Table of Contents

How Do We Deal With (Product) Backlog Explosion?

> Or “aggressive product backlog triage”.
> Or “why should we set up an 'attic' state in out backlog?”

NOTE: The thinking below, while written from the perspective Product Backlog can be applied to all different types of Backlogs - Product, Program, Portfolio, etc.

In Scrum we know that requirements are tracked in the product backlog (actually it tracks more than that, but this subset of product backlog items is certainly there). If we are not careful, this quickly leads to the idea that the product backlog is where we do what we traditionally called “requirements management”. We end up with the idea that “if we think it is a good idea it should be on the backlog” which results in an ever increasing list of requirements. My view is that this is a problem as, as the list increases, we have an increasing number of backlog items that will never actually get implemented. But we still apply time to managing this list, working on improving how we track this list and so on. As Peter Drucker says

> “There is nothing so useless as doing more efficiently what should not be done at all.”

Think about the effort wasted in managing an extensive list of requirements:

- Repetitive Review: Every planning session, you’ll revisit the list—again and again—wasting time on items that likely won’t make the cut. It’s like having 600 emails in your inbox instead of five. Even if you don’t realize it, those 600 weigh on you. (Don’t believe me? Try cutting your inbox in half and see how much better it feels.)
- Low-Priority Items Stay Low: If something didn’t make the cut this time, chances are it won’t next time either. The business keeps moving, and new high-priority items are always arriving.
- Loss of Focus: A massive list makes it harder to keep everything in mind, reducing your ability to focus on what truly matters.
- Operational Overhead: Managing a long list creates unnecessary complexity. Every action—prioritizing, categorizing, or updating—takes longer. And since large lists need additional organization, you end up maintaining a system just to manage them.
- False Sense of Progress: Being “on the list” doesn’t mean something will actually get done. Telling customers or stakeholders “it’s on the list” may feel reassuring, but the reality is that only the top items have any real chance of progress. Ironically, many probably understand that “it’s on the list” really means “it won’t happen,” but we keep up the illusion anyway.
- Duplicate Ideas Go Unnoticed: We assume that when a new idea comes in, we’ll recognize if it’s already on the list. In reality, this almost never happens. Either we fail to spot the duplicate, or the evolving nature of business means that what seems like a match has changed just enough to be treated as a separate item. The result? Redundant work and bloated lists.
- Work in Progress and Throughput: From queuing theory (Little’s Law), we know that reducing the size of a queue increases throughput. A massive backlog slows down actual delivery.
- Lack of Clarity: As lists grow, so do the layers of labels, special fields, and categorization rules meant to manage them. Without this insider knowledge, newcomers misunderstand the data,

leading to confusion rather than clarity.

In short, a large list doesn't just waste time—it creates unnecessary friction, delays, and cognitive load. The more you reduce it, the more effective your work becomes. If you have items in the list that are not actually going to be worked, then you have a set of effort which produces no value. How much effort you have wasted will depend on what you have done. If you just wrote a user story, the effort is low. If you've defined Acceptance Criteria (or Conditions of Satisfaction), or had the team estimate the item, then there is more wasted effort. You won't be able to get rid of all this wasted effort (for example, you might need an estimate on something to decide whether an item is worth considering) but if you do this for each and every item in the backlog it adds up to a lot of non-value producing effort. The question you have to ask yourself is this overhead of the big list worth the result. My view is that in most cases it is not, and that a more aggressive approach to maintaining the product backlog makes sense.

How do you clean up the list you have?

In the first instance you will have a natural reluctance to delete "a perfectly good record" and so perhaps the best idea is to set up an "attic". The idea behind the attic is that it is a special classification of product backlog items where, in the normal course of work, you never see the stuff that's in the attic. You then go through your product backlog and decide which items need to be put up into the attic. You will also need to make a call as to how (or perhaps a disingenuously whether) you will tell interested stakeholders about the disposal of their items.

The criteria for defining candidates for the attic will vary based on circumstances but here is a set of starting ideas.

If you have a huge long list, you start by conceptually putting everything in the attic, and then only bring those into the product backlog on a case by case basis, perhaps even as you actually work an item. I know this sounds aggressive. It works because generally people are better at selecting what gets started than deciding (and enforcing) what stops. Make sure you establish a limit on the amount of work you have in progress (WIP limit) to encourage flow of value.

Often you are not able to do this, perhaps for political reasons, or because there is a contract driving work, or you don't feel like the organization has the discipline, etc. For these situations you need to understand how much you will likely be able to deliver over a period of time so you can set an upper limit on how much should be in the product backlog. For example if you are working a single team, then use the velocity (or throughput, whatever you are comfortable with) information over a time period of "the next couple of releases" to set an upper limit either in terms of number of work items in the backlog. So if you are working on stories at the team level, and the team velocity is 50 on 2 week interations (sprints) and you have a yearly release cycle, you could set the upper limit to "this release plus a little of the next release" (50 (velocity) x 24 (sprints per year) x 1.5 (releases) 😀 900 points. Now you have a target size of product backlog based on "what might actually be delivered". For multiple teams on a single product, apply the same thinking.

How do we get to 900 points? You can start by looking at some large groups of potential attic candidates by establishing a criteria:

- Have a look at your product road-map. Are there things in the product backlog that are not related to your road-map? If yes, then think about putting these into the attic.
- Put the bottom half of your (prioritized) backlog in the attic. Trust me, even without looking at your backlog, unless you are already doing aggressive triage on the backlog, there is a bunch of stuff that you should never revisit. My assumption is that these are already toward the bottom since most people work actively on the top of the backlog. Put them in the attic.
- Set an “age” criteria. For example, “if the item was created more than 2 years ago” or “last viewed more than 1.5 years ago” then put them in the attic. The thinking here is that if its been there for a long time and we haven't done anything about it then it is unlikely this will change in the future.
- Set a “completeness” criteria. For example, at the Team level it might be “must be in use voice format, have Acceptance Criteria (or Conditions of Satisfaction), have investment allocation defined, and have an estimate”. The thinking here is that if the backlog item is not a good item, then the team won't work on it any way. This has the side effect of getting your important product backlog items “complete” as part of the process.
- Set a “type” criteria. For example, “we will only work 20% of capacity on defects” and use this formation to weed out long lists of specific types. The thinking here is similar to setting the upper limit in the first place.
- Set a “priority” or “value” criteria. For example, “we will not work any P4 / low priority items reported internally”. The thinking here is that it is of little importance, we aren't going to do anything about it.
- Or combinations of the above, or other criteria you have.

The really brave among you will set up a few of these criteria and simply apply it. After all, if we put things into the attic, its easy to get it back down from attic if we need to. What you'll mostly find is that when something important “comes back” again (and trust me, important items will come back) and you go up into the attic to find the relevant record, the attic item will not be quite the same thing and so you'll end up creating a new item that reflects current understanding.

After applying this kind of thinking you will probably find you still have too many items in the product backlog. The low hanging fruit is gone, so now you have have to go through the items individually and decide on disposal:

- Go through each of the items on in your product backlog. Ask yourself how the item fits into the vision and goals you have established for the product (you do have release goals, right?). If it doesn't fit in perhaps it is a candidate for the attic.
- Rigorously prioritize the product backlog. Many people explain to me their product backlog is not prioritized. While there are a number of reasons for this, if your product backlog is not prioritized per what you think is important, it is still prioritized in some way, and you are missing an important tool to get the best out of your team. If the priority is really not important then it won't matter if we put a line the cumulative points in the backlog sum to 900 and kill anything lower than that. Most people would balk at this approach which implies there really is an priority. Get your backlog prioritized so you can make these kinds of calls.

Once you have cleaned up your backlog, you now need to set things up so that you don't re-create the problem you had. Again, your approach will vary, but the base approach will come from the work you have done so far. Now that you have an upper limit on the number of items in the product backlog you can establish rules for accepting new items. For example you could say “new items can only be added to the product backlog if an existing item of equivalent size is removed from the backlog.” This approach

can be applied to individual product backlog items, but also to groups or items in the case where we have a new “epic” requirement that is more important that items previously on the list. The approach will force a certain degree of “readiness” criteria for new items coming in. For example, it is hard to make this call if you don't have estimates on new items. Once again, when something is removed you will need to determine how you communication this to relevant stakeholders.

How you do all this and who is involved will depend on circumstances. The on-going triage after you have done the clean-up work will need to happen regularly otherwise you build up too many new items for consideration. When this happens I've seen teams simply ignore the build up leading straight back to the original problem, only now it will be harder to go through the initial clean up (“we tried this before and it didn't work …”). One approach is to do this kind of work during a product backlog grooming session but this is not the only way of achieving the result.

Once you have gone through all this, what will you have?

- A product backlog that actually has a chance of being delivered
- A simpler backlog of important items that by comparison are easier to track mentally
- A process for on-going triage aimed at keeping the product backlog healthy
- A set of product backlog items that are ready for use by the team and that has had good discussion with the team
- The potential for more realistic communication with stakeholders (its up to you whether you take this potential)
- Less wastage as we make calls early on what does not get done, and so do not apply further effort to it
- Increased clarity as there is less need for special understanding of the product backlog

To me this seems like a worthwhile set of benefits over the traditional approach. And, after a period of time, you can delete all the items in the attic. This will happen when you realize that, in fact, no one has visited any of the items in the attic for a long time (just like the attic at home).

BlogEntry, FAQ, ProductBacklog, Lean, Waste, Triage, Attic, Backlog