

## 4.3: Adjusting Sprint Content

*One of the most common issues that arises with a Scrum Team is that the content of a Sprint needs to change during the Sprint. This happens for a number of reasons, and in this chapter we discuss this issue.*

### Table of Contents

<i>Finishing Early</i> .....	2
<i>Removing Something that's Not Necessary</i> .....	2
<i>The Team Can't do what It Committed To</i> .....	3
<i>Changing Something that we've Already Committed To</i> .....	4
<i>Negotiating New Work</i> .....	4
<i>Discussion</i> .....	7

If there's anything that I know for sure, it's that "stuff happens." As Robert Burns (1785) said, "The best laid schemes of Mice and Men oft go awry," and nowhere is this more true than with Sprint Planning. The Team can't finish what it committed to, it finishes early, new stuff shows up, existing stuff has to change, some of the Items the Team committed to aren't really important, and so on.

What do we do, oh, what do we do? I've seen more teams agonize over this than almost anything else in Scrum. Well, the short answer to "what do we do?" is that *this is the reality in front of the Team, so just deal with it*, but this is somehow unsatisfying. These teams want advice, not platitudes, so here goes.

There are five different situations here. Two of them are relatively easy to deal with:

- The Team finishing its commitments early, and
- The Team needs to remove something from the Sprint Backlog;

One of them is simple, but painful:

- The Team is not going to finish what it committed to;

And two of them are potentially difficult to deal with:

- The agreed-upon "doneness" Agreement for an item needs to change, and
- The Product Owner wants to add something to an "already full" Sprint.

In this chapter we discuss these issues, from easy to hard...

## Finishing Early

So, let's talk about the easy stuff first. One of the best things that can happen to a Scrum Team is that it finishes its work early in a Sprint. It amazes me that people are confused about what to do, but they are. So here goes...

if you finish early, it seems to me you have two choices: take a holiday, or do something new. I have seen teams take a half-day off if that's all the time they have left, and I applaud this. In most cases, however, you should just bring something new into the Sprint and do it.

By definition, the Back Burner consists of items that are no more than one conversation away from being committed to; that is, you need about half an hour of talking before you can start work on one of these.

So, the Team should work with its Product Owner to figure out which item on the Back Burner to do. It should be one that the Team thinks might fit into the time remaining in the Sprint, and they just commit to it and do it. All you have left to do is finalize the "doneness" Agreement and task it out, and then you can do it.

Yes, it is really just that simple... unless you can't find one that will fit in the time remaining in the Sprint. Then the Team can do one of two things: take a holiday, or just start the new story, knowing that it will spill over into the next Sprint. In this second case the story is actually part of the next Sprint, not this one.

Now, that really wasn't so hard, was it?

## Removing Something that's Not Necessary

The second easy case is only one step more complicated. If you realize that something on the Sprint Backlog doesn't really need to be done, and your Product Owner agrees, then you should probably just remove it.

If you've already started working on it, the problem is a little bit more complicated, as the Team needs to decide whether it's more effort to stop doing it (which will mean cleaning up whatever messes you've caused while working on it), changing it until it's right (see the section in this chapter on 'changing something you've already committed to'), or continuing to do it until it's finished or in a good state to stop.

If you haven't started it already, though, the decision should be easy. Just remove it from the Sprint Backlog and continue. If it turns out that you will now finish early, then the Team is in the same situation we just discussed before about finishing early. You

don't need to figure out what to replace the removed story with until you actually know that you have extra time – that's just thinking too hard 😊

## The Team Can't do what It Committed To

Now, for one of the more painful situations we have in Scrum. What if the Team just can't finish what it committed to? Since they actually committed to the “doneness” Agreements, what we're saying is they can't meet all the Agreements that they committed to. So, there are only two basic things the Team can do: remove some stories, or modify some Agreements. The second case actually has two options: we can split a story and remove part of it, or we can decrease quality and produce Technical Debt. So, there are three options in total: remove stories, split stories, or increase debt.

Before discussing these three options, remember that no matter what we do we must make it visible to our Stakeholders, or at least our Business Owner. I repeat, we must be open, honest, and visible to our Stakeholders about whatever it is we're going to do.

The best of the three options is to remove items from the Sprint. It's hard to split items successfully, and come up with new Agreements, and we don't want to create Technical Debt because it will haunt us forever. So, the best thing to do is remove some items from the Sprint Backlog.

Luckily for us, the items that we haven't started yet are probably the least important ones, and can easily be pushed into the next Sprint. This will have an effect on our velocity for this Sprint, and that's a shame, but it's the reality we're dealing with. And yes, I know it is painful, but this is Scrum, not the playground 😊

The second best thing to do can only be done if we have a large, decomposable, story. Generally speaking, we don't want to have items like this in our Sprint in the first place; but if we do, we could split one of them into two items and then remove one of them. This is not easy, but it can sometimes be done. The thing we need to worry about is that when we split the item, each of the sub-items still has a “doneness” Agreement that is good enough so that Technical Debt will not be created when we finally work on it.

The third case is the least palatable, and is the case of relaxing the “doneness” criteria and purposefully creating Technical Debt. We may need to do this because we *really* need to deliver the item, and the cost of not delivering it in this Sprint is higher than the future cost of the Technical Debt. This decision can't be made lightly, and is a decision that is “owned” by the Product Owner – and must/should be agreed to by the Business Owner, as well as other Stakeholders.

I hate this option! But some teams think it's a good idea. If you're one of those teams, just remember to add your Cleanup story to the Backlog (and it goes in the Back Burner so that it's always visible – right in your face – it never slips to the Fridge...). Make sure that it's visible, and well known, that you created Technical Debt, and that you have promised the system, in the Backlog, that we will fix it someday (see Chapter 4.5). Additionally, make sure that all your Stakeholders (or, at least, your Business Owner) know that you've done so, and that they agree it's the best, if not only, way to go.

## **Changing Something that is Already Committed To**

Often, when we're in the middle of a Sprint, we realize that our Agreement for some story isn't actually correct. Sometimes a developer realizes this, sometimes the Subject Matter Expert; maybe it's the Product Owner. Whatever. Since this is the reality we face we must do something about it. There are two cases: the case where the change is actually “internal scope creep” for the story, and the case where it's just a simple change that takes no additional effort.

The second case is pretty simple; all we have to do is change the Agreement. If the change is something as simple as “it's supposed to be green, not blue” and we haven't already started work on it, and then we just make the change – no big deal, right?

However, what if we've actually already made it blue and it will take a significant amount of work to make it green? Well, then it's not really a simple change anymore, is it? And so it will fall into the first case, that of a change that requires significant effort, and thus constitutes “internal scope creep.”

If we determine that the necessary change is “internal scope creep” for whatever reason, then we treat it just as we would any other significant addition to the Backlog; that is, we go into negotiations. This is what we will talk about next.

## **Negotiating New Work**

It is not unusual that external realities cause requirements to change faster than we want them to, putting pressure on us to modify our Sprint Backlog. Usually, this modification comes in the form of the Product Owner wanting us to add something to the Backlog – at least that's the way the problem is usually put to the Team: “here's something new we need to do...”

Now, it is clear that this new stuff must be something we need “right now,” or else we'd just wait and put it into the next Sprint. So, let's look at the cases we might have:

- We might have a brand-new piece of work, that just showed up, and it's more important than stuff we're working on in the Sprint;
- We might have an emergency, of a type that is recurring, like fixing a bug in an existing system or helping out on a sales call; or
- We might have a one-of-a-kind emergency, like helping out when the production server goes down.

Each of these cases is different, and I recommend different resolutions. What is common to the cases is the notion of having a “good faith” negotiation between the Product Owner and the rest of the Team. This negotiation should be “refereed” by the ScrumMaster, as managing the relationship between the PO and the rest of the Team is one of the ScrumMaster’s primary responsibilities. Note: in what follows we use the word “Team” to represent the Team *minus* the Product Owner, just to make the discussion less grammatically tortured.

So now let's look at the three cases.

Case one: we have a new important piece of work. This case is actually pretty straightforward. If there is work that we haven't started yet, and it's about the same amount of effort as the new story, and the new story can be integrated into the current Sprint's work without undue hardship, then we just make the swap – the new work for the existing work. The work we haven't started yet is pushed into the Back Burner, and the new work we want done gets moved into the Front Burner to be worked on in the Sprint.

Now, of course, this requires “good faith” negotiations between the Product Owner and the Team. Both sides are equal partners in this negotiation, and there should be no issues or problems with it. If there are problems with it, there is always the threat of an Abnormal Termination<sup>1</sup> to use.

The threat of an Abnormal Termination is used to try to get the Team and the Product Owner to negotiate in “good faith” – and can be used by either party. That is, the Team must be willing to consider the needs of the Product Owner, and the Product Owner must realize that the Team has the right to say “no” – and the threat of Abnormal Termination and restart provides the impetus to get together.

---

<sup>1</sup> An Abnormal Termination is “called” by the ScrumMaster when requested by either the PO or the Team. The Sprint is replanned, restarted, and all incomplete work is discarded. See the section on Abnormal Termination in Chapter 4.8 for more detail.

By the way, if this need to integrate new, important, work into your Sprint is a recurring problem, you may want to try mid-Sprint Planning meetings to help solve this problem (see the section on mid-Sprint planning meetings in Chapter 4.8). This may reduce the need for constant negotiations, and make it more normal and natural to integrate this new work. If you want to go all-out, and remove this problem altogether, try the Kanban(ish) variant of Scrum we discuss in Chapter 4.7, but don't try and do that until you are fairly mature as a team.

Case two: an emergency of recurring type. The first few times we see emergencies like this, we should treat them like new work; that is, we negotiate putting them into the Sprint. However, it doesn't make much sense to continue to negotiate once we realize that this happens "all the time;" that is, that this is a recurring issue. We don't want something that is just "business as usual" to cause upheaval on the Team – that's just silly.

Once we realize that this is common, there are a couple of things we can do. We can set aside time, as part of our process, to just work on this stuff; for example, we might say: "let's give ourselves four hours every day (as a Team) to work on bugs in the ABC system." What this does is separates these recurring emergencies from our Backlog altogether.

This has pluses and minuses. On the plus side, the issue is solved since there doesn't need to be a negotiation every time, and it only becomes an issue for the Team if the amount of recurring work exceeds the time we've set aside – in which case it becomes an impediment, and we have to do some negotiating at that point.

On the minus side, we don't get credit for the velocity that we are producing while doing this work. This is an issue because we might need to know how much total velocity the Team has so that it will know what it could do if it started doing different work altogether. The Team may also just want to get credit for the velocity, mightn't it?

What I recommend in these cases is what we call Placeholder stories, which are stories that hold Story Points to be used for as-yet-unknown work. See Chapter 4.4 on Placeholder stories for more details.

Case three: the one-of-a-kind emergency. In this situation what you do is pretty simple; you just drop everything and go fix the problem. Once the problem is fixed, the Team renegotiates the Sprint. As usual, the negotiation must be done in "good faith," and may result in an Abnormal Termination. This is the one case where I think the Abnormal Termination is a legitimate possibility, depending on how long it took to fix the emergency.

## Discussion

The issues we've discussed in this chapter are likely to happen to your Team. The basics of resolving them are pretty simple:

- Whatever you do, make it visible;
- Negotiate a “good faith;”
- Try to process-ize recurring stuff; and
- Do an Abnormal Termination as the last resort.

I hope this helps. Good luck.