Evolving sprint planning II – split only stories that you start working on

ontheagilepath.net/2012/11/evolving-sprint-planning-ii-split-only-stories-that-you-start-working-on.html

By Sebastian Radics

Background

During our agile journey we discussed and tried various ***improvements to*** the way we do our ***sprint planning II*** – the Scrum artefact to get to the tactical sprint level and decide how the implementation for stories looks like in detail.

Since about one year we made good experiences, only splitting stories on top of the sprint backlog – this post explains benefits and drawbacks of this approach.

How sprint planning II works

Based on our sprint flow we know what are the first stories we will work on in the sprint and how the team's resource distribution looks like.

We look at how long the team plans to work on the first story. If the sprint flow shows that it will take the team more than 1-2 days to implement the story and the team can work on this story together, we only split the first story by writting task cards (post-its for the scrum

board).

Sometimes the first story is too small to work on it with the whole team in parallel. If this is the case we split another story. Usually we don't have more than 2 stories splitted in the beginning.

As soon as its agreed by the team that a story implementation is being finished on this day the next story is splitted together.
Therefore the Scrum Master invites for a short story splitting session – mostly done in the team room and taking not longer than 15-30 minutes.

This means – we have many little sprint planning II sessions during the sprint.

What's the benefit

Goal is to work on as least stories in parallel as possible to:

- ensure working on the most important stories first and finish them during the sprint
- enable (enforce) team work to implement a story
    - ensure common understanding of the implementation – done by common planning, implementation, testing and shipping
    - avoid cherry picking for tasks – as this often occured when the current story was nearly finished and only default tasks (like preparing deployment, round trip testing, documentation,…) were left and a new story already attracted developers eyes 😉

- think about implementation details with the experience of previous stories and with least repetition necessary – sounds odd, but sometimes remembering task cards written 2 weeks ago can get a hard task and if stories belong thematically together implementation of story 1 can make tasks obsolete for story 2. At the end – ***DON'T PRODUCE WASTE***
- easy sprint overview

With this approach for sprint planning II the goal can be achieved – as:

- the whole team is focussed on implementing a story together – as a team
- cherry picking is limitted to one story 😉
- time investment is at the latest point in time and with this waste is reduced as all changes for the implementation caused by new information during the sprint are anticipated
- the board remains clean for the team and the clear focus on finishing stories is visible on the board too.
- the team decides when the next stories can be opened. There is a common understanding where we are in a story and why it can make sense to start working on the next story.

In addition:

- the bulky sprint planning II in the beginning of the sprint becomes smaller, more lightweight and energized. At least in the past we experienced less energy meetings as we already had a energy consuming sprint planning I meeting before.
- through the more lightweight approach – you automatically allow deeper insights and technical discussions for the story at hand, as it's not the goal to finish it for all stories but to focus on the one to start with.

What you need to prepare

- replace hour based burndown charts with story based burndown charts or the sprint flow – as decribed in Visualize sprint progress – an alternative to burndown charts
- ensure a common understanding how to work as a team on a story
- have a team are where splitting the next story is not a big deal with reservation of meeting rooms, changing locations,…

Drawbacks with this approach

- hour based burndown charts don't really work with it (as you can't count your hours in the beginning or you need to work with placeholders for not yet splitted stories)
- your sprint commitment needs to be done in sprint planning I and relies more on gut feeling (*with this we had less problems in the past*) as you don't have all stories technically clarified in detail.

- Necessary story splitting sessions (small kind of sprint planning II) during the sprint – a bit more coordination work for the Scrum Master and maybe a problem for teams that don't work onside together.
- You need to design your code to be ready for refactorings (this is not really a drawback – I know) … because stories later in the sprint, that maybe depend on earlier story can raise a need for adjusting the implementation.

Your opinion?

- What do you think about this approach? Does it sound useful for you or do you already use a similiar approach?
- Would you give it a try?
- How does your sprint planning II look like?