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

Version: 3.38 2020-03-11

The Scrum Anti-Patterns Guide

Table of Content

PREFACE 4

SCRUM MASTERY IN 300 WORDS 5

THE TOP THREE OBJECTIVES 5

SCRUM EVENT ANTI-PATTERNS 6

DAILY SCRUM ANTI-PATTERNS: 20 WAYS TO IMPROVE 6
THE DAILY SCRUM 6
THE PURPOSE OF THE DAILY SCRUM 6
DAILY SCRUM ANTI-PATTERNS 7
CONCLUSION 9
28 PRODUCT BACKLOG AND REFINEMENT ANTI-PATTERNS 10
THE PRODUCT BACKLOG 10
THE PRODUCT BACKLOG REFINEMENT ACCORDING TO THE SCRUM GUIDE 10
A TYPICAL PRODUCT BACKLOG REFINEMENT PROCESS 11
COMMON PRODUCT BACKLOG (REFINEMENT) ANTI-PATTERNS 12
CONCLUSION: 16
20 SPRINT PLANNING ANTI-PATTERNS 17
THE SPRINT PLANNING 17
THE PURPOSE OF THE SPRINT PLANNING 17
SPRINT PLANNING ANTI-PATTERNS 18
CONCLUSION 22
27 SPRINT ANTI-PATTERNS 23
THE SPRINT 23
SPRINT ANTI-PATTERNS 23
CONCLUSION 30
15 SPRINT REVIEW ANTI-PATTERNS 31
THE SPRINT REVIEW 31
THE PURPOSE OF SCRUM'S SPRINT REVIEW 31
WHAT DOES THE SCRUM GUIDE SAY ABOUT THE SPRINT REVIEW? 32
SPRINT REVIEW ANTI-PATTERNS 32
CONCLUSION 35
21 SPRINT RETROSPECTIVE ANTI-PATTERNS IMPEDING SCRUM TEAMS 36
INTRODUCTION 36
THE SCRUM GUIDE ON THE SPRINT RETROSPECTIVE 36
SPRINT RETROSPECTIVE ANTI-PATTERNS 37
CONCLUSION 42

SCRUM ROLE ANTI-PATTERNS 43

PRODUCT OWNER ANTI-PATTERNS 43
THE ROLE OF THE PRODUCT OWNER ACCORDING TO THE SCRUM GUIDE 43
PRODUCT BACKLOG AND REFINEMENT ANTI-PATTERNS 44
SPRINT PLANNING ANTI-PATTERNS 46
SPRINT ANTI-PATTERNS 47
PO ANTI-PATTERNS DURING THE DAILY SCRUM 48

Sprint Review Anti-Patterns 48
Conclusion 49
Scrum Master Anti-Patterns 50
Introduction 50
The Scrum Master According to the Scrum Guide 50
Possible Reasons Why Scrum Masters Leave the Path 51
The Scrum Master as Agile Manager 52
Scrum Master Anti-Patterns by Scrum Events 53
Conclusion 57
Development Team Anti-Patterns 58
Introduction 58
The Role of the Development Team in Scrum 58
Development Team Anti-Patterns by Scrum Event 59
Development Team Anti-Patterns — Conclusion 64
Scrum Master Anti-Patterns Derived from Job Ads 65
Analyzing a Job Advertisement for a Scrum Master or Agile Coach position 65
Scrum Master Anti-Patterns from Job Ads in 22 Examples 66
Conclusion 69
Scrum Stakeholder Anti-Patterns 70
The Stakeholder and Organizational Excellence in Legacy Organizations 70
Common Scrum Stakeholder Anti-Patterns 71

HOW TO DETECT SCRUM ANTI-PATTERNS 78

Use Burn-Down Charts to Discover Scrum Anti-Patterns 78
Introduction 78
Scrum Anti-Patterns Visualized by Burn-Down Charts 79
The Conclusion 84

ABOUT THE AUTHOR 84

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 Mastery in 300 Words

How to make Scrum work? Read on to learn more about my top three objectives for Scrum Masters striving to achieve Scrum Mastery.

The Top Three Objectives

Achieving Scrum Mastery is no rocket-science: Make sure that the Scrum Team delivers a valuable, potentially shippable Product Increment every single Sprint with the precision of a Swiss clockwork. Delivering at this level builds a happy customer base, trust within the organization as well as high morale among the team members. To do so, focus your activities on three objectives:

1. Defend the **Product Backlog** tooth and nail to ensure it represents the best possible use of the Development Team's work from a customer value perspective at any given moment—garbage in, garbage out. In other words, your Scrum Team's Product Backlog has to be actionable 24/7. By my standards for Scrum Mastery, that means that you need to be capable of running a meaningful Sprint Planning instantly.

2. Keep **technical debt** at bay from day #1: Make sure that the Development Team members are embracing Xtreme Programming techniques from TDD, pair programming, relentless refactoring to supporting an emergent architecture from the start. Also, fight for their Slack time—at least 20% of their theoretical capacity—uncompromisingly. Pay serious attention to the concept of 'Done,' as represented in the Definition of Done.

3. **Support the middle management** by educating them on how to become servant leaders, thus alleviating their fear of obsolescence. If the ability to pay for a mortgage is no longer an issue, personal agendas, managers might harbor, will be overcome, and we can address the necessary change within the organization collaboratively.

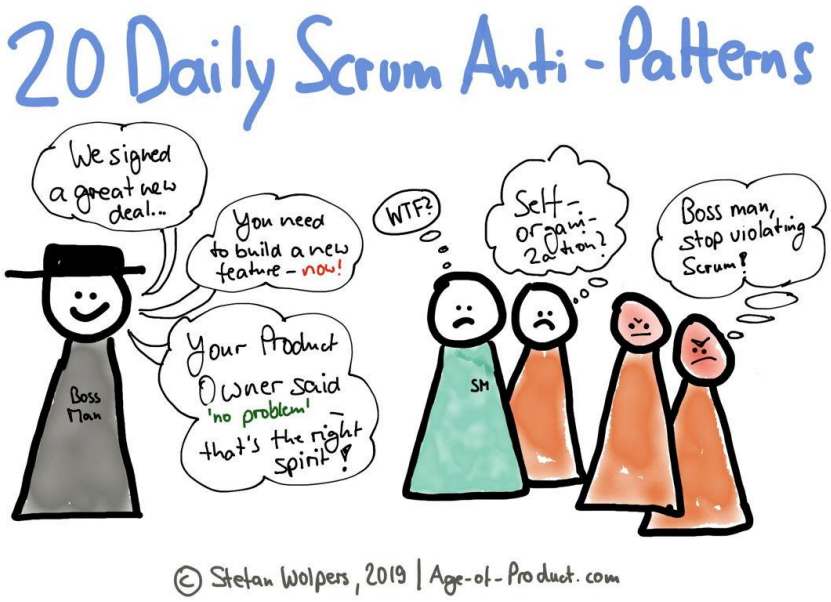
Of course, as so often, the devil is in the details. Some 160 of those devils we will address in this ebook.

Scrum Event Anti-Patterns

Daily Scrum Anti-Patterns: 20 Ways to Improve

The Daily Scrum

In my experience, the Daily Scrum is the Scrum event with the highest anti-pattern density among all events. Learn more about the Daily Scrum anti-patterns that threaten to derail your agile transition.



The Purpose of the Daily Scrum

The purpose of the Daily Scrum is clearly described in the Scrum Guide — no guessing is necessary:

"The Daily Scrum is a 15-minute time-boxed event for the Development Team. The Daily Scrum is held every day of the Sprint. At it, the Development Team plans work for the next 24 hours. This optimizes team collaboration and performance by inspecting the work since the last Daily Scrum and forecasting upcoming Sprint work. The Daily Scrum is held at the same time and place each day to reduce complexity."

"Daily Scrums improve communications, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision-making, and improve the Development Team's level of knowledge. This is a key inspect and adapt meeting."

Source: Scrum Guide 2017.

The Daily Scrum is an essential event for inspection and adaption, run by the Development Team, and guiding it for the next 24 hours on its path to achieving the Sprint Goal. The Daily Scrum is hence the shortest planning horizon in Scrum and thus mandatory.

Contrary to popular belief, its 15-minutes timebox is not intended to solve all the issues addressed during the Daily Scrum. It is about creating transparency, thus triggering the inspection. If an adaption of the plan or the Sprint Backlog, for example, is required, the Development Team is free to handle the resulting issues at any time. In my experience, most Daily Scrum anti-patterns result from a misunderstanding of this core principle.

Daily Scrum Anti-Patterns

Typically, a good Scrum Team won't need more than 10 to 15 minutes to inspect its progress towards the Sprint Goal. Given this short period, it is interesting to observe that the Daily Scrum is often so riddled with anti-patterns. The anti-patterns often cover a broad spectrum, ranging from behaviors driven by dysfunctional Scrum teams to apparent failures at an organizational level.

My unprioritized list of notorious Daily Scrum anti-patterns is as follows:

1. **No routine:** The Daily Scrum 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 the Daily Scrum. Think of it as a spontaneous drill: don't put too much thought into the stand-up, just do it. Skipping Daily Scrums can turn out to be a slippery slope: if you skip one Daily Scrums or two, why not skip every second one?)

2. **Status report:** The Daily Scrum is a status report meeting, and Development 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 Daily Scrum.

5. **Planning meeting:** The team hijacks the Daily Scrum to discuss new requirements, to refine user stories, or to have a sort of (sprint) planning meeting.

6. **Orientation lost:** The Daily Scrum serves one purpose as it answers a simple question: Are we still on track to meet the Sprint Goal? Or do we need to adapt the plan or the

Sprint Backlog or both? Often, the Development Team cannot answer that question immediately. (In that respect, visualizing the progress towards the Sprint Goal is a useful exercise. Removing the Development Team task of maintaining a mandatory burndown chart from the Scrum Guide a few years ago does not imply that a burndown chart is obsolete.)

7. **No use of work-item age:** A Development team member experiences difficulties in accomplishing an issue over several consecutive days and nobody is offering help. (This a sign that people either may not trust each other or do not care for each other. Alternatively, the workload of the Development Team has reached an unproductive level as they no longer can support each other.)

8. **Monologs:** Team members violate the timebox, starting monologues. (60 to 90 seconds per team member should be more than enough time on air.)

9. **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.)

10. **Disrespect I:** Other team members are talking while someone is sharing his or her progress with the team. (The use of talking tokens among adults to avoid this behavior does not qualify as a solution in my eyes.)

11. **Assignments:** The Product Owner – or even the "Scrum Master" – assigns tasks directly to team members.

12. **Cluelessness:** Team members are not prepared for the Daily Scrum. ("I was doing some stuff but I cannot remember what. Was important, though.")

13. **Let's start the shift:** The Daily Scrum acts as a kind of artificial factory siren to start the next shift. (This is a common Taylorism artifact where trust in the Development Team's capability to self-organize is missing.)

14. **Disrespect II:** Team members are late to the stand-up or do not show up at all.(This poses a massive risk for the Development Team as it is inspecting and probably adapting the plan based on incomplete information thus reducing the probability of achieving the Sprint Goal.)

15. **Excessive feedback:** Team members criticize other team members right away sparking a discussion instead of taking their critique outside the Daily Scrum.

16. **Overcrowded:** The Daily Scrum is ineffective due to a large number of active participants. (There is a reason why the Scrum Guide recommends to limit the number of Development Team members to nine.)

17. **Talkative chickens:** “Chickens” actively participate in the Daily Scrum. (Stakeholders are supposed to listen in but not distract the Development Team members during their inspection.)

18. **Command & control by the management:** Line managers are attending the Daily Scrum to gather “performance data” on individual team members. (This behavior is defying the very purpose of self-organizing teams.)

19. **"A word, please”:** Line managers are waiting until the Daily Scrum is over and then reach out to individual Development Team members for specific reporting from them. (Nice try. However, this hack is also unwanted behavior and distracts the Development Team.)

20. **Additional Work:** The Product Owner or even other stakeholders attempt to introduce new work to the current Sprint during the Daily Scrum. (This behavior may be acceptable for high priority bugs, although the Development Team should be aware of those before the Daily Scrum. Otherwise, the composition of the Sprint Backlog is the sole responsibility of the Development Team.)

Lastly, some teams like to have their Daily Scrum in Slack, particularly those that are not co-located. Using Slack does not manifest an anti-pattern per se; it is the prerogative of the Development Team to run the Daily Scrum in any fashion that serves its purpose: inspect the plan for the next 24 hours to meet the Sprint Goal. (I was even working with a co-located Scrum team that used Slack as their preferred way of having a Daily Scrum. It worked.)

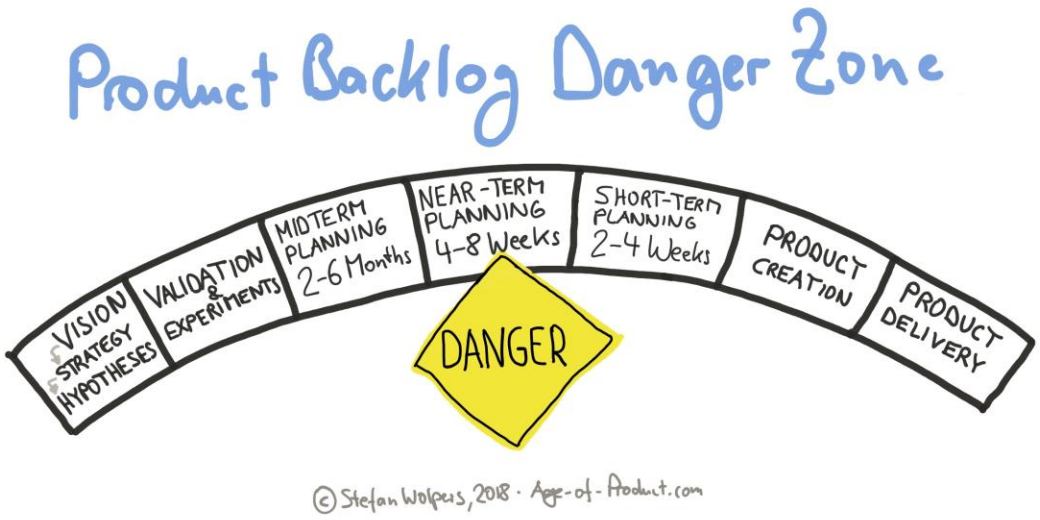
Conclusion

Given the importance of the Daily Scrum for the success of the Scrum Team’s effort of achieving the Sprint Goal, its anti-patterns density is no surprise. People seem to be either ignorant (or at least less well educated) about its purpose. Or they — intentionally or not — interfere with the Development Team’s self-organization. One way or another, it is a crucial responsibility of the Scrum Master to help all participants to overcome typical Daily Scrum anti-patterns.

28 Product Backlog and Refinement Anti-Patterns

The Product Backlog

Scrum is a simple, yet sufficient framework to build emerging products, provided you identify in advance what is worth building. 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; garbage in, garbage out—as the saying goes. The following article points at 28 Product Backlog anti-patterns — including the Product Backlog refinement process — that limit your Scrum 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:

"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 and ScrumInc*[1].

A Typical Product Backlog Refinement Process

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

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 timebox discussions. A typical timebox per Product Backlog item 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.

[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/.*

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.

Common Product Backlog (Refinement) Anti-Patterns

Despite being relatively straightforward, the process of creating and refining a Product Backlog often suffers from various anti-patterns. I have identified five different categories for Product Backlog anti-patterns:

General Product Backlog Anti-Patterns

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 PO is the only person 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. **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?)

3. **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.)

4. **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.)

5. **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.)

6. **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.)

7. **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 re-

finement 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.)

8. **No more than a title:** The Product Backlog contains user stories that comprise of little more than a title. (See above.)

9. **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.)

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

11. **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

1. **Roadmap?** The Product Backlog is not reflecting the roadmap. (The Product Backlog is supposed to be detailed enough 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.)

2. **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.)

3. **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.)

4. **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

1. **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 cognitive overload and makes alignment with the 'big picture' at portfolio management and roadmap planning level very tough.)

2. **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.)

3. **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.)

4. **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.)

5. **INVEST?** The Product Owner is not applying the INVEST principle by Bill Wake to user stories.

6. **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 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.)

7. **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).

8. **'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.)

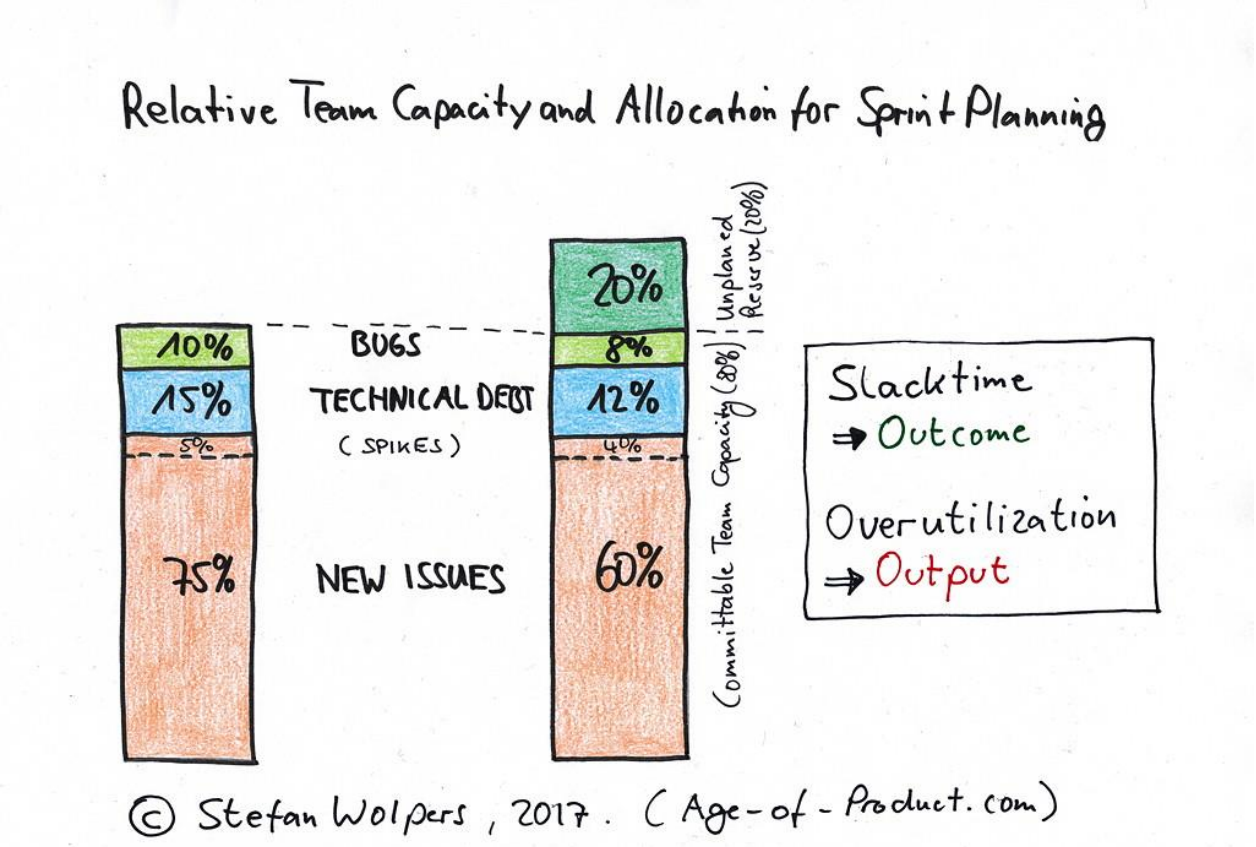
Product Backlog Anti-Patterns of the Development Team

1. **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?)

2. **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.)

3. **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 forecast. However, it cannot be addressed early enough. If a team's capacity 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

1. **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.)

2. **Too much refinement:** The team has too many refinement sessions, resulting in a too detailed backlog. (Too much refinement isn't healthy either.)

3. **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.)

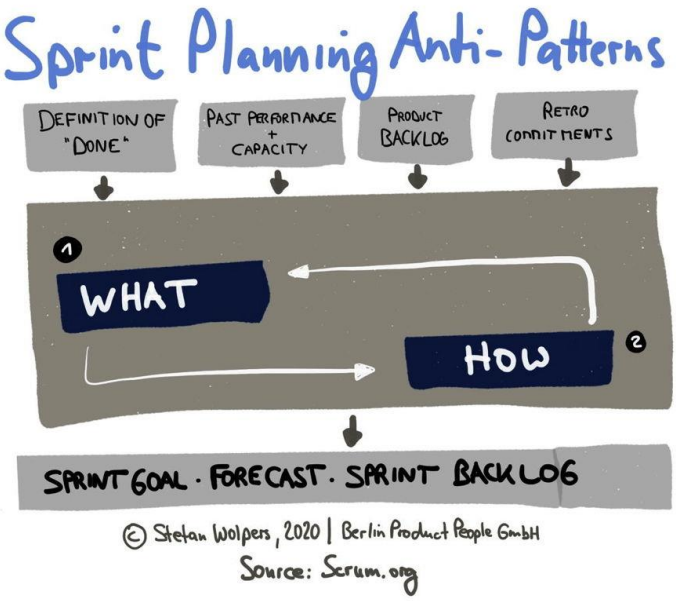
Conclusion:

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.

20 Sprint Planning Anti-Patterns

The Sprint Planning

Sprint Planning is a core event, defining how your customers' lives will improve with the next Product Increment. It is hence an excellent practice to invest upfront during the Product Backlog refinement and avoid the following 20 Sprint Planning anti-patterns.



The Purpose of the Sprint Planning

The purpose of the Sprint Planning is to align the Development Team and the Product Owner on what to build next, delivering the highest possible value to customers. The Product Owner introduces the business objective the upcoming Sprint is supposed to meet, the Scrum Team collaboratively creates a Sprint Goal, and the Development Team forecasts the work required to achieve the goal by picking the appropriate items from the Product Backlog, transferring them to the Sprint Backlog. Also, the Development Team needs to come up with a plan on how to accomplish its forecast as well as pick at least one high priority improvement issues from the Sprint Retrospective.

According to the Scrum Guide, the Sprint Planning answers two questions:

1. "What can be delivered in the Increment resulting from the upcoming Sprint?"
2. "How will the work needed to deliver the Increment be achieved?"

Source: Scrum Guide, 2017.

If the Scrum Team has been successfully utilizing the Product Backlog refinements to create and maintain an actionable Product backlog, the Sprint Planning consumes much less time than the eight hours, the Scrum Guides lists for a month-long Sprint. The Development Team and the Product Owner adjust the previously discussed scope of the upcoming Sprint to the available capacity.
Alternatively, a valuable new task may have appeared overnight, and the Product Owner wants this task to become a part of the next Sprint, too. Consequently, some other Product Backlog items need to be returned to the Product Backlog. A good team can handle that in ten to 30 minutes before moving on to the decomposition tasks and the initial planning of how the Development Team intends to accomplishing the work of the Sprint.

Sprint Planning Anti-Patterns

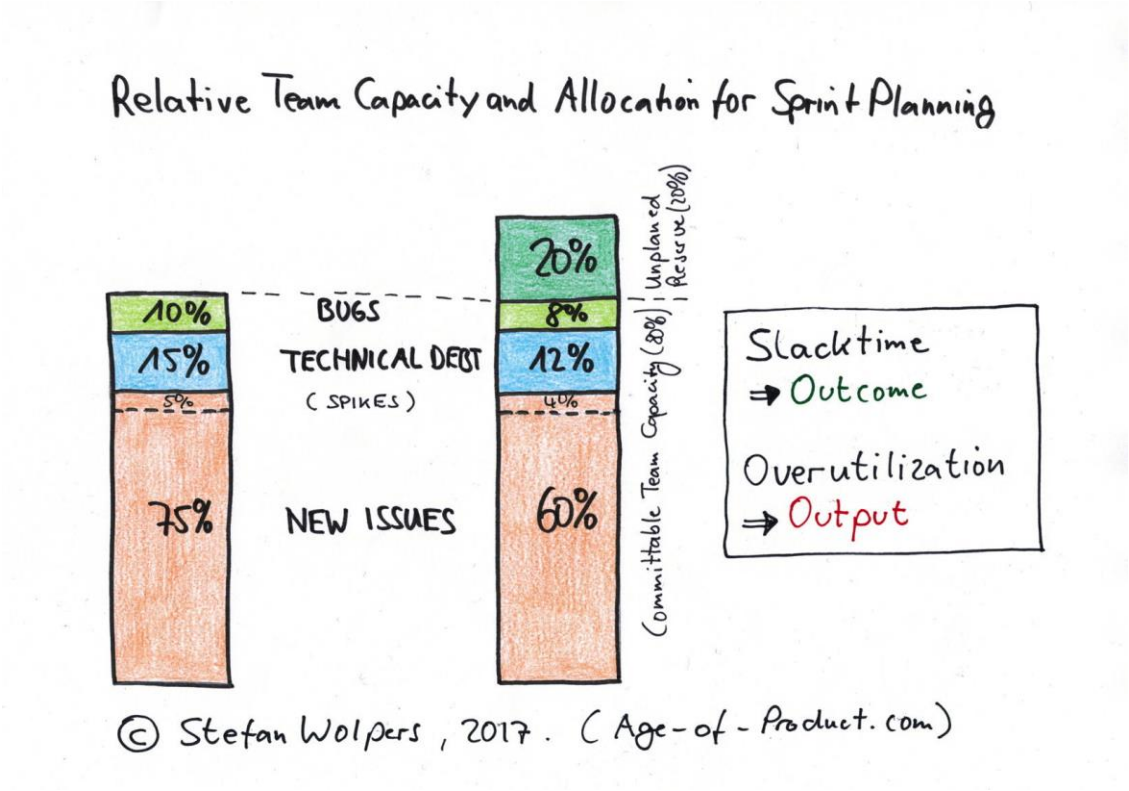
There are three categories of Sprint Planning anti-patterns. They concern the Development Team, the Product Owner, and the Scrum team:

Sprint Planning Anti-Patterns of the Development Team

- **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 about 20 % of resources are well-spent every sprint to fix bugs and refactor the codebase. If the Product Owner ignores the need for this, and the development team accepts this attitude, the Scrum Team will find itself in a downward spiral. Its future product delivery capability will decrease. Read more on technical debt and Scrum.)

- **No slack time:** 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 do pair programming, for example. 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 members. Overutilization will always push the individual team member to focus on his or her work. On the other side, slack time will allow the Scrum Team to act collaboratively and focus on the outcome.)



- **Planning too detailed:** During the Sprint Planning, the Development Team plans every single task of the upcoming Sprint in advance. (Don't become too granular. One-third of the tasks are more than sufficient to not just start with the Sprint, but also start learning. The Sprint Backlog is emergent, and 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 planning altogether. (Skipping planning is unfortunate, as it is also an excellent opportunity to talk about how to spread knowledge within the Development Team, where the architecture is heading, or whether tools are adequate. For example, the team should think about who will be pairing with whom on what task. The Development Team planning part is also well-suited to consider how to reduce technical debt, see above.)

- **Team leads?** The Development Team does not come up with a plan to deliver on its forecast collaboratively. Instead, a 'team lead' does all the heavy lifting and probably even 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 align the business objective of the upcoming Sprint with the overall product vision. (A serious goal answers the "What are we fighting for?" question. To a certain extent, it is also a negotiation between the Product Owner and the Development Team. It is focused and measurable, as the Sprint goal—based on the business objective—and Development Team's forecast go hand in hand.)

- **No business objective, no Sprint Goal:** The Product Owner proposes Product Backlog items that resemble a random assortment of tasks, providing no cohesion. Consequently, the Scrum Team does not create a Sprint goal. (If this is the natural way of finishing your Sprint Planning, 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, though, remember the sunk cost fallacy.)

- **Last minute changes:** The Product Owner tries to squeeze in some last-minute Product Backlog items that are not ready yet. (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 tasks 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 ordering the Product backlog 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 is the road to becoming a feature factory and deserves attention from the team's Scrum Master. It is violating the Development Team's prerogative to pick Product Backlog item for the Sprint Backlog as well as Scrum Values.)

- **No preparation:** The Product Owner does prepare the Product Backlog to provide useful Product Backlog items for selection by the Development Team. (Product Backlog needs to represent the best possible use of the Development Team's work from a customer value perspective at any given moment. In other words, your Scrum Team's Product Backlog

has to be actionable 24/7. By my standards, that means that you need to be capable of running a meaningful Sprint Planning instantly. Preparing a few basic Product Backlog items an hour before the beginning of the Sprint Planning is not enough.)

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 or the Sprint Goal at hand. (It is quite common, for example, to extend the Sprint length at the end of the year when most of the team members are on holiday. However, flexibly changing the Sprint length as "needed" is a dark pattern. Instead of changing the Sprint length to accommodate the Sprint Goal, the Scrum Team should invest more effort into sizing tasks in the right way.)

- **Over-commitment:** The Scrum Team regularly takes on way too many tasks and moves unfinished work directly to the next Sprint. (If two or three items spill over to the next Sprint while the Development Team meets the Sprint Goal, so be it. If regularly 30 to 40 percent of the original forecast 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. If the team considered Scrum still to be its choice, I would recommend to put more energy into Product Backlog refinement and creating meaningful Sprint Goals.)

- **Stage-Gate® by DoR:** The Scrum Team decides that a definition of ready is advantageous but handles it dogmatically, 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 corresponding Product Backlog item into the Sprint — that is fine. However, if the definition of ready is used a checklist, rejecting everything that is not 100 percent covered by it, then you are reintroducing waterfall through the backdoor, only this time it is the Development Team doing that. Read More: The Dangers of a Definition of Ready.)

- **Ignoring the DoR:** The Development Team does not reject Product Backlog items that do not meet its definition of ready. (This is the opposite side of being dogmatic about the application of DoR: unready tasks that will cause unnecessary disruptions during the Sprint—possibly endangering achieving the Sprint Goal—are allowed into it. Laissez-faire does not help either.)

- **Forecast imposed:** The Sprint forecast 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 forecast. 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 forecast on behalf of the Development Team.)

- **Planning ignored:** The Development Team is not participating collectively in the Sprint Planning. Instead, two team members, for example, the "tech lead" and "UX lead," represent the team. (As far as the idea of one or two "leading" teammates in a Scrum Team are concerned, there are none, see above. And unless you are using LeSS or Nexus—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.)

Sprint Planning Anti-Patterns of the Scrum Master

The Product Owner is responsible for the business objective of the upcoming Sprint and hence to provide an appropriately prepared Product backlog. At the same time, the Development Team is in charge of selecting the necessary Product Backlog items to meet the collaboratively created Sprint Goal. So, what are Sprint Planning anti-patterns of the Scrum Master, you may ask yourself?

I can think of one Sprint Planning anti-pattern that falls into the responsibility of the Scrum Master: not ensuring that an important improvement issue from the previous Sprint Retrospective to help the Scrum Team improve continuously becomes a part of the Product Backlog.

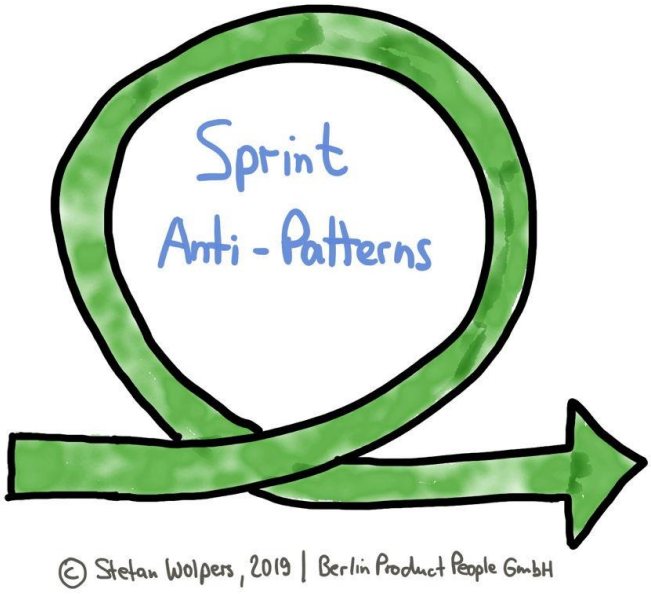
Conclusion

Sprint Planning is a core event, defining how your customers' lives will improve with the next Product Increment, and to not be taken lightly. Fortunately, most of the beforementioned Sprint Planning anti-patterns are simple to fix. Just take it to the team, respect Scrum Values, self-organization, and Scrum's built-in checks & balances.

27 Sprint Anti-Patterns

The Sprint

Welcome to Sprint anti-patterns article from our series on Scrum anti-patterns, covering not just the three Scrum roles, but also the stakeholders as well as the IT management.



Sprint Anti-Patterns

This list of notorious Sprint Anti-Patterns applies to all Scrum roles and beyond: the Product Owner, the Development Team, the Scrum Master, the Scrum Team itself, 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. (As the Sprint Backlog is emergent and new work may be identified as necessary to achieve the Sprint Goal, this attitude might leave the Development Team in the dark, risking the accomplishment of the Sprint Goal.)

- **A PO clinging to tasks**: The Product Owner cannot let go Product Backlog items once they become part of the Sprint Backlog. For example, the Product Owner increases the scope of a requirement. 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 part of the Sprint Backlog, 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 core Scrum principles.)

- **Delaying PO**: The Product Owner does not accept items from the Sprint Backlog once those are finished. Instead, he or she waits until the end of the Sprint. (The Product Owner should immediately check tasks that meet the acceptance criteria. Otherwise, the Product Owner will create an artificial queue within the Sprint, which will unnecessarily 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 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 of how to save the sprint. Lastly, misusing the cancellation 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 a waste. In this case, the Product Owner should consider canceling 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 the customers and thus to the organization. This goal requires focused work to accomplish the work deemed necessary to meet the Sprint Goal 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 maximum number of tasks a development team can work on at the same time. Exceeding this WiP number results in creating additional queues that consequently reduce the overall throughput of the Development Team. The cycle time, which is the period between starting and finishing a ticket, measures this effect.)

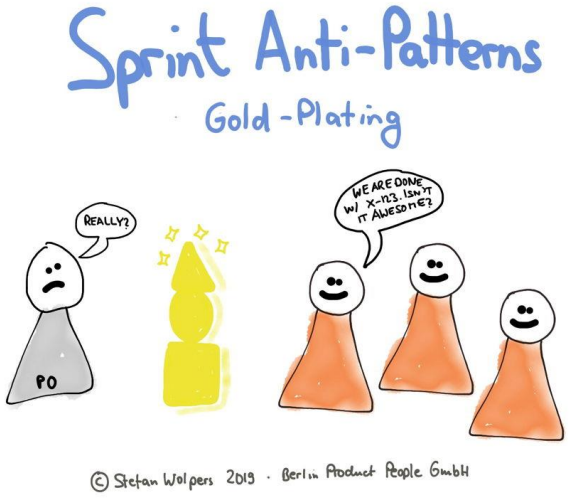
- **Cherry-picking:** The Development Team cherry-picks work. (This effect often overlays with the missing WiP issue. Human beings are motivated by short-term gratifications. It just feels good to solve yet another puzzle from the board, here: coding a new task. By comparison to this dopamine fix, checking how someone else solved another problem

during code review is less rewarding. Hence you often notice tickets queueing in the code-review-column, for example. It is also a sign that the Development Team is not yet fully self-organizing. Look also for Daily Scrum events that support this notion, and address the issue during the Sprint Retrospective.)

- **Board out-of-date:** The team does not update tickets on the Sprint board in time to reflect the current statuses. (The Sprint board, no matter if it is a physical or digital board, is not only vital for coordinating the Development 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 Development 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 for the return on investment the Development Team is creating—is unacceptable. This behavior also signals a substantial 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 Development Team is probably rather a group anyway.)

- **Gold-plating:** The Development Team increases the scope of the Sprint by adding unnecessary work to Product Backlog items of the Sprint Backlog. (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, creating a shared understanding from product vision down to the individual Product Backlog item, thus improving the trust level. If the Product Owner is not yet co-located with the Development Team, now would be the right moment to reconsider.)



Sprint Anti-patterns of the Scrum Master

- **Flow disruption:** The Scrum Master allows stakeholders to disrupt the flow of the Scrum Team during the Sprint. There are several possibilities for how stakeholders can interrupt the flow of the team during a sprint. Any of the examples will impede the team's productivity and might endanger the Sprint goal. 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. Particularly, he or she is not educating stakeholders on the negative impact of disruptions and how those may endanger the accomplishment of the Sprint goal.
    - The Scrum Master does not oppose line managers taking team members off the team assigning other tasks.
    - Lastly, the Scrum Master allows that either stakeholders or managers turn the Daily Scrum into a reporting session. This behavior will impede the Development Team's productivity by undermining its self-organization, reintroducing command & control through the backdoor.)

- **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 and offer his or her support. By the way, this is also the reason for marking tasks on a physical Sprint board with red dots each day if tasks do not move to the next column. (In other words: we are tracking the work item age.)

- **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:** 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.)

Note: I do not believe that it is the task of the Scrum Master to move tickets on a Sprint board. The Development Team members should do this during the Daily Scrum if they consider this to be helpful. 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 Development Team considers a burn-down chart helpful, the team members should also update the chart themselves. #justsaying #scrummasterisnotascribeorsecretary

Sprint Anti-patterns of the Scrum Team

- **The maverick & the Sprint Backlog:** Someone adds an item to the Sprint Backlog without consulting the Development Team first. (The control of the Sprint Backlog by the Development Team—in the sense of workload—is at the core of enabling the team to make a forecast. The scope is hence per se untouchable during the Sprint. Changes in 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 approval by the Development Team and probably a level of compensation. Another task of a similar size might need to go back to the Product Backlog. All these exceptions have in common that the Development Team has the final say collectively. No single teammate of the Scrum team can add or remove an item to or from the Sprint Backlog single-handedly.)

- **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. For that purpose, the Development Team creates a definition of "Done" to ensure that everyone understands the required quality level for a product Increment to be "shippable" to customers. Declaring buggy tasks "done" thus violates this core Scrum principle 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. Alternatively, the Development Team might have felt pressured to deliver to meet an (arbitrary) deadline, and they decided to cut corners by reducing quality. No matter the reason, in such a situation, there is plenty of work for the Scrum Master to accomplish.)

- **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 Scrum Team failed to create a shared understanding. There are plenty of reasons for this to happen, just to list a few:

    - 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 or customer interviews. There is a lack of understanding of the users' problems among 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 Scrum 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 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 over-extended. (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-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 a critical 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 hand.)

- **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 Scrum 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, rooted in utilization optimization at the team member level of the industrial paradigm. It also ignores the preferred team-building practice that Scrum Teams should select 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 the Scrum 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 specific tasks directly to engineers, thus bypassing the Product Owner and ignoring the Development Team's prerogative to self-organize. Alternatively, the manager removes an engineer from a team to work on such a task. (This behavior does not only violate core Scrum principles. It also indicates that the manager cannot let go of 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 for the Scrum Master 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 are 'serious bugs.' (Nice try #2. A special case is an "express lane" for bug fixes and other urgent issues. In my experience, every stakeholder will try and make his or her tasks eligible for that express lane.)

- **Disrupting the flow:** The stakeholders disrupt the flow of the Scrum Team. (See above, Scrum Master section.)

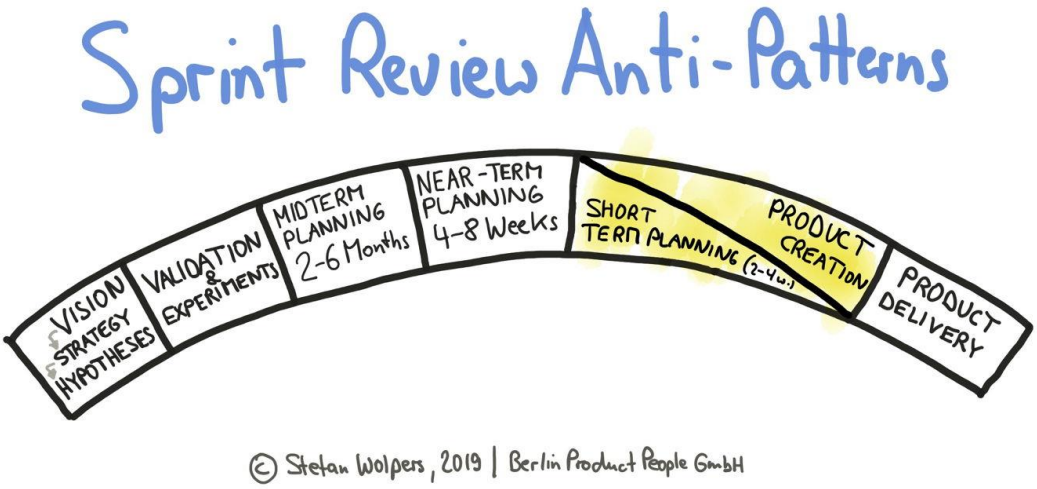
Conclusion

Although the Sprint itself is just a container for all other Scrum events, there are plenty of Sprint anti-patterns to observe. A lot of them are easy to fix by the Scrum Team or the Scrum Master. However, some sprint anti-patterns point at organizational issues that probably will require more than a Sprint Retrospective to change.

15 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 Empiricism at work: inspect the Product Increment and adapt the Product Backlog. 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, thus maximizing the value delivered to customers. 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 a Scrum Master as soon as possible.

What Does the Scrum Guide Say about the Sprint Review?

In contrast to other Scrum events, the Scrum Guide goes into detail regarding the Sprint Review. The Sprint Review includes the following elements (quote):

- Attendees include the Scrum Team and key stakeholders invited by the Product Owner;
- The Product Owner explains what Product Backlog items have been "Done" and what has not been "Done;"
- The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved;
- The Development Team demonstrates the work that it has "Done" and answers questions about the Increment;
- The Product Owner discusses the Product Backlog as it stands. He or she projects likely target and delivery dates based on progress to date (if needed);
- The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning;
- Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next; and,
- Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated releases of functionality or capability of the product.

"The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities."

Source: Scrum Guide 2017.

Sprint Review Anti-Patterns

Often, you can observe some of the following Sprint Review anti-patterns.

Sprint Review Anti-Patterns of the Product Owner

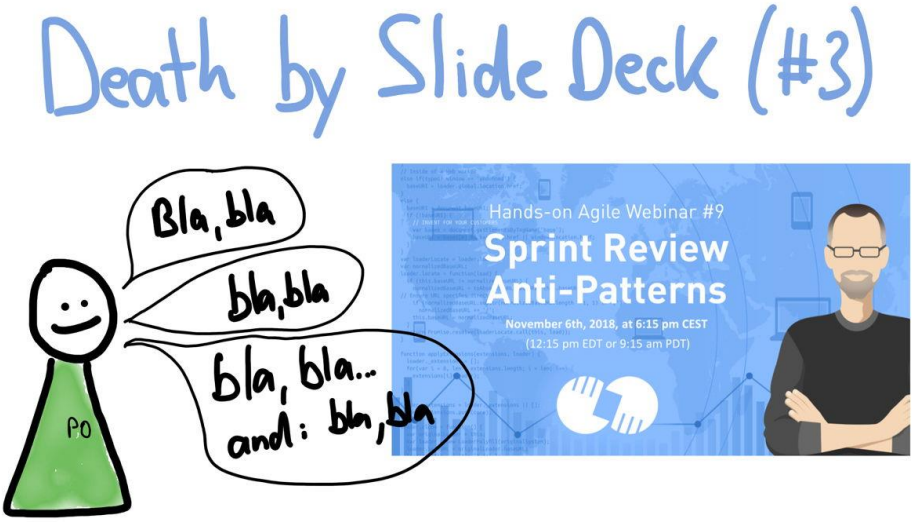
- **Selfish PO:** The Product Owner presents "his or her" accomplishments to the stakeholders. (Remember the old saying: There is no "I" in "team?")

- **"Acceptance" by the PO:** The Product Owner uses the Sprint Review to "accept" tasks/Product Backlog items. (An alignment — did the Development Team deliver the required functionality? — is useful and should be decoupled from the Sprint Review. The Product Owner should communicate with the Development Team when issues meet acceptance criteria.)

- **Unapproachable PO:** The Product Owner is not accepting feedback from stakeholders or the Development Team. (Such behavior violates the prime purpose of the Sprint Review event.)

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 discovery.)



- **Same faces again:** It is always the same folks from the Development Team who participate, not everyone. (Unless the organization scales Scrum based on LeSS or Nexus and we are talking about the overall Sprint Review, this Sprint Review Anti-pattern is a bad sign. To maximize the learning, the Sprint Review needs all Scrum Team members on deck. The challenge is that you cannot enforce the team's participation either, however. Instead, make it interesting enough that everyone wants to participate. If this is not happening, you should — as a Scrum Master — ask yourself how you have contributed to this situation.)

- **Side gigs:** The Development Team was working on issues outside the Sprint Goal, and the Product Owner learns about those for the first time during the Sprint Review.

- **Cheating:** The Development Team shows items that are not "done." (There is a good reason to show unfinished work on some occasions. Partially finished work, however, violates the concept of "Done," one of Scrum's first principles.)

Sprint Review Anti-Patterns of the Scrum Team

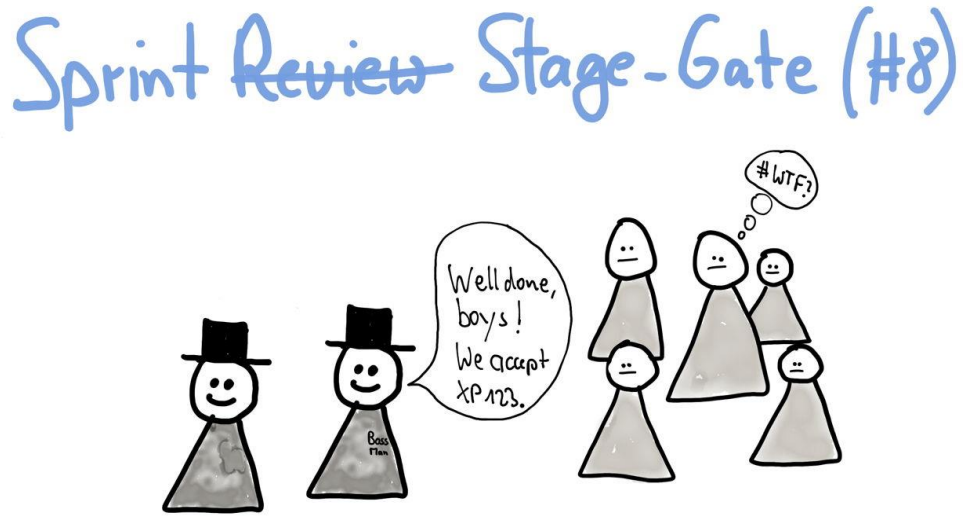
- **No Sprint Review**: There is no Sprint Review, as the Development Team did not meet the Sprint Goal. (A rookie mistake. Particularly in such a situation, a Sprint Review is necessary to create transparency.)

- **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, creating transparency and receiving 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. (My suggestion: Tell a compelling story at the beginning of the review to engage the stakeholders. Leave out those user stories that are probably not relevant to the story. Do not bore stakeholders by including everything that was accomplished. We are not accountants; the output is less relevant by comparison to the outcome.)

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 Sprint Review anti-pattern is typical for organizations that use an "agile"-waterfall hybrid. However, it is the prerogative of the Product Owner to decide what to ship when.)



- **No stakeholders:** Stakeholders do not attend the Sprint Review. (There are several reasons why stakeholders do not participate in the Sprint Review: they do not see any value in the event, or 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 event within the organization, at least in the beginning of using Scrum.)

- **No customers:** External stakeholders—also known as customers—do not attend the Sprint Review. (Break out of your organization's echo chamber, and invite some paying users to your Sprint Review.)

- **Starting over again:** There is no continuity in the attendance of stakeholders. (Longevity is not just beneficial at the team level, but also applies to stakeholder attendance. If they change too often, for example, because of a rotation scheme, their ability to provide in-depth feedback might be limited. If this pattern appears, the Scrum 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. Shift & Shareis an excellent Liberating Structure microstructure for that purpose.)

Conclusion

Scrum's Sprint Review is a critical Scrum event. It answers the question of whether the Scrum Team is still on track delivering the best possible value to the customers and the organization. Avoiding the before-mentioned Sprint Review's anti-patterns can hence significantly improve a Scrum Team's effectiveness.

21 Sprint Retrospective Anti-Patterns Impeding Scrum Teams

Introduction

What event could better embody Scrum's principle of empiricism than the Sprint Retrospective? I assume all peers agree that even the simplest retrospective—if only held regularly—is far more useful than having a fancy one once in a while, not to mention having none at all. Moreover, there is always room for improvement.

The Scrum Guide on the Sprint Retrospective

According to the Scrum Guide, the Sprint Retrospective serves the following purpose:

The purpose of the Sprint Retrospective is to:

Inspect how the last Sprint went with regards to people, relationships, process, and tools;
Identify and order the major items that went well and potential improvements; and,
Create a plan for implementing improvements to the way the Scrum Team does its work.

The Scrum Master encourages the Scrum Team to improve, within the Scrum process framework, its development process and practices to make it more effective and enjoyable for the next Sprint. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the definition of "Done", if appropriate and not in conflict with product or organizational standards.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

Source: Scrum Guide 2017.

By any standard, this is a compelling pitch as it addresses the why, the what, and the how of the retrospective.

Sprint Retrospective Anti-Patterns

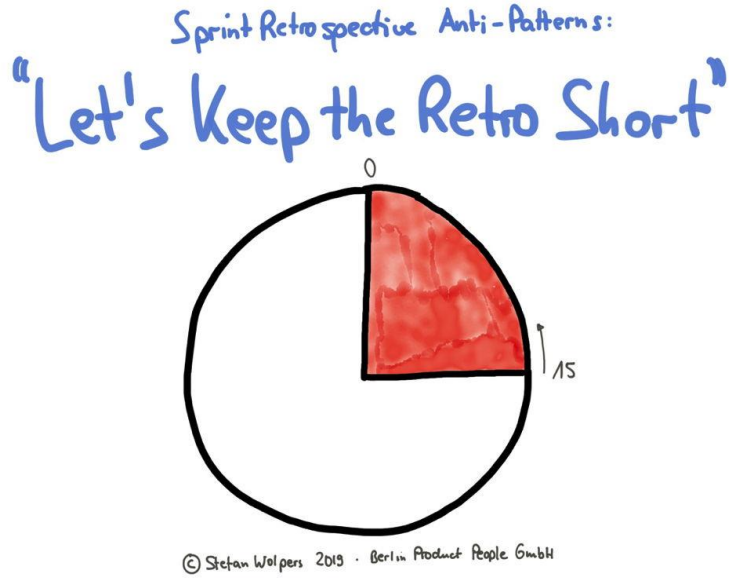
No matter the frequency of your retrospectives you should always watch out for the following Sprint Retrospective anti-patterns from the Scrum Team, the Development Team, the Scrum Master, as well as the organization:

Sprint Retrospective Anti-Patterns of the Scrum Team

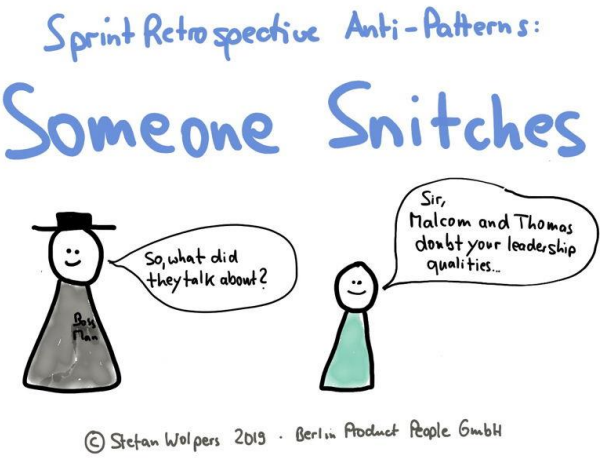
- **#NoRetro:** There is no retrospective as the team believes there is nothing to improve. (There is no such thing as an agile Nirwana where everything is just perfect. As people say: becoming agile is a journey, not a destination, and there is always something to improve.)

- **Dispensable buffer:** The team cancels retrospectives if more time is needed to accomplish the Sprint Goal. (The retrospective as a Sprint emergency reserve is a common sign of cargo cult Scrum. 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 a retrospective to achieve a Sprint Goal is a clear sign that the team does not understand basic principles, such as empiricism and continuous improvement. If the Scrum Team repeatedly does not meet the Sprint Goal, it should inspect what is going on here. Guess which Scrum event is designed for that purpose?)

- **Rushed retrospective**: The team is in a hurry and allocates much less than the necessary 60 to 180 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 sufficient time or consider stop having retrospectives at all. 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 rules 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—which 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 responsible instead.)

- **What improvement?** The team does not check the status of the action items from 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 folks still believe —for whatever reasons—that only the Development Team members 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 fails 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 force. My tip: Retromat's "Why are you here?" exercise is a good opener for a retrospective from time to time.)

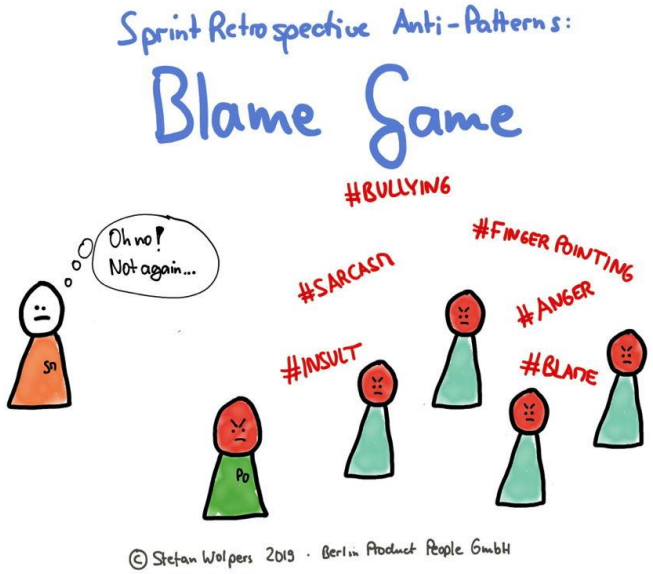
- **Groundhog day**: The retrospective never changes in composition, venue, or length. (There is a tendency in this case that the team will revisit the same issues over and over

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. Additionally, the Scrum Team is supposed to pick at least one important action item for the upcoming Sprint. This is why we have the retrospective before the Sprint Planning of the following Sprint. Postponing it into the next Sprint may interrupt the flow of the team and will probably leave one or more important team issues unattended for the length of a Sprint.)

- **#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 fails 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 less valuable. 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. Also, check out Liberating Structures' "Conversation Café" which addresses this issue with a simple protocol that fits well into retrospectives.)

- **Stakeholder alert**: Stakeholders participate in the retrospective. (There are several opportunities in Scrum that address the communication and information needs of stakeholders: the Sprint Review, the Daily Scrum, probably even the Product Backlog refinement, 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, consider having additional meetings if your team deems them necessary. 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 or likely lack of progress. Probably, the team members fear negative repercussions in the case of their absence, or you managed to hire a homogenous group of 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 team'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 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 among the worst Sprint Retrospective anti-patterns I can think of. 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 Scrum practices.

- **Let us see your minutes**: Someone from the organization—outside the Scrum 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.)

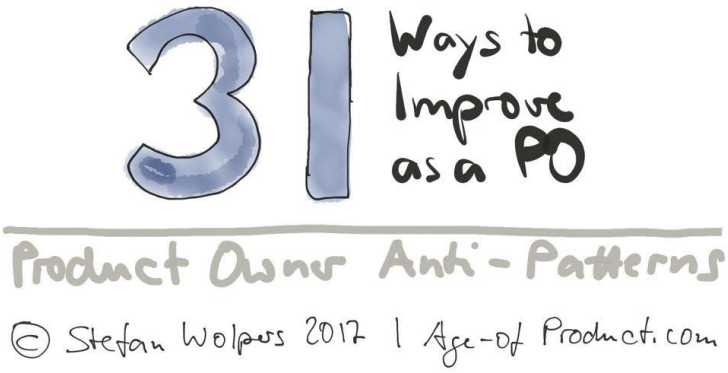
Conclusion

There are many ways in which a retrospective can be a failure even if it looks suitable 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 Role Anti-Patterns

Product Owner Anti-Patterns

If you are working as a Product Owner, there is—very likely—room for improvement. This list of some of the most common Product Owner anti-patterns might be a starting point. Hence, if you recognize some anti-patterns in your daily work, why don't you ask the rest of the Scrum Team for support? The Product Owner anti-patterns list is a good starting point for a retrospective.



The Role of the Product Owner According to the Scrum Guide

"The Product Owner is responsible for maximizing the value of the product resulting from work of the Development Team." The primary way of achieving this goal as a PO is the management of the Product Backlog. According to the Scrum Guide, this activity comprises:

- *Clearly expressing Product Backlog items;*

- *Ordering the items in the Product Backlog to best achieve goals and missions;*
- *Optimizing the value of the work the Development Team performs;*
- *Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,*
- *Ensuring the Development Team understands items in the Product Backlog to the level needed.*

Source: Scrum Guide 2017.

In my experience, most Product Owner anti-patterns result from a less than adequate handling of this Product Backlog management task.

Product Backlog and Refinement Anti-Patterns

You can spot most of the Product Owner anti-patterns in the PO's backyard — the Product Backlog and its refinement:

- **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 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.)

- **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.)

- **Prioritization by proxy:** A single stakeholder or a committee of stakeholder prioritize the Product Backlog. (The strength of Scrum is building on the strong position of the Product Owner. The PO is the only person 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.)
- **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:** 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 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.)

- **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 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 re-

searching them with a spike as a part of an iterative user story creation process.)

- **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).

Sprint Planning Anti-Patterns

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

- **What are we fighting for?** The Product Owner cannot align the business objective of the upcoming Sprint with the overall product vision. (A serious goal answers the "What are we fighting for?" question. To a certain extent, it is also a negotiation between the Product Owner and the Development Team. It is focused and measurable, as the Sprint goal—based on the business objective—and Development Team's forecast go hand in hand.)

- **No business objective, no Sprint Goal:** The Product Owner proposes Product Backlog items that resemble a random assortment of tasks, providing no cohesion. Consequently, the Scrum Team does not create a Sprint goal. (If this is the natural way of finishing your Sprint Planning, 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 ordering 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, though, remember the sunk cost fallacy.)

- **Last minute changes:** The Product Owner tries to squeeze in some last-minute Product Backlog items that are not ready yet. (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 tasks 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 ordering the Product backlog 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 is also a road to becoming a feature factory and deserves attention from the team's Scrum Master. It is violating the Development Team's prerogative to pick Product Backlog item for the Sprint Backlog as

well as Scrum Values.)

- **No preparation:** The Product Owner does not prepare the Product Backlog to provide useful Product Backlog items for selection by the Development Team. (Product Backlog needs to represent the best possible use of the Development Team's work from a customer value perspective at any given moment. In other words, your Scrum Team's Product Backlog has to be actionable 24/7. By my standards, that means that you need to be capable of running a meaningful Sprint Planning instantly. Preparing a few basic Product Backlog items an hour before the beginning of the Sprint Planning is not enough.)

Sprint Anti-Patterns

Another area prone to Product Owner anti-patterns is the sprint itself:

- **Absent PO**: The Product Owner is absent most of the Sprint and is not available to answer questions of the Development Team. (As the Sprint Backlog is emergent and new work may be identified as necessary to achieve the Sprint Goal, this attitude might leave the Development Team in the dark, risking the accomplishment of the Sprint Goal.)

- **PO clinging to tasks**: The Product Owner cannot let go Product Backlog items once they become part of the Sprint Backlog. For example, the Product Owner increases the scope of a requirement. 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 part of the Sprint Backlog, 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 core Scrum principles.)

- **Delaying PO**: The Product Owner does not accept items from the Sprint Backlog once those are finished. Instead, he or she waits until the end of the Sprint. (The Product Owner should immediately check tasks that meet the acceptance criteria. Otherwise, the Product Owner will create an artificial queue within the Sprint, which will unnecessarily 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 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 of how to save the sprint. Lastly, misusing the cancellation 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 a waste. In this case, the Product Owner should consider canceling the Sprint.)

PO Anti-Patterns during the Daily Scrum

By comparison to other Scrum events, the Daily Scrum is remarkably resilient to Product Owner anti-patterns:

- **Planning meeting:** The PO hijacks the Daily Scrum to discuss new requirements, to refine user stories, or to have a sort of micro (Sprint) planning meeting.

- **The talkative PO:** The Product Owner actively participate in the Daily Scrum. (POs and Stakeholders are supposed to listen in but not distract the Development Team members during their inspection and adaptation.)

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 Development Team and figure out collectively in what direction to take the product next, some Product Owners do not get the message:

- **Selfish PO**: The Product Owner presents "his or her" accomplishments to the stakeholders. (Remember the old saying: There is no "I" in "team?")

- **"Acceptance" by the PO**: The Product Owner uses the Sprint Review to "accept" tasks/Product Backlog items. (An alignment — did the Development Team deliver the required functionality? — is useful and should be decoupled from the Sprint Review. The Product Owner should communicate with the Development Team when issues meet acceptance criteria.)

- **Unapproachable PO**: The Product Owner is not accepting feedback from stakeholders or the Development Team. (Such behavior violates the prime purpose of the Sprint Review event.)

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 the Product Owner role. The list of Product Owner anti-patterns above may be a starting point.

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 post on Scrum anti-patterns how you can identify if your Scrum Master needs support from the team.

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 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 standard that an individual is not meeting. (See also the detailed lists of services rendered to the Product Owner, the Development Team, and the organization by the Scrum Master as defined by the Scrum Guide.)

Possible Reasons Why Scrum Masters Leave the 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 Scrum 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 Scrum cannot be pushed but needs to be pulled—that's the essence of self-organization.

- **Dogmatism**: Some Scrum Masters believe in applying the Scrum Guide literally which unavoidably will cause friction as Scrum is a framework, not a methodology.

- **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, probably 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 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. How hard can this probably be—the manual is merely 18 pages?" This conviction will bring out his or her true colors over time 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 its ship around the team will probably lose its Scrum Master. Note, that the Scrum Master cannot solve this issue by herself or himself. This is an effort of the whole Scrum Team.

- **The tactical Scrum Master**: These Scrum Masters drank HR's cool-aid that Scrum Master is a position, not a role. Moreover, there is a career path from Junior Scrum Master to VP of Agile Coaching. Consequently, they constrain their work strictly to the Scrum Team level until being promoted.

- **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 learning effort. Remember, it is a journey, not a destination, and you do not travel alone.

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, thus providing an appropriate level of business agility to the organization. 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 Master 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 neither a supervisor or a dispatcher.

- **Running meetings by 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.

- **Keeping the Scrum team dependent:** In this scenario, the Scrum Master pampers the team to a level that keeps the team dependent on his or her services: organizing meetings, purchasing stickies and sharpies, taking notes, updating Jira—you get the idea of this service level. More critical, however, is when the Scrum Master decides to keep the team in the dark about principles and practices to secure his or her job. This behavior is only a small step away from the dark side.

- **Pursuing flawed or dangerous metrics:** While utilizing predefined Jira reports is probably a sign of ignorance or laziness, keeping track of individual performance metrics—such as story points per developer per Sprint and reporting those to that person's line manager—is a critical warning sign. That is a classic supervisor hack to reintroduce command & control through the back door. It inevitably leads to cargo-cult Scrum.

- **Escalating under-performance:** During the Sprint, the Scrum Master reports to the management whether the Scrum team will meet the current forecast or not. 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."

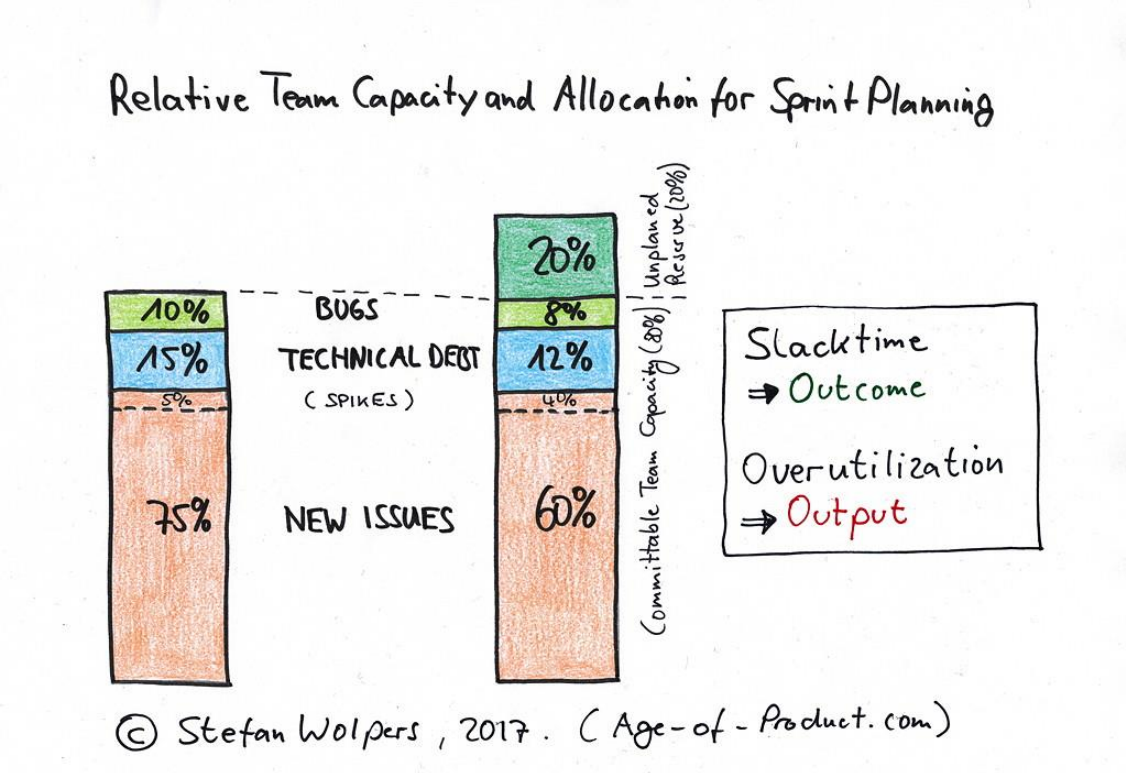
- **Focusing on team harmony:** The Scrum Master sweeps conflict and problems under the rug by not using Sprint Retrospectives to address those openly. This behavior is often a sign of bowing to politics and instead of using manipulation to meet organizational requirements that are opposing Scrum values and principles. If the organization values its underlings for following the 'rules' instead of speaking the truth why would you run Retrospectives in the first place? A 'Scrum Master' participating in cargo-cult Scrum is again a supervisor than an agile practitioner.)

Scrum Master Anti-Patterns by Scrum Events

The Sprint Planning

The following anti-patterns focus on the sprint planning:

- **100% utilization:** The Development Team regularly bows to the hard pushing Product Owner—"last Sprint, you delivered 25 story points, and now you are choosing only 17?"—and accepts more issues into the Sprint Backlog than it can stomach without the Scrum Master's invention. He or she does not address that this is a disrespectful behavior, negating the Development Team's prerogative of self-organization and ignoring its need for slack time. (The 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, or supporting each other. A level of 100% utilization always reduces the team's long-term productivity; it is classic taylorist line management thinking.) If at the end of a Sprint 50% of all issues spill over to the next Sprint and this is 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 to improve your customers' life. Perhaps, Kanban would be a good choice.



- **Unrefined Sprint Backlog items:** The Scrum Master does not address the acceptance of "unready" Product Backlog issues into the Sprint Backlog. This is a pretty sure way that the Development Team will not deliver the Sprint goal, rendering a core Scrum principle useless: providing a valuable, potentially shippable Product Increment at the end of the Sprint. (This refers to regular work, not emergencies.)

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 for how stakeholders can interrupt the flow of the team during a sprint. Any of the examples will impede the team's productivity and might endanger the Sprint goal. 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. Particularly, he or she is not educating stakeholders on the negative impact of disruptions and how those may endanger the accomplishment of the Sprint goal.
    - 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 priorities by the Product Owner.
- Lastly, the Scrum Master allows either stakeholders or managers to turn the Daily Scrum into a reporting session.

- **Assigning sub-tasks to developers:** The Scrum Master does not prevent the Product Owner—or anyone else—from assigning tasks directly to members of the Development Team. (It organizes itself without external intervention. And the Scrum Master is the shield of the team in that respect.)

- **Defining technical solutions:** An engineer turned Scrum Master is now 'suggesting' how the Development Team is implementing issues. (Alternatively, the Product Owner or an outsider is pursuing the same approach, for example, a technical lead.)

- **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 and offer his or her support. By the way, this is also the reason for marking tasks on a physical Sprint board with red dots each day if tasks do not move to the next column. (In other words: we are tracking the work item age.)

The Retrospective

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

- **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.)

- **Groundhog day:** The Retrospective never changes in composition, venue, or length. There is a tendency in this case that the Scrum team might revisit the same issues over and over again–it's groundhog day without the happy ending, though.

- **Let's have the retro 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. Additionally, the Scrum Team is supposed to pick at least one important action item for the upcoming Sprint. This is why we have the Retrospective before the Sprint Planning of the following Sprint. Postponing it into the next Sprint may interrupt the flow of the team and will probably leave one or more important team issues unattended for the length of a Sprint.)

- **#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.)

- **#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 fails 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 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.

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.

Development Team Anti-Patterns

Introduction

After covering the Scrum Master and the Product Owner, this article addresses Development Team anti-patterns, covering all Scrum Events as well as the Product Backlog artifact. Learn more about what to look out for if you want to support your fellow teammates.

The Role of the Development Team in Scrum

According to the Scrum Guide, the Development Team "consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. Only members of the Development Team create the Increment. Development Teams are structured and empowered by the organization to organize and manage their own work."

Development Teams are hence essential to Scrum's built-in checks & balances as the Development Team has the sole control over the Sprint Backlog and is watching over the Definition of Done. Generally, Development Teams need to have the following characteristics to be successful:

- *They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;*
- *Development Teams are cross-functional, with all the skills as a team necessary to create a product Increment;*

- *Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person;*
- *Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations, or business analysis; and,*
- *Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.*
-

Source: Scrum Guide 2017.

(Check out the complete Scrum Guide 2017 list on the Development Team by downloading the Scrum Guide Reordered.)

While the direction from the Scrum Guide sounds straight forward, in practice, some people get confused that the role "Development Team" is assigned to a group of people, thus creating a sub-team within the larger Scrum Team, while the roles of Scrum Master and Product Owner are assigned to individuals. Even more confusing might be the situation, when both Scrum Master and Product Owner are also actively creating the Product Increment, resulting in the Development Team being identical with the Scrum Team. Finally, the term Development Team seems to limit the role to technical people, for example, software engineers.

However, in my experience, given proper support by the Scrum Master, even lawyers and marketers can get comfortable with the designation "developer" when utilizing Scrum for their purposes. So, let's dive into an array of common Development Team anti-patterns that signal Scrum Masters that their team needs support.

Development Team Anti-Patterns by Scrum Event

The following list of Development Team anti-patterns addresses four Scrum events plus the Sprint itself:

Sprint Planning Anti-Patterns of the Development Team

- **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 events as practices such as Product Backlog refinement, 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 about 20 % of resources are well-spent every Sprint to fix bugs and refactor the codebase. If the Product Owner ignores the need for this work, and the Development Team accepts this attitude, the Scrum Team will find itself in a downward spiral, turning slowly but steadily into an

output-focused feature factory. Its future product delivery capability will decrease. Read more on technical debt and Scrum.)

- **No slack time:** 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 do pair programming, for example. 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 members. Overutilization will always push the individual team member into a less collaborative mindset, impeding self-organization. On the other side, slack time will allow the Scrum Team to act collaboratively and focus on the outcome.)

- **Planning too detailed:** During the Sprint Planning, the Development Team plans every single task of the upcoming Sprint in advance. (Don't become too granular. One-quarter of the tasks are more than sufficient to not just start with the Sprint, but also start learning. The Sprint Backlog is emergent, and 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 planning altogether. (Skipping planning is unfortunate, as it is also an excellent opportunity to talk about how to spread knowledge within the Development Team, where the architecture is heading, or whether tools are adequate. For example, the team might also consider who will be pairing with whom on what task. The Development Team planning part is also well-suited to consider how to reduce technical debt, see above.)

- **Team leads?** The Development Team does not come up with a plan to deliver on its forecast collaboratively. Instead, a 'team lead' does all the heavy lifting and probably even 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 Anti-patterns

Most of the following Development Team anti-patterns result from a lack of focus or procrastination:

- **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 the customers and thus to

the organization. This goal requires focused work to accomplish the work deemed necessary to meet the Sprint Goal 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 maximum number of tasks a development team can work on at the same time. Exceeding this WiP number results in creating additional queues that consequently reduce the overall throughput of the Development Team. The cycle time, which is the period between starting and finishing a ticket, measures this effect.)

- **Cherry-picking:** The Development 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 queueing in the code-review-column, for example. It is also a sign that the Development Team is not yet fully self-organizing. Look also for Daily Scrum events that support this notion, and address the issue during the Sprint Retrospective.)

- **Board out-of-date:** The team does not update tickets on the Sprint board in time to reflect the current statuses. (The Sprint board, no matter if it is a physical or digital board, is not only vital for coordinating the Development 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 Development 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 for the return on investment the Development Team is creating—is unacceptable. This behavior also signals a substantial 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 Development Team is probably rather a group anyway.)

- **Gold-plating:** The Development Team increases the scope of the Sprint by adding unnecessary work to Product Backlog items of the Sprint Backlog. (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, creating a shared understanding from product vision down to the individual Product Backlog item, thus improving the trust level. If the Product Owner is not yet co-located with the Development Team, now would be the right moment to reconsider.)

Anti-Patterns of the Daily Scrum

The Daily Scrum is the key "inspect & adapt event" for the Development Team. If the Daily Scrum is not working, at least in my experience, don't expect the Development Team to meet the Sprint Goal. Common Development Team anti-patterns of the Daily Scrum include:

- **No routine:** The Daily Scrum 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 the Daily Scrum. Think of it as a spontaneous drill: don't put too much thought into the stand-up, just do it. Skipping Daily Scrums can turn out to be a slippery slope: if you skip one Daily Scrums or two, why not skip every second one?)

- **Status report:** The Daily Scrum is a status report meeting, and Development Team members are waiting in line to "report" progress to the Scrum Master, the Product Owner, or maybe even a stakeholder.

- **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.")

- **Problem-solving:** Discussions are triggered to solve problems, instead of parking those so they can be addressed after the Daily Scrum.

- **Planning meeting:** The Development Team hijacks the Daily Scrum to discuss new requirements, to refine user stories, or to have a sort of (sprint) planning meeting.

- **Orientation lost:** The Daily Scrum serves one purpose as it answers a simple question: Are we still on track to meet the Sprint Goal? Or do we need to adapt the plan or the Sprint Backlog or both? Often, the Development Team cannot answer that question immediately. (In that respect, visualizing the progress towards the Sprint Goal is a useful exercise. Removing the Development Team task of maintaining a mandatory burndown chart from the Scrum Guide a few years ago does not imply that a burndown chart is obsolete.)

- **No use of work-item age:** A Development team member experiences difficulties in accomplishing an issue over several consecutive days and nobody is offering help. (Often, this result is a sign that people either may not trust each other or do not care for each other. Alternatively, the workload of the Development Team has reached an unproductive level as they no longer can support each other. Note: Of course, the Scrum Guide does not mention the 'work item age.' However, it has proven to be a useful practice.)

- **Cluelessness:** Team members are not prepared for the Daily Scrum. ("I was doing some stuff, but I cannot remember what. Was important, though.")

- **Excessive feedback:** Team members criticize other team members right away sparking a discussion instead of taking their critique outside the Daily Scrum.

Development Team Anti-Patterns: Sprint Review

- **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 discovery.)

- **Same faces again**: It is always the same folks from the Development Team who participate, not everyone. (Unless the organization scales Scrum based on LeSS or Nexus and we are talking about the overall Sprint Review, this Sprint Review Anti-pattern is a bad sign. To maximize the learning, the Sprint Review needs all Scrum Team members on deck. The challenge is that you cannot enforce the team's participation either, however. Instead, make it interesting enough that everyone wants to participate. If this is not happening, you should — as a Scrum Master — ask yourself how you have contributed to this situation.)

- **Cheating**: The Development Team shows items that are not "done." (There is a good reason to show unfinished work on some occasions. Partially finished work, however, violates the concept of "Done," one of Scrum's first principles.)

- **No Sprint Review**: There is no Sprint Review, as the Development Team did not meet the Sprint Goal. (A rookie mistake. Particularly in such a situation, a Sprint Review is necessary to create transparency.)

Sprint Retrospective Anti-Patterns of the Development Team

- **#NoRetro**: There is no retrospective as the Development 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 Goal. (The retrospective as a Sprint emergency reserve is a common sign of cargo cult Scrum. 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 a retrospective to achieve a Sprint Goal is a clear sign that the team does not understand basic principles, such as empiricism and continuous improvement. If the Scrum Team repeatedly does not meet the Sprint Goal, it should inspect what is going on here. Guess which Scrum event is designed for that purpose?)

- **Extensive whining**: The Development 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.)

- **Product Owner non-grata**: The Product Owner is not welcome to the retrospective. (Some folks still believe —for whatever reasons—that only the Development Team members 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 fails together. How is that supposed to work without the Product Owner?)

Anti-Patterns at the Product Backlog Level

- **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 Development 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.)

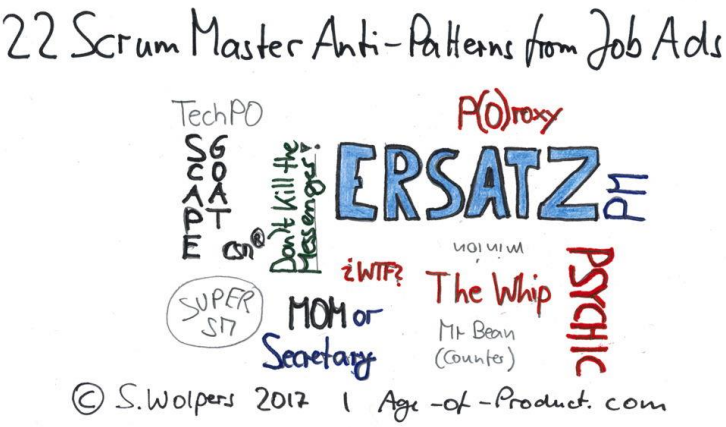
- **Submissive team:** The Development Team submissively follows the demands of the Product Owner. (Challenging the Product Owner whether his or her selection of issues is the best use of the Development Team's time is the noblest obligation of every team member: why shall we do this?)

Development Team Anti-Patterns — Conclusion

Given the essential role the Development Team has to live up to make Scrum successful, it is no surprise that there are plenty of Development Team anti-patterns to observe. The good news is, though, that many of those are entirely under control of the Scrum Team. All it takes to tackle these anti-patterns for the team is starting to inspect and adapt. Why not use your next retrospective for this?

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.



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 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 on 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 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?)

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 sand 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.

Conclusion

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.

Scrum Stakeholder Anti-Patterns

The Stakeholder and Organizational Excellence in Legacy Organizations

Regularly, InfoQ applies the 'Crossing the Chasm' metaphor to engineering practices, thus covering a part of the agile movement to create learning organizations. Its recent 'Culture & Methods – the State of Practice in 2019' edition found that new converts to Scrum, for example, will recruit themselves mostly from the late majority and laggards. (The early majority of organizations are already adopting BDD/DDD or Pragmatic Agility.)

Those laggards — or legacy organizations — are easy to spot: Some form of applied Taylorism, usually a strict hierarchy to command & control functional silos with limited autonomy, made it into the postindustrial era. Often, these organizations were once created to train farm boys into assembly-line workers within a standardized industrial process churning out standardized products in the name of output optimization. Human beings became cogs in the machinery, rewarded for functioning well without asking questions. Too bad, when nowadays diversity, autonomy, mastery, and purpose become the driving factors in a highly competitive environment where more of the same for everyone is no longer creating value.

The conflict at the stakeholder level in such legacy organizations is apparent: mostly, the stakeholder is a manager of a functional silo with objectives that do not necessarily align with those of a product or Scrum Team. Where the organization needs to morph into a kind of 'team of teams' structure with a shared understanding of purpose and direction as well as the need to create value for the customers at heart, the reality of a legacy organization attempting to become agile is often very different. For managers, it means moving:

- From WIIFM (what-is-in-for-me syndrome) to team playing — the team wins, the team loses,
- From career planning as an individual to servant leadership in a team of teams structure,
- From knowing it all and being the go-to person to solve problems to trusting those closest to the problem to come up with a solution,
- From 'failure is no option' to embracing failure as a means to learn effectively,
- From claiming success as a personal contribution to stepping back and letting the responsible team shine.

Abandoning yesterday's game – and probably its symbols of power, too — and accepting that an agile transition may provide job security, but most certainly not role security is a monumental undertaking for the majority of the management of a legacy organization. Probably, many of these managers will not adapt and even quit the organization sooner or later.

Common Scrum Stakeholder Anti-Patterns

After defining the context, let us consider some Scrum stakeholder anti-patterns in detail. Most often, Scrum stakeholder anti-patterns result from a training and coaching void accompanied by not changing individual career objectives. Thus, they manifest themselves in the continued pursuit of local optima or personal agendas. In both situations, the incentive structure of an organization most likely still fosters a predictable behavior that contradicts the organization's goals at a system level.

Charlie Munger: "Never, ever, think about something else when you should be thinking about the power of incentives."

The following list of Scrum stakeholder anti-patterns addresses Scrum events, system-related issues as well as issues of individual players.

Scrum Stakeholder Anti-Patterns at Scrum Event Level

The Sprint

Anti-patterns of this sort point at stakeholders' ignorance of the core idea of Scrum — self-organizing teams:

- **Pitching developers:** The stakeholders try to sneak in small tasks by pitching them directly to developers, bypassing the Product Owner. (Nice try #1.)

- **Everything's a bug:** The stakeholders try to speed up delivery by relabeling their tasks are 'serious bugs.' (Nice try #2. A special case is an "express lane" for bug fixes and other urgent issues. In my experience, every stakeholder will try and make his or her tasks eligible for that express lane.)

- **Flow disruption:** The Scrum Master allows stakeholders to disrupt the flow of the Scrum Team during the Sprint. There are several possibilities for how stakeholders can interrupt the flow of the team during a sprint. Any of the examples will impede the team's productivity and might endanger the Sprint goal. 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. Particularly, he or she is not educating stakeholders on the negative impact of disruptions and how those may endanger the accomplishment of the Sprint goal.
    - 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 priorities by the Product Owner.
    - Lastly, the Scrum Master allows either stakeholders or managers to turn the Daily Scrum into a reporting session.

Product Backlog and Refinement Anti-Patterns

These stakeholder anti-patterns result from ignoring the role of the Product Owner, turning him or her into a mere scribe. Two important anti-patterns of this kind are:

- **Requirements handed down:** 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: Product Backlog item creation is a Scrum Team exercise.)

- **Prioritization by proxy:** A single stakeholder or a committee of stakeholder prioritize the Product Backlog. (The strength of Scrum is building on the strong position of the Product Owner. The PO is the only person to decide what tasks become Product Backlog items. Moreover, the Product Owner also decides on the ordering of the Product Backlog. Take away that empowerment, and Scrum turns into a pretty robust waterfall 2.0 process.)

The Daily Scrum

Most anti-patterns in this category result from perceived information needs — think of them as withdrawal symptoms:

- **Status report:** The Daily Scrum is a status report meeting, and Development Team members are waiting in line to "report" progress to a stakeholder.
- **Talkative chickens:** "Chickens" actively participate in the Daily Scrum. (Stakeholders are supposed to listen in but not distract the Development Team members during their inspection.)
- **Command & control by the management:** Line managers are attending the Daily Scrum to gather "performance data" on individual team members. (This behavior is defying the very purpose of self-organizing teams.)
- **"A word, please":** Stakeholders are waiting until the Daily Scrum is over and then reach out to individual Development Team members for specific reporting from them. (Nice try. However, this hack is also unwanted behavior and distracts the Development Team.)
- **Direct assignment of tasks:** A stakeholder assigns tasks directly to a Development Team member.

Sprint Planning Anti-patterns of Stakeholders

Forecast imposed: The Sprint forecast 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 forecast. Or a stakeholder points at the team's previous velocity demanding to take on more user stories. ("We need to fill the free capacity.") Or the 'tech lead' of the Development Team is making a forecast on behalf of the Development Team.)

The Sprint Review

Again, this category is often a combination of ignorance, fighting a perceived loss of control or pulling rank to override scrum principles:

- **Scrum à la stage-gate®**: The Sprint Review is a kind of stage-gate® approval process where stakeholders sign off features. (This Sprint Review anti-pattern is typical for organizations that use an "agile"-waterfall hybrid. However, it is the prerogative of the Product Owner to decide what to ship when.)
- **No stakeholders**: Stakeholders do not attend the Sprint Review. (There are several reasons why stakeholders do not participate in the Sprint Review: they do not see any value

in the event, or 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 event within the organization, at least in the beginning of using Scrum.)

- **No customers**: External stakeholders—also known as customers—do not attend the Sprint Review. (Break out of your organization's echo chamber, and invite some paying users to your Sprint Review.)

- **Starting over again**: There is no continuity in the attendance of stakeholders. (Longevity is not just beneficial at the team level, but also applies to stakeholder attendance. If they change too often, for example, because of a rotation scheme, their ability to provide in-depth feedback might be limited. If this pattern appears, the Scrum 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. Shift & Share is an excellent Liberating Structure microstructure for that purpose.)

The Sprint Retrospective

Here, it is mainly about control and line management issues:

- **Stakeholder alert:** Stakeholders participate in the Retrospective. (There are several opportunities in Scrum that address the communication and information needs of stakeholders: the Sprint Review, the Daily Scrum, probably even the Product Backlog refinement, 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, consider having additional meetings if your team deems them necessary. However, the Retrospective is off-limits to stakeholders.)

- **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 to ensure that Scrum Team members will also point at critical issues in the future.)

Scrum Stakeholder Anti-Patterns at System Level

These anti-patterns result mainly from a half-hearted approach to becoming an agile organization. Typically, it ends in the form of cargo-cult agile:

- **Lack of transparency**: The organization is not transparent about vision and strategy hence the Scrum Teams are hindered to become self-organizing.

- **Lack of leadership**: Senior management is not participating in agile processes, for example, the Sprint Reviews, despite being a role model. Instead, they do expect a different form of (push) reporting.

- **Cargo-cult agile by cherry picking**: Agile processes are either bent or ignored whenever it seems appropriate, for example, the Product Owner role is reduced to a project manager role. Or stakeholders are bypassing the Product o Owner to get things done and get away with it in the eyes of the senior management, as they would show initiative. There is a lack of discipline to support the agile transition.

- **Agile on a tight budget**: The organization does not spend enough time and budget on proper communication, training, and coaching to create a shared understanding of purpose and direction among all members of the organization.

- **Telling people how to do things**: In the good old times on the shop floor, it was a valuable trait to train newcomers or workgroups in the art of assembling a Model T — as the manager probably did herself. Nowadays, as we invest most of our time building products that have never been built before this attitude becomes a liability. Just let the people closest to the job at hand figure out how to do this. Guidance by objectives and providing support when requested or needed will be appreciated, though.

- **Steering meetings**: Unimpressed by the agile ways of working, the manager insists on continuing the bi-weekly steering meetings to ensure that the team will deliver all her requirements in time. This one has a quick remedy, though: just do not participate in meetings that have no value for the team.

- **Limited to non-existing feedback loops**: The sales organization and other functional silos guard the direct access to customers, thus preventing the product teams from learning.

Sprint Anti-patterns of the IT Management

Also, there are some typical anti-patterns of those stakeholders closest to the Scrum Teams — the IT management:

- **All hands to the pumps w/o Scrum:** The management temporarily abandons Scrum in a critical 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 hand.)

- **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 Scrum 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, rooted in utilization optimization at the team member level of the industrial paradigm. It also ignores the preferred team-building practice that Scrum Teams should select 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 the Scrum 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 specific tasks directly to engineers, thus bypassing the Product Owner and ignoring the Development Team's prerogative to self-organize. Alternatively, the manager removes an engineer from a team to work on such a task. (This behavior does not only violate core Scrum principles. It also indicates that the manager cannot let go of 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 for the Scrum Master from a higher management level to deal with.)

Personally Motivated Scrum Stakeholder Anti-Patterns

There are numerous ways of how stakeholders can impede the progress of a product team. Four of the most common ones are as follows:

- **The 'My budget' syndrome**: Stakeholders do not compete for a Scrum Team capacity but claim that they allocate "their" budget on feature requests as they see fit. (That process leads to the creation of local optima at a silo or departmental level. The effect can be observed particularly in organizations, that tie additional benefits to individuals. Instead, resources need to be allocated in the spirit of optimization for the whole organization. **Note**: 'Pet projects' also fall in this category.)

- **'We know what to build'**: The is no user research, nor any other interactions of the product delivery organization with customers. (There are several reasons causing this phenomenon ranging from a founder or entrepreneur who pursues his or her product vision without engaging in customer discovery activities. Or the product delivery organization is solely briefed indirectly by key account managers. Probably, the sales department deems a direct contact of Scrum Team members with customers too risky and hence prevents it from happening. What these patterns share is either a bias that is hurting the learning effort or a personal agenda. While the former can be overcome by education, the latter is more difficult to come by as the culprits typically reject the idea that they are guided by selfish motives. For becoming an effective product delivery organization it is essential that the team directly communicates with customers at a regular base.)

- **Selling non-existing features**: What features do you need us to provide to close the deal? Sales managers chase sales objectives by asking prospects for a feature wish-list and pro-

vide those to the product delivery organization as requirements. (The problem with customers is that they usually lack the depth of knowledge required to provide useful answers to this question. Most of the time, they also lack the level of abstract thinking necessary to come up with a viable, usable, and feasible solution. As the saying goes: if the only tool you are familiar with is a hammer every problem will look like a nail. Pursuing the sales process in such a way will lead the product into a feature comparison race to the bottom, probably inspired by bonuses and personal agendas. This is the reason why product-people like to observe customers in their typical environment using a product to avoid misallocating resources on agenda-driven features. At a systems level, reconsidering individual monetary incentives for salespeople is helpful, too. In a learning organization, teams win not individuals.)

- **Bonus in limbo**: We are nearing the end of a quarter. Bonus relevant KPIs (key performance indicators) are at risk of not being met. The responsible entity demands product changes or extensions in the hope that those will spur additional sales. (This behavior is comparable with the 'what features do you need to close the deal' anti-pattern, but it demanded in more pressing fashion, typically four weeks before the end of a bonus period.)

- **Financial incentives to innovate**: The organization incentivizes new ideas and suggestions monetarily. (Contributing to the long list of ideas and thus the hypotheses short-list should be intrinsically motived. Any possible personal gain might inflate the number of suggestions without adding value.)

For more information watch the webinar: Product Discovery Anti-patterns.

Conclusion

There are a lot of different reasons why Scrum stakeholders do not act as expected. Some result from organizational debt, particularly in legacy organizations from the industrial area. Some are intrinsically motivated, for example, by personal agendas, while others originate from a lack of training or anxieties. Whatever the reason, though, Scrum stakeholder anti-patterns need to be overcome to turn an agile transition into a success. Otherwise, you might end up in some form of cargo-cult agile or Scrumbut.

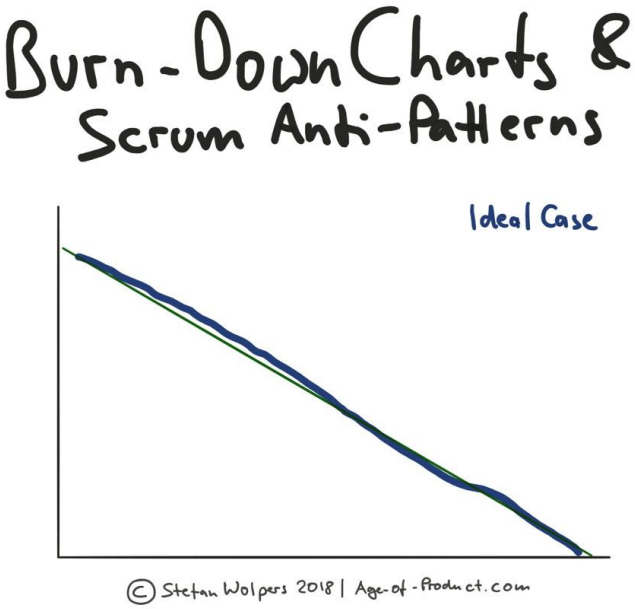
How to Detect Scrum Anti-Patterns

Use Burn-Down Charts to Discover Scrum Anti-Patterns

Introduction

A burn-down chart tracks the progress of a team toward a goal by visualizing the remaining work in comparison to the available time. So far, so good. More interesting than reporting a status, however, is the fact that burn-down charts also visualize scrum anti-patterns of a team or its organization.

Learn more about discovering these anti-patterns that can range from systemic issues like queues outside a team's sphere of influence and other organizational debt to a team's fluency in agile practices.



Scrum Anti-Patterns Visualized by Burn-Down Charts

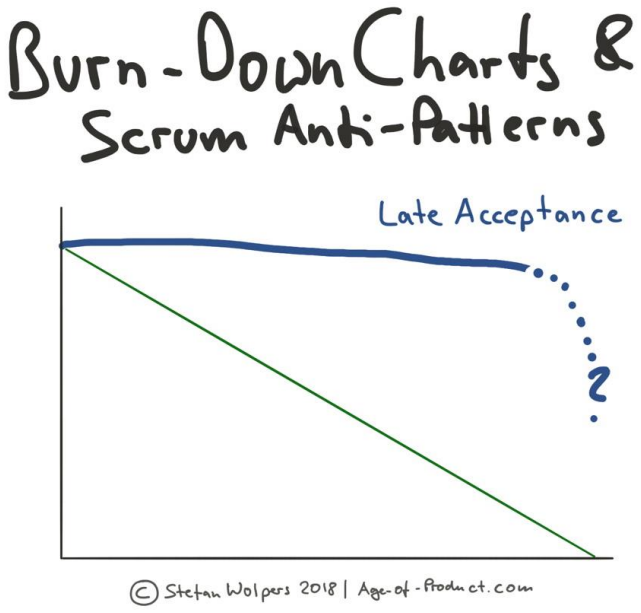
Burn-down charts have become popular to provide team members as well as stakeholders with an easy to understand status whether a sprint goal will be accomplished. (Critics of the burn-down chart may note, though, that a scrum team should have a gut feeling anyway whether the sprint goal is achievable.)

Hence, this post is focusing on another useful aspect of burn-down charts: they are equally well suited to provide additional insights into all kind of impediments, both at a team level and at an organizational level.

The following graphs visualize four of the typical anti-patterns that can be easily detected with burn-down charts:

1. Late Acceptance

The product owner accepts or rejects tasks only late in the sprint:



This behavior may be rooted in various issues, for example:

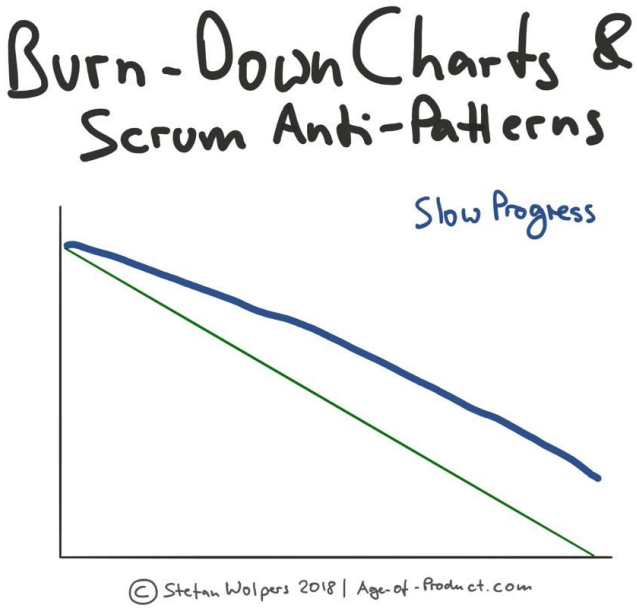
- **The absent product owner**: The product owner is rarely available for the team to clarify matters and accept work. This is creating an artificial queue that has a diminishing effect on the team's ability to deliver value by delaying the necessary clarification of tasks or the shipment of tasks themselves. (Note: LeSS susceptible for this effect when the product owner when not willing to delegate responsibility.)

- **The proxy product owner**: The team is working in a remote setup and the product owner is not onsite with the rest of the team. (Note: A proxy product owner is usually not a solution as he or she will just increase the time for feedback and add to the communication problems.)

- **Consequences**: There will likely be a spill-over to the next sprint as the feedback loop does not provide enough time to fix issues during the sprint. The team will probably not meet the sprint goal. If this not an isolated incident but a persistent pattern, action needs to be taken.

2. Slow Progress

In this case, the graph is located above the line of the expected progress for the complete sprint length:



There are several reasons why this might be the case:

- **The ambitious team**: The sprint goal is too ambitious and the team realizes only during the sprint that it will not deliver the sprint goal. (Note: It is okay to aim high and fail, however, it should not be the regular pattern as it is negatively influencing the trust of the organization in the team.)

- **The submissive team**: The sprint goal is too ambitious from an engineering perspective. However, instead of speaking up, the team tries to make it happen thus failing at the end of the sprint.

- **Capacity issues**: The capacity of the team changes after the sprint starts, for example, team members get sick, or they give notice and leave the team. (Note: Admittedly, this is hardly plannable anyway.)

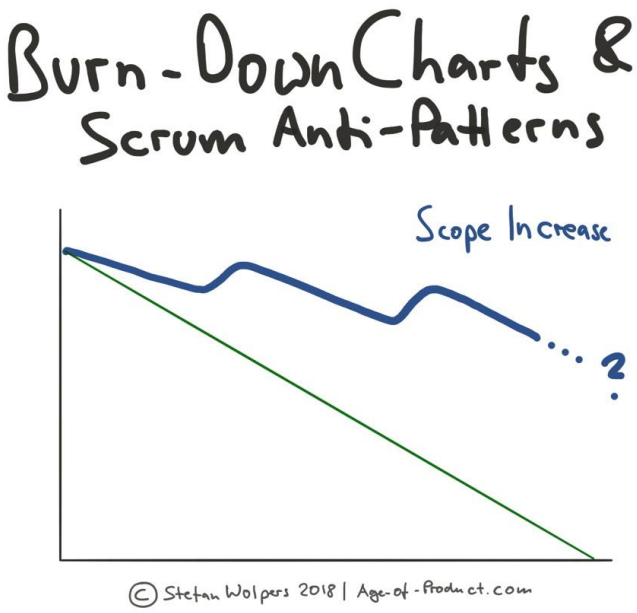
- **Change of priorities**: The team needs to address a critical issue—probably a bug—which leaves less capacity to accomplish the original sprint goal. (Note: Depending on the magnitude of the disturbance it might be useful to consider canceling the spirit. At least, the

team needs to reduce original sprint scope—which may require a mid-sprint re-planning to determine whether a reduced sprint backlog will still deliver the original sprint goal.)

- **Outside dependencies**: The team faces dependencies outside its sphere of influence not foreseeable during sprint planning. (Note: A classic systemic dysfunction.)

3. Scope Increase

The scope of work increases over the course of the sprint:



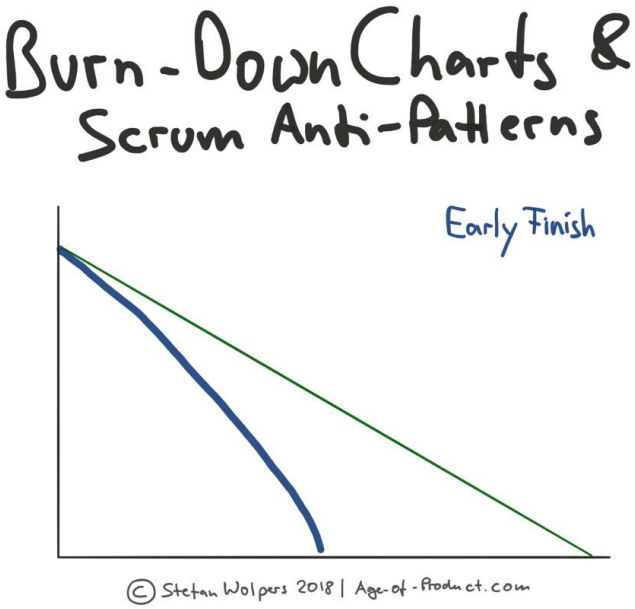
Most of the time, this pattern can be attributed to inadequate preparation:

- **Refinement failure**: The scrum team fails to refine tasks accurately only to discover that the effort to create a valuable product increment is higher than originally expected. (Note: If this happens multiple time during the sprint then the team accepted stories into the sprint the team has not fully understood. This points at serious issues with the product backlog refinement process or the collaboration with the product owner in general.)

- **Dynamic sprint backlog**: Urgent tasks are pressed or find their way into the sprint without compensation. (Note: Depending on the magnitude of the tasks, canceling the current sprint and focusing on the apparently more valuable new issues might be the better alternative. Unless, of course, those new issues are hacking the scrum process of sprint planning. There are several examples of this behavior: A manager pulls strings to get his or her task into a sprint or tasks are disguised as critical bugs that need to be fixed immediately.)

4. Early Finish

The team accomplishes the sprint goal way earlier than expected:



Of course, an early finish is the anti-anti-pattern if the team figured out how to deliver a task with much less effort than expected. Or the sprint goal could be achieved with fewer tasks that planned.
However, the positive news might also hint at some problems. Again, the reasons for this phenomenon are multi-faceted. My two top-candidates are:

- **The overly cautious team**: The team probably overestimated the effort to be on the safe side with its prediction. (Note: This could indicate that the management tracks, for example, velocity as an important metric for the contribution of the team members despite its limited usefulness. Or the organization is output oriented and does not accept 'failure' as an option. In these cases, the organization is setting the wrong incentives. See also the Hawthorne effect.)

- **A hack for slack time**: The team included buffer time to be able to address technical debt, its need for pairing or other issues that do not regularly receive attention and hence managed to finish early. (Note: This might indicate that the current allocation of resources is neglecting the long-term health of the team as well as the code base. Also, watch out for the feature factory syndrome where team utilization and output matter more than the long-term outcome.)

Note: These anti-patterns are only recognizable if the team provides the necessary transparency.

The Conclusion

It is a good idea to use burn-down chart patterns for the next retrospective as they easily identify team problems or systemic dysfunctions. And utilizing burn-down charts in that capacity does not even require switching to story points per se—equally sized stories can just be counted to create a dimension for the y-axis.

Enhancing burn-down charts with additional data, for example, context and occurrences, as well as lead time and cycle time values, will increase the benefit of burn-down charts even more.

Speaking of which: At the team level, I would suggest creating a rotating scheme of team members to update the burn-down chart daily. It is a team exercise and not the job of the scrum master.

Lastly, no matter what purpose you are using burn-down charts for, avoid falling into a common trap: Start counting subtasks. This accounting will quickly lead you on the track of abandoning your definition of done. Instead, you will start marking tasks as 90 % complete. Welcome to cargo cult agile—how would that differ from the waterfall approach?

About the Author

Stefan is a Professional Scrum Trainer with Scrum.org, an Agile Coach, and Scrum Master.

He is specializing in coaching agile practices for change, for example, agile software development with Scrum, LeSS, Kanban, and Lean Startup, as well as product management.

He also serves as one of the XSCALE Alliance stewards and coaches organizations in business agility. Additionally, he is a licensed facilitator of the Agile Fluency™ Team Diagnostic.

He has served in senior leadership positions several times throughout his career. His agile coaching expertise focuses on scaling product delivery organizations of fast-growing, venture-capital funded startups, and transitioning existing product teams in established enterprise organizations.

Stefan is also curating the popular 'Food for Agile Thought' newsletter for the global Agile community with 24,000-plus subscribers. He blogs about his experiences on Age-of-Product.com and hosts the most significant global Slack community of agile practitioners with more than 6,300 members.

His ebooks on agile topics have been downloaded more than 41,000 times. Lastly, Stefan is the organizer of the Agile Camp Berlin, a Barcamp for 200-plus agile practitioners.

Read more about Stefan at Scrum.org, and connect with him via LinkedIn, or Twitter, or privately via email.