

Product Backlog: The Foundation of Your Agile Success

An organized and categorized backlog is the foundation of your Agile success. The backlog is used to prioritize customer requests and ensure the team is working on the most important features for your business. Agile practices work because they deliver value to the customer in increments, and give customers the opportunity to offer real and relevant feedback to your team at defined intervals throughout the development cycle. Without an up-to-date backlog, Agile teams are forced to pause between each development interval while decisions are made about what to work on next. Worse yet, an outdated backlog may cause your team to spend effort on features that are no longer the best value for the business.

Getting Started

Before your team can go Agile, the product owner must build a backlog for the product or project. If you're starting a project from scratch, this process is straightforward. Sit down with the customer and end users to discuss what critical problems the project will solve or challenges the customer is hoping to overcome with this product. It should only take a few hours to outline enough requirements for the first few sprints.

If you're working on an existing product, building the initial backlog is more complex. You likely have thousands of feature requests, bugs, and other tasks that have been cataloged over the years. The problem isn't filling the backlog, it's deciding which items are the most important to work on next. Start by adding known higher-priority tasks that support your release and sprint goals. This should give the team enough to choose from for the first couple of sprints. After the first sprint is planned, the product owner will iterate over the backlog again and pull in more issues, tasks, and ideas from the master list.

Add Tasks to the Backlog

Three types of work items fit in the backlog.

Bugs

Bugs and defects are problems found by development, testing, and end users. In a waterfall process, testing is typically the last step of the development lifecycle and it's quite common to push a release live with a large collection of minor (and sometimes moderate!) defects. Bugs pile up over the years, and should be included in the backlog and prioritized accordingly.

Technical Debt

Over time, the direction and scope of a product usually changes. Performance and scalability expectations change. New technology or best practices become available. While it's easy to get buy-in that updating the existing solution to address these types of issues is a good idea, in practice it can be difficult for the product owner to prioritize them over highly visible features requested by customers. These types of backlog items are often referred to as 'technical debt' because they so frequently accumulate over time. Examples of technical debt include upgrading to the latest third-party libraries, making architectural changes to support better scalability and configurability, or refactoring the source code for easier maintainability in the future. These tasks need to be included in the backlog and prioritized along with defects so they have visibility in the planning cycle.

New Features

Feature requests come from a variety of sources, including end users, sales, support, and product management. They can be the hardest to prioritize as you balance the competing needs of keeping your existing customer base satisfied, satisfying the needs of near-term sales opportunities, and working toward a longer-term vision of your product. It's critical that your product owner routinely monitors these sources and arbitrates potentially conflicting requests to ensure the backlog contains the features that will attract new customers and build loyalty with existing customers.

Organize the Backlog

Next, you need to classify and prioritize the backlog. It doesn't do any good to have a backlog if you can't quickly analyze it by different criteria. Classification and prioritization are largely manual processes, but this goes faster than you might expect.

When first getting started, don't worry about organizing your entire backlog. Find the tasks that are clearly high-priority, mark them as such, and ask the development team to tag them as outlined below. Slowly expand classification and prioritization to include more and more of the backlog as needed, based on your sprint schedule.

Classification

We recommend starting with the following two classification categories: functional area and theme. Functional area is simply the area of the product a defect, technical debt, or new feature applies to. Developers can help classify work items because they have an in-depth understanding of the underlying codebase in the project. For example, if refactoring the user security model is a top priority, then planning a sprint to address that specific area of the product is easier if you've already tagged your backlog by functional area.

Classifying by theme is really about grouping tasks together to make sprint planning a smoother process. Tagging each task with a theme gives you greater flexibility when planning a sprint. Let's say that customers are struggling to find data in the system you're building. If you have tagged your backlog with different theme tags—usability, for example—building a sprint to address immediate customer issues is easy. Simply pull out everything tagged 'usability,' and use that subset of tasks to plan your sprint.

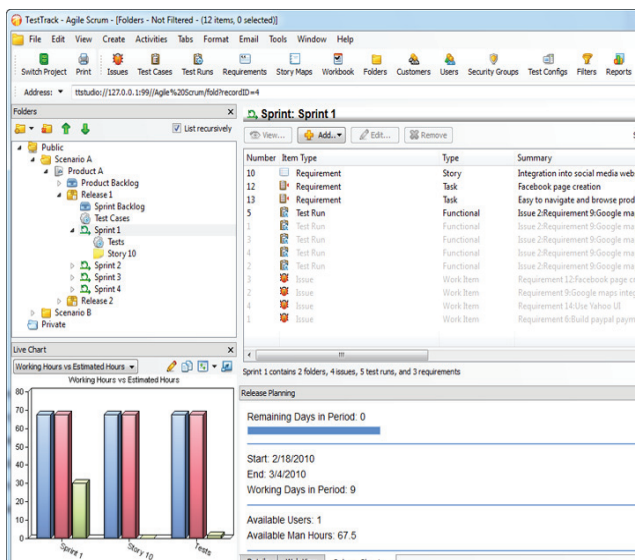


Figure 1: Issues classified with TestTrack folders



Never-realized low priority items!

Simple, non-critical items always seem to end up at the bottom of the pile. This is where classification helps by making it much easier to find related items, regardless of priority, during sprint planning.

Prioritization

Next, you'll want to prioritize the backlog, which is typically based on business value. This is where product owners use their understanding of the needs of their customers and the business. In the best cases, you can apply a direct business value to an item (e.g., "accounts we'll win if we implement this"). However, in reality, you often can't clearly quantify an item.

Some people like to prioritize every item from 1 to N. Others prefer a bucket approach, whether high, medium, low, or a scale from 1 to 10. In general, the bucket approach is easier, especially if you're starting from scratch with a large backlog. After tasks are prioritized into buckets, you can refine the buckets as many times as you want. An organized product backlog, with a good classification and prioritization scheme, is essential to your Agile success.



Ever-changing priorities!

One of the goals of an Agile methodology is to make sure the product reflects what the customer needs. However, it's easy to fall into a 'loudest customer first' approach, or the related 'this week's sales opportunity' prioritization. If you use a bucket prioritization approach, you can ensure you never have more than a certain number of items in the top priority bucket. If you add something to the top priority bucket, you have to take something else out.

Populating the Sprint Backlog

The sprint backlog comprises the tasks pulled from the product backlog for a specific sprint. The team has committed to delivering every task in the sprint backlog within the next sprint's timeframe, so the team should be responsible for choosing tasks for the sprint. A common hurdle seen in organizations just starting out with Agile methods is that management still wants to define what goes into every release.

With Agile, it is important that the team defines what will be delivered in every sprint. This might seem like a minor point, but it's critical to transitioning to an Agile mindset. If management sets the deliverables, the team is not forced to commit to and own the sprint. Your product owner must prioritize and communicate the product backlog effectively. If that's done on the front-end, the development team will naturally balance the demands of the customer and schedule.

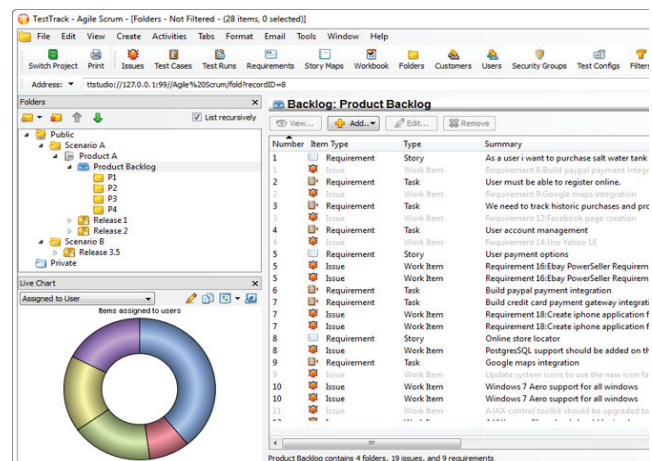


Figure 2: Sprint backlog including requirements, test runs, and issues

Once the sprint backlog is defined, the Scrum Master takes ownership of those tasks from the product owner. A burn down chart is used to gauge developer progress against the sprint deadlines. If the team took too much backlog into the sprint, the team negotiates with the product owner to take tasks out of the sprint and put them back into the product backlog. If the sprint is progressing faster than planned, the team can negotiate with the product owner to finish early or to pull more tasks into the sprint to meet the original schedule.



Make sure testing tasks are in the sprint backlog!

After the team commits to the sprint, everything in the sprint should be in the sprint backlog. It's common for new Agile teams to focus on developer tasks and leave QA out of the loop. This provides an incomplete view of the sprint because testing effort is not included. QA is still left until the end of the development process, and because the results are the same, management is left wondering why Agile is better.

Maintaining the Backlog

Building and organizing the backlog is always a work in progress. Following an Agile methodology is about making sure that what you're working on is what the customer actually needs. The product owner must commit to maintaining the backlog on a regular basis for a few reasons.

First, customer needs change over time, and the backlog must stay in sync with those changes. For instance, flip phones were all the rage a few years back, then it was mobile phones with QWERTY keyboards, and now it's touch screen smart phones.

The backlog for the input device of a mobile phone these days is much different than it was a couple years ago. The product owner is responsible for staying on top of those trends and adjusting priorities accordingly.

In addition, implementing features can change or eliminate the need for other seemingly unrelated features. For example, implementing user authentication with a certain protocol could make the selection of other protocols unnecessary. Again, the product owner is responsible for managing those changes, and better categorization can make that process easier.

Finally, the backlog is open to everyone. All team members and project stakeholders can add ideas and tasks to the backlog. The product owner is responsible for taking those new ideas, vetting them against existing ideas, and classifying and prioritizing them.



Idea dumping ground!

Without steady attention and grooming, backlogs can turn into a dumping ground where good ideas go to die. Product owners should review their backlogs regularly. You should also have some mechanism in place to escalate aging requests.

In a Nutshell

Proper planning and organization is critical to your Agile success, and that's where backlogs come into play. Here's what we learned:

- The product owner is responsible for the product backlog, prioritizing customer requests, and ensuring the team is working on the most important tasks.
- The backlog can include bugs, technical debt, and new feature requests, and should be organized into classification categories, such as functional area and theme.
- The product owner should then prioritize tasks in the backlog, typically based on business value.
- The tasks in the product backlog are then used to populate the sprint backlogs.
- The product backlog must be maintained on a regular basis to allow for project changes, such as changing customer needs.