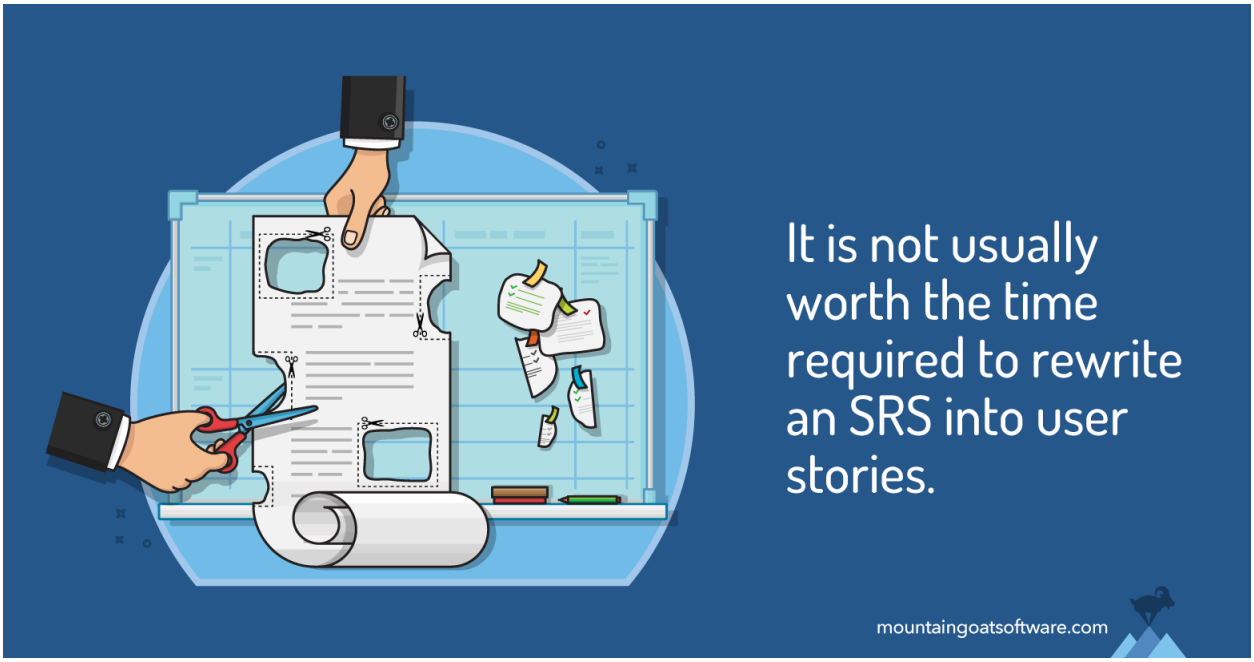

Can a Traditional SRS Be Converted into User Stories?
By Mike Cohn

A lot of traditionally managed projects begin with a Software Requirements Specification (SRS). Then sometime during the project, a decision is made to adopt an agile approach. And so a natural question is whether the SRS can serve as the newly agile project's product backlog. Some teams contemplate going so far as rewriting the SRS into a product backlog with user stories. Let's consider whether that's necessary.

But before taking up this question, I want to clarify what I mean by a Software Requirements Specification or SRS. I find this type of document to vary tremendously from company to company. In general, though, what I'm referring to is the typical document full of "The system shall ..." type statements.

But you can imagine any sort of traditional requirements document and my advice should still apply. This is especially the case for any document with numbered and nested requirements statements, regardless of whether each statement is really written as "the system shall ..."

Some Drawbacks to Using the SRS as a Product Backlog

On an agile product, the product backlog serves two purposes:

- It serves as a repository for the work to be done
- It facilitates prioritization of work

That is, the product backlog tells a team what needs to be done and can be used as a planning tool for sequencing the work. In contrast, a traditional SRS addresses only the issue of what is to be done on the project.

There is no attempt with an SRS to write requirements that can be implemented within a sprint or that are prioritized. Writing independent requirements is a minor goal at best, as shown by the hierarchical organization of most SRS documents, with their enumerated requirements such as 1.0, 1.1, 1.1.1, and so on.

These are not problems when an SRS is evaluated purely as a requirements document. But when the items within an SRS will also be used as product backlog items and prioritized, it creates problems.

With an SRS, it is often impossible for a team to develop requirement 1.1.1 without concurrently developing 1.1.2 and 1.1.5. These dependencies make it not as easy as picking a story at a time from a well-crafted product backlog.

Prioritizing sub-items on an SRS is also difficult, as is identifying a subset of features that creates a minimum viable product.

A Software Requirements Specification is good at being a requirements specification. It's good at saying what a system or product is to do. (Of course, it misses out on all the agile aspects of emergent requirements, collaboration, discovery, and so on. I'm assuming these will still happen.)

But an SRS is not good for planning, prioritizing and scheduling work. A product backlog is used for both of these purposes on an agile project.

My Advice

In general, I do not recommend taking the time to rewrite a perfectly fine SRS. Rewriting the SRS into user stories and a nice product backlog could address the problems I've outlined. But, it is not usually worth the time required to rewrite an SRS into user stories.

Someone would have to spend time doing this, and usually that person could spend their time more profitably. I would be especially reluctant to rewrite an SRS if other teammates would be stalled in starting their own work while waiting for the rewritten product backlog.

A ScrumMaster or someone on the team will have to find ways of tracking progress against the SRS and making sure requirements within it don't fall through the cracks. Usually enlisting help from QA in validating that everything in an SRS is done or listed in the product backlog is a smart move.

One additional important strategy would be educating those involved in creating SRS documents to consider doing so in a more agile-friendly manner for future projects. If you can do this, you'll help your next project avoid the challenges created by a mismatch between agile and starting with an SRS document.

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.