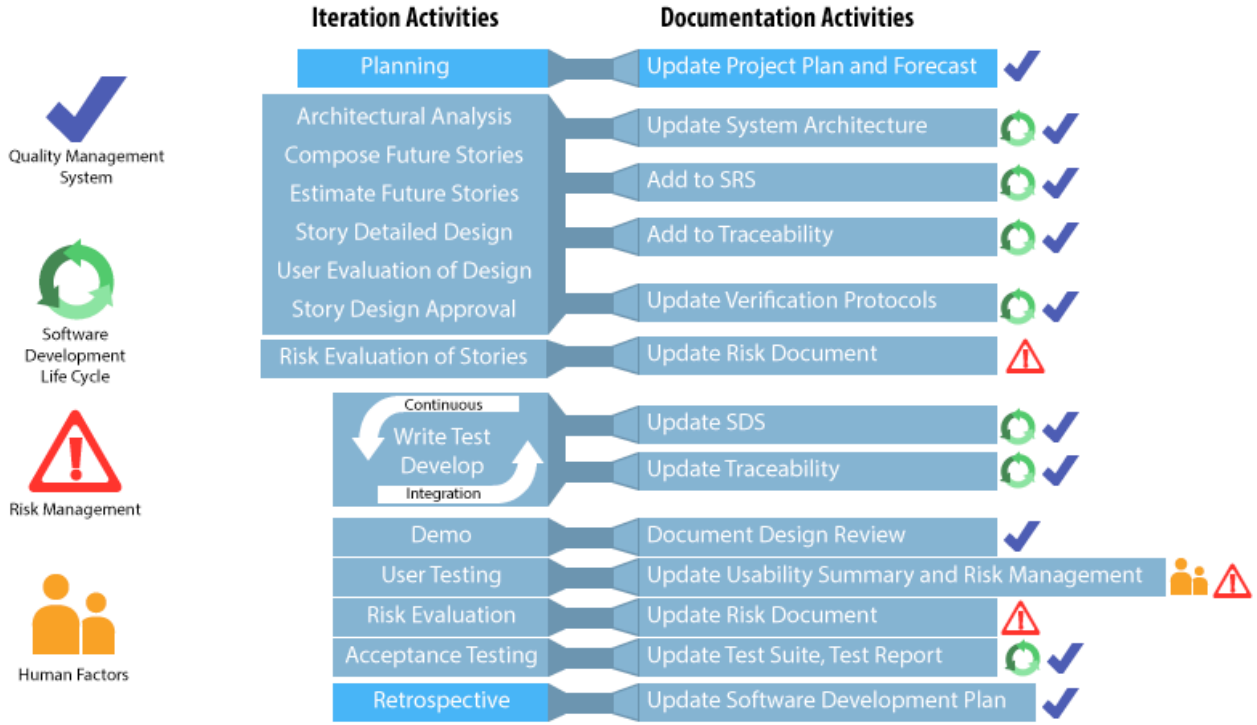
Design Control

The FDA requires evidence of design control, or how the requirements set has evolved during the development of the product, and require risk management to be integrated into the design process. Part of this process involves producing work plans, requirements specifications, hazard analysis, design documents, test cases, test plans, and test results as evidence of compliance.

Incorporating agile into a quality management system starts with the software development plan, documenting how you will:

- Create user stories and FURPS+[20], requirements and get feedback on these from users, risk management, human factors and other stakeholders
- Create task flows, wireframes and detailed design specifications
- Perform risk analysis, including human factors analysis
- Develop user acceptance criteria
- Select user stories for iterations (preferably using risk based prioritization)
- Perform development iterations, including length of iterations and the activities to be performed within each iteration, including development practices (pair programming, test driven development, continuous integration, refactoring) and review practices (stand-ups, iteration reviews, demos, architectural reviews, retrospectives, burn down and bug reporting)
- Augment informal review practices through independent review, verification and validation activities on a story, increment and release level. (In our experience, this is best achieved by embedding quality assurance and human factors with the team.)

In an agile process, documentation is updated on a per iteration or per increment basis, with some documentation



updated automatically through application of the agile process and development practices such as test driven development.

The diagram above illustrates how documentation can be updated during the course of a development iteration:

Design Reviews

In Agile, Design reviews are a natural part of the process: Iteration reviews, demos, architectural reviews, and retrospectives are an integral part of every development iteration. These frequent reviews help catch many issues early. Design reviews are also performed on a increment and release level.

Software Verification provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase.

Software Validation is confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.

In traditional waterfall, verification and validation activities often happen at the end, after all development is complete, which diverges from the FDA's recommendations:

"[The FDA] does recommend that software validation and verification activities be conducted throughout the entire software lifecycle."[22]

One of the biggest advantages of Agile is the integration of testing with development throughout the development process. Testing teams are full participants in requirements, architectural and design reviews, and collaborate with human factors engineers to create acceptance criteria for user stories and implement test cases. By testing stories as soon as they are implemented, testing teams uncover defects and discrepancies early in the process, when they are easiest to fix and have the least impact on other parts of the system. This process also has the benefit of parallelizing independent verification and validation with development, thereby compressing the schedule.

In Agile, verification starts at the user story level, providing traceability from user story to code, code to unit test, and user story to acceptance test. Stories are validated through acceptance tests and product demos. Acceptance tests, with descriptions of preconditions and expected behavior, can be executed by automated testing tools, which can be integrated into a continuous integration server to increase testing coverage. With certain tools, Static Analysis can also be automated as part of continuous integration. This automation helps teams execute tests more efficiently, generate test results automatically, and ensures accuracy of tests. It also frees up testing teams to use other techniques like feature testing, exploratory testing, scenario testing and system testing.

Verification and Validation

Implementing an iterative development method within the organization has helped us build a structure for consistent deliverables and focus on where issues may be happening within a project much sooner than we may have otherwise.

This has helped us react during a time in the project when there was still some flexibility as to what types of intervention may make a difference in pulling a project back on track. On top of that, it has helped us find a better and earlier linkage in developing Verification tests that can aid in identifying software issues earlier, rather than seeing issues for the first time during formal testing when the overhead of making a change is much more costly." [21]

Manager of Software Development

Thomas Price,

Covidien

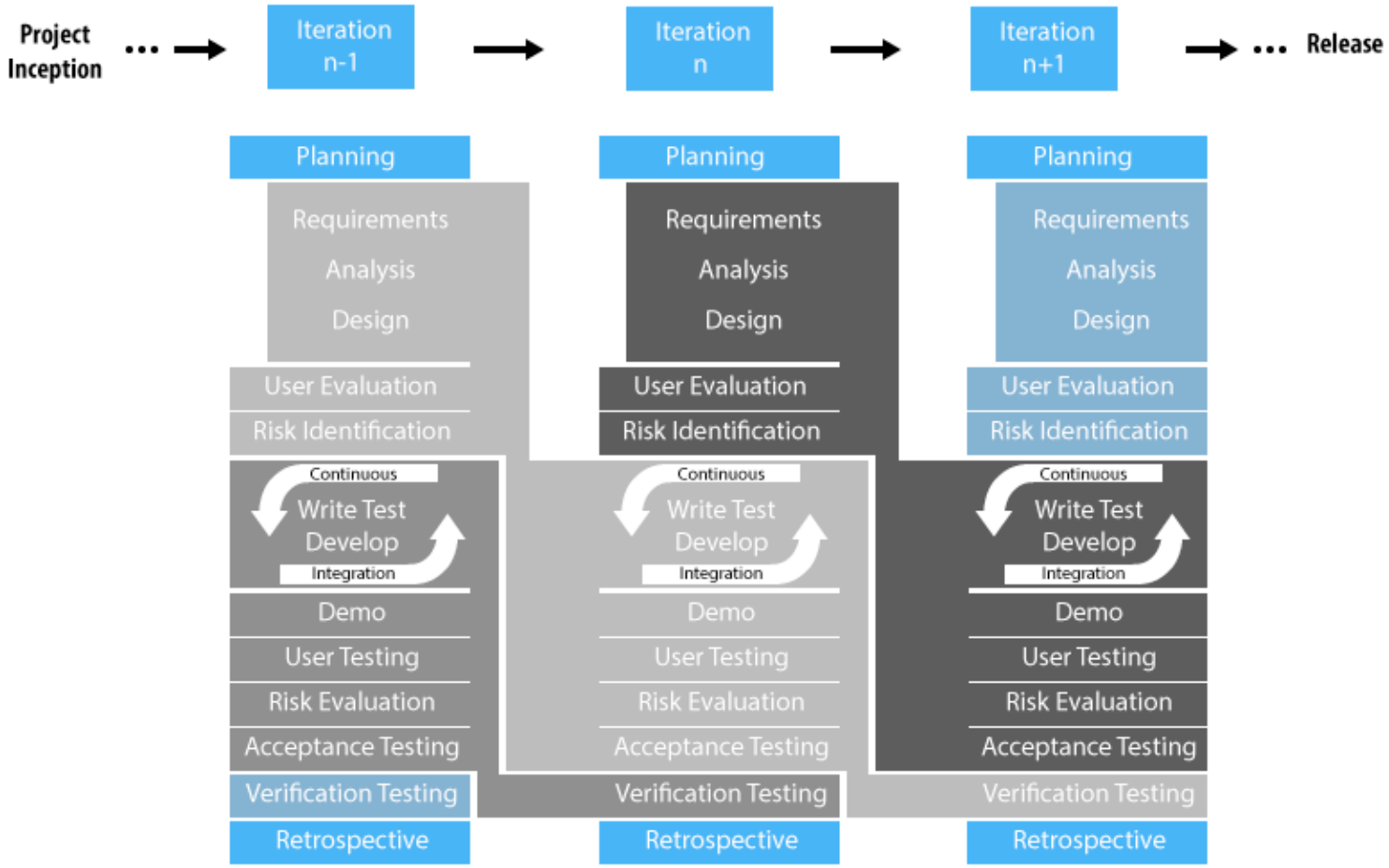


Figure 03: The Agile Process

At an increment level, verification includes traceability between the product requirements document and the Software Requirements Specification. Validation includes testing of features through feature acceptance tests. These can be automated using automated testing tools, and persist at the system level regression test repository, and should be included in the continuous integration environment once complete.

Other tests, such as formative usability tests, can also be done at an increment level. Another strong advantage of agile is that these formative usability tests can be run with actual working software rather than prototypes. Feedback from these tests can be incorporated in risk analysis and user experience design activities to further mitigate risks and ensure that the product satisfies end user needs.

Summary

Agile methodology, when properly adapted to the FDA's quality systems regulations, can provide superior results to currently prevailing waterfall development methods, especially for complex systems with significant software components.

Agile methodology is designed to get feedback early and often during the course of product design and use this feedback to continuously improve the product. As a result, it is well suited to the development of complex systems with emergent requirements. The feedback loop can be adapted to incorporate risk management, human factors and verification and validation. This fast feedback cycle, coupled with agile practices such as test driven development and continuous integration, allow product organizations to:

- Adapt to change
- Rapidly reduce requirements uncertainty
- Speed development and reduce waste
- More closely align to user needs, improve user interfaces and thus adoption
- Reduce risks and defects

Thus, companies able to adopt agile methodology give themselves a better opportunity to rise to the challenges posed by the rapidly changing regulatory, reimbursement and technology landscape and turn these to competitive advantage and market success.

References

1. Understanding Barriers to Medical Device Quality, FDA's Center for Devices and Radiological Health (CDRH), October 2011 p 28. http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm277272.htm
2. "The majority of companies have increased significantly in every measure related to the use of software in products. Interestingly, the life sciences industry (which includes medical device manufacturing) has increased the ratio of software engineers more than others, and 92% of respondents in that industry say they plan to further increase software use over the next five years." Tech-Clarity Perspective: Developing Software-Intensive Products: Addressing the Innovation-Complexity Conundrum, 2012, pp. 4-5.
3. Understanding Barriers to Medical Device Quality, FDA's Center for Devices and Radiological Health (CDRH), October 2011 p 3. http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm277272.htm
4. Understanding Barriers to Medical Device Quality, FDA's Center for Devices and Radiological Health (CDRH), October 2011 p 31. http://www.fda.gov/AboutFDA/CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHReports/ucm277272.htm
5. General Principles of Software Validation; Final Guidance for Industry and FDA Staff, January 2002, section 3.1. http://www.fda.gov/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm085281.htm
6. 2005 CHAOS Report, the Standish Group.
7. Why Agile Software Development Techniques Work: Improved Feedback, by Scott Ambler. http://www.ambysoft.com/essays/whyAgileWorksFeedback.html
8. 2011 CHAOS Manifesto, the Standish Group.
9. Agile Development Methods for Space Operations, Jay Trimble and Chris Webster, NASA Ames Research Center.
10. Adopting Agile in the U.S. Department of Defense.
11. GE Becomes More Agile, Rachel King, Wall Street Journal, May 30, 2012.
12. "It is easiest to describe the processes in this standard in a sequence, implying a 'waterfall' or 'once-through' life cycle model. However, other life cycles can also be used. Some development (model) strategies as defined at ISO/IEC 12207 [9] include (see also Table B.1): ... Evolutionary: The 'evolutionary' strategy also develops a SYSTEM in builds but differs from the incremental strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, customer needs and SYSTEM requirements are partially defined up front, then are refined in each succeeding build." IEC 62304 - Medical device software - Software life cycle processes, 2006. Annex B, pp. 75-76.
13. Design Control Guidance for Medical Device Manufacturers, US Food and Drug Administration, p.5
14. AAMI TIR45:2012 Guidance on the use of AGILE practices in the development of medical device software, Association for the Advancement of Medical Instrumentation, August 2012.
15. Recognized Consensus Standards, January 15th, 2013, U.S. Food and Drug Administration - Recognition Number 13-36: AAMI TIR45:2012, Guidance on the use of AGILE practices in the development of medical device software. http://

www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfstandards/detail.cfm?id=30575

16. ISO 14971: 2007 Medical devices -- Application of risk management to medical devices.
17. AAMI TIR45:2012 Guidance on the use of AGILE practices in the development of medical device software, Association for the Advancement of Medical Instrumentation, August 2012. Figure 4-Mapping IEC 62304's activities into Agile's incremental/evolutionary lifecycle, p. 23.
18. Agile Executive Forum 2013.
19. Beyond the Hype: Evaluating and Measuring Agile, Quantitative Software Management, Inc., 2011.
20. FURPS+ is a model for classifying requirements. FURPS is an acronym that stands for Functionality, Usability, Reliability, Performance and Supportability. The + in FURPS+ stands for design constraints, implementation requirements, interface requirements and physical requirements. For a more detailed discussion of FURPS+, see http://www.ibm.com/developerworks/rational/library/4706.html
21. Software Design at Any Price, Interview with Thomas Price for the 2013 Software Design for Medical Devices Conference.
22. General Principles of Software Validation: Final Guidance for Industry and FDA Staff, , January 2002, p. 1.

About the Author

Bernhard Kappe is founder and president of Pathfinder Software, where he applies his expertise in lean startup and agile development to help medical companies launch successful products faster. He has a passion for helping companies deliver more from their innovation pipelines and for launching software that makes a difference in the lives of patients, physicians, and caregivers.

He speaks on agile and product innovation at a number of medical device and health it conferences, and blogs on medical software development at

http://blog.pathfindersoftware.com

He can be reached via:

email: bkappe@pathf.com

linkedin: http://www.linkedin.com/in/bernhardkappe

twitter: @bernhardkappe

telephone: 312-372-1058 x6002

About Pathfinder Software

Are you solving tough problems in healthcare that users care about?

Pathfinder partners with large enterprise and start-up organizations to drive innovation and launch the right software products that tackle big problems in healthcare. And, that make a difference in the lives of clinicians, caregivers and patients.

If you're exploring solving tough problems, desire speed to market in an FDA regulated environment, and want to leverage breakthrough technology and user experience, we'd like to have a conversation with **YOU**.

Contact Pathfinder Software

888-986-2914

PathfinderSoftware.com

@PathSoft

linkedin.com/company/pathfinder-software

v1.03