

4.2: Commitment-Based Planning

This chapter gives more details about the Commitment-Based planning practice that was mentioned in the last chapter.

Table of Contents

<i>General Strategy</i>	1
<i>Stories should be “ready” for planning</i>	2
<i>Prioritization</i>	3
<i>The Team Commits</i>	4
<i>Commitment and Technical Debt</i>	5
<i>Plan of Action</i>	6
<i>Discussion of Commitment-Based Planning</i>	7

In the last chapter we talked about the planning that takes place between sprints, and we mentioned both Velocity-Based and Commitment-Based planning. In this chapter I want to discuss Commitment-Based planning in more detail, as well as discuss why I think it’s better than Velocity-Based planning for most agile teams.

First of all, remember that planning, for an agile team, is actually a commitment that the Team makes. This commitment is based on a negotiation between the Product Owner and the rest of the Team, where the Product Owner is speaking for all the Stakeholders that are outside the Team.

In the sections that follow, we discuss the general strategy for Commitment-Based planning, and the major issues that arise when doing it.

General Strategy

Commitment-Based planning begins with the Product Owner having a collection of Stories sufficient to fill the Sprint. Typically, you would expect up to two or three Sprint’s worth of stories to be available, with a minimum of 1¼ sprint’s worth. See the chapter on Backlog Grooming (xx) for discussion of getting stories ready for planning.

At the beginning of planning, the Team discusses the overall goals and priorities, and the PO selects a single story to consider (the highest priority). The Team (including the Product Owner) negotiates the “doneness” Agreement for this single story, and the Team (without undue influence from the Product Owner, or consideration of the Story’s

size in SPs) commits to this single story. What they are actually committing to is the “doneness” Agreement, which we typically simply refer to as the story’s Agreement.

The Team may not be able to commit to a story, or might not even be able to agree on “done.” This makes the story in question an epic, by definition, and the Team must decide what to do. Typical choices include committing to an Analysis Story to figure out what to do about the epic, or extracting a smaller story from the epic to do instead (putting the remainder back on the backlog), or skipping the story altogether and moving to the next one.

After a Story is committed to, the Team (with the PO in the lead) has the option to reprioritize the Story list, and the Team takes the next one to consider. Once again, the Team comes to the “doneness” Agreement and commits to adding the Story to the list of already-committed-to Stories. This process is repeated until the Sprint is “full” and the Sprint Plan is complete.

This process is quite simple, but leads to a number of issues that are tightly intertwined. We describe and discuss these issues in the following sections.

Stories should be “ready” for planning

There are a number of things you should do before you can even begin planning. The most important thing you can do is make sure that your Product Owner is prepared, and understands what the stories are about. Remember that the Product Owner is a role here, so what we’re actually saying is that someone on the Team knows about each story; that is, each story has its own champion (Story Owner) who represents the Stakeholder’s needs/wants to the Team. This may require that the Product Owner (person) coordinates the efforts of all the Story Owners.

This preparation is probably the result of significant Backlog Grooming that has happened previously, which has moved the stories to the Back Burner. Everyone must make sure that the Product Owner is familiar with what has happened in the grooming so far, and that each story has its Story Owner. The Product Owner must be familiar enough with all the stories in order for him or her to prioritize the stories correctly.

Of course, it’s not just the Product Owner that must be prepared; the stories themselves must be ready, and good enough, for planning. This means that they’re going to be small enough so that we feel confident that they’re not epics, and that the definition of “done” is well enough known to the Story Owner so that the Team can know what

they're getting into with a minimum of conversation. This confidence about the definition of "done" can happen in a couple of ways:

- If the story is a "common" one, the Team may do nothing until the meeting. That is, the Team will almost surely know what to do, because they have seen stories like this one before, at another time and another place... this process of recognizing similar stories is called storyotyping, and is the topic of a later section in this book. What is important to know about storyotyping for now is that it gives us common "doneness" criteria that we can begin negotiations with.
- They could have done some analysis already. That is, the Team (perhaps the Story Owner) could've worked with their Stakeholders in order to already have a good idea about what "done" means for the story. These discussions are usually with Stakeholders that are Subject Matter Experts (SMEs).

In either case, this definition of "done" has not been finalized – that happens during the Sprint Planning meeting. What has happened so far is that the "doneness" criteria have been thought about, and notes may have been made for consideration and discussion during the meeting.

Prioritization

As the Team conducts its planning meeting, they move down the list of stories discussing and committing to them one at a time. While doing this, they are constantly reprioritizing the list. In fact, they may even be adding new stories to the list as they go based on discussions they're having about the realities of the situation.

Technically, this prioritization/reprioritization is "owned" by the Product Owner, but in reality it is done by the Team as a whole, with the Product Owner having the final say – especially about Business Value. There are a number of things to consider when prioritizing and reprioritizing this list. Most of these factors are in play all the time, and influence which stories will be prioritized in the Sprint, and in what order.

Because prioritization is so important and complicated, we have separated out the considerations for prioritization into its own chapter, where it is discussed along with Grooming (chapter 3.9). For the purposes of Commitment-Based planning, it suffices to say that the list may be re-prioritized after each story is committed to, and the way it is done is up to the Product Owner and how he/she works with the Team. They need to remember to prioritize necessary Chores, either directly, through adoption, or with

some version of the “70/30 Rule” – all of this is discussed in the Prioritization and Grooming chapter.

One of the interesting twists on prioritization that comes about because we are doing Commitment-Based planning is what happens when the Sprint is nearly “full.” It be that after a number of stories have been committed to, some members of the Team will say something like “We think we might be able to fit one more Small story in here” or “we really shouldn’t have any more database-centric stories, since our database skills are being stretched already.” In situations like these we may reprioritize the remaining backlog in order to bring stories up that will potentially fit into this sprint based on these constraints.

There is no universal truth here; all of these prioritization issues are important at one time or another. Prioritization is a distinctly subjective, and human, endeavor, and should be undertaken with a lot of discussions, arguments, and analysis. There is nothing simple about it. Good luck!

The Team Commits

As the Team moves down the list during planning, it is committing to the stories as it goes. This is actually a commitment to all the stories in the list so far, so it’s more complex than just committing to each story individually.

The word “commitment” is a very *heavy* word for some people and a very *light* word for others. In other words, for some people a commitment is a promise that one would kill oneself to keep, and for others it’s merely an agreement to try. As mentioned in the last chapter, it’s a real good idea for your Team’s definition of commitment to be same as the Organization’s definition. Make sure that the Team, the individuals on the Team, and the Stakeholders each have the same meaning in mind, as it can get really unhappy otherwise. Nobody wants to be involved in a conversation that goes “You promised you’d get it done!” and “No, we didn’t, we just said we try!” Bad day for everybody...

As the Team moves down the list, it does whatever it needs to do in order to be able to commit to the story. This may include tasking it out and adding Task Hours, but it doesn’t have to. It could include a detailed plan about how the Team will actually get it done, but it doesn’t have to. It could include doing some quick analysis or experimentation in order to reduce risk of commitment, but it doesn’t have to. All that is required is commitment. Everything else is just noise.

The Product Owner can be a problem when the Team is committing. Since the PO has actual power, he/she must go to great lengths to make sure that the Team's commitment is genuine, and not coerced by the PO. This issue is a constant one for the ScrumMaster, and a frequent topic during Sprint Retrospectives.

The Team ignores the Story's SP size. However, when discussing the "doneness" Agreement, it is reasonable to make statements like "I thought you said this was a Small, and these acceptance criteria sure make it look like a Large." This won't influence the Team's commitment, but helps keep the PO "honest," and may also lead to a resizing of the story – this could be important when doing Velocity calculations and release monitoring (see section 7).

If the last story that is investigated doesn't fit, the Team should quit planning. We don't want too many "planned out" stories on the Backlog, as this is considered inventory, and hence, waste, based on Lean principles. Practically speaking, stories that are planned out are too easy to choose in future planning – just because they look ready to go – even if they aren't really the most important. However, be realistic. If the Sprint gets nearly full. If a large story won't fit, try a small one. If this one more won't fit, then quit, don't keep searching for one that does. ☺

Commitment and Technical Debt

As we're moving along, committing and reprioritizing, we may have to make a decision to do a story with less quality than we know we should because we need it and it won't fit if we "do it right." In other words, we are committing to producing Technical Debt on purpose, and are intentionally "watering down" the "doneness" Agreement. There are people that argue that if we don't have time to do it right then we shouldn't do it – that not giving enough time and effort to the story proves that we don't really want it enough.

This is an honorable concept, but it's not realistic. Often, "doing the right thing is not the right thing to do" for some legitimate business reason, so we must have a process to deal with it – we can't ignore this reality and still call ourselves agile. Whether we like it or not, there are times that we will knowingly create Technical Debt in order to provide Business Value, and we do it if the tradeoff seems reasonable to the Product Owner, the Team, and the Stakeholders. The bottom line is that the Product Owner has the right to do this, and all we can hope is that he/she doesn't do it lightly...

In situations like this we want make sure that we add a Cleanup story to the backlog, and put (and keep) it in the Back Burner, so that it's not hidden from us in the future.

This story will be a constant reminder that we made this compromise, and we hope it will make us feel guilty enough that we'll actually fix the issue. See Chapter 4.5 for more guidance on Cleanup stories.

Plan of Action

As the Team is moving down the list, we must remember that what is important is that the Team commits to the collection of stories as it goes. That is, at any given point the Team is committing to add the next story to the list of stories it has already committed to.

In order to do this that we may need to have a plan of action, which may look like an actual "old-fashioned" plan the way we used to make them. The Team may need to have conversations that go like "Amir and Dan will work on this one first, and then switch to this other one. While this is going on Amy and Pam will work on these two other stories, and Fred and Mark will swarm as necessary."

The Team may need to task out the stories in order to have a more fine-grained plan of action. In this case we wind up with a collection of tasks for the stories, but we must remember that it is not the tasks we are committing to, it is the stories and, specifically, the stories' Agreements.

If the Team does task out the stories, it may even want to estimate the effort for the tasks in something real, like task hours, and make a Sprint TaskHour BurnDown graph as we do work (see Chapter 4.6). Even if the Team doesn't task out the stories it may need to apply effort estimates to the stories themselves, often in terms of days or hours of work. Just remember that this is not the same as the sizing of the stories (in Story Points) that we did earlier – that is about the story itself, not the effort it will take.

This plan of action could be documented, or a might not be. The point is not to actually have the plan, but to have the Team believe that there is a way that they can get it done so that it will be able to commit. It's all about commitment, not about the plan of action. When you are doing the work and the Sprint, you will know that your Team has a little "waterfall" in them when they start talking about the plan of action rather than the stories themselves. You need to make sure that the Team stays focused on the Agreement and the stories, not the plan of action.

Discussion of Commitment-Based Planning

The reasons that we like Commitment-Based planning rather than other planning methods are:

- It minimizes the number of already planned out stories on the backlog which, in lean terms, is minimizing our inventory. This is important because an already planned out, tasked out, and estimated story just sits there saying “I’m all ready to go” when in reality it might not be ready, or it might not be the right story to go next, soon, or even at all. Things happen, things change, and what we thought we knew in previous planning meetings is likely to be false as we move along.
- This method of planning does not rely on our sizing of the stories, which means that our commitments are based on the realities of the moment rather than what we used to think when we sized the stories. Also, since we’re committing to the stories, we may need to actually do some investigation of the code quality, to find out if the right people are available, and so on. These are things that we could not possibly have known about when we originally did the Story Sizing.

The main problem with Commitment-Based planning is that the Product Owner is a member of the Team, and we need to avoid the possibility of undue influence by the Product Owner when it comes to committing to the stories. This is a job for the ScrumMaster to manage, and is a risk we take when we do Commitment-Based planning. However, we believe that this risk is worth the reward of doing planning based on the actual facts of the moment, and we believe that if a Team knows that it will be doing Commitment-Based planning does not have to think so hard about the stories early on.