

# CONCEPTS



January 26 2012

## *Agile Requirements Definition and Management (RDM)*

How Agile requirements help drive better results

By **Jason Moccia**

### Executive Summary

One of the myths of Agile software development is that documentation is not required or useful. It is true that one of the core values within the Agile Manifesto is Working Software over Comprehensive Documentation. However, note the word “over” in this statement. The Manifesto is not saying “no” documentation, it is saying there is a preference for working software over documentation. The goal is to remove impediments from the system and leave things that add value. If your organization is creating lengthy documents to produce software and you are still struggling to release software on time and within budget then ask yourself this question, “What value are the documents adding?” The value question is an essential part of Agile as well as Lean Thinking. Lean Thinking was popularized by Toyota and has been widely adopted across many industries. Its premise is to eliminate waste from the system and to get down to the essence of what it takes to drive value to the customer. It also focuses on self-organizing and self-correcting teams to drive quality and efficiency in the system.

The concept of Agile Requirements Definition and Management (RDM) is not a new concept. However, the struggles to figure out how traditional requirements cycles fit within an Agile framework remains convoluted. For a system to work, an organization needs to think about the entire lifecycle and not just the software development portion. If you start at a high-level within your company and analyze what documentation is being produced in order to create finished code, you’ll most likely find that over 50% of the documentation created was not used. Why are we creating waste?

One theory relates to the complexity factor. As a system starts to mature it starts to move toward complexity. It becomes increasingly difficult to understand, organize, manage, and maintain these systems. People have a tendency to add more to the system to help fill gaps and create protocol to maintain and control it. As organizational turnover occurs and the rules, regulations, and business climate change, the system must also adapt. This causes the complexity of the system to grow exponentially until it gets to the point where it is difficult and/or prohibited from functioning properly. This is precisely why the best systems, or for that matter products, are simple and streamlined. This is exactly why Lean Thinking removes waste from the system. Requirements Definition and Management is no different in its vulnerabilities to the complexity factor. This is why companies sometimes end up with requirements specifications that are hundreds of pages long and extremely difficult to follow and manage.

If you’re looking to adopt Agile and want to run a leaner operation, you have to take a holistic view of the organization. Requirements definition in Agile has to be looked at through a separate lens, not strictly in conjunction with the development team. The same principles used in Agile software development can be applied to requirements definition and management as well. Let’s look at a quick example of how this works. There are various Agile frameworks out there, but the most popular is Scrum. Other frameworks include XP, Crystal, and Kanban to name a few, but Scrum is the most commonplace so we will use this for demonstration purposes. It is also important to mention that some of these frameworks can be combined. For example, portions of Kanban can be used within Scrum to control team capacity constraints by limiting work in progress (WIP).

Scrum allows development teams to build software incrementally over 2-4 week events called Sprints [see Figure 1]. Requirements are fed into a Product Backlog prior to Sprint inception, which gets decomposed into Sprint Backlogs Items through Sprint Planning. The development team starts by discussing what needs to be developed in a given Sprint based on the organizational needs and

strategy. The work items are pulled from the Product Backlog and directed by a Product Owner, with the process being controlled and managed by a ScrumMaster. The goal for the business is to make sure they feed the Product Backlog and can support and describe what needs to be built by the development team prior to the Sprint starting. The problem is that most organizations struggle with keeping pace and/or don't have the right level of detail defined in the Product Backlog to properly tie into the development Sprints.

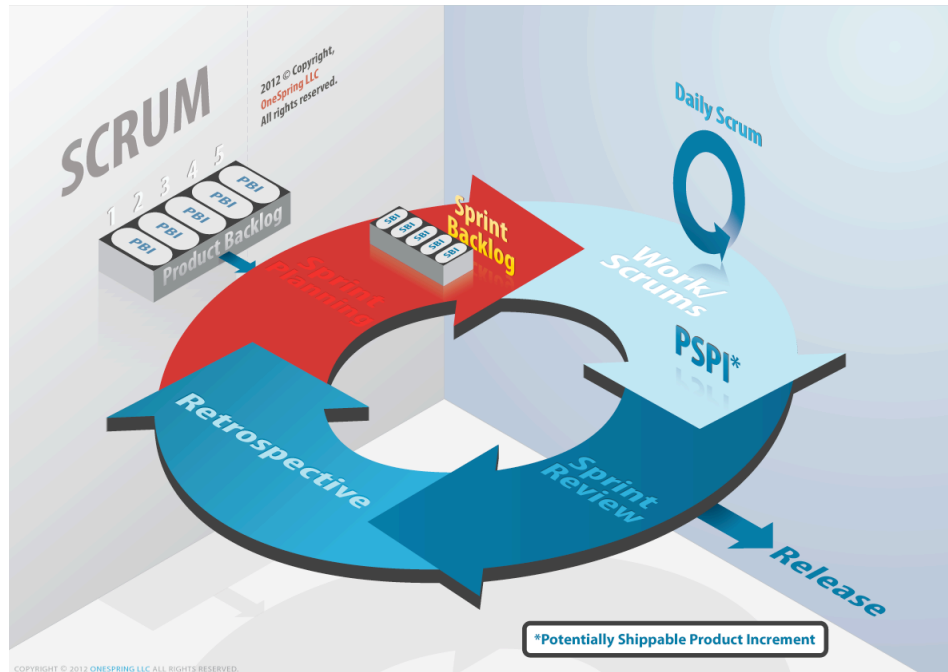


Figure 1: Scrum / Sprint

Agile Requirements Definition and Management was specifically design to solve this problem by outpacing the development team. This is accomplished by feeding the Product Backlog faster than the development team can produce code. The framework can be used for just-in-time requirements definition or to build a repository of requirements for future use. In either case, if a team is using Scrum they are working from the Product Backlog. Since the Product Backlog is a “backlog” of work, the required pace of filling the backlog is driven by the designated Sprint timeframe. The goal is to make sure the business (i.e. Product Owner) can clearly articulate what needs to be built and that what is define is of high quality. To accomplish this the Requirements Cycle follows a Scrum-like process that mirrors the development cycle but stays two to three steps ahead [See Figure 2]. The goal is to create a process by which requirements can be thoroughly vetted, organized and communicated that is iterative, timely, and quality-focused.

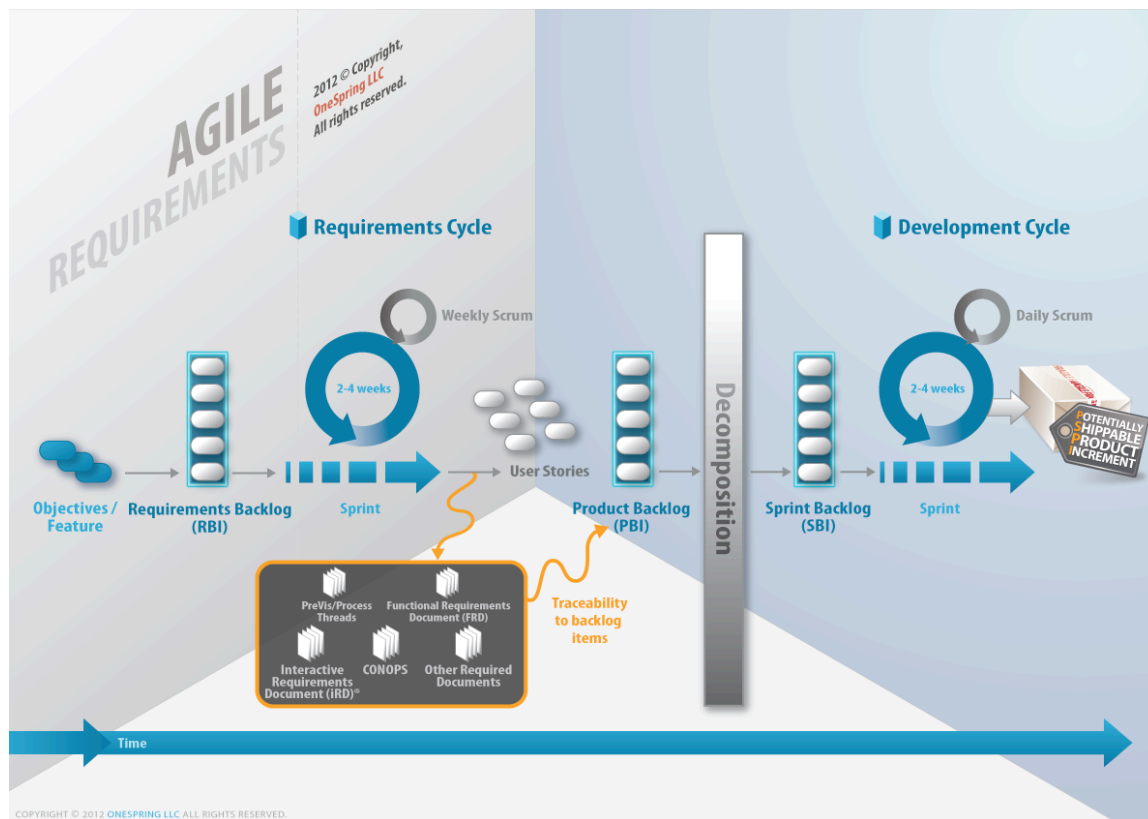


Figure 2: Agile Requirements Definition and Management (RDM)

It all starts by identifying and filling a Requirements Backlog. This type of backlog is a list of items that need to be defined in order to fill the Product Backlog. The end goal could be User Stories, Visualizations, Functional Requirements, etc. The requirements team decides based on the business strategy and objectives what needs to be defined and built through requirements planning and prioritization. Like the development team, the requirements team would plan their Sprint, perform the work, and review the outputs. If the outputs meet expectations then they can be moved to the Product Backlog. In many cases, organizations will have documents that need to be created to pass certain tollgates or organizational milestones. These items can also be put in the Requirements Backlog but may not end up in the Product Backlog. Instead, these documents in many cases become reference material for the development team to pull from. This is where traceability from the Product Backlog to any external documents becomes important to establishing project continuity.

Another important portion of RDM is called Decomposition. Decomposition is the process by which the Product Backlog Items are communicated and refined in collaboration with the development team. Decomposition can be used in several ways. One such way is to setup a culture of collaboration where the development teams are brought into the requirements phase to refine the Product Backlog. In Scrum, this is commonly referred to as grooming the backlog. Another way to use Decomposition relates to procurement and/or timing delays. Some projects experience gaps in time between when requirements are defined and when development starts. There are many reasons why this happens but it is important to note that this occurs on a regular basis. The larger the gap in time between definition of requirements and development, the more risk that occurs with developing the right product. Loss of vital team members, knowledge and overall team availability all are at risk the longer the gap. Decomposition in this case is used as a way to pick up where things left off by using the Product Backlog to communicate and share requirements.

Agile is quickly becoming the most popular way of developing software because it fosters continuous improvement, time-boxed development cycles, and delivering value to the end users faster. That value will be driven to a large extent by the quality and clarity of requirements that feed the software

development process. An agile, lean, and timely approach to requirements as the starting point will help to ensure that your process is optimized.

There are many flavors of Agile on the market today, I've discussed but a few of them in this article. The key is to figure out what works for your organization and to start experimenting. The faster you dive into trying to be more Agile, the faster you will start seeing the benefits it brings.

#### About the Author

Jason Moccia has over 14 years of experience in the software development field and is a co-founder and managing partner of OneSpring LLC ([www.onespring.net](http://www.onespring.net)), where he oversees the Federal business practice as well as Operations. OneSpring helps companies to work smarter by providing an entirely new approach to software requirement definition.

In addition to Mr. Moccia's leadership role within OneSpring, he has also worked as a senior business analyst with numerous Fortune 1000 companies—including, but not limited to, Ernst & Young, General Electric, SAIC, Florida Power & Light, InterContinental Hotels, Deloitte, and SunTrust. Mr. Moccia is a Certified ScrumMaster (CSM) and holds a Bachelor of Science degree in Business Administration from the University of Florida with a focus in Management Information Systems (MIS).