



**The Scrum Anti-Patterns Guide—
A Hands-on Manual from the Trenches
by Stefan Wolpers**

Version: 1.6 2017-12-05

The Scrum Anti-Patterns Guide



Inhaltsverzeichnis

PREFACE	4
SCRUM CEREMONY ANTI-PATTERNS	5
17 STAND-UP ANTI-PATTERNS	5
THE DAILY SCRUM	5
STAND-UP ANTI-PATTERNS – FROM DYSFUNCTIONAL SCRUM TEAMS TO ORGANIZATIONAL FAILURES	5
CONCLUSION	7
28 PRODUCT BACKLOG AND REFINEMENT ANTI-PATTERNS	8
THE PRODUCT BACKLOG	8
THE PRODUCT BACKLOG REFINEMENT ACCORDING TO THE SCRUM GUIDE	8
A TYPICAL PRODUCT BACKLOG REFINEMENT PROCESS	8
COMMON PRODUCT BACKLOG (REFINEMENT) ANTI-PATTERNS	9
CONCLUSION:	13
19 SPRINT PLANNING ANTI-PATTERNS	14
THE SPRINT PLANNING	14
THE PURPOSE OF THE SPRINT PLANNING	14
SPRINT PLANNING ANTI-PATTERNS	15
CONCLUSION	18
27 SPRINT ANTI-PATTERNS	19
THE SPRINT	19
SPRINT ANTI-PATTERNS	19
CONCLUSION	25
14 SPRINT REVIEW ANTI-PATTERNS	26
THE SPRINT REVIEW	26
THE PURPOSE OF SCRUM'S SPRINT REVIEW	26
SPRINT REVIEW ANTI-PATTERNS	27
CONCLUSION	29
21 SPRINT RETROSPECTIVE ANTI-PATTERNS IMPEDING SCRUM TEAMS	30
INTRODUCTION	30
SPRINT RETROSPECTIVE ANTI-PATTERNS	30
CONCLUSION	34
SCRUM ROLES	35
PRODUCT OWNER ANTI-PATTERNS	35
PRODUCT BACKLOG AND REFINEMENT	35
SPRINT PLANNING ANTI-PATTERNS	37
SPRINT ANTI-PATTERNS	38
STAND-UP	39
SPRINT REVIEW ANTI-PATTERNS	39
CONCLUSION:	40
SCRUM MASTER ANTI-PATTERNS	41
INTRODUCTION	41
THE SCRUM MASTER ACCORDING TO THE SCRUM GUIDE	41
POSSIBLE REASONS WHY SCRUM MASTERS LEAVE THE AGILE PATH	42

The Scrum Anti-Patterns Guide



THE SCRUM MASTER AS AGILE MANAGER	43
SCRUM MASTER ANTI-PATTERNS BY SCRUM CEREMONY	45
THE CONCLUSION	48
SCRUM MASTER ANTI-PATTERNS DERIVED FROM JOB ADS	49
ANALYZING A JOB ADVERTISEMENT FOR A SCRUM MASTER OR AGILE COACH POSITION	49
SCRUM MASTER ANTI-PATTERNS FROM JOB ADS IN 22 EXAMPLES	50
CONCLUSION—SCRUM MASTER ANTI-PATTERNS FROM JOB ADS	53
<u>ABOUT THE AUTHOR</u>	54

The Scrum Anti-Patterns Guide



Preface

Welcome to my hands-on guide on scrum anti-patterns, detailing over 160 anti-patterns that you might observe in practice.

Please note that I will not be able to automatically provide you with new versions of this ebook if you unsubscribe from the *Food for Agile Thought* newsletter. In doing so, you also delete your email address from the list of readers of this ebook.

Thank you for your understanding!

Best,
Stefan



Scrum Ceremony Anti-Patterns

17 Stand-up Anti-Patterns

The Daily Scrum

The daily stand-up is the ceremony with the highest anti-pattern density among all scrum ceremonies. Learn more about the stand-up anti-patterns that threaten to derail your agile transition.

Stand-up Anti-Patterns – From Dysfunctional Scrum Teams to Organizational Failures

Typically, a good scrum team needs about five to ten minutes for a stand-up. Given this short period, it is interesting to observe that the daily stand-up is the scrum ceremony with the highest potential anti-pattern density. The anti-patterns range from behaviors driven by dysfunctional teams to apparent failures at an organizational level.

My favorite stand-up anti-patterns are as follows:

1. **No routine:** The stand-up does not happen at the same time and the same place every day. (While routine has the potential to ruin every retrospective, it is helpful in the context of stand-ups. Think of it like a spontaneous drill: don't put too much thought into the stand-up, just do it. Skipping stand-ups can turn out to be a slippery slope. And skipping may only be acceptable the day after the sprint planning. However, please keep in mind that every team member can veto skipping the stand-up.))
2. **Status report:** The stand-up is a status report meeting, and team members are waiting in line to "report" progress to the scrum master, the product owner, or maybe even a stakeholder.
3. **Ticket numbers only:** Updates are generic with little or no value to others. ("Yesterday, I worked on X-123. Today, I will work on X-129.")
4. **Problem solving:** Discussions are triggered to solve problems, instead of parking those so they can be addressed after the stand-up.
5. **Planning meeting:** The team hijacks the stand-up to discuss new requirements, to refine user stories, or to have a sort of (sprint) planning meeting.

The Scrum Anti-Patterns Guide



6. **No red dots:** A team member experiences difficulties in accomplishing an issue over several consecutive days, and nobody is offering help. (This a sign that that people either do not trust each other, or that the utilization of the team is maximized.)
7. **Monologs:** Team members violate the time-boxing, starting monologues. (60 to 90 seconds per team member should be more than enough time on air.)
8. **Statler and Waldorf:** A few team members are commenting every issue. (Usually, this is not just a waste of time, but also patronizing as well as annoying.)
9. **Disrespect I:** Other team members are talking while someone is sharing his or her progress with the team. (Similarly irritating is the need to use speak tokens among adults to avoid this behavior.)
10. **Assignments:** The product owner – or scrum master – assigns tasks directly to team members.
11. **Cluelessness:** Team members are not prepared for the stand-up. (“I was doing some stuff but I cannot remember what. Was important, though.”)
12. **Let’s start the shift:** The stand-up acts as a kind of artificial factory siren to start the next shift. (This is a common Taylorism artifact where trust in the team is missing.)
13. **Disrespect II:** Team members are late to the stand-up. (**Note:** if the time for the stand-up was not chosen by the team it otherwise indicates distrust on the management side.)
14. **Excessive feedback:** Team members criticize other team members right away sparking a discussion instead of taking their critique outside the stand-up.
15. **Overcrowded:** Stand-ups are ineffective due to the large number of active participants.
16. **Talkative chickens:** “Chickens” actively participate in the stand-up. (I think it is generally acceptable if stakeholder ask a question during the stand-up. However, they are otherwise supposed to merely listen in.)
17. **Anti-agile:** Line managers are attending stands-up to gather “performance data” on individual team members. (This behavior is defying the very purpose of self-organizing teams.)

Depending on the context, it could also be an anti-pattern if the product owner – or even another stakeholder – is introducing new tickets to the current sprint during the stand-up. This behavior may be acceptable for priority one bugs. (Although the team should be aware

The Scrum Anti-Patterns Guide



of those before the stand-up.) However, it is an unacceptable behavior – and thus an anti-pattern – for changing priorities on the fly in the middle of a sprint.

Lastly, some teams like to have stand-ups in Slack, particularly those that are not co-located. Again, depending on the context, this does not need to manifest an anti-pattern per se. I was even working with a co-located team that used Slack as their preferred way of having a stand-up. It worked.

Conclusion

A lot of agile practitioners tend to consider stand-ups to be a candidate for waste. However, from a scrum master or agile coach perspective stand-ups offer the highest yield of anti-patterns – given the effort is so small by comparison to other ceremonies.

The Scrum Anti-Patterns Guide



28 Product Backlog and Refinement Anti-Patterns

The Product Backlog

Scrum is a practical framework to build products, provided you identified in advance what to build. But even after a successful product discovery phase, you may struggle to make the right thing in the right way if your product backlog is not up to the job.

The following article points at the most common product backlog anti-patterns – including the product backlog refinement process – that limit your team’s success.

The Product Backlog Refinement According to the Scrum Guide

First of all, let’s have a look at the current issue of the Scrum Guide on the product backlog refinement:

“Product Backlog refinement is the act of adding detail, estimates, and order to items in the Product Backlog. This is an ongoing process in which the Product Owner and the Development Team collaborate on the details of Product Backlog items. During Product Backlog refinement, items are reviewed and revised. The Scrum Team decides how and when refinement is done. Refinement usually consumes no more than 10% of the capacity of the Development Team. However, Product Backlog items can be updated at any time by the Product Owner or at the Product Owner’s discretion.

Higher ordered Product Backlog items are usually clearer and more detailed than lower ordered ones. More precise estimates are made based on the greater clarity and increased detail; the lower the order, the less detail. Product Backlog items that will occupy the Development Team for the upcoming Sprint are refined so that any one item can reasonably be “Done” within the Sprint time-box. Product Backlog items that can be “Done” by the Development Team within one Sprint are deemed “Ready” for selection in a Sprint Planning. Product Backlog items usually acquire this degree of transparency through the above-described refining activities.

The Development Team is responsible for all estimates. The Product Owner may influence the Development Team by helping it understand and select trade-offs, but the people who will perform the work make the final estimate.”

Source & Copyright: [©2016 Scrum.Org](http://2016.Scrum.Org) and [ScrumInc](http://ScrumInc.com)¹.

A Typical Product Backlog Refinement Process

Based on the Scrum Guide, a typical process looks as follows:

¹ Offered for license under the Attribution Share-Alike license of Creative Commons, accessible at <http://creativecommons.org/licenses/by-sa/4.0/legalcode> and also described in summary form at <http://creativecommons.org/licenses/by-sa/4.0/>.

The Scrum Anti-Patterns Guide



1. The product owner prioritizes the backlog in advance, so it reflects the best possible use of the development team's resources:
 - a. The product owner creates and pre-populates the two upcoming sprints with user stories, using the team's project management software (or a spreadsheet or any other organizational tool the team applies).
 - b. The product owner maintains this pattern continuously.
 - c. The product owner also adds new user stories that he or she may have identified since the previous refinement session.
2. The product owner and the development team are jointly working on user stories:
 - a. The product owner provides the answer to the 'why' question (business purpose),
 - b. The team answers the 'how' question (technical implementation),
 - c. And both collaborate on the 'what' question: what scope is necessary to achieve the desired purpose?
3. The whole team agrees to time-box discussions. A typical time-box per user story would be around five minutes on average per cycle.
4. The product owner provides the acceptance criteria to user stories.
5. The development team defines what is required to consider a user story to be ready for becoming a sprint backlog item.
6. The product owner clarifies questions of the team or invites subject matter experts to refinement sessions who can answer the team's questions.
7. Consecutive refinement cycles last until each user story meets the definition of ready, or is no longer pursued.
8. Lastly, the team may estimate the available user stories that meet the "definition of ready" criteria. The product owner can now choose from those user stories to become part of an upcoming sprint.

Common Product Backlog (Refinement) Anti-Patterns

Despite being a relatively straightforward, the process of creating and refining a product backlog often suffers from various anti-patterns. I have identified four different categories:

General Product Backlog Anti-Patterns

- **Prioritization by proxy:** A single stakeholder or a committee of stakeholder prioritizes the product backlog. (The strength of Scrum is building on the strong position of the product owner. The product owner is the only persons to decide what tasks become product backlog items. Hence, the product owner also decides on the priority. Take away that empowerment, and Scrum turns into a pretty robust waterfall 2.0 process.)

The Scrum Anti-Patterns Guide



- **100% in advance:** The scrum team creates a product backlog covering the complete project or product upfront because the scope of the release is limited. (Question: how can you be sure to know today what to deliver in six months from now?)
- **Over-sized:** The product backlog contains more items than the scrum team can deliver within three to four sprints. (This way the product owner creates waste by hoarding issues that might never materialize.)
- **Outdated issues:** The product backlog contains items that haven't been touched for six to eight weeks or more. (That is typically the length of two to four sprints. If the product owner is hoarding backlog items, the risk emerges that older items become outdated, thus rendering previously invested work of the scrum team obsolete.)
- **Everything is estimated:** All user stories of the product backlog are detailed and estimated. (That is too much upfront work and bears the risk of misallocating the scrum team's time.)
- **Component-based items:** The product backlog items are sliced horizontally based on components instead of vertically based on end-to-end features. (This may be either caused by your organizational structure. Then move to cross-functional teams to improve the team's ability to deliver. Otherwise, the team – and the product owner – need a workshop on writing user stories.)
- **Missing acceptance criteria:** There are user stories in the product backlog without acceptance criteria. (It is not necessary to have acceptance criteria at the beginning the refinement cycle although they would make the task much easier. In the end, however, all user stories need to meet the definition of ready standard, and acceptance criteria are a part of that definition.)
- **No more than a title:** The product backlog contains user stories that comprise of little more than a title. (See above.)
- **Issues too detailed:** There are user stories with an extensive list of acceptance criteria. (This is the other extreme: the product owner covers each edge case without negotiating with the team. Typically, three to five acceptance criteria are more than sufficient.)
- **Neither themes nor epics:** The product backlog is not structured by themes or epics. (This makes it hard to align individual items with the “big picture” of the organization. The product backlog is not supposed to be an assortment of

The Scrum Anti-Patterns Guide



isolated tasks or a large to-do-list.)

- **No research:** The product backlog contains few to no spikes. (This often correlates with a team that is spending too much time on discussing prospective problems, instead of researching them with a spike as a part of an iterative user story creation process.)

Product Backlog Anti-Patterns at Portfolio and Product Roadmap Level

- **Roadmap?** The product backlog is not reflecting the roadmap. (The product backlog is supposed to be detailed only for the first two or three sprints. Beyond that point, the product backlog should rather focus on themes and epics from the product roadmap. If those are not available, the product backlog is likely to granular.)
- **Annual roadmaps:** The organization's portfolio plan, as well as the release plan or product roadmap, are created once a year in advance. (If the product backlog stays aligned to these plans, it introduces waterfall planning through the backdoor. Agile planning is always "continuous". At the portfolio level, the plan needs to be revised be least every three months.)
- **Roadmaps kept secret:** The portfolio planning and the release plan or product roadmap are not visible to everybody. (If you do not know where you are going any road will get you there. This information is crucial for any scrum team and needs to be available to everybody at any time.)
- **China in your hands:** The portfolio planning and the release plan or the product roadmap are not considered achievable and believable. (If this is reflected in the product backlog, working on user stories will probably be a waste.)

Product Backlog Anti-Patterns of the Product Owner

- **Storage for ideas:** The product owner is using the product backlog as a repository of ideas and requirements. (This practice is clogging the product backlog, may lead to a cognitive overload and makes alignment with the 'big picture' at portfolio management and roadmap planning level very tough.)
- **Part-time PO:** The product owner is not working daily on the product backlog. (The product backlog needs to represent at any given time the best use of the development team's resources. Updating it once a week before the next refinement session does not suffice to meet this requirement.)

The Scrum Anti-Patterns Guide



- **Copy & paste PO:** The product owner creates user stories by breaking down requirement documents received from stakeholders into smaller chunks. (That scenario helped to coin the nickname “ticket monkey” for the product owner. Remember: user story creation is a team exercise.)
- **Dominant PO:** The product owner creates user stories by providing not just the ‘why’ but also the ‘how’, and the ‘what’. (The team answers the ‘how’ question – the technical implementation –, and both the team and the PO collaborate on the ‘what’ question: what scope is necessary to achieve the desired purpose.)
- **INVEST?** The product owner is not applying the [INVEST principle by Bill Wake](#) to user stories.
- **Issues too detailed:** The product owner invests too much time upfront in user stories making them too detailed. (If a user story looks complete, the team members might not see the necessity to get involved in a further refinement. This way a “fat” user story reduces the engagement level of the team, compromising the creation of a shared understanding. By the way, this didn’t happen back in the days when we used index cards given their physical limitation.)
- **What team?** The product owner is not involving the entire scrum team in the refinement process and instead is relying on just the “lead engineer” (or any other member of the team independently of the others).
- **‘I know it all’ PO:** The product owner does not involve stakeholders or subject matter experts in the refinement process. (A product owner who believes to be either omniscient or a communication gateway is a risk to the Scrum team’s success.)

Product Backlog Anti-Patterns of the Development Team

- **Submissive team:** The development team submissively follows the demands of the product owner. (Challenging the product owner whether his or her selection of issues is the best use of the development team’s time is the noblest obligation of every team member: why shall we do this?)
- **What technical debt?** The development team is not demanding adequate resources to tackle technical debt and bugs. (The rule of thumb is that 25% of resources are allocated every sprint to fixings bugs and refactor the code base.)
- **No slack:** The development team is not demanding 20% slack time from the product owner. (This is overlapping with the sprint planning and the team’s commitment. However, it cannot be addressed early enough. If a team’s capacity

The Scrum Anti-Patterns Guide



is always utilized at 100 %, its performance will decrease over time. Everyone will focus on getting his or her tasks done. There will be less time to support teammates or to pair. Small issues will no longer be addressed immediately. And ultimately, the 'I am busy' attitude will reduce the generation of a shared understanding among all team members why they do what they are doing.)

Product Backlog Anti-Patterns of the Scrum Team

- **No time for refinement:** The team does not have enough refinement sessions, resulting in a low-quality backlog. (The Scrum Guide advises spending up to 10% of the Scrum team's time on the product backlog refinement. Which is a sound business decision: Nothing is more expensive than a feature that is not delivering any value.)
- **Too much refinement:** The team has too many refinement sessions, resulting in a too detailed backlog. (Too much refinement isn't healthy either.)
- **No DoR:** The scrum team has not created a 'definition of ready' that product backlog items need to match before becoming selectable for a sprint. (A simple checklist like the 'definition of ready' can significantly improve the scrum team's work. It will increase the quality of both the resulting user stories as well as the general way of working as a team.)

Conclusion:

Even in the case, you have successfully identified what to build next, your product backlog, as well as its refinement process, will likely provide room for improvement. Just take it to the team.

What product backlog and refinement anti-patterns are missing? Please share with us in the comments.

The Scrum Anti-Patterns Guide

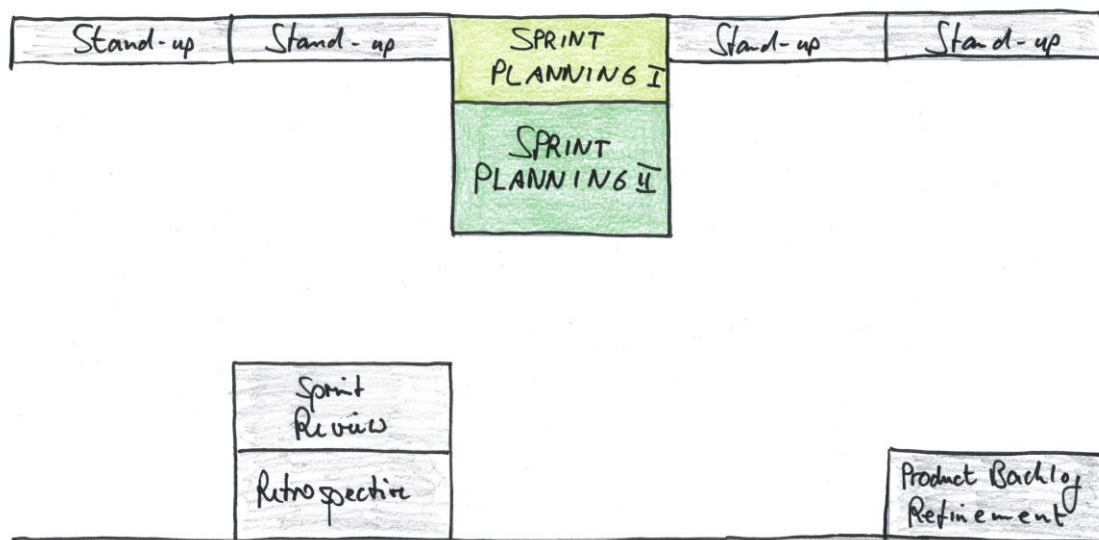


19 Sprint Planning Anti-Patterns

The Sprint Planning

Scrum's sprint planning is a simple ceremony. Invest upfront during the product backlog refinement, and you will keep it productive. Avoiding the following 19 sprint planning anti-patterns will help, too.

Scrum Sprint Planning Anti-Patterns



© Stefan Wolpers (2017) | Age-of-Product.com

The Purpose of the Sprint Planning

The purpose of Scrum's sprint planning is to align the development team and the product owner. Both need to agree on the shippable product increment of the next sprint. The idea is that the development team's commitment reflects the product owner's sprint goal. Also, the team needs to come up with a plan on how to accomplish its commitment. (Or forecast if you prefer that term.)

If the scrum team has been successfully using product backlog refinements in the past the sprint planning part 1 will be short. The development team and the product owner will

The Scrum Anti-Patterns Guide



adjust the discussed scope of the upcoming sprint to the available capacity. Maybe, someone from development team will not be available next sprint. So, one or two tasks will have to go back to the product backlog.

Or a valuable new task appeared overnight, and the product owner wants this task to become a part of the next sprint backlog. Consequently, some other user story needs to go back to the product backlog. A good team can handle that in five to ten minutes before moving on to sprint planning part 2. During sprint planning II the team breaks down the first set of sprint backlog items into subtasks.

Sprint Planning Anti-Patterns

There are three categories of sprint planning anti-patterns. They concern the development team, the product owner, and the scrum team.

Sprint Planning Anti-Patterns of the Development Team

- **Any absentees?** The team members do not determine their availability at the beginning of the sprint planning. (Good luck with making a commitment in this situation.)
- **Capacity?** The development team overestimates its capacity and takes on too many tasks. (The development team should instead take everything into account that might affect its ability to deliver. The list of those issues is long: public holidays, new team members, and those on vacation leave, team members quitting, team members on sick leave, corporate overhead, scrum ceremonies and other meetings to name a few.)
- **Ignoring technical debt:** The development team is not demanding adequate capacity to tackle technical debt and bugs during the sprint. (The rule of thumb is that 25% of resources are allocated every sprint to fix bugs and refactor the code base. If the product owner ignores this practice, and the development team accepts this violation the scrum team will find itself in a downward spiral. Its future product delivery capability will decrease.)
- **No slack:** The development team is not demanding 20% slack time from the product owner. (If a team's capacity is always over-utilized, its performance will decrease over time. This will particularly happen in an organization with a volatile daily business. As a consequence, everyone will focus on getting his or her tasks done. There will be less time to support teammates or to pair. The team will no longer address smaller or urgent issues promptly. Individual team members will become bottlenecks, which might seriously impede the flow within the team. Lastly, the 'I am busy' attitude will reduce the generation of a shared understanding among all team

The Scrum Anti-Patterns Guide



members. Overutilization will always push the individual team member to focus on his or her output. On the other side, slack time will allow the scrum team to act collaboratively and focus on the outcome.)

- **Planning too detailed:** During sprint planning II, the development team plans every single subtask of the upcoming sprint in advance. (Don't become too granular. Two-thirds of the sub-tasks are more than sufficient, the rest will follow naturally during the sprint. Doing too much planning upfront might result in waste.)
- **Too much estimating:** The development team estimates sub-tasks. (That looks like accounting for the sake of accounting to me. Don't waste your time on that.)
- **Too little planning:** The development team is skipping the sprint planning II altogether. (Skipping the sprint planning II is unfortunate, as it is also a good situation to talk about how to spread knowledge within the development team. For example, the team should think about who will be pairing with whom on what task. The sprint planning II is also a well-suited to consider how to reduce technical debt.)
- **Team leads?** The development team does not come up with a plan to deliver on its commitment collaboratively. Instead, a 'team lead' assigns tasks to individual team members. (I know that senior developers do not like the idea, but there is no 'team lead' in a scrum team. **Read More:** [Why Engineers Despise Agile](#)).

Sprint Planning Anti-Patterns of the Product Owner

- **What are we fighting for?** The product owner cannot provide a sprint goal, or the chosen sprint goal is flawed. (An original sprint goal answers the "What are we fighting for?" question. It is a negotiation between the product owner and the development team. It is focused and measurable, as sprint goal and team commitment go hand in hand. Lastly, the sprint goal is a useful calibration for the upcoming sprint.)
- **Calling Kanban 'Scrum':** The sprint backlog resembles a random assortment of tasks, and no sprint goal is defined. (If this is the natural way of finishing your sprint planning I, you probably have outlived the usefulness of Scrum as a product development framework. Depending on the maturity of your product, Kanban may prove to be a better solution. Otherwise, the randomness may signal a weak product owner who listens too much to stakeholders instead of prioritizing the product backlog appropriately.)
- **Unfinished business:** Unfinished user stories and other tasks from the last sprint spill over into the new sprint without any discussion. (There might be good reasons for that, for example, a task's value has not changed. It should not be an automatism,

The Scrum Anti-Patterns Guide



though, remember the [sunk cost fallacy](#).)

- **Last minute changes:** The product owner tries to squeeze in some last-minute user stories that do not meet the definition of ready. (Principally, it is the prerogative of the product owner to make such kind of changes to ensure that the development team is working only on the most valuable user stories at any given time. However, if the scrum team is otherwise practicing product backlog refinement sessions regularly, these occurrences should be a rare exception. If those happen frequently, it indicates that the product owner needs help with prioritization and team communication. Or the product owner needs support to say ‘no’ more often to stakeholders.)
- **Output focus:** The product owner pushes the development team to take on more tasks than it could realistically handle. Probably, the product owner is referring to former team metrics such as velocity to support his or her desire. (This would be an opportunity for the candidate to step in and address the issue.)

Sprint Planning Anti-Patterns of the Scrum Team

- **Irregular sprint lengths:** The scrum team has variable sprint cadences. For example, tasks are not sized to fit into the regular sprint length. Instead, the sprint length is adapted to the size of the tasks. (It is quite common to extend the sprint length at the end of the year when most of the team member are on holiday. However, there is no reason to deviate from the regular cadence during the rest of the year. Instead of changing the sprint length, the scrum team should invest more effort into sizing epics and user stories in the right way.)
- **Over-commitment:** The scrum team regularly takes on way too many tasks and moves unfinished work simply to the next sprint. (If two or three items spill over to the next sprint, so be it. If regularly 30-40 percent of the original commitment is not delivered during the sprint the scrum team may have created a kind of ‘time-boxed Kanban.’ Maybe, this is the right moment to ask the scrum team whether moving to Kanban might be an alternative.)
- **Stage-gate by DoR:** The definition of ready is handled in a dogmatic way thus creating a stage-gate-like approval process. (That is an interesting topic for a discussion among the team members. For example, should a valuable user story be postponed to another sprint just because the front end designs will not be available for another two working days? My suggestion: take it to the team. If they agree with the circumstances and accept the user story into the sprint — that is fine. **Read More:** [The Dangers of a Definition of Ready](#).)
- **Ignoring the DoR:** The development team is not rejecting user stories that do not meet the definition of ready. (This is the opposite side of being dogmatic about the

The Scrum Anti-Patterns Guide



application of DoR: not-ready user stories that will cause unnecessary disruptions during the sprint are allowed into it. Laissez-faire does not help either.)

- **Forecast imposed:** The sprint commitment is not a team-based decision. Or it is not free from outside influence. (There are several anti-patterns here. For example, an assertive product owner dominates the development team by defining its scope of the commitment. Or a stakeholder points at the team's previous velocity demanding to take on more user stories. ("We need to fill our free capacity.") Or the 'tech lead' of the development team is making a commitment on behalf of the team. Whatever the reason is, the candidate should address the underlying issues.)
- **Planning ignored:** The development team is not participating collectively in the sprint planning. Instead, two team members, for example, the tech and UX leads, represent the team. (As far as the idea of one or two 'leading' teammates in a scrum team is concerned, there are none, see above. And unless you are using LeSS – no pun intended – where teams are represented in the overall sprint planning, the whole scrum team needs to participate. It is a team effort, and everyone voice hence needs to be heard.)

Conclusion

Scrum's sprint planning is a simple ceremony. Invest upfront during the product backlog refinement, and you will keep it productive. Most of the beforementioned sprint planning anti-patterns are simple to fix. Just take it to the team.

The Scrum Anti-Patterns Guide

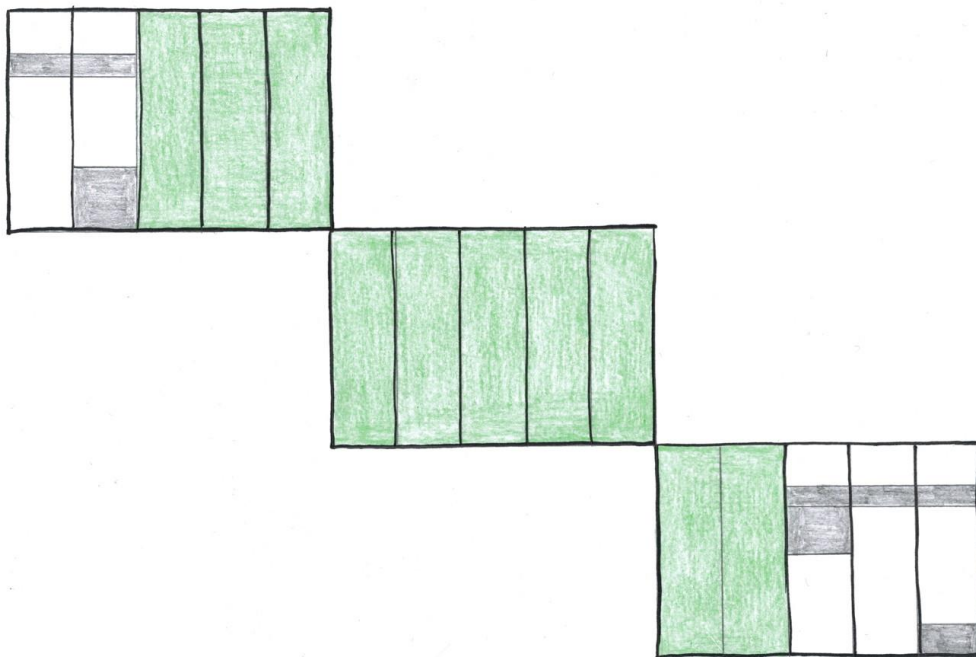


27 Sprint Anti-Patterns

The Sprint

The sprint is neither an official scrum ceremony nor an artifact. Obviously, it is not a role either. It is merely a time-box. Still, there are plenty of sprint anti-patterns to make your life as a scrum team harder than necessary.

27 Sprint Anti-Patterns Holding Scrum Teams Back



© Stefan Wolpers (2017) | Age-of-Product.com

Sprint Anti-Patterns

This list of notorious sprint anti-patterns applies to the development team, the product owner, the scrum master, the scrum team, as well as stakeholders and the IT management:

Sprint Anti-patterns of the Product Owner

- **Absent PO:** The product owner is absent most of the sprint and is not available to answer questions of the development team. (That creates a micro-waterfall)

The Scrum Anti-Patterns Guide



approach for the duration of the sprint.)

- **PO clinging to tasks:** The product owner cannot let go product backlog items once they become sprint backlog items. For example, the product owner increases the scope of a user story. Or, he or she changes acceptance criteria once the team accepted the issue into the sprint backlog. (There is a clear line: before a product backlog item turns into a sprint backlog item the product owner is responsible. However, once it moves from one backlog to the other, the development team becomes responsible. If changes become acute during the sprint the team will collaboratively decide on how to handle them.)
- **Inflexible PO:** The product owner is not flexible to adjust acceptance criteria. (If the work on a task reveals that the agreed upon acceptance criteria are no longer achievable or waste, the scrum team needs to adapt to the new reality. Blindly following the original plan violates a core scrum principle.)
- **Delaying PO:** The product owner does not accept sprint backlog items once those are finished. Instead, he or she waits until the end of the sprint. (In the spirit of continuous integration, the product owner should immediately check tasks that meet the acceptance criteria. Otherwise, the product owner will create an artificial queue which will increase the cycle-time. This habit puts also reaching the sprint goal at risk.)
- **Misuse of sprint cancellation:** The product owner cancels sprints to impose his or her will onto the team. (It is the prerogative of the product owner to cancel sprints. However, the product owner should not do this lightly without a serious cause. The product owner should also never abort a sprint without consulting the development team first. Probably, the team has an idea how to save the sprint. Lastly, misusing the cancelation privilege also indicates a serious team collaboration issue.)
- **No sprint cancellation:** The product owner does not cancel a sprint whose sprint goal can no longer be achieved. (If the product owner identified a unifying sprint goal, for example, integrating a new payment method, and the management then abandons that payment method mid-sprint, continuing working on the sprint goal would be waste. In this case, the product owner should cancel the sprint.)

Sprint Anti-patterns of the Development Team

- **No WiP limit:** There is no work in progress limit. (The purpose of the sprint is to deliver a potentially shippable product increment that provides value to either the customers or the organization. This goal requires getting something done by the end of the sprint. The flow theory suggests that the productivity of a team improves with a work-in-progress (WiP) limit. The WiP limit defines the largest number of tasks a

The Scrum Anti-Patterns Guide



team can work on at the same time. Exceeding this WiP number results in creating extra queues that reduce the throughput of the team. The cycle time which is the period between starting and finishing a ticket measures this effect.)

- **Cherry-picking:** The team cherry-picks work. (This effect often overlays with the missing WiP issue. Human beings are motivated by short-term gratifications. It just feels good to solve yet another puzzle from the board, here: coding a new task. By comparison to this [dopamine fix](#), checking how someone else solved another problem during code review is less rewarding. Hence you often notice tickets queuing in the code-review-column, for example.)
- **Board out-of-date:** The team does not update tickets on the board in time to reflect the current statuses. (The board, no matter if it is a physical or digital board, is not only vital for coordinating a team's work. It is also an integral part of the communication of the scrum team with its stakeholders. A board that is not up-to-date will impact the trust the stakeholders have in the scrum team. Deteriorating trust may then cause counter-measures on the side of the stakeholders. The (management) pendulum may swing back toward traditional methods as a consequence. The road back to PRINCE II is paved with abandoned boards.)
- **Side-gigs:** The team is working on issues that are not visible on the board. (While sloppiness is excusable, siphoning off resources, and by-passing the product owner – who is accountable of the scrum team's return on investment – is unacceptable. This behavior also signals a massive conflict within the “team.” Given this display of distrust—why didn't the engineers address this seemingly important issue during the sprint planning or before—the team is probably rather a group anyway.)
- **Gold-plating:** The team increases the scope of the sprint by adding unnecessary work to sprint backlog items. (This effect is often referred to as scope-stretching or gold-plating. The development team ignores the original scope agreement with the product owner. For whatever reason, the team enlarges the task without prior consulting of the product owner. This ignorance may result in a questionable allocation of resources. However, there is a simple solution: the developers and the product owner need to talk more often with each other. If the product owner is not yet co-located with the development team now would be a good moment to reconsider.)

Sprint Anti-patterns of the Scrum Master

- **Flow disruption:** The scrum master allows stakeholders to disrupt the flow of the development team during the sprint. (There are several possibilities how stakeholders can interrupt the flow of the team during a sprint. For example:

The Scrum Anti-Patterns Guide



- The scrum master has a laissez-faire policy as far as access to the development team is concerned.
 - The scrum master does not object that the management invites engineers to random meetings as subject matter experts.
 - Lastly, the scrum master allows that either stakeholders or managers turn the daily scrum into a reporting session.
Any of these behaviors will impede the team's productivity. It is the scrum master's obligation to prevent them from manifesting themselves.)
- **Lack of support:** The scrum master does not support team members that need help with a task. (Often, development teams create tasks an engineer can finish within a day. However, if someone struggles with such a task for more than two days without voicing that he or she needs support, the scrum master should address the issue. By the way, this is also the reason for marking tasks on a physical board with red dots each day if they do not move to the next column.)
 - **Micro-management:** The scrum master does not prevent the product owner—or anyone else—from assigning tasks to engineers. (The development team organizes itself without external intervention. And the scrum master is the shield of the team in that respect.)
 - **#NoRetro:** The scrum master does not gather data during the sprint that supports the team in the upcoming retrospective. (This is self-explanatory.)

Note: I do not believe that it is the task of the scrum master to move tickets on a board. The team members should do this during the stand-up. It is also not the responsibility of the scrum master to update an online board so that it reflects the statuses of a corresponding physical board. Lastly, if the team considers a burn-down chart helpful, the team members should also update the chart after the stand-up.

Sprint Anti-patterns of the Scrum Team

- **The maverick & the sprint backlog:** Someone adds an item to the sprint backlog without consulting the team first. (The fixed scope of a sprint backlog—in the sense of workload—is at the core of enabling the team to make a commitment or forecast. The scope is hence per se untouchable during the sprint. Changes of the composition of sprint backlog are possible, for example, when a critical bug pops up after a sprint's start. However, adding such an issue to the sprint backlog requires compensation. Another task of a similar size needs to go back to the product backlog. All these exceptions have in common that the scrum team decides collectively on them. No single teammate can add or remove an item to or from the sprint backlog.)

The Scrum Anti-Patterns Guide



- **Hardening sprint:** The scrum team decides to have a hardening or clean-up sprint. (That is a simple one: there is no such thing as a hardening sprint in scrum. The goal of the sprint is the delivery of a valuable potentially shippable product increment. Often, scrum teams agree in advance on a standard of what “done” means – also known as DoD or definition of done –. Declaring buggy tasks “done” thus violates core principles of the team’s way of collaboration. Hardening sprints are commonly a sign of a low grade of adoption of agile principles by the team or the organization. This is probably because the team is not yet cross-functional. Or quality assurance is still handled by a functional, non-agile silo with the product delivery organization.)
- **Delivering Y instead of X:** The product owner believes in getting X. The development team is working on Y. (This is not merely a result of an inferior product backlog refinement. This anti-pattern indicates that the team failed to create a shared understanding. There are plenty of reasons for this to happen, for example:
 - The product owner and the development team members are not talking enough during the sprint. (The product owner is too busy to answer questions from the team or attend the daily scrum. Or, the team is not co-located, etc.)
 - No development team member has ever participated in user tests. There is a lack of understanding the users’ problems among the engineers. (This is the reason why engineers should also interview users regularly.)
 - The product owner presented a too granular user story, and no one from the development team cared enough to have a thorough look. The user story seemed ready.
 - Probably, the user story was missing acceptance criteria altogether, or existing acceptance criteria missed the problem. No matter the reason, the team should address the issue during the next retrospective.)
- **No sense of urgency:** There is no potentially shippable product increment at the end of the sprint. There was no reason to cancel the sprint either. It was just an ordinary sprint. (This is a sign that the scrum team lacks the sense of urgency to deliver at the end of the sprint. If it is acceptable to fail on delivering value at the end of the sprint, the whole idea behind scrum is questioned. Remember, a scrum team trades a commitment or forecast for inclusion in decision-making, autonomy, and self-organization. Creating a low-grade time-boxed Kanban and calling it “scrum” will not honor this deal. Therefore, it is in the best interest of the scrum team to make each sprint’s outcome releasable even if the release will not materialize.)
- **New kid on the block:** The scrum team welcomed a new team member during the sprint. They also forgot to address the issue during sprint planning thus ending up overcommitted. (While it is acceptable to welcome new teammates during a sprint, the team needs to account for the resulting onboarding effort during the sprint planning and adjust its capacity. The new team member should not be a surprise. However, if the newbie turns out to be a surprise it is rather an organizational anti-

The Scrum Anti-Patterns Guide



pattern.)

- **Variable sprint length:** The scrum team extends the sprint length by a few days to meet the sprint goal. (This is just another way of cooking the agile books to match a goal or a metric. This is not agile; it is just inconsequential. Stop lying to yourself, and address the underlying issues why the team outcome does not meet the sprint goal. **Note:** I would not consider a deviating sprint length during the holiday season at the end of the year to be an anti-pattern.)

Sprint Review Anti-patterns of the IT Management

- **All hands to the pumps w/o scrum:** The management temporarily abandons scrum in an all-hands-to-deck situation. (This is a classic manifestation of disbelief in agile practices, fed by command & control thinking. Most likely, canceling sprints and gathering the scrum teams would also solve the issue at hands.)
- **Reassigning team members:** The management regularly assigns team members of one scrum team to another team. (Scrum can only live up to its potential if the team members can build trust among each other. The longevity of teams is hence essential. Moving people between teams, on the contrary, reflects a project-minded idea of management. It also ignores the preferred team building practice that scrum teams should find themselves. All members need to be voluntarily on a team. Scrum does rarely work if team members are pressed into service.

Note: It is not an anti-pattern, though, if two teams decide to exchange teammates temporarily. It is an established practice that specialists spread knowledge this way or mentor other colleagues.)

- **Special forces:** A manager assigns tasks directly to engineers, thus bypassing the product owner. (This behavior does not only violate core scrum principles. It also indicates that the manager cannot let go command and control practices. He or she continues to micromanage subordinates although a scrum team could accomplish the task in a self-organized manner. This behavior demonstrates a level of ignorance that may require support from a higher management level to deal with.)

Sprint Review Anti-patterns of Stakeholders

- **Pitching developers:** The stakeholders try to sneak in small tasks by pitching them directly to developers. (Nice try #1.)
- **Everything's a bug:** The stakeholders try to speed up delivery by relabeling their tasks as 'serious bugs'. (Nice try #2. A special case is an "express lane" for bug fixes and other urgent issues. Every stakeholder will try and make his or her tasks eligible

The Scrum Anti-Patterns Guide



for that express lane.)

- **Disrupting the flow:** The stakeholders disrupt the flow of the scrum team. (See above, scrum master section.)

Conclusion

Although the sprint itself is just a time-box there are plenty of sprint anti-patterns to observe. A lot of them are easy to fix by the scrum team. Other sprint anti-patterns, however, point at organizational issues that probably will require more than a retrospective to change.



14 Sprint Review Anti-Patterns

The Sprint Review

Are we still on the right track? Answering this question in a collaborative effort of the scrum team as well as internal (and external) stakeholders is the purpose of the sprint review. Given its importance, it is worthwhile to tackle the most common sprint review anti-patterns.

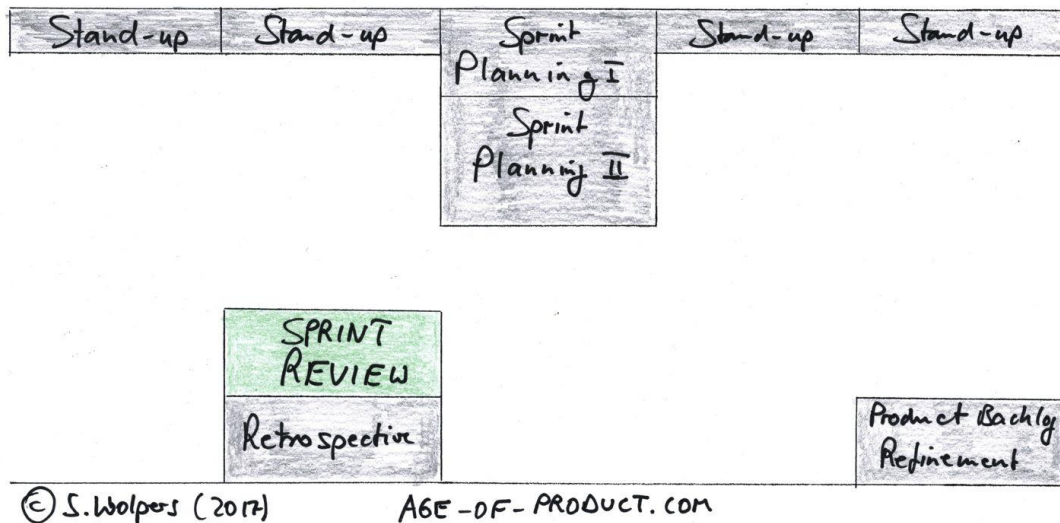
The Purpose of Scrum's Sprint Review

The sprint review is about Scrum's core principle: inspect & adapt. The development team, the product owner, and the stakeholders need to figure out whether they are still on track delivering value to customers. It is the best moment to create or reaffirm the shared understanding among all participants whether the product backlog is still reflecting the best use of the scrum team's resources. It is also because of this context that calling the sprint review a "sprint demo" does not match its importance for the effectiveness of the scrum team.

The sprint review is thus an excellent opportunity to talk about the general progress of the product. The sprint review's importance is also the reason to address sprint review anti-patterns as soon as possible.



14 Scrum Sprint Review Anti-Patterns



Sprint Review Anti-Patterns

Typically, you can observe the following sprint review anti-patterns.

Sprint Review Anti-patterns of the Product Owner

- **Selfish PO:** The product owner presents “his or her” accomplishments to the stakeholders. (Remember the old saying: There is no “I” in “team”?)
- **Delayed sprint acceptance:** The product owner uses the sprint review to accept user stories. (This should be decoupled from the sprint review. The product owner should accept user stories the moment they meet the acceptance criteria.)
- **Unapproachable PO:** The product owner is not accepting feedback from the stakeholders. (Such a behavior violates the prime purpose of the sprint review ceremony.)

The Scrum Anti-Patterns Guide



Sprint Review Anti-patterns of the Development Team

- **Death by PowerPoint:** Participants are bored to death by PowerPoint. (The foundation of a successful sprint review is “show, don’t tell,” or even better: let the stakeholders drive the ceremony.)
- **Same faces again:** It is always the same representatives from the development team who participate. (Unless the organization works with several teams based on LeSS (large scale Scrum), this is not a good sign. The challenge is that you cannot enforce the team’s participation either, though. Instead, make it interesting enough that everyone wants to participate. **Note:** If the team does not attend religiously in full strength in each sprint review it is not an anti-pattern per se. However, there should be some rotation among participating team members.)
- **Side gigs:** The development team was working on issues outside the sprint scope. The product owner learns about those for the first time during the sprint review.
- **Cheating:** The development team demos items that are still buggy. (There is a good reason to show unfinished work on some occasions. Buggy work on the other side violates the DoD at an unacceptable level.)

Sprint Review Anti-patterns of the Scrum Team

- **Following a plan:** The scrum team does not use the sprint review to discuss the current state of the product or project with the stakeholders. (Again, getting feedback is the purpose of the exercise. A we-know-what-to-build attitude is bordering on hubris. **Read More:** [Sprint Review, a Feedback Gathering Event: 17 Questions and 8 Techniques.](#))
- **Sprint accounting:** Every task accomplished is demoed, and stakeholders do not take it enthusiastically. (Tell a compelling story at the beginning of the review to engage the stakeholders. Leave out those user stories that are not relevant to the story. Do not bore stakeholders by including everything that was accomplished. We are not accountants.)

Sprint Review Anti-patterns of the Stakeholders

- **Scrum à la stage-gate:** The sprint review is a kind of stage-gate approval process where stakeholders sign off features. (This anti-pattern is typical for organizations that use an agile-waterfall hybrid. Otherwise, it is the prerogative or the product owner to decide what to ship when.)

The Scrum Anti-Patterns Guide



- **No stakeholders:** Stakeholders do not attend the sprint review. (There are several reasons why stakeholders do not go to the sprint review: they do not see any value in the ceremony. It is conflicting with another important meeting. They do not understand the importance of the sprint review event. No sponsor is participating in the sprint review, for example, from the C-level. To my experience, you need to “sell” the ceremony within the organization.)
- **No customers:** External stakeholders – also known as customers – do not attend the sprint review. (Break out of your organization’s filter bubble, and invite some paying users of your product.)
- **Starting over again:** There is no continuity in the attendance of stakeholders. (Longevity is not just a team issue, but also applies to stakeholders. If they change too often, for example, because of a rotation scheme, how can they provide in-depth feedback? If this pattern appears the team needs to improve how stakeholders understand the sprint review.)
- **Passive stakeholders:** The stakeholders are passive and unengaged. (That is simple to fix. Let the stakeholders drive the sprint review and put them at the helm. Or organize the sprint review as a science fair with several booths.)

Conclusion

Scrum’s sprint review is a simple yet meaningful ceremony. It answers the question whether the scrum team is still on track delivering value to the customers and the organization. Avoiding the sprint review’s anti-patterns can significantly improve a team’s effectiveness.

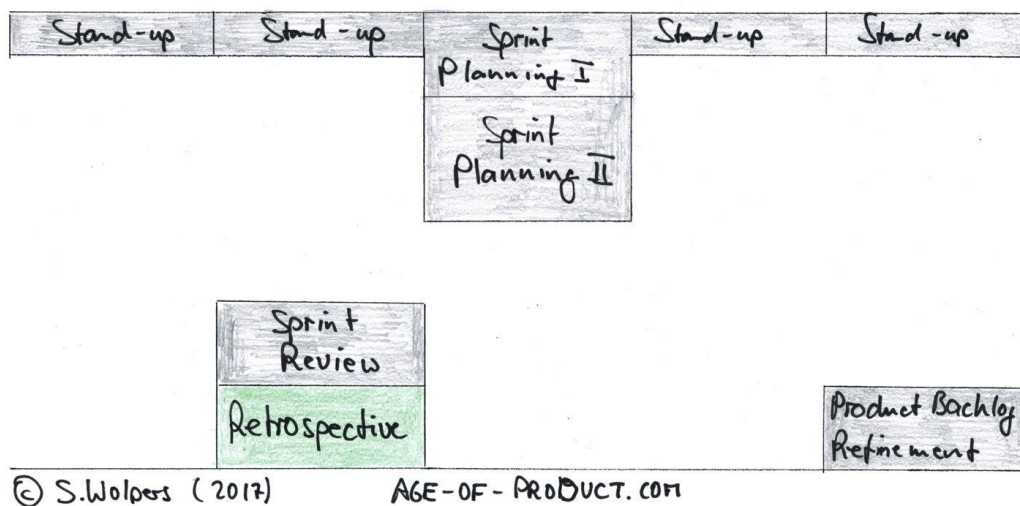


21 Sprint Retrospective Anti-Patterns Impeding Scrum Teams

Introduction

What ceremony could better embody scrum's 'inspect and adapt' mantra than the sprint retrospective? I assume all agile peers agree that even the simplest retrospective—if only held regularly—is far more useful than having a fancy one once in a while, or in the worst case having none at all. And there is always room for improvement. Learn more about 21 common sprint retrospective anti-patterns.

21 Sprint Retrospective Anti-Patterns



Sprint Retrospective Anti-Patterns

No matter the frequency of your retrospectives you should always watch out for the following sprint retrospective anti-patterns from the scrum team, the development team, the scrum master, as well as the organization:

Sprint Retrospective Anti-Patterns of the Scrum Team

The Scrum Anti-Patterns Guide



- **#NoRetro:** There is no retrospective as the team believes there is nothing to improve. (There is no such thing as an agile Nirwana where everything is just perfect. As people say: becoming agile is a journey, not a destination, and there is always something to improve.)
- **Dispensable buffer:** The team cancels retrospectives if more time is needed to accomplish the sprint commitment/forecast. (The retrospective as a sprint emergency reserve is a common sign of cargo cult agile. I believe, it is even a worse anti-pattern than not having a retrospective because there is presumably nothing to improve. That is just an all too human fallacy bordering on hubris. However, randomly canceling the retrospective to achieve a sprint goal is a clear sign that the team does not understand basic agile principles, such as continuous improvement. If the scrum team repeatedly does not meet a sprint goal, it should inspect what is going on here. Guess which scrum ceremony is designed for that purpose?)
- **Rushed retrospective:** The team is in a hurry and allocates much less than the necessary 60 to 90 minutes for a retrospective. (That is a slippery slope and will probably end up with a ritualized ceremony of little value. Most team members will likely regard it as a waste sooner or later. Do it right by allocating whatever time is needed or consider stop having retrospectives. And while you are at it, why don't you abandon scrum altogether?)
- **Someone sings:** Someone from the participants provides information on the retrospective to an outsider. (For retrospectives, the [Vegas rule](#) applies: what is said in the room stays in the room. There is no exception from this rule.)
- **Extensive whining:** The team uses the retrospective primarily to complain about the situation and assumes the victim's role. (Change requires reflection, and occasionally it is a good exercise to let off steam. However, not moving on once you have identified critical issues and trying to change them defies the purpose of the retrospective. Limiting the number of stickies to 2-3 per participant may help to change this attitude. You may also consider balancing good and negative feedback by handing out an equal number of green and red stickies. Looks to be a bit too enforcing for my taste, though.)
- **UNSMART:** The team chooses to tackle UNSMART actions. (Bill Wake created the SMART acronym for reasonable action items: S – Specific, M – Measurable, A – Achievable, R – Relevant, T – Time-boxed. If the team picks UNSMART action items, though, it sets itself up for failure and may thus contribute to a bias that “agile” is not working. **Read More:** [INVEST in Good Stories, and SMART Tasks.](#))
- **#NoAccountability:** Action items were accepted before. However, no one was chosen to be responsible for the delivery. (If the “team” is supposed to fix X, probably everyone will rely on his or her teammates to handle it. Make someone

The Scrum Anti-Patterns Guide



accountable instead.)

- **What action?** The team does not check the status of the action items from the previous retrospectives. (The sibling of autonomy is accountability. If you are not following up on what you wanted to improve before why care about picking action items in the first place?)

Sprint Retrospective Anti-Patterns of the Development Team

Product owner non grata: The product owner is not welcome to the retrospective. (Some purists still believe that only the development team and the scrum master shall attend the team's retrospective. However, the Scrum Guide refers to the [scrum team, including the product owner](#). It does so for a good reason: the team wins together, and the team loses together. How is that supposed to work without the product owner?)

Sprint Retrospective Anti-Patterns of the Scrum Master

- **Waste of time:** The team does not collectively value the retrospective. (If some team members consider the retrospective to be of little or no value it is most often the retrospective itself that sucks. Is it the same procedure every time, ritualized, and boring? Have a meta-retrospective on the retrospective itself. Change the venue. Have a beer- or wine-driven retrospective. There are so many things a scrum master can do to make retrospectives great again and reduce the absence rate. And yes, to my experience introverts like to take part in retrospectives, too.)
- **Prisoners:** Some team members only participate because they are forced to join. (Don't pressure anyone to take part in a retrospective. Instead, make it worth their time. The drive to continuously improve as a team needs to be fueled by intrinsic motivation, neither by fear nor by order. Tip: Retromat's "[Why are you here?](#)" exercise is a good opener for a retrospective from time to time.)
- **Groundhog day:** The retrospective never changes in composition, venue, or length. (There is a tendency in this case that the team will revisit the same issues over and over again – it's groundhog day without the happy ending, though.)
- **Let's have it next sprint:** The team postpones the retrospective into the next sprint. (Beyond the 'inspect & adapt' task, the retrospective shall also serve as a moment of closure that resets everybody's mind so that the team can focus on the new sprint goal. That is the reason why we have the retrospective before the planning of the follow-up sprint. Postponing it into the next sprint may interrupt the flow of the team. It also delays tackling possible improvements by up to a sprint.)

The Scrum Anti-Patterns Guide



- **#NoDocumentation:** No one is taking minutes for later use. (A retrospective is a substantial investment and should be taken seriously. Taking notes and photos supports this process.)
- **No psychological safety:** The retrospective is an endless cycle of blame and finger pointing. (The team wins together, the team loses together. The blame game documents both the failure of the scrum master as the facilitator of the retrospective as well as the team's lack of maturity and communication skills.)
- **Bullying:** One or two team members are dominating the retrospective. (This communication behavior is often a sign of either a weak or uninterested scrum master. The retrospective needs to be a safe place where everyone—introverts included—can address issues and provide his or her feedback free from third party influence. If some of the team members are dominating the conversation, and probably even bullying or intimidating other teammates, the retrospective will fail to provide such a safe place. This failure will result in participants dropping out of the retrospective and render the results obsolete. It is the main responsibility of the scrum master to ensure that everyone will be heard and has an opportunity to voice his or her thoughts. By the way, equally distributed speaking time is according to Google also a sign of a high-performing team. **Read More:** [What Google Learned From Its Quest to Build the Perfect Team.](#))
- **Stakeholder alert:** Stakeholders participate in the retrospective. (There are plenty of scrum ceremonies that address the communication needs of stakeholder: the sprint review, the product backlog refinement, the daily scrums, not to mention opportunities of having a conversation at water coolers, over coffee, or during lunchtime. If that spectrum of possibilities still is not sufficient, feel free to have additional meetings. However, the retrospective is off-limits to stakeholders.)
- **Passivity:** The team members are present but are not participating. (There are plenty of reasons for such a behavior: they regard the retrospective a waste of time, it is an unsafe place, or the participants are bored to death by its predictiveness. Probably, the team members fear negative repercussions in the case of their absence, or you managed to hire a homogenous group of East-European introverts. In other words: there is no quick fix, and the scrum master needs to figure out what kind of retrospective works in his or her organization's context.)

Sprint Retrospective Anti-Patterns of the Organization

- **No suitable venue:** There is no adequate place available to run the retrospective. (The least appropriate place to have a retrospective is a meeting room with a rectangular table surrounded by chairs. And yet it is the most common venue to have a retrospective. Becoming agile requires space. If this space is not available, you

The Scrum Anti-Patterns Guide



should become creative and go somewhere else. If the weather is fine, grab your stickies and go outside. Or rent a suitable space somewhere else. If that is not working, for example, due to budget issues, remove at least the table so you can sit/stand in a circle. Just be creative. **Read More:** [Agile Workspace: The Undervalued Success Factor.](#))

- **Line managers present:** Line managers participate in retrospectives. (This is the worst anti-pattern I can think off. It turns the retrospective into an unsafe place. And who would expect that an unsafe place triggers an open discussion among the team members? Any line manager who insists on such a proceeding signals his or her lack of understanding of basic agile practices. Note: If you are small product delivery team at a start-up and your part-time scrum master (or product owner) also serves in a management function, retrospectives might be challenging. In this case, consider hiring an external scrum master to facilitate meaningful retrospectives.)
- **Let us see your minutes:** Someone from the organization—outside the team—requires access to the retrospective minutes. (This is almost as bad as line managers who want to participate in a retrospective. Of course, the access must be denied.)

Conclusion

There are many ways in which a retrospective can be a failure even if it looks favorable at first glance. The top three sprint retrospective anti-patterns from my perspective are: not making the retrospective a safe place, unequally distributed speaking time, and a ritualized format that never changes.

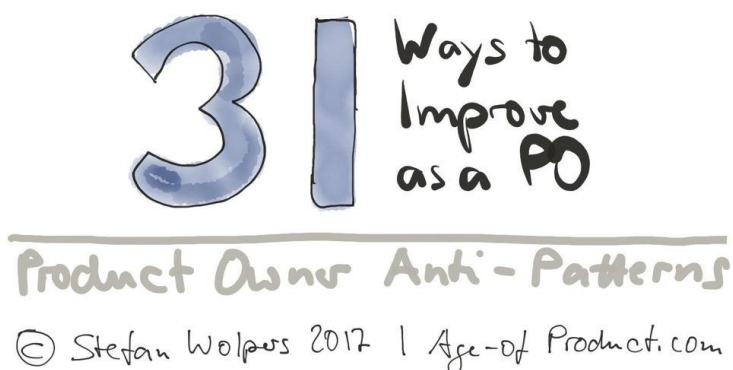


Scrum Roles

Product Owner Anti-Patterns

If you are working as a product owner, there is — very likely — room for improvement. I curated 30-plus strong list of the most common product owner anti-patterns to help you up your game.

If you like to improve on those you recognize why don't you ask the scrum master for support? The product owner anti-patterns list is a good starting point for a retrospective.



Product Backlog and Refinement

You can spot most of the product owner anti-patterns in the PO's backyard — the product backlog and its refinement:

The Scrum Anti-Patterns Guide



1. **Prioritization by proxy:** A single stakeholder or a committee of stakeholder prioritizes the product backlog. (The strength of Scrum is building on the strong position of the product owner. The product owner is the only persons to decide what tasks become product backlog items. Hence, the product owner also decides on the priority. Take away that empowerment, and Scrum turns into a pretty robust waterfall 2.0 process.)
2. **Over-sized:** The product backlog contains more items than the scrum team can deliver within three to four sprints. (This way the product owner creates waste by hoarding issues that might never materialize.)
3. **Outdated issues:** The product backlog contains items that haven't been touched for six to eight weeks or more. (That is typically the length of two to four sprints. If the product owner is hoarding backlog items, the risk emerges that older items become outdated, thus rendering previously invested work of the scrum team obsolete.)
4. **Missing acceptance criteria:** There are user stories in the product backlog without acceptance criteria. (It is not necessary to have acceptance criteria at the beginning the refinement cycle although they would make the task much easier. In the end, however, all user stories need to meet the definition of ready standard, and acceptance criteria are a part of that definition.)
5. **No more than a title:** The product backlog contains user stories that comprise of little more than a title. (See above.)
6. **Issues too detailed:** There are user stories with an extensive list of acceptance criteria. (This is the other extreme: the product owner covers each edge case without negotiating with the team. Typically, three to five acceptance criteria are more than sufficient.) **Talkative chickens:** The product owner — at least in my eyes — is more a “chicken” than a team member in a stand-up. Talking too much may hence be an issue. (The product owner who is also a part-time scrum master is a different scenario, though.)
7. **Storage for ideas:** The product owner is using the product backlog as a repository of ideas and requirements. (This practice is clogging the product backlog, may lead to a cognitive overload and makes alignment with the ‘big picture’ at portfolio management and roadmap planning level very tough.)
8. **Part-time PO:** The product owner is not working daily on the product backlog. (The product backlog needs to represent at any given time the best use of the development team's resources. Updating it once a week before the next refinement session does not suffice to meet this requirement.)



9. **Copy & paste PO:** The product owner creates user stories by breaking down requirement documents received from stakeholders into smaller chunks. (That scenario helped to coin the nickname “ticket monkey” for the product owner. Remember: user story creation is a team exercise.)
10. **Dominant PO:** The product owner creates user stories by providing not just the ‘why’ but also the ‘how’, and the ‘what’. (The team answers the ‘how’ question – the technical implementation –, and both the team and the PO collaborate on the ‘what’ question: what scope is necessary to achieve the desired purpose.)
11. **INVEST?** The product owner is not applying the [INVEST principle by Bill Wake](#) to user stories.
12. **Issues too detailed:** The product owner invests too much time upfront in user stories making them too detailed. (If a user story looks complete, the team members might not see the necessity to get involved in a further refinement. This way a “fat” user story reduces the engagement level of the team, compromising the creation of a shared understanding. By the way, this didn’t happen back in the days when we used index cards given their physical limitation.)
13. **What team?** The product owner is not involving the entire scrum team in the refinement process and instead is relying on just the “lead engineer” (or any other member of the team independently of the others).
14. **‘I know it all’ PO:** The product owner does not involve stakeholders or subject matter experts in the refinement process. (A product owner who believes to be either omniscient or a communication gateway is a risk to the Scrum team’s success.)

Sprint Planning Anti-Patterns

The number two area on my list of product owner anti-patterns is the sprint planning itself:

1. **What are we fighting for?** The product owner cannot provide a sprint goal, or the chosen sprint goal is flawed. (An original sprint goal answers the “What are we fighting for?” question. It is a negotiation between the product owner and the development team. It is focused and measurable, as sprint goal and team forecast go hand in hand. Lastly, the sprint goal is a useful calibration for the upcoming sprint.)
2. **Calling Kanban ‘Scrum’:** The sprint backlog resembles a random assortment of tasks, and no sprint goal is defined. (If this is the natural way of finishing your sprint planning I, you probably have outlived the usefulness of Scrum as a product development framework. Depending on the maturity of your product, Kanban may

The Scrum Anti-Patterns Guide



prove to be a better solution. Otherwise, the randomness may signal a weak product owner who listens too much to stakeholders instead of prioritizing the product backlog appropriately.)

3. **Unfinished business:** Unfinished user stories and other tasks from the last sprint spill over into the new sprint without any discussion. (There might be good reasons for that, for example, a task's value has not changed. It should not be an automatism, though, remember the [sunk cost fallacy](#).)
4. **Last minute changes:** The product owner tries to squeeze in some last-minute user stories that do not meet the definition of ready. (Principally, it is the prerogative of the product owner to make such kind of changes to ensure that the development team is working only on the most valuable user stories at any given time. However, if the scrum team is otherwise practicing product backlog refinement sessions regularly, these occurrences should be a rare exception. If those happen frequently, it indicates that the product owner needs help with prioritization and team communication. Or the product owner needs support to say 'no' more often to stakeholders.)
5. **Output focus:** The product owner pushes the development team to take on more tasks than it could realistically handle. Probably, the product owner is referring to former team metrics such as velocity to support his or her desire.

Sprint Anti-Patterns

Another area prone to product owner anti-patterns is the sprint itself:

1. **Absent PO:** The product owner is absent most of the sprint and is not available to answer questions of the development team. (That creates a micro-waterfall approach for the duration of the sprint.)
2. **PO clinging to tasks:** The product owner cannot let go product backlog items once they become sprint backlog items. For example, the product owner increases the scope of a user story. Or, he or she changes acceptance criteria once the team accepted the issue into the sprint backlog. (There is a clear line: before a product backlog item turns into a sprint backlog item the product owner is responsible. However, once it moves from one backlog to the other, the development team becomes responsible. If changes become acute during the sprint the team will collaboratively decide on how to handle them.)
3. **Inflexible PO:** The product owner is not flexible to adjust acceptance criteria. (If the work on a task reveals that the agreed upon acceptance criteria are no longer achievable or waste, the scrum team needs to adapt to the new reality. Blindly

The Scrum Anti-Patterns Guide



following the original plan violates a core scrum principle.)

4. **Delaying PO:** The product owner does not accept sprint backlog items once those are finished. Instead, he or she waits until the end of the sprint. (In the spirit of continuous integration, the product owner should immediately check tasks that meet the acceptance criteria. Otherwise, the product owner will create an artificial queue which will increase the cycle-time. This habit puts also reaching the sprint goal at risk.)
5. **Misuse of sprint cancellation:** The product owner cancels sprints to impose his or her will onto the team. (It is the prerogative of the product owner to cancel sprints. However, the product owner should not do this lightly without a serious cause. The product owner should also never abort a sprint without consulting the development team first. Probably, the team has an idea how to save the sprint. Lastly, misusing the cancellation privilege also indicates a serious team collaboration issue.)
6. **No sprint cancellation:** The product owner does not cancel a sprint whose sprint goal can no longer be achieved. (If the product owner identified a unifying sprint goal, for example, integrating a new payment method, and the management then abandons that payment method mid-sprint, continuing working on the sprint goal would be a waste. In this case, the product owner should cancel the sprint.)

Stand-up

By comparison to other Scrum ceremonies, the stand-up is remarkably resilient to product owner anti-patterns. There is only one serious:

1. **Planning meeting:** The product owner hijacks the stand-up to discuss new requirements, to refine user stories, or to have a sort of (sprint) planning meeting.
2. **A talkative (PO) chicken:** The product owner — at least in my eyes — is more a “chicken” than a team member in a stand-up. Talking too much may hence be an issue. (The product owner who is also a part-time scrum master is a different scenario, though.)

Sprint Review Anti-Patterns

Finally, there is the sprint review. Despite that it is an outstanding opportunity for the product owner to improve the collaboration with both stakeholders and the scrum team, some POs simply do not get the message:

1. **Selfish PO:** The product owner presents “his or her” accomplishments to the stakeholders. (Remember the old saying: There is no “I” in “team”?)

The Scrum Anti-Patterns Guide



2. **Delayed sprint acceptance:** The product owner uses the sprint review to accept user stories. (This should be decoupled from the sprint review. The product owner should accept user stories the moment they are meeting the acceptance criteria.)
3. **Unapproachable PO:** The product owner is not accepting feedback from the stakeholders. (Such a behavior violates the prime purpose of the sprint review ceremony.)

Conclusion:

Admittedly, the product owner role is the most challenging scrum role, and the higher the expectations are, the easier it is to fail them. Nevertheless, the concept of continuous improvement also applies to exercising the product owner role. The list of product owner anti-patterns above may be a starting point.

The Scrum Anti-Patterns Guide



Scrum Master Anti-Patterns

Introduction

Scrum Master Anti-Patterns: The reasons why scrum masters violate the spirit of the Scrum Guide are multi-faceted. They run from ill-suited personal traits and the pursuit of individual agendas to frustration with the team itself.

Read on and learn in this final post on scrum anti-patterns how you can identify if your scrum master needs support from the team to up his or her agile game.

SCRUM MASTER

20

ANTI-PATTERNS

© STEFAN WOLPERS 2017 · AGE-OF-PRODUCT.COM

The Scrum Master According to the Scrum Guide

Before we start dissecting probable reasons and manifestations of scrum master anti-patterns let us revisit how the Scrum Guide defines the role of the scrum master:

The Scrum Anti-Patterns Guide



“The Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide. Scrum Masters do this by helping everyone understand Scrum theory, practices, rules, and values.

The Scrum Master is a servant-leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.”

Source: [Scrum Guide 2017](#).

The keystone of the definition of the scrum master role is the servant leadership aspect. In most cases of scrum master anti-patterns, it is precisely this part that the individual is not living up to.

Possible Reasons Why Scrum Masters Leave the Agile Path

The reasons why scrum masters violate the spirit of the Scrum Guide are multi-faceted. They run from ill-suited personal traits via the pursuit of own agendas, to frustration with the team itself. Some often-observed reasons are:

- **Ignorance or laziness:** One size of agile fits every team. Your scrum master learned the trade in a specific context and is now rolling out precisely this pattern in whatever organization he or she is active no matter the context.
- **Lack of patience:** Patience is a critical resource a successful scrum master needs to field in abundance. Of course, there is no fun in readdressing the same issue several times, rephrasing it probably, if the solution is so obvious—from the scrum master's perspective. So, why not tell them how to do it 'right' all the time, thus becoming more efficient? Too bad, that agile cannot be pushed but needs to be pulled.
- **Dogmatism:** Some scrum masters believe in applying the Scrum Guide literally which unavoidably will cause friction.
- **Laissez-faire turned into indifference:** Pointing the team in a direction where the team members themselves can find a solution for an issue is good leadership. Letting them run without guidance, however, quickly turns into indifference, or worse, into an I-do-not-care mentality.
- **Dolla, dolla, bill ya'll—the scrum master imposter:** Secretly, the scrum master is convinced that this agile/scrum thingy is a fad, but recognizes that it is a well-paid one: “I will weather the decline in demand for project managers by getting a scrum master certificate.” This conviction will bring out his or her true colors over time

The Scrum Anti-Patterns Guide



inevitably.

- **Pearls before swine — the frustrated scrum master:** The scrum master has been working his or her butt off for months, but the team is not responding to the effort. The level of frustration is growing. There are a lot of potential reasons for a failure at this level: the lack of sponsoring from the C-level of the organization, a wide-spread belief that 'agile' is just the latest management fad, and thus ignorable. The team composition is wrong. There is no psychological safety to address problems within the team, and the company culture values neither transparency nor radical candor. Or individual team members harbor personal agendas unaligned with the team's objective — just to name a few. If the scrum team does not manage to turn the ship around the team will probably lose the scrum master. Note, that the scrum master cannot solve this issue by herself or himself. The cooperation of the team is required.
- **Lastly, the rookie:** If you apply [Occam's razor](#) to the situation, you may also conclude that your scrum master has not yet defected to the dark side. He or she might merely be inexperienced. Given that we all need to learn new skills regularly, cut him or her some slack in this case, and reach out to support the effort.

The Scrum Master as Agile Manager

In my eyes, 'agile management' is an oxymoron. The primary purpose of any agile practice is empowering those closest to a problem with finding a solution. In other words, the team shall become self-organizing over the course of time. Self-organizing teams need coaches, mentors, and (servant) leaders, however, not a manager in the taylorist meaning of the word.

Watch out for the scrum master anti-patterns corresponding to this 'agile manager' attitude:

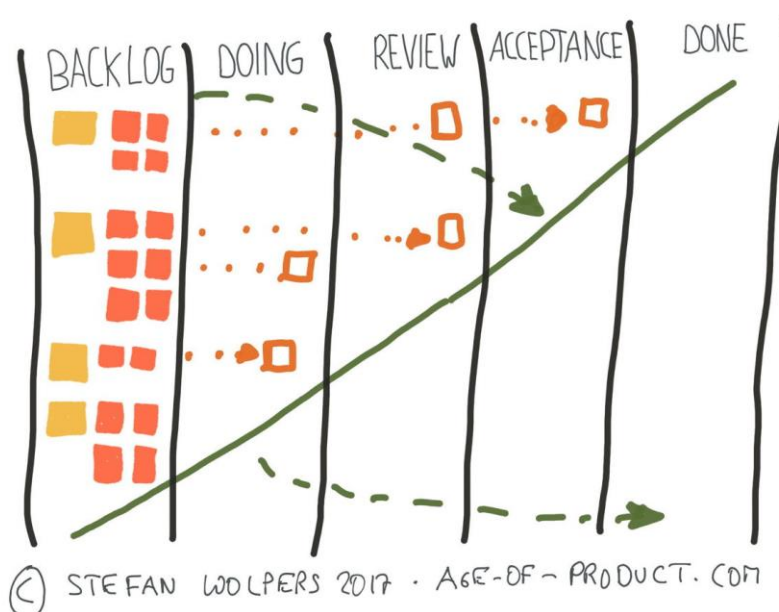
- **Agile management:** Self-organization does not mean the absence of management: Why would a scrum team assume, for example, responsibility for pay-role? Would that help with creating value for the customer? Probably less so. Hence, being a self-organizing team does not mean the absence of management per se. It does mean, however, that there is no need for micromanagement comparable to practices at a General Motors assembly plant in 1926. The scrum master is not a supervisor.
- **Running meetings be allowing someone to speak:** When team members seek eye-contact with the scrum master before speaking out the scrum master already left the facilitation role in favor of the supervisor mode.
- **Burn-down chart enforcer:** The scrum master focuses his or her work on producing a daily update to the burn-down chart. If the team considers a burn-down

The Scrum Anti-Patterns Guide



chart useful and the scrum master accepts to update it, so be it. I still believe, though, that a current sprint board serves the same purpose at a glance without adding a new administrative layer. However, if the burn-down is solely maintained to track the output for reporting purposes the team needs to challenge this attitude.

NO NEED FOR BURN-DOWN CHARTS



© STEFAN WOLPERS 2017 · AGE-OF-PRODUCT.COM

- **Pursuing flawed metrics:** The scrum master keeps track of individual performance metrics such as story points per developer per sprint, probably to report to that person's line manager. That is a classic supervisor hack to reintroduce command & control through the back door. It inevitably leads to cargo-cult scrum.
- **Escalating under-performance:** The scrum master reports to higher levels that the scrum team will not meet the current sprint commitment or forecast. I took this from a job offer I received: "You will coordinate and manage the work of other team members, ensuring that timescales are met and breaches are escalated." Perhaps, we should also reintroduce running the gauntlet for underperformers while we are at it?
- **Focusing on team harmony:** The scrum master sweeps conflict and problems under the rug by not using retrospectives to address those openly. This behavior is often a sign of bowing to politics and instead using manipulation to meet

The Scrum Anti-Patterns Guide



organizational requirements that are opposing scrum principles and values. If the organization values its underlings for following the ‘rules’ instead of speaking the truth why would you run effective retrospectives in the first place? A ‘scrum master’ participating in cargo-cult agile is again a supervisor than an agile practitioner.)

Scrum Master Anti-Patterns by Scrum Ceremony

Scrum Master Anti-Patterns During the Sprint Planning

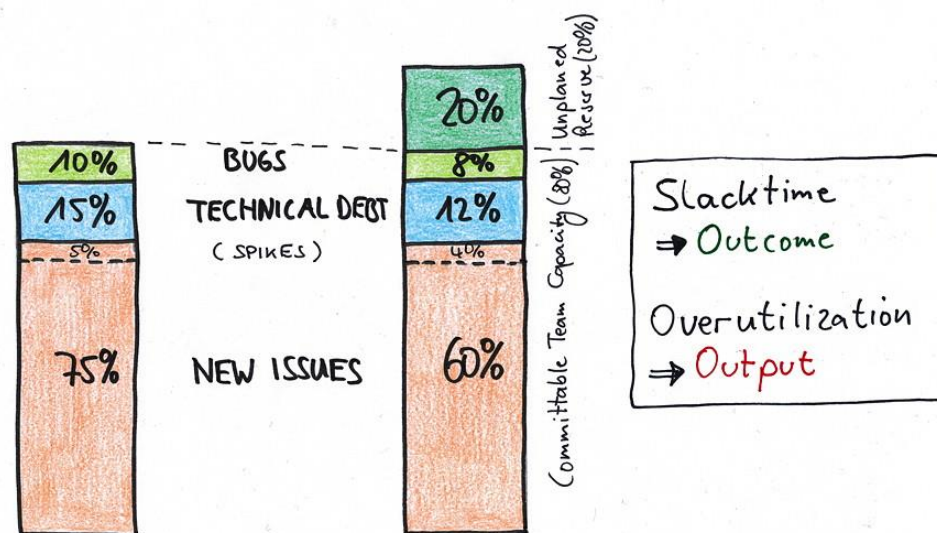
The following anti-patterns focus on the sprint planning:

- **Oversized sprint backlog without objection:** The team regularly accepts more issues into the sprint backlog than it can stomach without the scrum master’s invention. If at the end of a sprint 50% of all issues spill over to the next sprint and this a pattern then your team is not practicing scrum. Probably, it is a sort of time-boxed Kanban—which would be okay, too. Just make up your mind how you intend improving your customers’ life. Perhaps, Kanban would be a good choice.
- **Unrefined stories accepted into the sprint backlog:** The scrum master does not address the acceptance of issues into the sprint backlog violating the team’s definition of ready. This is a sure way that the scrum team will not deliver the sprint goal, rendering a scrum principle useless: providing a potentially shippable product increment at the end of the sprint. (This refers to regular work, not emergencies.)
- **100% utilization:** The product owner squeezes additional (functional) work into the sprint backlog, and the scrum master does not address the necessity of slack time. (The scrum team’s effectiveness will be significantly impeded if the team does not address technical debt every sprint. It will also suffer if there is no time for pairing, for example. A level of 100% utilization always reduces the team’s long-term productivity. A utilization rate of 100% is classic taylorist line management thinking.)

The Scrum Anti-Patterns Guide



Relative Team Capacity and Allocation for Sprint Planning



© Stefan Wolpers, 2017. (Age-of-Product.com)

Scrum Master Anti-Patterns During the Sprint

The following anti-patterns focus on the mishandling of the sprint itself:

- **Flow disruption:** The scrum master allows stakeholders to disrupt the flow of the scrum team during the sprint. There are several possibilities how stakeholders can interrupt the flow of the team during a sprint. Any of the examples will impede the team's productivity. The scrum master must prevent them from manifesting themselves:
 - The scrum master has a laissez-faire policy as far as access to the development team is concerned.
 - The scrum master does not oppose line managers taking team members off the team assigning other tasks.
 - The scrum master does not object that the management invites engineers to random meetings as subject matter experts.
 - The scrum master turns a blind eye to mid-sprint changes of the sprint backlog lacking the approval of scrum team.
 - Lastly, the scrum master allows that either stakeholders or managers turn the daily scrum into a reporting session.

The Scrum Anti-Patterns Guide



- **Assigning sub-tasks to developers:** The scrum master does not prevent the product owner—or anyone else—from assigning tasks directly to engineers. (The development team organizes itself without external intervention. And the scrum master is the shield of the team in that respect.)
- **Defining technical solutions:** The engineer turned scrum master and is now ‘suggesting’ how the scrum team is implementing issues.
- **Lack of support:** The scrum master does not support team members that need help with a task. Often, development teams create tasks an engineer can finish within a day. However, if someone struggles with such a job for more than two days without voicing that he or she needs support, the scrum master should address the issue. By the way, this is also the reason for marking tasks on a physical board with red dots each day if tasks do not move to the next column.

Scrum Master Anti-Patterns During the Retrospective

The final set of anti-patterns addresses the sprint retrospective:

- **Groundhog day:** The retrospective never changes in composition, venue, or length. There is a tendency in this case that the team will revisit the same issues over and over again – it’s groundhog day without the happy ending, though.
- **Let’s have it next sprint:** The scrum master postpones the retrospective into the next sprint. Beyond the ‘inspect & adapt’ task, the retrospective shall also serve as a moment of closure that resets everybody’s mind so that the team can focus on the new sprint goal. That is the reason why we have the retrospective before the planning of the follow-up sprint. Postponing it into the next sprint may interrupt the flow of the team. It also delays tackling possible improvements by up to a sprint.
- **#NoRetro:** The scrum master does not gather data during the sprint that supports the team in the upcoming retrospective. This could also be a sign of frustration, see above.
- **#NoDocumentation:** The scrum master does not take minutes for later use. A retrospective is a substantial investment, and the scrum team should take it seriously. Taking notes and photos supports this process.
- **No psychological safety:** The retrospective is an endless cycle of blame and finger pointing without intervention from the scrum master. The team wins together and the team loses together. The blame game documents both the failure of the scrum

The Scrum Anti-Patterns Guide



master as the facilitator of the retrospective as well as the team's lack of maturity and communication skills.

- **Bullying is accepted:** One or two team members are dominating the retrospective. This communication behavior is often a sign of either a weak or uninterested scrum master. The retrospective needs to be a safe place where everyone—introverts included—can address issues and provide his or her feedback free from third-party influence. If some of the team members are dominating the conversation, and probably even bullying or intimidating other teammates, the retrospective will fail to provide such a safe place. This failure will result in participants dropping out of the retrospective and render the results obsolete. It is the primary responsibility of the scrum master to ensure that everyone will be heard and has an opportunity to voice his or her thoughts. By the way, equally distributed speaking time is according to Google also a sign of a high-performing team.

Read More: [What Google Learned From Its Quest to Build the Perfect Team.](#)

- **Stakeholder alert:** The scrum master permits stakeholders to participate in the retrospective. There are plenty of scrum ceremonies that address the communication needs of stakeholders: the sprint review, probably the product backlog refinement, the daily scrums, not to mention opportunities of having a conversation at water coolers, over coffee, or during lunchtime. If that spectrum of possibilities still is not sufficient, feel free to have additional meetings. However, the retrospective is off-limits to stakeholders, and the scrum master needs to enforce this rule.

The Conclusion

There are plenty of possibilities to fail as a scrum master. Sometimes, it is the lack of organizational support. Some people are not suited for the job. Others put themselves above their teams for questionable reasons. Some scrum masters simply lack feedback from their scrum teams and stakeholders. Whatever the case may be, though, try and lend your scrum master in need a hand to overcome the misery. Scrum is a team sport.

The Scrum Anti-Patterns Guide



Scrum Master Anti-Patterns Derived from Job Ads

Job ads for scrum master or agile coach positions reveal a great insight into an organization's progress on becoming agile. Learn more about what makes job ads such a treasure trove with the following 22 scrum master anti-patterns. To gain these, I analyzed more than 50 job ads for scrum master or agile coach positions.

22 Scrum Master Anti-Patterns from Job Ads



Analyzing a Job Advertisement for a Scrum Master or Agile Coach position

Probably, you are considering a position as a scrum master or agile coach in a particular organization. I suggest that before going all in (the application process), you should consider analyzing the job description for scrum master anti-patterns first.

How Large Organizations Create Job Ads

Usually, the organization's HR department will create the final text of the job advertisement and post it to the chosen job sites. Hopefully, and depending on their process and level of collaboration (and agile mindset) in the organization, the team for which the new position

The Scrum Anti-Patterns Guide



was advertised may have participated in creating the job ad. This certainly avoids advertising a wrong description to prospective candidates.

Too often, however, advertisements may read like a copy and paste from positions that an organization's HR believes to be similar to that of a scrum master (for example, a project manager). Or, sometimes, the HR department copies from other scrum master job ad which they believe correctly reflect the requirements of the organization. So, don't be too surprised to see a job advertisement that reads like a list of scrum master anti-patterns.

Red Flags: A Sign of Cargo Cult Agile or just an Organization at the Beginning of the Agile Transition?

This is often the case when an organization's HR does not have a lot of experience in hiring agile practitioners because they are in the early stages of the agile transition. Therefore, an unusual job description does not imply that the organization is not trying to become agile, it may just mean that the HR department has not yet caught up with the new requirements. Such an advertisement can actually help raise the topic and be of benefit during the job interview.

Be aware, however, that if an organization which claims to be agile is using this kind of advertisement despite being well underway on its agile transition, it then raises a red flag: miscommunication in the hiring process may indicate deeper issues or problems at the organizational level. It could be as critical as someone at management level, to whom the new scrum master would likely report, having no clue what becoming agile is all about.

Scrum Master Anti-Patterns from Job Ads in 22 Examples

As mentioned previously, here are some examples of scrum master advertisement anti-patterns (from more than 50 actual job descriptions) that should raise a red flag:

1. **Ersatz PM:** The scrum master position is labeled as "Project manager/Scrum master", "Agile Project Manager", or "Agile scrum master". (Are there un-agile scrum masters mentioned in the Scrum Guide?)
2. **The whip:** The scrum master is expected to communicate the company priorities and goals. (Product backlog-wise priorities are the job of the product owner. Scrum-wise it is a good idea that the scrum master spreads scrum values and, for example, coaches the scrum team to become self-organizing. Whether this is aligned with the company goals remains to be seen.)
3. **Technical PO:** The scrum master is also supposed to act as a (technical) product owner. (There is a reason why scrum knows three roles and not just two. Avoid

The Scrum Anti-Patterns Guide



assuming more than one role at a time in a scrum team.)

4. **Outcome messenger:** The scrum master reports to stakeholders the output of the scrum team (velocity, burndown charts). (Velocity—my favorite agile vanity metric.) (Read More: [Agile Metrics — The Good, the Bad, and the Ugly.](#))
5. **SuperSM:** The scrum master is supposed to handle more than one or two teams simultaneously. (Handling two scrum teams is already challenging, any number beyond that is not feasible.)
6. **Scrum secretary:** The scrum master is supposed to do secretarial work (room bookings, facilitation of ceremonies, ordering office supplies). (Read More: [Scrum Master Anti-Patterns: Beware of Becoming a Scrum Mom \(or Scrum Pop\).](#))
7. **Scrum mom:** The scrum master is removing impediments on behalf of the team. (How is the scrum team supposed to become self-organizing if the scrum master handles all obstacles?).
8. **Team manager:** The scrum master is responsible for team management. (If nothing else helps read the manual Scrum Guide: Is there anything said about team management by the scrum master?)
9. **Delivery manager:** The scrum master is responsible for the “overall delivery of the committed sprint”. (I assume the organization does not understand scrum principles very well. The forecast and the sprint goal seem to be particularly challenging.)
10. **CSM®, CSP® & CST®:** CSM or equivalent certification is listed as mandatory. (A typical save-my-butt approach to hiring. A CSM certification only signals that someone participated in a workshop and passed a multi-choice test.)
11. **Delivery scapegoat:** The scrum master is expected to accept full responsibility of the delivery process. (That is rather the responsibility of the scrum team.)
12. **Proxy PO:** The scrum master is expected to drive functional enhancements and continuous maintenance. (Maybe someone should talk to the product owner first?)
13. **Keeper of the archives:** The scrum master is expected to maintain relevant documentation. (Nope, documentation is a team effort.)
14. **The PM Reloaded:** The scrum master organizes the scrum team’s work instead of the project manager. (Why use scrum in the first place if creating self-organizing teams is not the goal?)

The Scrum Anti-Patterns Guide



15. **Risk detector:** The scrum master is expected to monitor progress, risks, resources, and countermeasures in projects. (The scrum master is neither a project manager nor a risk mitigator. (Risk mitigation is a side-effect of becoming a learning organization built around self-organizing teams.))
16. **Scrum minion:** The scrum master is expected to prepare steering team and core team meetings. (The last time I checked the Scrum Guide there was no 'steering team' mentioned.)
17. **WTF?** The scrum master is expected to perform the role for “multiple flavors of agile methodologies”. ([Multiple what?](#))
18. **Psychic:** The scrum master is expected to participate in “project plan review and provide input to ensure accuracy”. (The scrum master is neither a project manager nor capable of predicting the future any better than another human being.) <
19. **Bean counter:** The scrum master is expected to “review and validate estimates for complex projects to ensure correct sizing of work”. (Well, reviewing estimates might be the job of the scrum team during the product backlog refinement process if they see value in that. However, there is no review by the scrum master.)
20. **Discoverer:** The scrum master is expected to provide “design thinking sessions”. (I love covering the product discovery process, too. However, this should be a joint effort with the product owner and the rest of the team.)
21. **Techie:** The scrum master is expected to “walk the product owner through more technical user stories”. (Nope, that is the job of the developers. The product backlog refinement meetings are ideal for this purpose.)
22. **Siloed in doing agile:** There is no mention of the scrum master either coaching the organization, or coaching the product owner.

My favorite anti-pattern is:

“...working reliably on projects within a given time and budget frame whilst maintaining our quality standards.”

In other words: “Actually, we’re happy with our waterfall approach but the C-level wants us to be agile.”

Let’s close this section with an exemplary job advertisement, posted by Zalando in 2016 for a (senior) agile coach position: [\(Senior\) Agile Coach](#).

The Scrum Anti-Patterns Guide



Conclusion—Scrum Master Anti-Patterns from Job Ads

The job ad of the organization of your interest is a best-of of scrum master anti-patterns. Should you in this case immediately drop your interest in becoming a member of that organization? I don't think so. An extensive list of red flags can be beneficial, too. For example, the HR department might merely be misaligned with the scrum team in question as the organization is still in the early day of its agile transition. That sounds like an attractive opportunity to me.

On the other hand, the organization might just try to attract talented people by sugar-coating its otherwise command & control like management style with some glitzy agile wording. Continuing the application process under these conditions might indeed be a waste of your time. A short phone call/interview will bring clarity.

The Scrum Anti-Patterns Guide



About the Author

Stefan Wolpers has worked many years as a product manager, product owner, and agile coach (Scrum, Lean Startup, Lean Change). He's founded multiple companies, and has led the development of B2C and B2B software for, primarily, startups, but also for other organizations — including a former Google subsidiary.

Despite originally studying chemistry Stefan has never worked in a laboratory, and instead continued his education in business administration and law. Following school he discovered a passion for software and, in 1996, launched the first online ecommerce platform to feature SAP R/3 connectivity — only to learn that the early bird does not necessarily catch the worm. After moving from his home town of Hamburg to Berlin, Germany, he created Susuh GmbH, a marketplace for local services. Other ventures followed, and in 2011 he founded [Startup Camp Berlin](#) — one of the largest German startup conferences today.

Stefan's latest project, [Age of Product](#), focuses on the exchange of knowledge between the people involved in product development: product managers, product owners, Scrum Masters, designers, and developers. The goal is to help those involved in product development with lessons learned and best practices for continuous agile product discovery and delivery.

Connect with Stefan via [Twitter](#), [LinkedIn](#), or privately via [email](#).