Implementing the Product Backlog

A practical overview on how to setup an effective Product Backlog

Auteur Ron Eringa

Implementing the Product Backlog

A practical overview on how to setup an effective Product Backlog

Introduction

One of the most important artifacts in Scrum is the Product Backlog. The Product Backlog is owned by the Product Owner and should be the single source for any changes that you make to your product.

The Scrum Guide tells us WHAT a Scrum Team should do with the Product Backlog to guarantee that we are continuously focusing on building the highest value features each Sprint. What the Scrum Guide doesn't tell us is HOW to implement this Backlog and how it is used on a day to day basis. This is actually a good thing, since we don't want to be too prescriptive towards our community, do we?

This means we are creating the boundaries for a successful Scrum Implementation without telling people how they should do their work (**Individuals and Interactions** over **Processes and Tools**).

This whitepaper contains a number of best practices on HOW your Product Backlog can be implemented. This paper is for those who are struggling with Product Backlog implementation or are searching for new ideas to improve their current implementation.

What is a Product Backlog Item?

Scrum Guide:
"The Product Backlog lists all features, functions, Requirements, enhancements, and fixes that constitute the changes to be made to the product in future Releases. Product Backlog Items have the attributes of a description, order, Estimate, and Value."

Injecting passion, Agility and quality into your organisation.

RON ERINGA
AGILE COACH

After being graduated from the Fontys University in Eindhoven, I worked as a Software Engineer/Designer for ten years. Although I have always enjoyed technics, helping people and organizations are my passion. Especially to deliver better quality together. When people focus towards a common goal, interaction is increasing and energy is released. This is the beauty of the Agile way of working.

Personally I have about seven years of experience with all kinds of Agile practices, like Scrum, Lean, Kanban, TDD, and Continuous Integration. During my experience I had roles as a Coach, Trainer, Scrum Master, Product Owner, and Agile Project Manager. In addition, I also speak regularly at various Agile conferences and seminars.

Themes, Epics, Stories, and Defects
A Product Backlog Item defines all work that needs to be done to make an incremental update to your product. The amount of work related to a Product Backlog Item can differ from a few hours to months. A Product Backlog typically contains small Items with high priorities on the top and big chunks with low priorities on the bottom.

Scrum does not prescribe how a Product Backlog Item should look like, since we don't want to tell people how they should do their work. However there are a number of good practices in eXtreme Programming (XP) that can be used to make the definition of a Product Backlog Item more explicit.

While Scrum only talks about Product Back Items, the Agile community (www.mountaingoatsoftware.com/blog/stories-epics-and-themes) has more manifestations for a Product Backlog Item:

- User Story: User Stories describe a new functionality from the perspective of the person who desires the new functionality. A User Story should be small enough for a team to be picked up in a Sprint.

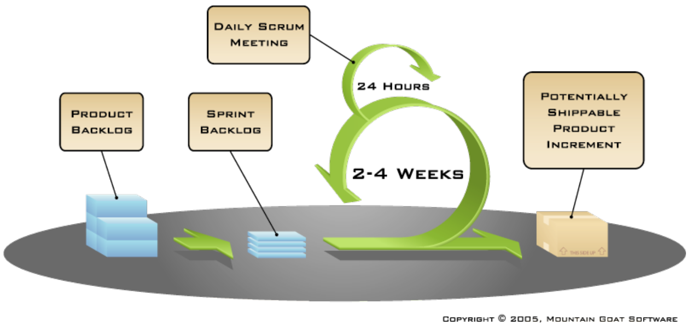
- Epic: An Epic is a large User Story. It is a label that is used to indicate that a User Story can not be picked up in a Sprint, since it is simply too large/complex and needs to be broken down into smaller User Stories.

- Theme: A Theme is a group of User Stories. Sometimes it is useful to speak in groups of User Stories, for example when we are talking to customers about the content of a Release.

Just like Themes, Epics, and User Stories (which are typical terms that come from XP) a team can also put Defects on their Product Backlog. This means that Defects can be just another implementation of a Product Backlog Item and can be treated as a Product Backlog Item in terms of Planning, Estimation, and Priority.

Structuring the Product Backlog

In most overviews and explanations of the Scrum Framework the Product Backlog is depicted as a simple 2 dimensional list with small Items on the top, medium Items in the middle, and large Items on the bottom.


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

1. Standard Scrum picture with a simple representation of the Product Backlog

This is a simple, straightforward way to explain the functionality of the Product Backlog.

However, once people start to implement the actual Product Backlog by using tools, they quickly find out that this 2 dimensional list quickly transfers into a complex tree that:

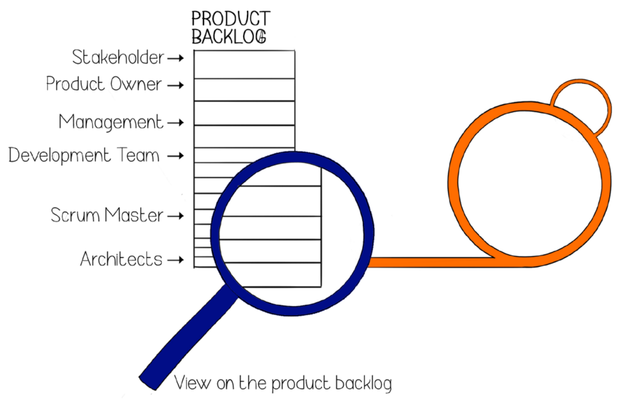
- Evolves over time.
- Contains detailed and high/medium level Items.
- Contains different type of Items (Epic, Stories, and Defects).
- Tends to get even more difficult when it is divided over multiple teams.
- Can take a lot of time to keep it up to date.

2. Example: Product Backlogs can become complex and hard to maintain

Different Users, different Requirements

Since the Product Backlog is the single source for maintaining all product changes, a lot of people will be interacting with the Product Backlog, each with their own Interest and Requirements. The Product Backlog needs to be transparent and should facilitate the collaboration between the Stakeholders, Product Owner, and Development Team.

Scrum Guide:
"Product Backlog Management includes ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next."



3. A Product Backlog has many Users, each with their own view

Development Teams
Development Teams tend to focus on the highest priority User Stories on the top of the Product Backlog. Together with the Product Owner it is their task to breakdown large Epics, that don't fit in a Sprint into smaller User Stories that are ready to be picked up in a Sprint. The Development Team helps the Product Owners to elaborate the details.

Some example Requirements that Development Teams have for the Product Backlog:

- As a Development Team we would like to be able to create, change, and delete User Stories on the Product Backlog, so that we can help the Product Owner in preparing the content for the upcoming Sprints.
- As a Development Team we would like to have a good overview of the total Product Backlog (containing ID, Label, and Size), so that our Backlog Refinement sessions are very effective.
- As a Development Team we would like to effectively split User Stories into smaller User Stories without losing the higher level information, so we can help the Product Owner in prioritizing important features over less important ones.
- As a Development Team we would like to add details (Story Points, Description, Attachments and Acceptance/ Demo Criteria) to our User Stories, so that we can use this information during our estimation sessions.

Stakeholders
Stakeholders tend to focus on larger Epics and only a subset of the Product Backlog (the part that represents their Business representation). Most of the times they want to see the progress of the stories and not so much all the details, where a Development Team is focusing on.

Some example Requirements that Stakeholders have for the Product Backlog:

- As a Stakeholder I would like to see the progress of the Development Team(s), so that I can prepare for new product functionality that will become available.
- As a Stakeholder I would like to see what stories a team has picked up, so I can determine my involvement in the current Sprint.
- As a Stakeholder I would like to see the Acceptance Criteria of an Epic/User Story, so that I can verify if my functional behavior is according to my expectations.
- As a Stakeholder I would like to see which teams are allocated to the creation of my functionality, so that I can determine on which teams I should focus in the coming Sprint(s).
- As a Stakeholder I would like to see the relation between my User Stories and those of other Stakeholders, so that I discuss Business Value and priorities.
- As a Stakeholder I would like to have a place where I can capture new ideas, so that the Product Owners can determine which new functionality should be added to the Product Backlog.

Product Owners
The Product Owner is the most intensive user of the Product Backlog. He should be the one who keeps it lean and mean, so that everyone understands the content. The Product Owner is responsible to make the Product Backlog reflect the latest status of all requested functionality from the Stakeholders. He uses the Product Backlog refinement sessions as an input for this.

Some example Requirements that Product Owners have for the Product Backlog:

- As a Product Owner I would like to have an overview of how high level Epics are translated into low level User Stories, so that I can consider risks and relations between different functionality.
- As a Product Owner I would like to plan my Sprints and Releases using the content of the Product Backlog, so that I can manage the expectations of my Stakeholders.
- As a Product Owner I would like to visualize the progress (for example, by using a Release Burndown) using the sum of all Product Back Items, so I can align Project/ Release Plans based on the latest Estimates.
- As a Product Owner I would like to spend my time efficiently on administrating the Product Backlog, so that I can focus on collaborating with the Development Team and the Stakeholders.
- As a Product Owner I would like to define a Portfolio (or Themes) containing a subset of Epics/User Stories on my Product Backlog, so that I can make better strategic decisions.
- As a Product Owner I would like to specify the Business Value of the User Stories on my Product Backlog, so that I can make better priority decisions.

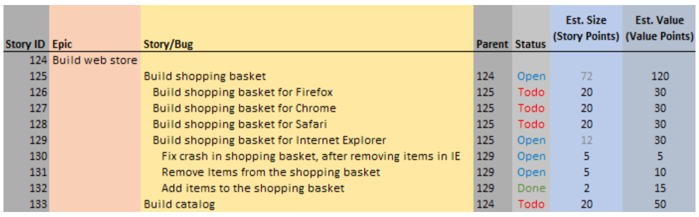
For a good explanation on how a Product Owner uses the Product Backlog, check out the video by Henrik Kniberg called "Agile Product Ownership in a Nutshell".

Visualizing the Product Backlog

When implementing the Product Backlog there are different techniques that you can use to represent the content of your Product Backlog. A common practice is to have some kind of central repository (differing from a simple spreadsheet to a complex Agile ALM tool) to store the actual Product Back Items. On top of that you can use different techniques to represent the content of this repository.

Hierarchical trees
The most common way to structure multiple levels of detail in a Product Backlog is by using a tree structure, comparable to the directory structures used in most Operating Systems. An implementation like this could be realized by using simple spreadsheet tools.

When starting with a new Scrum Team in a small setting, a spreadsheet is a simple and cheap way to maintain a Product Backlog. However, once your Product Backlog becomes more complex, it will become more and more difficult to maintain a Product Backlog in a spreadsheet.

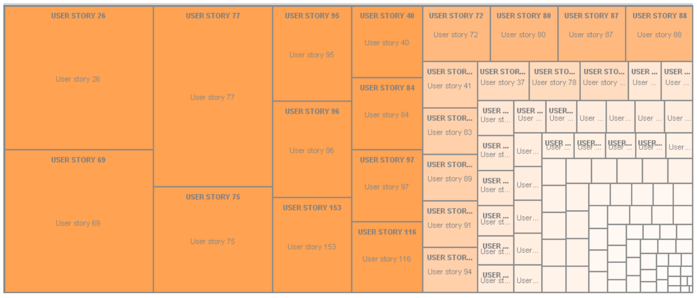


Story ID	Epic	Story/Bug	Parent	Status	Est. Size (Story Points)	Est. Value (Value Points)
124	Build web store					
125		Build shopping basket	124	Open	72	120
126		Build shopping basket for Firefox	125	Todo	20	30
127		Build shopping basket for Chrome	125	Todo	20	30
128		Build shopping basket for Safari	125	Todo	20	30
129		Build shopping basket for Internet Explorer	125	Open	12	30
130		Fix crash in shopping basket, after removing items in IE	129	Open	5	5
131		Remove Items from the shopping basket	129	Open	5	10
132		Add items to the shopping basket	129	Done	2	15
133		Build catalog	124	Todo	20	50

4. Example: Product Backlog using hierarchical tree representation

TreeMapping
TreeMaps can be used to visualize the size of different Items in a list. A TreeMap is a very convenient tool for a Product Owner if he wants to make priority-decisions in his Product Backlog based on the size or Value of a Story. Unfortunately the big Agile tool vendors have not yet discovered this handy way of representing and hence maintaining a Product Backlog, but there are tools that can be used to create such visualizations:

- IBM's Manyeyes
- TreeMapGViz in Google Docs



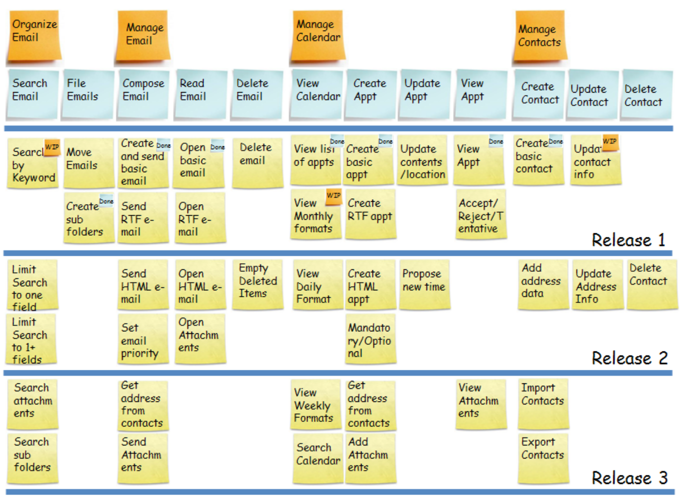
5. Example: Product Backlog using TreeMap representation

For more details on:

- TreeMapping, see Wikipedia
- For a practical example how TreeMaps help you manage files on your harddisk (in the analogy of PBI's in a Product Backlog), see SequoiaView
- How to visualize your Product Backlog with a TreeMap, see this article by Mike Cohn.

Story Mapping
User Story Mapping is a way to visualize your Product Backlog in a way that:

- You are able to identify relations between User Stories.
- You are able to effectively plan Releases.
- You can see the status of your Product Backlog in a quick overview.
- You can easily track all changes in your Product Backlog.
- You can see how an Epic or Theme is translated into smaller Stories on the Product Backlog.



6. Example: User Story Map

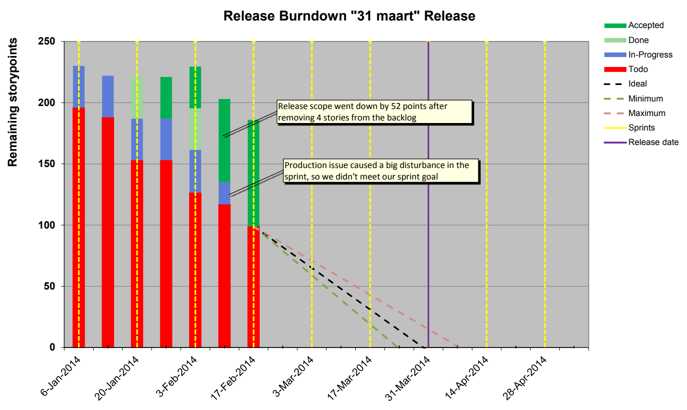
By using a Story Map you can help your Team and your Stakeholders to oversee the big picture in your Product Backlog.

Visualizing a Release Burndown

A Release Burndown is a great tool to keep the Stakeholders informed of the progress in your Product Backlog. Once you have planned a Release, the Release Burndown keeps you up to date on the progress towards the Release.

Many Agile ALM tools have a default, built-in possibility to visualize the Release Burndown, each with their own features and configurable options.

A number of best practices are gathered in the Release Burndown below.



7. Example: Release Burndown

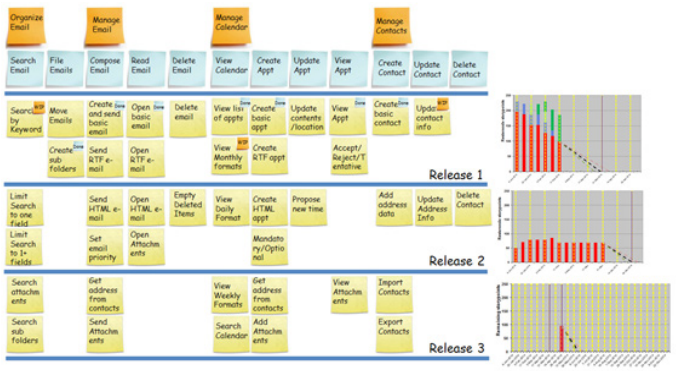
Clarification:

- The red bars in the Release Burndown indicate the remaining work in the Product Backlog.
- The blue bars indicate the work encapsulated in the current Sprint.
- The light green bars indicate the work that is finished in a Sprint.
- The dark green bars indicate the accepted (by the Stakeholders during Sprint Review) User Stories.
- The dotted yellow lines indicate the Sprints.
- The purple line indicates the Release Date.
- The dotted lines indicated by 'Ideal', 'Minimum' and 'Maximum' indicate the possible Release Date (based on the Minimum, Ideal, and Maximum Velocity).
- Since the bars are cumulated the total of each bar indicate possible Scope changes in the total Backlog.
- With the annotations you can indicate big changes in the Scope of the Product Backlog. This is a convenient way to visualize the history of the Product Backlog.

The interpretation from the example above:

"This Release Burndown is updated weekly. We are now on Feb 17 2014 with three more Sprints (of two weeks) to go for this Release. Based on our current Velocity we will probably finish all User Stories planned for this Release. In the last/current Sprint we had to remove some User Stories from the Release Scope in order to make the end date for this Release (so we probably have a fixed time restriction for this Release). When we have no setbacks we might finish the Release Scope half way the last Sprint, but if we have some setbacks we might overrun the Release Date and finish a week later than the last Sprint."

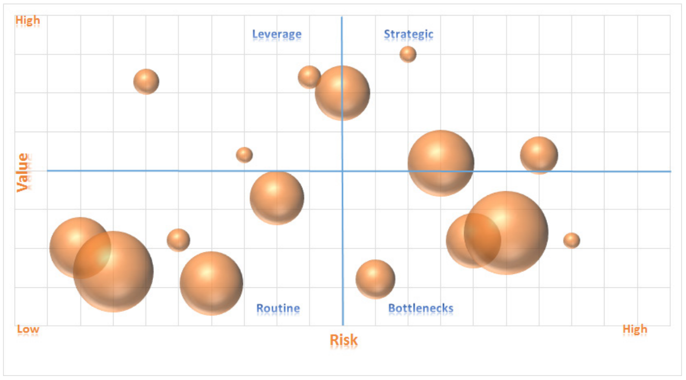
Combining a Story Map with a Release Burndown
It is a good idea to combine a Story Map with a Release Burndown. Since the Story Map indicates which stories are on the Backlog and for which Sprint/Release they are planned. Combined with a Release Burndown, Stakeholders can immediately see what the estimated end dates are for that Release.



Prioritize on Risk, Value, and Complexity
Product Owners are responsible for prioritizing the Product Backlog. There are many factors that can influence the priority of an Item on the Product Backlog. Some of the most used factors are:

- Risk. Risk can be related to product availability, number of suppliers, ease or cost of switching supplier, or the availability of substitute products or services.
- Value. Value can be related to Profit Impact, Volume or Value purchased, impact on Supply Chain "Value-add", Business Growth Potential, or Dependency.
- Complexity. In Scrum we often use Story Points, a relative factor to express the complexity or effort for creating a Product Backlog Item.

In many marketing departments the Risk-Value Matrix (also called "Kraljic Model") is used to determine the priority of new features in a product portfolio. This model can be helpful for a Product Owner to determine the priority of the Items on his Product Backlog. [Download the Risk-Value Matrix.]



The Risk-Value Matrix works as following:

The Items in the 'Routine' quadrant are 'Low Risk, Low Value' adding. This means that probably all your competitors are able to reproduce the functionality and besides that these features are not the ones your customer is waiting for.

- The Items in the 'Leverage' quadrant are 'Low Risk, but High Value'. These are the features that:

  - can be built in, since the technology is simple and plenty available or
  - do not have a major impact on the architecture of the system

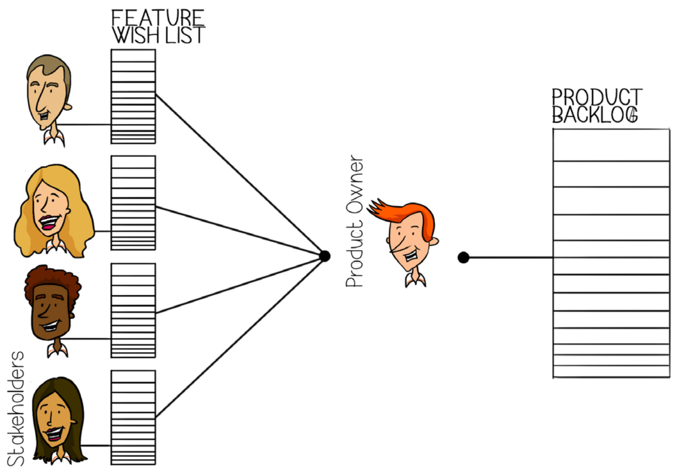
- The Items in the 'Bottleneck' quadrant are 'High Risk, but Low Value'. These are the features that may cost a little and delivering some nice to have features, but who might have a big impact on the architecture of the system.

- The Items in the 'Strategic' quadrant are 'High Risk, High Value'. These are the features that cost a lot (for example by using a new technology), but might be delivering a lot of Value as well.

The size of the Items in the Matrix reflect the amount of estimated effort to build the Item.

A Product Owner would typically focus on the small Items in the 'Leverage quadrant' and avoid the Items in the 'Bottleneck quadrant', while once in a while adding some of the features in the 'Strategic quadrant' on the top of the Product Backlog. Since most Agile Product Backlog tool vendors have not yet discovered the use of the Risk-Value Matrix in combination with the Product Backlog, this practice can be done most easily by using a spreadsheet.

Adding new ideas to the Product Backlog
As already mentioned, the Product Backlog is the single source for any changes that you make to your product. But as Henrik Kniberg mentions in his Product Owner movie, your Stakeholders are not going to be limited to one or two new ideas to put on the Product Backlog. Therefore, to keep focus, the Product Owner must say no to some of the new ideas to keep the Product Backlog lean and maintainable. As a counter reaction to this movement I often see Stakeholders keeping their own wish list with new features. As long as the Product Owner does not agree to put it on the Product Backlog these wish lists keep growing in size and number (typically each Stakeholder keeps his own separate list). It is the task for the Product Owner to keep all his Stakeholders happy and accept new requests once in a while.



Some organizations have evolved to a situation where there is a single 'Wish List' and by using a voting mechanism, they determine the next Items that should move to the Product Backlog. It might even be the case that some customers' vote weighs heavier than others.

In this way it is easier for the Product Owner to determine the Business Value, once the Item moves to the Product Backlog (the Business Value might be related to the number of votes and the importance/'weight' of the customer).

A number of tool vendors of Visual Management and ALM tools have understood the need for a creating voting functionality that can be used for this purpose. Some examples:

- Rally Idea Manager
- Trello card voting
- Jira, watching & voting

Tools

One of the values of the Agile Manifesto is **Individuals & Interactions** over **Processes & Tools**. Honoring this value I've tried to stay away as far as possible from mentioning tools in this paper. However, at some point in time you're going to have to answer the question which tool to use in order to make use of the best practices depicted.

From my experience, the best tools out there are brown papers, sticky notes, whiteboards, scotch tape, and scissors. The next phase in tools is almost always a spreadsheet in order to avoid a lot of manual work. And once you have reached the point that Product Backlogs are implemented by multiple teams and you want some uniformity in your organization it's a wise decision to start using digital ALM tools that make it easier to work with a Product Backlog.

The question of which tool fits best in your organization is not an easy one and differs in each situation.

Need help?

Hopefully this paper has helped you a little bit more on getting an idea of what to expect when you are implementing your Product Backlog. If you recognize the described Requirements and challenges and if you want help in finding the right practices and tools for your purpose, don't hesitate to contact me: r.eringa@prowareness.nl.