Software Process Management

Agile Best Practices for Medical Device Development

Manage Compliance with Agile QMS

Andreas Birk

SWPM White Paper

April 2021

Developed in collaboration with Intland Software

INTLAND SOFTWARE

Summary

What is the current best practice for agile development of MedTech products? How can regulatory compliance be achieved effectively?

This white paper introduces and explains key agile practices that are specifically important in the regulated healthcare domain. It focuses particularly on:

- Definition of Done
- Nonfunctional requirements
- Incremental compliance
- Agile QMS

Additional agile practices include: Backlog Constraints, Solution Intent, Agile Release Train, Requirements Tracing, Test Automation, and Behavior-Driven Development. The white paper provides profiles of selected practices and recommends useful additional information sources.

Contents

Access additional insights provided by our partner Intland Software:

Unlocking the Power of Agile in Medical Device Development

3-part series of white papers & webinars

Access insights

Agile Success Stories in Healthcare

Build on Experiences and Lessons Learned

Approaching Agile in Medical Device Development

Unlock the Power of Agile in Regulated Environments

Agile Best Practices for Medical Device Development

Manage Compliance with Agile QMS

Agile in Regulated MedTech Environments

Medical device development must pay attention to regulatory compliance. This contrasts with many areas where agile methods have emerged. Common agile practices typically focus more on functional aspects, while regulatory compliance involves nonfunctional requirements and process requirements like risk management procedures.

This white paper presents agile practices that are particularly well-suited for regulated MedTech environments. At its center are the practice of Definition of Done, how to address nonfunctional requirements, and building compliance incrementally.

Definition of Done and nonfunctional requirements

The Definition of Done is a core practice of the most widely used agile method Scrum. At the same time, Definition of Done can do an excellent job representing regulatory compliance requirements in agile projects.

The section on the Definition of Done describes this practice and explains its important role for compliance management. Another section presents additional practices for managing nonfunctional requirements.

Incremental compliance

Agile builds on an iterative development approach. It provides a context in which regulatory compliance can be addressed and satisfied in an incremental manner.

Additional agile practices for MedTech

The concluding section presents several other agile practices relevant to MedTech. It also provides links to important information sources on agile practices.

Agile Practices

Agile development does not know normative, prescribed processes. Agile methods usually define frameworks that enable basic collaboration, coordination, and accomplishment of technical tasks. Teams are encouraged to continuously find better ways of working together and of providing value to customers.

Practice

The term *practice* is commonly used to describe a specific way of working or accomplishing a task. It can be a method like Sprint planning, a template like Role-Feature-Reason for defining user stories, or a habit like daily standup meetings.

Best practice

We use the term best practice to highlight that a particular practice appears more useful than others.

It is important to know that best practice always depends on a specific point in time and, often, on a specific context: Today's best practice might be outdated in a year from now. And only a few practices will work in each and every situation. So, it must always be understood as *current* best practice, and *adapted to the situation* at hand.

Interrelated practices

Most practices assume that other specific practices are in place, too. For instance, user stories represent individual requirements. They need Epics for clarifying the larger picture. A product backlog and an agile board help to manage user stories throughout the development lifecycle.

Because of these interdependencies between agile practices: Whenever you modify an agile practice in your team or organization, you should check whether other related practices must be adapted, too.

Definition of Done

Definition of Done is one of the most central and important practices in agile development. It is also very important for achieving regulatory compliance. However, most introductions to agile methods don't focus specifically on the Definition of Done, and hands-on practical advice is rare.

This section points out why Definition of Done is central to agile development, how it can be implemented, and what its role is in regulatory compliance.

What is Definition of Done?

The Definition of Done is a set of criteria that each piece of work within an agile iteration (in Scrum called *Sprint*) must fulfill in order to become part of the iteration's product release (in Scrum called *Potentially Shippable Increment, PSI*).

The Definition of Done can relate to (1) each individual work item of an iteration (in Scrum called *Backlog Item*, typically defined in the form of a User Story) or (2) the iteration (or *Sprint*) as such. Figure 1 shows example criteria that a team might define for its work items and iterations.

Criteria for Potentially Shippable and Definition of Done(*) on *Backlog Item* Level:

- Code and tests checked in*
- Unit tests complete and pass*
- Code documentation complete
- Integration succeeds*
- Integration tests complete and pass*
- Static analyses complete and pass
- Packaging and staging succeed*
- System tests complete and pass*
- Acceptance tests complete and pass

Criteria for Potentially Shippable and Definition of Done(*) on *Iteration* Level:

- Performance tests complete and pass
- Risk management complete and documented
- All defects closed or prioritized for later iterations
- Installation packages available*
- Operations documentation complete*
- User documentation complete
- Marketing material complete

Figure 1: Example criteria of Potentially Shippable and Definition of Done on the levels of backlog item and iteration. These lists contain all the steps that an organization might need or find important to ship a release. Criteria marked by an asterisk (*) are what the teams can currently accomplish within each sprint. They form the current Definition of Done (cf. [1] and [2]).

Teams create their product's Definition of Done

The Definition of Done must be created by the team as an agreement between the developers and the product owner. In multi-team settings, all teams jointly define the product's overall Definition of Done. It shall then also reflect the organization's quality standards.

A Definition of Done is valid and remains unchanged for each development iteration of a product. It represents the team's commitment about when the iteration's work can become part of the product increment.

Since the Definition of Done determines whether work counts for the iteration, it is fundamental to estimation, planning, and monitoring. A team's velocity metric only includes "done" work. Therefore, the Definition of Done must be clear before meaningful work estimation and planning for an upcoming Sprint can start.

Implementing the Definition of Done

How is a Definition of Done implemented in agile development projects? Teams can choose from the following techniques. At best, they are applied in combination:

- Written and printed lists: presenting the Definition of Done at the workplace
- Central online list and documentation, detailed and binding
- Tasks attached to each backlog item (for backlog item Definition of Done)
- Separate tasks in the Sprint backlog (for iteration Definition of Done)

Written and printed lists

The Definition of Done should always be visible and available at the team members' workplaces. So we need paper-based, real physical lists that communicate the essence of the Definition of Done. They should contain the criteria in a short and succinct form. These can be hand-written flipchart papers on the wall, sticky notes alongside the physical Scrum board, or printed checklists on each developer's desk.

Central online list and documentation

The Definition of Done must be clear and unambiguous. Everyone in the organization must know or at least be able to look up the exact criteria. As the Definition of Done evolves over time, the most up to date and currently binding definition must always be accessible.

All this calls for more than just a simple list of short single-line criteria. You may put this kind of information into the team's or organization's central document repository or the development organization's Wiki system.

This documentation can even become a central part of the repository, in accordance with the Definition of Done's central role in the agile development workflow. For instance, you may link each criterion with the recommended tools and work instructions for accomplishing it, from module naming conventions to usage of the automated test environment.

Tasks attached to each backlog item

Definition of Done criteria related to backlog items is best represented as tasks or as a checklist with each individual backlog item, like a User Story or Defect Report. When using an agile workflow tool, these tasks or checklists should also be defined there. When using physical tools like sticky notes on a Scrum board, printed checklists may be the preferred solution.

Ideally, the fulfillment of each Definition of Done criterion should be documented, such as closing tasks or ticking checklist items. This adds to the team's work progress transparency. It also is a basis for documenting fulfillment of company standards and regulatory compliance.

Separate tasks in the Sprint backlog

Definition of Done criteria that are related to the entire iteration instead of each individual backlog item should be managed as separate tasks or even as user stories. They are treated as first-class citizens of the Sprint backlog and must be completed by the end of the Sprint.

Evolving the Definition of Done

In larger and more complex product and development contexts, the initial Definition of Done rarely contains all the criteria needed for shipping the product to the end customer. Some work remains *undone* even though the teams have completed their increments. Such a Definition of Done is considered *weak* and *not perfect*.

The organization must find ways to accomplish the undone work remaining from the iterations. Agile teams must incrementally build the capabilities needed to perform this work as part of the iterations. They must evolve and adapt their Definitions of Done accordingly.

Larman and Vodde propose the following techniques for reducing the amount of undone work [2]:

- **Automation:** For instance, automating tests
- **Harmonization:** Focusing on optimal standard solutions and workflows

- **Environment:** For instance, more effective test equipment
- **Parallelization:** Starting activities as early as possible
- **Cross-functionality:** Building up all skills for completing the work within each agile team

Definition of Done from a compliance perspective

Regulatory compliance sets mandatory criteria for products and their development processes. Definition of Done sets criteria—both product- and process-related—that each piece of an agile increment must fulfill. This makes Definition of Done the primary instrument for bringing compliance criteria to agile development.

Definition of Done

Practice Profile

Description: The Definition of Done is a set of criteria that each piece of work within an agile iteration must fulfill in order to become part of the iteration's product release.

Important Aspects:

- Agile teams define their Definitions of Done autonomously. Compliance management can influence the Definition of Done through collaborating with the team and its Product Owner, highlighting regulatory demands, and through defining suitable requirements for regulatory compliance.
- A team's Definition of Done can represent regulatory compliance requirements in agile development.
- If multiple teams work on the same product, their Definitions of Done should be coordinated. The Definition of Done also represents corporate standards, including regulatory compliance regulations.
- The concept of "undone" work can drive continuous improvement in Agile, using the Definition of Done as a vehicle.

Information Sources:

The Scrum Guide [3] defines concisely the role that the Definition of Done plays in the agile workflow of the Scrum method.

Mitch Lacey [1] gives very illustrative advice how an agile team can define its Definition of Done by brainstorming and consolidating a list of criteria for Potentially Shippable (Chapter 7. How Do You Know You're Done?).

Craig Larman and Bas Vodde provide detailed practical advice on using and enhancing the Definition of Done. They define the concept of "undone" work and describe how teams can use it to drive agile improvement. Their book [2] provides a detailed explanation (Chapter 10. Definition of Done). The LeSS website contains a short overview description [4].

However, some aspects of this agile way of compliance management contrast with established traditional approaches. This section points them out and provides recommendations for how compliance managers can cope with this challenge.

Team autonomy

A Definition of Done is worked out and agreed between product owners and developers. Compliance management is not entitled to mandate any criteria directly. So the general approach is: Compliance management must point out to product owners and development teams what are the important regulatory compliance requirements and constraints.

Product owners shall act as close allies of compliance management. Developers can be expected to propose the concrete measures through which compliance requirements shall be satisfied.

Corporate quality standards

Larger organizations shall maintain a central Definition of Done, which must be fulfilled by all teams. In addition, each team is allowed to set its specific Definition of Done criteria, and to add its own measures to fulfill the central criteria.

Compliance management can use this constellation to negotiate and coordinate a system of corporate quality standards. They shall address all important compliance requirements. The organization-wide process of coordinating them can be used as a vehicle for making all teams aware of compliance matters.

Documenting fulfillment of Definition of Done criteria

Regulatory requirements always demand, in one way or the other, that important development steps and decisions are documented. It is not sufficient that agile teams know and follow compliance requirements through their Definitions of Done. They also must be able to demonstrate whether and how they have satisfied those requirements.

For this reason, it is important to set up and track explicit development tasks for each Definition of Done criterion on each backlog item and iteration. Tool-based task tracking along with revision-safe data storage will usually do the job in an effective and efficient manner.

Nonfunctional Requirements in Agile

Standards and regulatory systems like ISO 13485 for medical devices mandate that development organizations shall systematically manage customer and product requirements. This includes nonfunctional product characteristics like performance and safety.

Agile development manages requirements information typically in the form of user stories, which address mostly the functional aspects of a product. In contrast, agile methods still often neglect the management of nonfunctional requirements (NFR). So this section aims to point out practices available for the agile management of NFR in order to support controlled development and regulatory compliance.

Agile practices for managing NFR

Current state of the art in the agile management of nonfunctional requirements includes five practices:

- User stories for NFR
- NFR as Backlog Constraints
- NFR in Definition of Done
- Managing Backlog Constraints in (Agile) Specifications
- Managing Backlog Constraints in Solution Intent

User Stories for NFR

While user stories are particularly well-suited for defining functional product characteristics, they can also be used for expressing nonfunctional aspects. Mike Cohn has described this in an illustrative and concise blog article [5].

An example of a NFR user story is: “As a Compliance Manager, I want that the product logs important operational data on its core functions, so that we can analyze and control the product’s conformance to its key requirements.”

In essence, the advice goes as follows:

- Identify who has a particular interest in a specific nonfunctional product characteristic. Note this person or stakeholder in the “As a …” clause of the user story template.

- Identify the nonfunctional characteristic and the object that bears it. Note them in the “I want …” clause of the template.
- Identify why the nonfunctional property is important to the stakeholder. Note it in the “So that …” clause.

Managing NFR as user stories is particularly useful for developing core features that establish the nonfunctional characteristic in the product.

NFR as Backlog Constraints

Non-functional requirements often express constraints on many functional requirements simultaneously. An example is the user training demanded by ISO 13485: For each critical core functionality of the product, training must be available or planned. These NFR should be attached to the backlog to make clear they apply to many or all backlog items.

Teams are free to decide how exactly they establish and represent backlog constraints. However, the approach should be transparent, efficient, and document relevant actions for demonstrating compliance. It is suitable to use practices like Definition of Done, Agile Specification, and Solution intent. These are described in the following.

Backlog Constraints have come to be known through the Scaled Agile Framework SAFe®. More detailed information can be found on the SAFe® website [6] and its article on nonfunctional requirements [7].

NFR in Definition of Done

A Definition of Done must be fulfilled by each backlog item of an iteration or by the iteration’s increment as such in order to count as completed. So NFR that apply to many backlog items can be included in the Definition of Done to ensure they will be addressed.

As an example: ISO 13485 training requirement can be formulated as a Definition of Done criterion as “Training for new functionality available or planned.”

As long as only a few such criteria must be included in the Definition of Done, each one can be listed as an individual entry. However, for more complex products, this list of criteria grows fast. Then it will become hard to oversee and manage the Definition of Done.

Larger collections of NFR should be grouped, and managed outside the Definition of Done, which then only refers to these groups. These NFR collections can be managed as (agile) specification documents or Solution Intent (see below). Other agile practices and Test Automation in particular can be very important to efficiently manage NFR collections and Definition of Done.

Backlog Constraints in (agile) specifications

Managing larger groups of NFR as backlog constraints works best when they are organized in a separate kind of document. A proven blueprint are traditional requirements specification documents.

Some people may object that specification documents were "not agile" and should be avoided. However, this is not quite true. The Agile Manifesto states "we *favor* working software over comprehensive documentation". If we need control over NFR, and structured documentation can give it to us, then such documentation is fine.

What is important: manage specification documents in a lean and agile manner that avoids unnecessary overhead. Well-structured corporate or team Wiki pages, best integrated with your agile tool environment, can do a perfect job here. Dean Leffingwell and Don Widrig recommend this approach in their book chapter on nonfunctional requirements in agile development [8].

You can be most effective when you know how to flexibly use accomplishments from plan-based development under the umbrella of Agile: Use the best from both worlds! Find more on this capability of hybrid agile/non-agile development in another, accompanying white paper on approaches to Agile. Over time, traditional practices may evolve into new agile ones. An example is Solution Intent as pointed out in the next section.

Backlog Constraints in Solution Intent

Solution Intent is another practice that has emerged from the Scaled Agile Framework (SAFe®). It contains and organizes various kinds of information in a way that suits agile development: specifications including NFR, design, and tests. Solution Intent shall be "the single source of truth about the solution".

Important to regulated environments is that information in a Solution Intent shall be traceable. This means that requirements information is connected with related design and test documentation. Also, decisions are documented, including those relevant for demonstrating regulatory compliance.

The SAFe® web page on Solution Intent provides many useful recommendations on how to structure and manage the information [9]. However, there are no prescriptions on how an organization or team should implement their Solution Intent. Generally, it is a good idea to use a well-balanced blend of physical media (e.g., cards and charts) with modern tools for software engineering and collaboration (e.g., Wiki pages, workflow management, and model-based generation and automation).

Build Compliance Incrementally & Agile QMS

Incremental Compliance

Still too often projects address regulatory compliance relatively late in the development lifecycle. Agile development opens the opportunity for compliance management to receive development teams' attention early.

Compliance after development

Figure 2 (a) shows a situation still frequently found today: most, if not all activities for achieving regulatory compliance are conducted after the actual product development has been completed. This causes delays in delivery and a partial rework of an otherwise finished product.

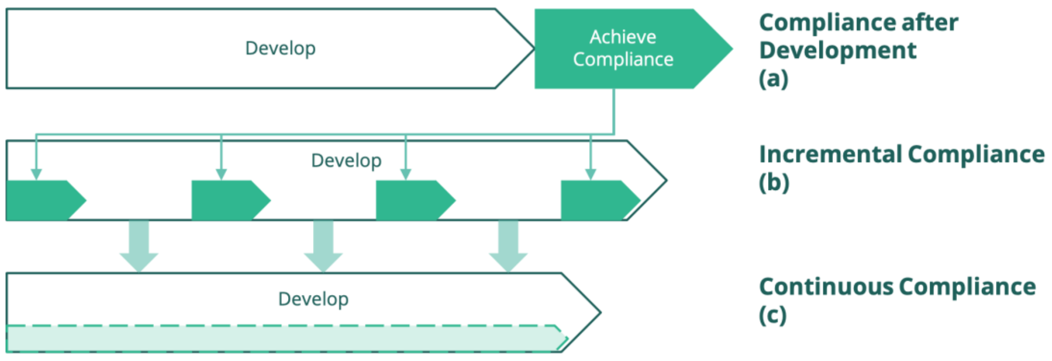
Develop
Achieve Compliance
Compliance after Development (a)
Develop
Incremental Compliance (b)
Develop
Continuous Compliance (c)

Figure 2: Relations between achieving compliance and development.

Incremental and continuous compliance

It is more efficient, if compliance-related activities can be accomplished in closer connection with the other development activities (Figure 2 (b)). Ideally, compliance is taken care of continuously in every relevant development activity from day one, as illustrated in Figure 2 (c).

Agile development offers the means and, in its purest form, even mandates such incremental and continuous approaches to regulatory compliance. The following agile practices accomplish this:

- Compliance criteria rooted in the Definition of Done: The related compliance measures will thus be accomplished for each individual backlog item or at least once per iteration.

- Gradually strengthening a weak Definition of Done by reducing "undone" work: Compliance-related activities that used to be conducted outside iterations will stepwise be integrated with other development activities.
- Plan compliance-related product functionality and technical capabilities as part of the regular product backlog: These compliance features will be implemented in agile iterations like every other product functionality.

Agile QMS

A Quality Management System (QMS) is the core instrument through which medical device development achieves and maintains regulatory compliance. QMS have traditionally formed around plan-based management approaches.

However, there is no reason why a QMS could not be implemented in an agile manner. As Agile has brought many advances to software development already, agile principles and practices may also contribute to a next-generation QMS.

Important characteristics of an Agile QMS are:

- The Agile QMS is integrated throughout the entire organization
- Quality is rooted in development teams
- The Agile QMS is built on lean/agile practices instead of process control

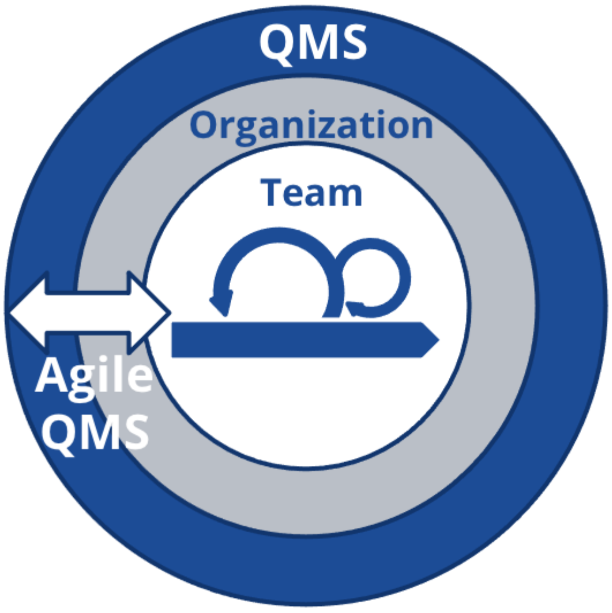

Figure 3: Agile QMS aims at integrating quality matters firmly into the development organization and its teams by using suitable agile practices.

Additional Agile Practices for MedTech

Compliance management can choose from a wide range of practices that help establish regulatory compliance in agile product development. This section lists such important practices and useful information sources.

Agile practices

Agile Release Train

Agile Release Trains (ART) are organizational structures that gather all teams and stakeholders needed to develop a set of related product solutions. As an agile practice that has originally been defined in SAFe® [6] they can be used in MedTech to orchestrate the interaction between the traditional compliance manager role and agile cross-functional teams.

A detailed introduction to Agile Release Trains provides the article in the SAFe® Glossary [10].

Requirements Tracing

Requirements Tracing is fundamental for demonstrating that regulatory compliance requirements are fulfilled and implemented in a product.

For instance, trace links can mark the path from a legal regulation like EU Regulation 2017/745 via derived functional requirements, their implementations and tests cases to successful integrations and test runs. This way they document that and how a particular product satisfies the regulatory demands.

The box below presents an overview and introductory information on requirements tracing.

Test Automation

Test Automation is a wide field, with many powerful solutions relevant to compliance management. The basic strategy is that automated tests demonstrate fulfillment of compliance criteria after every relevant product change.

Once the infrastructure for automated tests has been established, each individual test cycle has a low direct cost. The benefits can outweigh the costs significantly.

A very good practical introduction to test automation and its potential is provided in the books by Lisa Crispin and Janet Gregory [11][12].

Requirements Tracing

Practice Profile

Synonyms: Trace Links, Tracing, Requirements Tracing, Traceability, Requirements Traceability

Description: Requirements tracing documents originate and use relationships between important artifacts of the software and system development lifecycle.

Important Aspects:

- Effective requirements tracing demands suitable tool support, at least a requirements management tool, and ideally an integrated infrastructure covering the entire product lifecycle.
- Traceability matrices have many limitations; requirements tracing must go far beyond trace matrices.
- For compliance management, requirements tracing is an important tool that helps to demonstrate compliance and root compliance matters firmly in the development process.

Information Sources:

The Wikipedia article on requirements traceability [13] gives a systematic introduction and overview.

Karl Wiegers and Joy Beatty [14] treat requirements tracing in a very comprehensive and practical manner (see Chapter 29: Links in the requirements chain).

Behavior-Driven Development

Behavior-Driven Development (BDD) Behavior-Driven Development (BDD) is a method that combines acceptance testing, test automation, and a specific approach to requirements definition. Its elements encompass the entire development lifecycle of agile iterations.

BDD is sometimes referred to as Specification by Example. It defines user stories details in the form of test scenarios. An underlying automation framework makes it possible that textual requirements and test definitions can be executed and run automatically.

The method requires some upfront investment and continuous maintenance of its technical infrastructure. However, it can make test-driven development highly effective.

The Agile Alliance's glossary article on Behavior-Driven Development [15] provides a good starting point for learning about this practice. It also refers to useful further reading.

Useful information sources

Some information sources are particularly useful when looking for agile practices relevant to MedTech environments:

Agile Alliance: https://www.agilealliance.org/

The Agile Alliance is a community and network on agile development. It maintains a rich information repository. The Agile Glossary contains brief and concise characterizations of many agile practices. Blog posts and videos offer additional detailed information and experiences.

LeSS - Large-Scale Scrum: https://less.works

LeSS is a method for scaling Scrum. It proposes several very interesting and well-thought-through practices. They are characterized briefly on the LeSS website. Books from LeSS's authors Craig Larman and Bas Vodde provide valuable additional information. A good starting point is their 2016 book *Large-Scale Scrum: More with LeSS* [2].

SAFe® - Scaled Agile Framework®: https://scaledagileframework.com

SAFe® is another leading framework for scaling agile development. It is particularly relevant for continuous product development. Many useful agile practices have evolved from SAFe®. The SAFe® website contains practical descriptions and additional information about these practices.

Another good information source are Dean Leffingwell's books on Scaling Software Agility [16] and Agile Software Requirements [8], that provided the basis for SAFe®.

Access additional insights provided by our partner Intland Software:

Agile Success Stories in Healthcare

White paper & webinar

Access insights

Software Process Management

Agile Success Stories in Healthcare

Build on Experiences and Lessons Learned

Andreas Birk

SWPM White Paper

June 2021

Developed in collaboration with Intland Software

INTLAND SOFTWARE

References

[1] M. Lacey, *Scrum field guide: The agile advice for your first year and beyond*, 2nd Ed. Boston, MA: Addison-Wesley, 2016.

[2] C. Larman and B. Vodde, *Large-scale Scrum: More with LeSS*. Upper Saddle River, NJ: Addison-Wesley, 2016.

[3] J. Sutherland and K. Schwaber, “The Scrum Guide^{TM}: The definitive guide to Scrum: The rules of the game,” Scrum.org and ScumInc., Nov. 2020. [Online]. Available: http://www.scrumguides.org/.

[4] C. Larman and B. Vodde, “Definition of Done,” *Large-Scale Scrum (LeSS)*, 2021. https://less.works/less/framework/definition-of-done (accessed Apr. 26, 2021).

[5] M. Cohn, “Non-functional requirements as user stories.” https://www.mountaingoatsoftware.com/blog/non-functional-requirements-as-user-stories (accessed Apr. 26, 2021).

[6] Scaled Agile, Inc., “Scaled Agile Framework® (SAFe®),” *Scaled Agile Framework*. https://scaledagileframework.com (accessed Apr. 26, 2021).

[7] “Nonfunctional Requirements,” *SAFe® Glossary*. https://www.scaledagileframework.com/nonfunctional-requirements/ (accessed Apr. 26, 2021).

[8] D. Leffingwell, Agile software requirements: Lean requirements practices for teams, programs, and the enterprise. Boston, MA: Addison Wesley, 2011.

[9] “Solution Intent,” *SAFe® Glossary*. https://www.scaledagileframework.com/solution-intent/ (accessed Apr. 26, 2021).

[10] “Agile Release Train,” *SAFe® Glossary*. https://www.scaledagileframework.com/agile-release-train/ (accessed Apr. 26, 2021).

[11] L. Crispin and J. Gregory, *Agile testing: A practical guide for testers and agile teams*. Amsterdam: Addison-Wesley Longman, 2008.

[12] J. Gregory and L. Crispin, *More agile testing: Learning journeys for the whole team*. Upper Saddle River, NJ: Addison-Wesley Professional, 2014.

[13] “Requirements traceability,” Wikipedia (EN), 2021. https://en.wikipedia.org/wiki/Requirements_traceability (accessed Apr. 26, 2021).

[14] K. Wiegers and J. Beatty, Software requirements, 3rd Ed. Redmond, WA: Microsoft Press, 2013.

[15] "Behavior Driven Development (BDD)," *Agile Alliance Glossary*, 2021. https://www.agilealliance.org/glossary/bdd/ (accessed Apr. 26, 2021).

[16] D. Leffingwell, Scaling software agility: Best practices for large enterprises. Boston, MA: Addison-Wesley, 2007.

Imprint

SWPM White Paper, ID 2021-02, V 1.1, April and July 2021

Dr. Andreas Birk, Software.Process.Management,
Usedomstr. 15, 70439 Stuttgart, Germany

https://swpm.de, phone: +49 711 6645 324, email: info@swpm.de

VAT ID no.: de814817588

This SWPM White Paper has been developed in collaboration with Intland Software GmbH (https://intland.com).

Access additional insights provided by our partner Intland Software:

Unlocking the Power of Agile in Medical Device Development

Software Process Management

Agile Success Stories in Healthcare

Build on Experiences and Lessons Learned

Andreas Birk

SWPM White Paper

June 2021

Developed in collaboration with Intland Software

Software Process Management

Approaching Agile in Medical Device Development

Unlock the Power of Agile in Regulated Environments

Andreas Birk

SWPM White Paper

March 2021

Developed in collaboration with Intland Software

INTLAND SOFTWARE

Software Process Management

gile Best Practices for edical Device Development

nage Compliance with Agile QMS

s Birk

White Paper

021

Developed in collaboration with Intland Software

INTLAND SOFTWARE

3-part series of white papers & webinars

How can you best approach Agile in medical device development?

Your guide to taking the first steps on the path to Agile in MedTech development!

Access insights

What are key agile practices for the regulated healthcare domain?

Find out how your organization can achieve regulatory compliance incrementally!

Access insights

What experiences have world-class healthcare companies had on their journeys to Agile?

Build on case studies and lessons learnt from leading MedTech companies!

Access insights