Abstract
In safety-critical environments where traceability is a key concern, transitioning to an agile methodology can be challenging. Many teams find it difficult to translate familiar, traditional terms and processes into agile. Some struggle with how to meet regulatory requirements while adhering to the new methodology. Others are met with resistance from within the organization.

This white paper discusses how to overcome these barriers and successfully transition to agile in a safety-critical environment.
Introduction
In the ongoing struggle to simultaneously improve product quality, reduce production costs, and get to market faster, many companies in safety-critical industries are integrating agile into their product development processes.

These agile initiatives can be met with resistance from executives, managers, and team members who believe agile to be a poor fit for environments in which traceability is crucial. In reality, agile works very well in safety-critical environments, and has been adopted by several regulated companies. Agile has even been embraced by regulatory agencies, such as the U.S. Food and Drug Administration (FDA). In fact, in early 2012, the FDA accepted an Association for the Advancement of Medical Instrumentation (AAMI) report as a Recognized Consensus Standard. AAMI TIR45:2012, Guidance on the Use of Agile Practices in the Development of Medical Device Software affects 510(k), IDE, PMA, HDE, and software validation processes for “medical devices that contain software, accessories to medical devices that contain software, and ‘standalone software’ that meets the definition of a medical device or accessory.”

Agile can be better than traditional, waterfall methodologies for matching business needs with development realities, and for reducing wasted time and effort as much as possible. Projects that involve continuous product development and enhancement—like the projects used to bring medical devices to market—are a natural fit for an agile methodology.

Audience
This white paper is primarily intended for executives and managers working in safety-critical environments who:

- Recognize the value of agile development methodologies
- Have committed to transitioning to agile
- Are having trouble with this transition

The barriers encountered may include pushback from others within the company, struggles adapting to agile practices, problems with traceability under the new process, or a combination of these difficulties.
Common Concerns Leading to Pushback

Let’s begin by looking at some of the common objections to adopting agile. Many of these worries persist despite evidence to the contrary. It may help ease the transition to know these common concerns can be overcome.

**Concern 1: Agile Doesn’t Produce the Required Documentation**

Some read the Manifesto for Agile Software Development and focus on “working software over comprehensive documentation.” Many feel they are required to have comprehensive documentation to pass an audit, and therefore agile will not work for their process.

This is a misinterpretation of the Agile Manifesto. What the manifesto actually says is that working software is more valuable than comprehensive documentation, not that it should be eliminated.

Auditors do require documentation to prove that safety-critical products have been thoroughly tested for reliability, and that all known defects have been eliminated. These documents are required because they contribute to the production of a safe, effective product. Like anyone else, however, auditors do not want to wade through piles of useless paper.

Agile proponents argue that teams should only produce valuable reports and should eliminate wasteful documentation. They also recognize that more documentation is necessary in safety-critical, regulated environments. For example, some agile developers have argued that the traceability matrix is unnecessary in an agile environment unless it is also a regulated environment.

Because agile generates less documentation, it can be challenging to maintain the required records and traceability matrices needed to satisfy industry regulations. A product development solution can help with this, and in some cases, can automatically generate the required documentation.

**Concern 2: Agile Won’t Work with Our Product**

One of the underlying principles of agile is that working software is the primary measure of progress. To that end, most agile methodologies call for each sprint to result in a working, demonstrable version of the product. The thinking behind this principle is to have some way to measure how “done” a project is, to avoid always being 80 percent done. Each sprint needs to produce measurable outcomes, so progress can be tracked accurately.

However, in some types of projects—such as embedded software—sprints may not result in features that can be easily demonstrated to or evaluated by users. Some use this as an argument against adopting agile.
It's a valid argument, and purely agile methodologies are seldom used for just this reason. Instead, teams often take what they can use from agile and merge it with the best parts of their old methodology, creating an agile/waterfall hybrid. The point isn't to go agile no matter what; the point is to improve the product development process. A hybrid approach may be what some teams need to fit agile into their unique circumstances. Perhaps using an agile methodology at the team level and waterfall at the project level will be effective.

For products that have both hardware and firmware, waterfall may work best for developing the hardware, while agile best serves the software developers. That is Schneider Electric’s hybrid approach, which Erick Bergmann and Andy Hamilton described in a presentation at the Agile 2013 Conference. (A PDF of their PowerPoint presentation is available on the conference web site.)

**Concern 3: Senior Management Won’t Support a Decrease in Productivity**

Another objection to implementing agile is the expected dip in productivity while the team gets up to speed on the new process. Many may fear the impact a temporary period of low productivity will have on their performance reviews—especially if productivity issues were the reason for the process change in the first place.

There are two ways to counter this objection: by communicating management’s acceptance, and through the use of a pilot program.

All senior managers need to accept that productivity will be down during the transition phase. Although teams learning agile show little velocity in the first few sprints, their productivity under the old process probably wasn’t very good, either. Otherwise, why make the change?

Once management is on board, their acceptance needs to be loudly and repeatedly communicated to the team to allay any doubts the team may have. When team members know they will not be held accountable for low productivity during the transition, they should be much more open to the new agile process.

Beginning with a pilot program is a way to ease management and teams concerned about productivity as they switch to agile. A well-planned, limited pilot program can help minimize the impact on productivity by keeping it confined to the pilot group. Not only will it make senior management happier, but it’s also easier to manage the training for smaller teams and keep them focused.

Smaller pilot programs are also more easily measured, and results are seen more quickly. Once the pilot team begins to show progress, it’s easier to sell the rest of the company on the new process.
Adapting to Agile
For teams used to waterfall, the conversion to agile can be challenging. Many find it difficult to translate familiar, traditional terms and processes into agile. Others struggle with how to maintain the traceability needed to meet regulatory requirements while adhering to the new methodology. This section shows, at a high level, how agile works within a safety-critical environment.

Translating Traditional Terms into Agile
Mapping traditional terms to agile can help teams with the transition by providing them with an agile Rosetta Stone. Each organization will have its own variations on certain terms, but Table 1 provides a starting point for developing an organization-specific map.

Table 1: Traditional terminology mapped to agile terms

<table>
<thead>
<tr>
<th>TRADITIONAL TERM</th>
<th>AGILE EQUIVALENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status report</td>
<td>Burndown/Burnup chart</td>
</tr>
<tr>
<td>Status meeting</td>
<td>Daily standup</td>
</tr>
<tr>
<td>Validation criteria</td>
<td>Definition of Done</td>
</tr>
<tr>
<td>Feature list</td>
<td>Product backlog</td>
</tr>
<tr>
<td>Customer</td>
<td>Product owner</td>
</tr>
<tr>
<td>Project manager</td>
<td>Scrum Master</td>
</tr>
<tr>
<td>Next planned features</td>
<td>Sprint backlog</td>
</tr>
<tr>
<td>Feature planning meeting</td>
<td>Sprint planning meeting</td>
</tr>
<tr>
<td>Post mortem</td>
<td>Sprint retrospective/review</td>
</tr>
<tr>
<td>Phase</td>
<td>Sprint/Iteration</td>
</tr>
<tr>
<td>Requirement</td>
<td>User Story</td>
</tr>
<tr>
<td>Estimate</td>
<td>Story points</td>
</tr>
<tr>
<td>Feature planning for phase</td>
<td>Sprint planning</td>
</tr>
<tr>
<td>Feature planning</td>
<td>Release planning</td>
</tr>
<tr>
<td>Development or team problem</td>
<td>Impediment</td>
</tr>
<tr>
<td>Post mortem</td>
<td>Sprint review</td>
</tr>
</tbody>
</table>
Let’s take a closer look at a few of these agile terms and how they differ from their waterfall counterparts.

To begin with, status reporting tends to be very streamlined in an agile environment. A **daily standup** meeting is focused on what the team did yesterday, what they’re planning to do today, and any roadblocks they have. This is in contrast to a more traditional weekly status meeting, which often either focuses on more nebulous “percent done” items (e.g., task x is 45% done) or long-term plans (e.g., We’re going to be working on feature y over the next few weeks).

Similarly, **burn down charts** reflect what work has been completed, and extrapolates from that a prediction of when the sprint will be completed. This concept of looking at only completed tasks, rather than a more difficult to measure percent done, is critical to improving predictability.

**User stories** are another interesting area. In traditional agile, user stories often replace requirements and requirement documents. These lightweight descriptions focus on describing the who, what, and why of a requirement. Requirement documents most often look at exactly how something should work, with less description of the underlying user need. In a safety-critical environment, these two concepts are married together so that developers can understand the business need (the user story), while keeping parts of the requirement document for verification and reporting.

Finally, the word “**sprint**” shows up a lot in the table on the previous page, as well as in agile literature in general. Sprints are often described as short iterations (a few weeks long at most) that result in a “potentially shippable” product. In complex, safety-critical environments, creating a potentially shippable product in a few weeks is often unrealistic. However, the underlying concept is to focus on short iterations that have visible progress, so that product owners can validate the direction of the team. Again, the goal is to avoid the danger of being 80 percent done, with the final 20 percent needing 80 percent of the total work put into the project. Or even worse, having the work that is done being out of synch with what the product owner has in mind.
When you compare traditional, waterfall development with agile, it looks something like this:

**Figure 1:** A side-by-side comparison of waterfall and agile development methodologies

### Agile Teams
Because agile teams have to design, build, and test a working product in a short period of time, team members need to encompass the entire spectrum of skills. There is no agile design team, agile development team, or agile testing team. There is just an agile team that does it all.

Agile teams usually consist of:
- A product owner
- Developers
- Testers
- A facilitator (Scrum Master)

In regulated environments, quality assurance team members will also assist with verification, validation, traceability and other necessary documentation activities to ensure regulatory requirements are met.
This kind of cross-functional team is the heart of agile development. It is also the biggest stumbling block when companies transition from waterfall to agile, because it requires a major reorganization of the product development team. This kind of disruption to the status quo may cause some discomfort and resistance from individual team members, which can take valuable time away from the important work of making progress on the project.

If a team member's resistance stems from a lack of confidence in agile, the problem may go away after the team completes a few sprints and the value of the methodology begins to become apparent. But if a team member is too negative or actively tries to sabotage the process, the Scrum Master may have to involve the member's manager to help rectify the situation.

While excessive negativity and sabotage should not be tolerated, skepticism can be useful. Skeptics might ask the most valuable questions about the process, or have ideas for improving or customizing the methodology to work better with the company's unique needs.

### Risk Management and Mitigation
Risk management in an agile environment is a little different than it is with waterfall, although the same basic steps are followed. The steps will vary a bit from industry to industry, but they do not differ wildly.

For example, IEC 61508, “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems,” is a basic functional safety standard applied to a broad range of industries. The core activities of a functional safety program as defined by IEC 61508 are:

1. Hazard identification
2. Risk assessment
3. Risk mitigation
4. Assessment of residual risk
5. Safety requirements specifications
6. Hardware and software reliability specifications

When adopting agile in a safety-critical environment, the steps most affected are hazard identification, risk assessment, and risk mitigation. These steps should be performed during the retrospective phase of each sprint, while maintaining a backlog of requirements with associated risk scores for each. This score is then used in evaluating and selecting requirements for the next sprint. We'll discuss this in more depth later in this paper.
Combining Requirements and User Stories

In agile, user stories replace requirements to define a product’s features. User stories help developers and testers to better understand how the end user will interact with particular features, providing a context that regimented requirements lack.

But safety-critical product development is still largely driven by requirements, mainly because of existing development practices and regulatory statutes. Most teams and customers also understand how to write, manage, and build software to requirements.

As a result, many agile teams find they have to work with both user stories and requirements. User stories give teams greater flexibility in designing and implementing features, but can be more difficult to objectively assess for quality because they tend to be less precise.

For requirements and user stories to coexist successfully, teams need to structure their projects to take advantage of the best qualities of both. An effective way to do this is to use requirements as the agreement between the team and the end users, and user stories to build a context around those requirements.

Requirements, which are structured and objective, are best used to define exactly what will be implemented. This includes non-functional aspects of the product, such as security and performance.

User stories can provide context around requirements by helping the team understand how a feature will be used. User stories can give the development team a deeper understanding of the feature and its purpose. These stories can be managed with requirements without much additional overhead, as long as they each use different ways of representing the same features. This could mean that each user story supports multiple requirements, or each requirement uses more than one user story.

Developers will rely on user stories when designing and implementing features, referring to requirements for clarification when necessary. Testers will write and run test cases to support requirements because requirements more formally represent the agreement with the users. If testers also refer to the user stories, they will better understand how to structure the tests, ensuring important features are accurately tested.
User stories and requirements have to be linked together for this approach to work. Team members must be able to determine at a glance what user story supports which requirement and vice versa.

**Figure 2:** User stories and requirements have to be linked together.

And of course, linking is important for traceability. Traceability helps teams determine what else must change if a requirement or user story changes. What test cases are affected? How will the source code change? If artifacts are linked to requirements, and the requirements to user stories, teams can immediately see a change's impact.

Using both requirements and user stories results in a better understanding of the use of the application and the needs of the users. Project teams can work with them to have a better understanding of the business problem while delivering on the needs of the users.

**Conclusion**

While the transition to an agile methodology can be challenging, agile has been proven successful in regulated, safety-critical environments. Companies like Schneider Electric and others have proven agile’s effectiveness.

Because agile generates less documentation, it can be challenging to maintain the required records and traceability matrices needed to satisfy auditors. A product development management solution can facilitate traceability in an agile environment, however.

With an agile methodology, strong traceability, and a comprehensive product development solution, safety-critical companies can quickly realize agile’s full benefits of reduced cost, higher quality, and faster time to market.
About Seapine Software
With over 8,500 customers worldwide, Seapine Software, Inc. is the leading provider of quality-centric product development solutions. Headquartered in Cincinnati, Ohio, with offices in Europe, Asia-Pacific, and Africa, Seapine’s development solutions help organizations ensure the consistent release of high quality products, while providing traceability, metrics and reporting, and compliance.

Learn more at seapine.com