Time to abandon Scrum sprints? 14+7 arguments for your next steps towards agility

ontheagilepath.net/2015/05/time-to-abandon-scrum-sprints-147-arguments-for-your-next-steps-towards-agility.html

By Sebastian Radics

Starting with Scrum back in 2008 I was a strong believer in the concept of using sprints.

It provided a nice frame for starter teams, ensured regular feedback cycles and was a key to remove old delivery and product building barriers.

Today I question the concept of sprints. Having learned more about predictability, concepts of lean and flow I'm sure you can achieve even better results, if you start to abandon sprints and optimize flow.

Read on for a check with 14 reasons why using sprints is considered to be beneficial – challenged with the no sprint approach. Followed by 7 reasons why to abandon sprints.

14 reasons for sprints – Let's check the no sprint approach

In 14 reasons why sprints make sense my comrade Stefan Nowaczynski discussed why it's beneficial to use sprints (thanks for this valuable input).
I'd like to consider all 14 points and check how it works in an agile environment without sprints. (Follows the pattern – show the argument for having sprint and describe how it works without sprints).

1. The product owner knows when a functionality is ready

Instead of using the sprint for predictability you consider cycle time distributions and throughput measurements for previous stories.
As mentioned in Unleash predictability by using Actionable agile metrics a great way is to establish SLAs based on e.g. the 80% percentile for all story completion times (using the Scatterplot and Cycle Time histograms as easy to gather metrics).

This way the product owner knows how many stories in a given time interval can be completed and how long a story takes on average. It's an inherent predictability with less overhead than having artificial sprint boundaries.

2. Customers do know when a functionality is ready
Like mentioned in point 1.
In addition you replace the sprint review by a feature review, done on demand and as soon as possible after a potential shippable increment is implemented.
The better you can involve your customer, the earlier you can generate feedback and start earning money.
3. Less coordination overhead
It gets even less using on demand prioritization, grooming and pull and remove long and sometimes exhausting sprint preparations. Nicely described in So long Scrum, hello Kanban.
4. The team knows when it will receive feedback
Lets do get feedback even earlier using ScrumBan – as it's implied in the workflow steps. Just add a step validation or review and generate feedback as soon as a story is ready. The side effect in working with cycle times and SLAs is that you increase "pressure" to generate feedback and ship it.
5. By using the velocity the possible amount of features to be delivered in a cadence is known
Velocity is replaced via inherent throughput and cycle time (see point 1).
6. Early feedback ensures in time corrections and show that a feature is ready/done
Done by workflow steps in Kanban (see point 4). Benefit is less delays than using sprints.
With sprints you (often) generate feedback at the end of the sprint (via sprint review) and assuming corrections/changes you prolong delivery for another sprint time period.
Having it in your workflow it get's blocked because of missing parts and increases "pressure" on the whole system as WIP limits start firing. Stop starting, start finishing!
7. The team has an overview about upcoming topics for the next sprint
In Kanban you use a selected column – best having a minimum/maximum limit. It's the task of the product owner to replenish the selected column properly.
This way the team always knows what to pull next. You can establish grooming (product backlog refinement) meetings – but do it as soon as necessary following your flow of stories.
Half way through the list – let's consider the next 7 pro sprint arguments and ways to work without it.
8. The product owner can create a roadmap based on estimations and velocity
Create a roadmap based on your SLAs (throughput and average cycle time). For larger topics (epics) just ask the team for a *rough* orientation how many stories this epic will be.
9. You find errors earlier
See point 4 and 6. Quality assurance is embedded in the workflow.
10. Changes are communicated in time
Changes get communicated even faster see point 4 and 6. As soon as story enters a review state feedback generation is triggered and necessary course corrections can be

done.
The cost of delay gets even more reduced than using fixed length iterations.
11. Trust through reliable cadence of delivery
The same is true using Kanban and SLA's. It based on past data and get's more an more reliable by an increasing number of finished stories (and therewith measurement points).
I can tell the customer – with a probability of e.g. 80% this story gets delivered in x days (when once selected).
12. Progress is visible and you can feel it
You get even more visibility using an always filled Kanban board and toughly managing your throughput and cycle time.
Immediate workflow feedback, team huddles to solve flow interruptions and a real drive on getting things done – highly visible via board, cumulative flow diagrams and scatterplots (see Unleash predictability by using Actionable agile metrics)
13. Higher quality work
Sprint end pressure leads to possible cheating and hidden quality lacks (e.g. described in So long Scrum, hello Kanban).
Workflow inherited QA, feedback and validation and a steady pace ensure even higher quality because the exhausting rushing elements get removed.
14. Work is more fun, is enjoyable and the team is valued
So long Scrum, hello Kanban and Kanban is the new Scrum show examples of even more fun using Kanban.
In my opinion – the team's value is shown through high quality products, delivered in a predictable and fast manner.
Continuous and reliable customer collaboration and feedback is a core agile value – and it's embedded and even faster established using Kanban (ScrumBan).

Followed by 7 reasons why to abandon sprints

1. Cycle time delays and WIP problems

Striving for a predictable process we should have a predictable cycle time. Aiming for one piece flow (like Toyota) moves us to reduce work in progress (WIP) as much as possible. Using sprints you increase cycle times – by adding queueing times aka waste.

When a story is done and needs to wait until the sprint end to get the feedback and go – it's a non value adding time.
If you already ship it before the sprint end – why having sprints at al 😉

2. Artificial slicing and goals

You need to slice your stories to fit in a sprint. If you already have 10 stories and your sprint is nearly full and you could just add another 1/2 story — what will happen? Either artificial slicing or moving the item to the next sprint.
Not necessary without sprint. You pull the next one as soon as slots get available. You slice having predictability in mind (knowing that too big stories increase risks of delays and are less predictable).
Sprint goals – sometimes seem artificial to me to. Commitment on stories is too pushy – let's invent sprint goals as focus point for commitments.
In my opinion – lets define goals for small pieces of added customer value. Independent whether it takes one week, 2 days or 5 weeks. What's the next value we can create and this is a next goals to reach.

3. Delayed feedback

Feedback should be generated as soon as something is finished. Not waiting for the sprint end but immediately. Work even more with your customer on site and adapt for changes before working on more topics going in the "wrong" direction.

4. Artificial customer demos

Let's have customer demos when there is a value adding piece of software available or generating feedback helps to get course corrections right aways.
Sometimes building some fundamental elements takes a little longer and generating feedback + demoing in between does not make sense. Just wait until its done.
On the other hand – many stories can be demoed immediately when finished. Why waiting?

5. Deferring revenue

Shipping as fast as you can will reduce your cost of delay (of not delivering a feature).
Waiting for a sprint end to deliver (and generating feedback) does not make sense. Ship it and stop starting + start finishing!

6. Higher process complexity

Sprints lead to artificial slicing of topics, rushing at sprint ends (if not done properly ;-), delays and artificial elements in your flow. It adds more complexity by giving a higher weight to estimations and planning.
Why doing it, when you can achieve even better results without having sprints?

7. Backlog waste

Sprints log in a backlog (sprint backlog) for the sprint interval's time. What if changes occur in between? You need to check the sprint goal and maybe have to cancel the

sprint. Why – because Scrum says so and increasing the pain of interruption has an educational aspect.
Ok.
But let's assume a mature environment. Agile thinking is already common and you adapt to the real world having sometimes interruptions you cannot predict 😉
In Kanban you adapt immediately but without having the burden of planning a whole new sprint. Just add it to the selected column and ensure it gets pulled next. If necessary introduce an expedite lane (but be aware how it influences your predictability!)

But

Working without sprints requires mature teams and agility experts (lean delivery agents, agile coaches, Kanban experts,… and more role names are guiding).
It's important to focus on value creation, flow and customer collaboration – removing waste from the system and challenging "lazy" areas immediately.
Excuses that there is too much interruption? Customers are not available? Product slicing is difficult or impossible? WIP limits don't matter?
All areas to challenge – what to be honest – is a benefit of enforcing it via sprints.
Learning to do it workflow inherited is the next step – and maybe yours too?
Please challenge it! Why do you still consider using sprints? Or already working without it and you have supporting arguments that I missed? I'm happy to read you comments…

Further readings

- Kanban is the new Scrum by Abby Fichtner
- So long Scrum, hello Kanban by Alex Salazar
- 14 reasons why sprints make sense by Stefan Nowaczynski
- Scrum and Scrumban as it's next evolutionary step
- Unleash predictability by using Actionable agile metrics