

## Create a Lean, Mean Requirements Machine

Much ink has been shed discussing the many ways in which requirements should be gathered. The following is a guide to Agile requirements-gathering.

### Waterfall vs. Agile

First, some background. In a waterfall software development project, the vast majority of requirements are gathered at the start of the project. This takes an extraordinary amount of time, and results in a collection of documents, diagrams, and flow charts that become stale before the project is ever delivered. Said succinctly, this is the wrong way to gather software requirements.

Think of it this way: if you were building a house or a bridge, a thorough requirements-gathering process would make sense because it is nearly impossible to make adjustments to the design halfway through construction. Software, on the other hand, has a unique advantage in that it is flexible; requirements can shift along the way as business needs change.

In an Agile software development project, requirements are iterative. In other words, the broad scope of the project is defined upfront; each iteration has its own unique set of requirements. This approach has several benefits – most notably, a more efficient change control process. For instance, if your competitor releases a new feature or function, your team can quickly change course to compete with this feature by pushing other requirements down in order of priority.

Because Agile is built around the idea of a fast, customer-focused, and iterative delivery process, some would go so far as to argue that the words, “agile,” and, “requirements,” should never be uttered in the same sentence. While the word, “requirements,” technically does not exist in Agile’s glossary of terms, the truth is that “requirements” are fulfilled by way of alternative artifacts known as User Stories.

## User Stories

Requirements of an Agile project are gathered through user stories. Rather than a traditional “requirements” document, user stories are typically captured on index cards or sticky notes and arranged on walls or tables to facilitate planning and discussion. User stories are short and simple descriptions of a feature told from the perspective of the end user. The following template is usually followed:

*As a <user type>, I want to be able to <goal> so that <reason>.*

Here are three examples of user stories:

- As a manager, I want to be able to review my employees’ time sheets so that I can monitor their productivity and allocate resources appropriately.
- As a marketer, I want to be able to select a date range when reviewing my website traffic so that I can compare it to previous date ranges and make adjustments to my marketing strategy if necessary.
- As a system administrator, I want to be able to preview changes I make before publishing them in my live environment.

In the same way that a fiction story builds and grows, user stories develop and change with time, chapter-by-chapter, release-by-release. By shifting the focus away from requirements and functionality, and reframing the conversation into *actual* user needs and desires, product owners can better prioritize development.

**User stories effectively replace the need for a requirements document.** User stories typically live in the product backlog, and most Scrum projects rely on the product backlog to prioritize the list of features and functionality to be developed.