How to Create a Product Backlog: An Agile Experience

In: Agile Business Analysis, Requirements Models and Specifications By: Laura Brandenburg

The product backlog is really the core deliverable that maintains and evolves the requirements in an agile environment. Ownership by the agile business analyst (or a product owner with BA responsibilities) is critical.

A product backlog contains a complete list of all requirements under consideration (written using a user story syntax – more on that below), rank ordered, and matrixed with other key characteristics that facilitate planning and prioritization.

Let's look at how the product backlog emerged for one particular project and how decisions were made about what to include in the product backlog.

Switching from Traditional to Agile – And Getting Started on the Product Backlog

This project was initiated with a traditional approach. The business analyst created a fairly traditional **scope statement and features list**. The features list was fundamentally business-driven. When I started work on the project we had not yet defined how we'd manage requirements communication with the out-sourced development team.

At the project kick-off with the development team, we reviewed a fairly final draft of the scope statement and started discussions around detailed requirements communication. The development team is part of an agile shop, so regardless of who did it, a product backlog and user stories would be created. The team agreed that it made sense for the business analyst to own creating and maintaining the product backlog and user stories.

Blending Requirements and Design in the Product Backlog

Identifying a user story blends elements of requirements and design. As I worked through my list, driven by my understanding of what the business

wanted, I kept running up against the question of, “How will it make sense to build this in a delivery cycle?”

To balance these two perspectives, I drafted the product backlog, then we tore it apart as a team into deliverable nuggets of functionality. For us, the question the product backlog answered was, “G*iven what we know about scope, what is the best way to deliver in an agile environment?*” So, we were blending agile with more traditional methodologies a bit to the benefit of the business team (that cares about scope) and the technology team (that has optimized their development process to deliver in sprints).

Using the User Story Syntax in the Product Backlog

The second challenge I encountered really centered around the syntax of a user story. In the past, I’ve been an **“ability to” analyst**…never again. This time I used the following syntax to write requirements in my original scope statement.

“[**Somebody**] does [**something**] with [**some information**]”.

This provides a much more powerful way to capture features and also ensures you are capturing all aspects of the feature. And the standard user story format?

“As a [**user**], I can [**do something**] so that [**perceived benefit**].”

These were surprisingly similar. I really played around with the benefits statement and found that sometimes it added real value to the story and other times real confusion. I decided to consider it optional for this project.

But the standard user story format was missing the “some information” component that had really helped flesh out many of my requirements. Therefore, most of the product backlog ended up in the following blended format (items in parens are optional):

“As a [**user**] (*with some information*), I can [**do something**] (*with some information*) (*for some perceived benefit*).”

Evolving the Product Backlog

Over the course of the project, the product backlog continued to evolve and was "groomed." As we drilled into the requirements behind some of the earlier user stories, we discovered they were bigger than anticipated. We broke apart user stories into two or more product backlog items. On occasion, we combined stories.

As new stories were added, we assigned them story points (a kind of estimate) and a general priority. We rank ordered the next 20 or so user stories. As stories were targeted for a specific sprint, we added that information.

The backlog became a tool to scope and re-scope the project and it proved exceedingly flexible. Since we were using Team Foundation Server (TFS) to manage our user stories, I could download the complete product backlog and with some simple formatting in Excel share it with the business team. Then I could input my changes and update the user stories. Eventually, we even got the syncing functionality to work so I could make updates in Excel that were reflected back in TFS.

Managing a product backlog is a key way for the business analyst to add value to an agile project. It's a wonderful tool for keeping business needs in sync with development deliverables and for streamlining how the requirements get managed throughout the development process.

http://www.bridging-the-gap.com/an-agile-experience-my-first-product-backlog/