CONFESSIONS OF A

# SCRUM MASTER

**for the Agile Scrum Master, Product Owner, Stakeholder and Development Team**

by **Paul VII**

InfoQueue
ENTERPRISE SOFTWARE DEVELOPMENT SERIES

Pashun Publishing

# Confessions of a Scrum Master

For the Agile Scrum Master, Product Owner, Stakeholder, and Development Team

**By**

**Paul VII**

# Table of Contents

# Introduction

> “Life is not perfect and the greatest lessons of all are learned by making mistakes and improving.” Paul VII

Although I have always endorsed *The Scrum Guide* as the authority on scrum and the official rulebook, some things can only be learned from experience. This is because although *The Scrum Guide* does teach you the scrum rules, nothing can completely prepare you for what to do when things do not go according to plan.

In my years as a software engineer, scrum master, and agile portfolio manager, it has been a combination of my experiences and implementing continuous improvement that have by far provided me with the most learning. The experiences have been both good and bad. They have involved mistakes by my colleagues and, more importantly, mistakes by me. I have found that our own mistakes are often the biggest wake-up calls of all. However, I have also learned that as long as we truly inspect and adapt, we always grow hugely from our mistakes.

I wrote this book because I have found that my experiences and mistakes are also valuable to others, allowing them to foresee some situations before they occur and benefit from my experience.

This book takes the form of short stories, because they are stories of my life. In addition, this format should help entertain you while enlightening you.

Once you have read this book you will have gained:

- Insight into real life through common, difficult situations that have occurred in real corporate organisations
- Confessions that reveal mistakes I have made on the job and how I have dealt with them
- Lessons learned by an industry expert that can be applied to any project, in any organisation

These are my confessions.

# Scrum: Introduction and Recap

Note: This scrum intro is taken from one of my previous books.

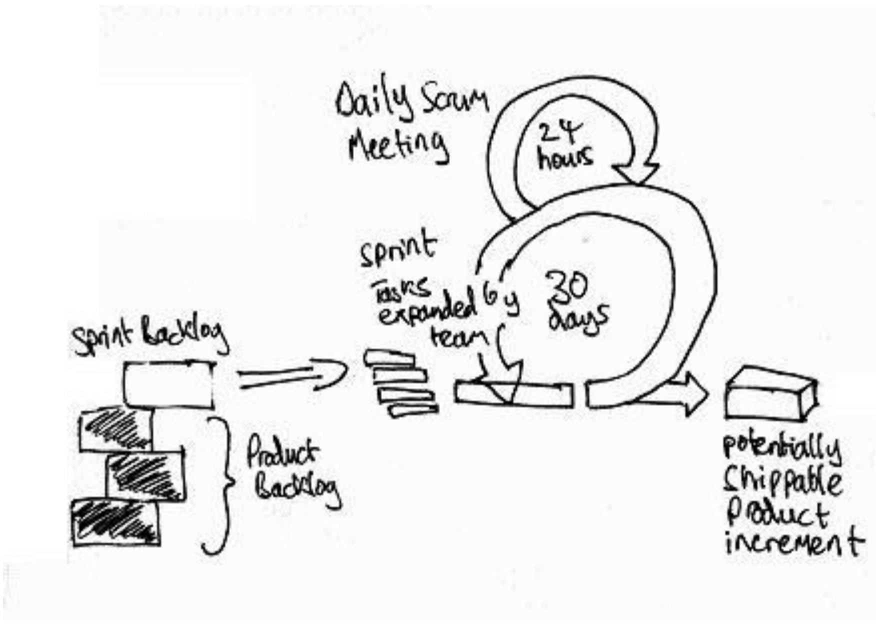
## Scrum theory

“History repeats itself, unless you do something about it!”
Paul VII

Scrum is based on empirical process-control theory. The idea is very simple so do not let the name worry you. It consists of three principles: transparency; inspection; and adaptation. The idea is that the scrum team agree to be transparent (honest) in all that they do on the project.

Being transparent means that functionality is not done until it meets the development team’s definition of done. Transparency builds trust between the team members. Once the team have agreed on transparency, they agree to consistently check on progress (inspection) and make improvements based on what they have seen (adaptation). These can be improvements in practices, sticking to values, communication, or other aspects. This is powerful stuff in industry, the ability to consistently inspect and adapt. In that way, a team are improving time and time again before, during, and after the release of a product. This is something that was not possible with the waterfall model of development.

## Scrum skeleton



The scrum skeleton is a quick and easy way to explain the process to someone, so I will use it to explain the process to you.

On the left side of the skeleton, we see the product backlog, which is nothing more than a list of all the features (and their acceptance criteria) that the business desires for the product. The team takes on a subset of that backlog, called the sprint backlog, which they break down into tasks and work on in an iteration called a sprint. A sprint is a period of time less than thirty days and in that time, the team work on their tasks until they develop a working increment of the product.

These are like the miniature phases of the waterfall within the Waterfall Model (as described in my book the Scrum Mega Pack). There is some gathering of requirements and updating of specifications before the sprint, followed by design, implementation, and testing.

Above the large sprint circle, you will see a smaller circle. This represents the fact that every day the team meet to check progress and adapt their plan for the day in a daily scrum meeting.

At the end of a sprint, the team deliver a potentially shippable increment of the product. The business can review the increment in a sprint review and then release the new feature(s) to the world if they so wish.

The team then discuss (transparently) their progress during the sprint in a sprint retrospective (inspect) so they can improve (adapt) on things that need improvement or retain things that are going well. The cycle then begins again and repeats until the product owner has nothing more to add to the product backlog.

The scrum skeleton demonstrates the simplicity and power of scrum as a miniature factory, churning out shippable features each sprint.

These concepts can apply to any team or indeed individual who needs to deliver a project.

## The Characters

So now, it is time for me to say something I have always wanted to say in a book. Are you ready? Here goes! The names of individuals have been changed in this book to protect the identities of those involved. However, the situations are real and based solely on my point of view.

The characters in this book are a mixture of characters from projects that I have worked on within the software-engineering industry. The overview of the characters is brief, just enough to give you a feel for their key attributes. Obviously there was a lot more depth to them in real life (so please do not take negative comments as a declaration of evil character).

### Jon

An experienced software developer and employee of the company for over four years, he is a hard-working person though very set in his ways. Jon is renowned throughout the organisation for having “concerns” and ensuring that these concerns take centre stage in projects. This can unfortunately railroad the team if left unchecked.

### Sammy

A quality-assurance tester with one year’s experience at the company, Sammy is friendly and adaptive to change.

He has a keen eye for detail and is responsible for driving quality on projects.

**Seb**

A software developer only recently hired, Seb is a “glass half empty” kind of guy. He has a good sense of humour when he is in the mood, but when he is not in the mood, he sees the world in a negative light. Unfortunately, most of the time he is not in the mood.

**Claire**

Claire is an experienced developer with a varied background in programming. Although she has an amiable side, she is also quick-tempered at times. She takes any perceived challenge to her solution very seriously.

**Martha**

A senior product owner who recently joined the company, she has excellent knowledge of product management but only basic knowledge of scrum. Unfortunately, she is not a good listener and comes across as arrogant. Due to her ability to be polite when she feels it necessary as well as her experience, she has obtained a senior role within the company.

### Orbury

A product owner with excellent knowledge of scrum and a passion for products, Orbury works with the team and not against them. With this in mind, he is also human and sometimes gets frustrated (who doesn’t?).

### Gordon

A software developer with a balanced attitude towards processes and practices, Gordon is very much gets to the point. Unlike many of the developers, his gut feeling tells him that the scrum rules will help the team. He finds it difficult to go against the team as he has only recently started at the company.

### Judy

A quality-assurance tester who is adaptive to change and well versed in scrum, Judy has been at the company longer than anyone else. Ironically, Judy speaks very little about her views. She does, however, speak up when there is a quality issue.

## Confession 1: Tools vs. People

I had only been at the company for two months but we had just completed a high-profile project on time and within budget using scrum. What made this unique was that many scrum masters had struggled to deliver projects smoothly. Even if they did meet the deadline, it was usually artificial since they usually struggled with an extremely high number of technical issues or bugs and the scrum process was rarely followed to the letter. I had been given lots of excellent feedback from all the managers in the department, which really encouraged me, made me feel confident, and gave me the start I needed in the company.

Although the project team were set to continue with another release, my programme manager had other ideas for me: “Paul, I have another project for you, and this one has very high visibility with the board of directors.” She explained how the new team had been working without a scrum master, how the team was set to gain another four members, and how the project needed to deliver within three months. From experience, I knew this was a very short space of time to inspect and adapt with a predominantly new team. The programme manager continued to brief me on how the product owner, Rob, was a very difficult person to deal with. He had a reputation for throwing his toys out of the pram. I decided to bear her

advice in mind but give the product owner a chance.
A week later, I met with Rob. He gave me an excellent overview of the product and had been very well organised in mapping out the dependencies.

“So, can I see the product backlog?” I asked. Rob began to talk me through the fact that the product backlog actually existed in two places. There was a backlog stored in the online tool, which allowed one of the team members to work from home. There was also a printed copy of the backlog in a room near the team’s area. This was because one of the two developers was very passionate and opinionated about working with paper story cards.

Although I began my career with paper cards, I had quickly seen and coached on the benefits of using a software tool (when used to the best of its ability). I was well versed in the agile principle of “people and communication over processes and tools” and I worked well with both paper and electronic product backlogs so this was not a problem to me. However, my gut told me that two versions of a large backlog were going to become a problem.
The next day I met the programme manager again and told her how well the meeting had gone. Rob seemed like a cool guy to me. She warned me not to take him at face value (and she turned out to be right, but I will save that for another story). She proceeded to tell me that the new

starters had joined the team, and that I would meet them at the daily scrum (stand-up) later that morning.

I walked upstairs and kicked off the stand-up meeting at 10:30 on the dot. This is where I met the team, which consisted of Jon, Seb, Claire, and Gordon. Seb, Claire and Gordon were new to the company, having started only one week earlier. Jon was a veteran within the company and had been in many scrum teams. He was known not only for his skill as a software engineer, but also as an opinionated man with a reputation for taking issue with projects. The others were not shy either but for now all were pretty quiet at the daily scrum. None of them raised issues and all appeared to know what they were doing for the day. I took some time to build a rapport with them and although they were not the most talkative bunch, we seemed to get on well.

Later that day, there was a meeting to discuss the product backlog with the director of product. The product owner was away so I used the printed backlog as a reference and held the meeting. The team were also present since the purpose of the meeting was to discuss progress thus far. In talking through the backlog, Jon’s ears pricked up.

Looking at the board where the backlog stories were stuck, he exclaimed, “Those stories are wrong!” The room turned to him. “We did those ages ago. And some of them were re-written weeks ago!” The director took notice, but there was no hint of anger. I concealed my embarrassment. It turned out that there had been a mismatch between the printed and electronic versions of the backlog. While the electronic backlog was more up to date, the paper backlog was obsolete. We concluded the meeting and I set about thinking how to resolve the problem.

In all my time at the company, an electronic-tool-based backlog had proved very effective. Stakeholders had visibility, the product owner could make updates from anywhere, team members could work from home, and most importantly (as with paper cards), there was only one version of the backlog.

Another point related to the product owners. They had a reputation for reluctance to manage their backlog and even a reluctance to perform their scrum duties and turn up to meetings. Using paper cards put a much greater reliance on their being on time before we could make any progress. It meant we had to call our often-offsite product owner before we could even discuss upcoming features.

I decided that on such a mission-critical project, with only four sprints in which to deliver, we should not take the chance of using paper. I felt that if the success of the project depended on such a critical factor, I needed to make this decision quickly. I played with the idea of raising it for debate but after some thought I decided that the risk of stirring up unnecessary fuss and worry about using the tool was too great and not worth it. Besides, the team had been using the tool anyway. All I was going to do was use it exclusively. I set about telling Martha, the substitute product owner who was covering for Rob, my plan and why it was important.

“As long as we get the work done, I honestly don’t care!” she said.

Later that week, we had our first planning meeting. I knew this would be challenging for us. Not only did we have a team of mostly new developers and a new substitute product owner, we also had a number of people who were new to scrum and who were quite opinionated, one of whom was pretty set in his ways. I knew that this was also probably the first time that anyone except me had used a tool for planning. However, I also knew that the new product owner had no interest in the backlog and that the paper backlog was resulting in cards lost and misplaced. (Note: I had been spending some time building a relationship with the product owner and explaining the benefits of our practices. However, Martha was well known for letting messages go in one ear and out of the other.) We also needed to support one member working from home that the tool was already helping.

I kicked off the meeting bang on time. The team were in pretty good spirits so I had high hopes.

“Right, guys, let’s have some fun. As you can see here is our backlog,” I said, pointing to the screen with the software backlog. Jon looked at the screen in a less than enthusiastic way. I began to explain the planning game to the team. Martha, the substitute product owner, talked through each story.

We came to a point at which the story needed updating. Using the tool, we decided that the story needed to be dragged out of the sprint backlog, as Martha did not want the work done within the sprint. She dragged the story out using the mouse. A swirly graphic appeared as the computer processed the move. Suddenly, there was an outburst.

“This is driving me nuts!” said Jon. We all looked at him. “In agile, we’re not meant to do things that slow us down. That thing is slowing us down!”

Now although I knew very well that there was some truth in what Jon was saying, his approach was abrupt and negative. I felt that he was picking on a loading icon as proof that planning was being slowed down. Despite my feeling, I quietly pointed out that although I took his point, overall planning was smooth; it just took some getting used to.

This, however, was not enough for Jon. He continued to speak about the virtues of agile and about how frustrating this was. The other developers, new to the company and eager to bond with their fellow teammate, backed him up. I could see that the temperature was rising for the team. I was not angry about Jon’s suggestion but his approach did frustrate me. I concealed my feelings.

“OK!” I said. “I was simply trying to explain why this might appear slower. I was also trying to avoid duplication. If you are more comfortable with paper cards, let’s use paper cards!” I left the room to get the paper so that we could re-write the stories.

When I came back, everyone had stopped complaining. They seemed almost shocked that I took their point on board. The planning meeting continued and we planned the sprint.

The next day, I took some time to think about what had happened and how the team may have felt. However Jon expressed it, this was possibly an issue for the whole team. I felt satisfied with my decision to go back to paper cards, even though I knew the benefits of the tool. Weeks later, the team actually did begin to see the positives of the tool, and decided themselves not to use it in meetings but to use it outside of meetings. Nevertheless, this event was significant for my career. From my point of view, taking the decision to use a tool instead of paper cards was in the best interests of the team. It avoided duplication, made life easier for the less-than-enthusiastic product owner, and allowed the team to work from home. However, the frustration it caused in that first meeting negated a lot of the good it brought. In an effort to help the team, I had put a tool before them. If I had given the reasons up front and allowed them to make a decision, I would have strengthened our relationship and possibly created a different outcome for that meeting.

**Lessons learned:**

- Avoid duplicate backlogs: I could have explained from the outset to the team that we should always have only one version of the truth. If you need a paper copy of the electronic backlog, I advise you to always make sure that the electronic copy is up to date, and to consider the printouts temporary. Otherwise, find a method to make a paper backlog work. If the product owner is not around, maybe the backlog can be left near the team (this is not ideal, which is why the electronic backlog seemed to work well).
- Good relationships solve problems: Working harder on my relationship with the product owner and team would have made it easier to explain my reasons for an electronic backlog. Instead, I was concerned that I would cause a fuss because my team were at times

very negative. I advise you to spend time building solid relationships so that people have open ears when you discuss important or difficult topics.

- Team decisions: I could have involved the team much earlier. I advise you to involve the team in decisions of this nature that affect them. There is nothing wrong with giving them a recommendation. The main task for a scrum master is to uphold the rules in the scrum guide. Getting the whole scrum team to collaborate is a big part of this. How to store the backlog is something the product owner should be most involved in and the team should also feel comfortable enough to raise any blockers. A scrum master is then able to approach the issue from the point of view of a blocker that needs to be resolved.
- Explain benefits: I did not explain the benefits of the approach until the approach was challenged. If you are passionate and confident about something, explain its benefits to the team before asking for it to be implemented. This includes the team, empowers them, and creates a collaborative environment.

For basic and advanced training in a book and a complete overview of scrum, see the Scrum Mega Pack:

http://www.pashunconsulting.co.uk/scrum_mega_pack.html

## Confession 2: Release-Planning Peril

The sprint-planning meeting is the most crucial meeting when it comes not only to getting commitment from the team but also to giving everyone an understanding of the features that they will be working on. In a similar vein, the release-planning meeting is a helpful way to give the whole scrum team an overview of the next release. The following story shows how one or two bad decisions can put this important meeting at risk.

The team were ready for their first release-planning meeting. Given that we were planning the first-ever release, this meeting was more important for us than it was for many teams at the time. I knew from experience that the senior stakeholders would want to track progress tightly. However, this did not worry me since I knew exactly how to coach the product owner on how to communicate progress and building a release plan would give us the best possible starting point.

The new product owner, Martha, did a good job of organising key people for the meeting. The creative director would attend for the first part of the meeting to explain the concept and set the vision for the first release of the product. He would then leave the rest of the room to discuss and estimate the release backlog.

It so happened that the scrum team was an unusual combination of job titles from the business’s perspective. We had a new product owner taking over from an old one. Orbury (at that time an assistant product manager) was also assisting the handover. This gave the impression of three product owners at times. There were also three designers, two testers, and five developers as well as myself, the scrum master. In short, there were quite a large number of people at this time-boxed meeting.

I was also working with a team that absolutely detested meetings. I could understand this since I had been a developer and understood what attracted them to the job. They were very much concerned with getting things done, working out logic, and problem solving. They were not concerned with lengthy discussions and comments. As a developer, I just happened to be different in that I enjoyed that.

Martha told me in no uncertain terms that she was feeling pressured to get estimates for the business in that meeting. I spent some time explaining that we could not get any reliable estimates until we had run at least two or three sprints; that way we could measure our average velocity. However, Martha did not appear to be concerned with this. She was focused on using this estimation session as a means to derive and forecast a launch date.

This all led to a tense feeling in the room as the product manager’s and assistant’s anxiety showed through.

Once the creative presentation and discussion was complete, I decided to be as positive as possible and began the meeting on an upbeat note.

“Right!” I said. “Let’s start planning! Martha, can you read the stories? And team, let’s be ready to discuss before you estimate. We will have to time-box the meeting so please be mindful of the time.” Martha started the process, reading each story title and acceptance criteria before the team discussed and estimated.

The team had been in the company no longer than three weeks. They had little idea how scrum worked. Frankly, many of them had very little interest and at times had made it clear that they were less than enthusiastic about the process. Given their outspoken characters, I knew that I had to handle any disagreements very diplomatically.

After thirty minutes or so, we had discussed and estimated only a few stories in the backlog. The product managers were becoming conscious of the passing time. Orbury stirred in his seat. He was frustrated from the moment we got in the room since he had just given up smoking. Martha was not the most patient person at the best of times. She began hurrying the team, and the product manager she was taking over from also expressed a need to keep the meeting moving. I could see that the temperature was rising and that the product managers were feeling the most pressure from the stakeholders. The team were feeling pressure from the product owners and this was not helping the estimation session.

The situation unravelled when one of the developers, Claire, decided that we needed a new story.

“Ah, I’ve noticed something,” she said. “When we write all the code for these stories, we will end up needing to create a new type of database and a handler. We should create a story to write the whole database and handler upfront. That will save us time later.”

My agile principles came to mind. I knew that the most agile way to bring in large-scale changes was in small increments. In other words, instead of creating a large story for a database and handler, it would be better to build just what is necessary (part of the database and handler) to complete the story.

I said, “I see what you mean, Claire. You could do it that way. What I would recommend rather than a separate technical story card is to include the setup of the database in one of the epics. When you take on your first story, you then do just enough work to get the first story done including any set up. This way you take the hit for getting the database and handler created in the first story but delay any unnecessary setup till you take on subsequent stories. That would be the more agile way to do it and would still allow us to deliver a working increment from the get go."

Claire was not convinced. Despite being cautious and diplomatic, my words were doing nothing but frustrating her. How dare I tell her how to code, she thought.

"Are you telling me I can't create that story?" she asked angrily.

"No, Claire. I am just giving some advice on how best to go about this in an agile way," I replied.

Jon decided to get involved. "There is so much development experience in this room. We would be fools not to listen to it. We know how to write code and we have been doing it for years!" he exclaimed.

"Exactly!" Claire chipped in. Her words managed to convince everyone that I was actually a bully in the shape of a scrum master.

The developers and product managers began to join the conversation. I could sense that the rest of the people in the room were becoming frustrated as time ticked on. The atmosphere was becoming tenser. I could see that there was not enough time to coach the team and finish estimating in one meeting. I decided that allowing Claire to raise the story was definitely not a showstopper and was not worth fighting in that meeting. Instead, I would coach the team on agile principles outside of the meeting.

“Okay, Claire!” I said. “As I say, this is the more agile way to do it, but let us discuss it after the meeting. You can create the story and we will also bring it up in our retrospective.” Claire wrote the story card feeling mildly satisfied but still mildly frustrated. We continued discussing and estimating more stories.

Now, it so happened that the team were very new to the project, new to the process, and new to the first release. This meant that there was a long period of discussion for each story. The meeting, scheduled for four hours, was quickly running out of time. The product managers and assistant were growing more frustrated by the minute.

“Arrghh! Guys, I’ve got to get a damn cigarette, I can’t wait any longer,” Orbury exclaimed. He stormed out.

We agreed to have a five-minute break. The long meeting was not helping the situation. I had booked it for four hours because we wanted to get estimates that same day. There was now only one hour left and we had covered only a third of the backlog. I could see that we were not going to finish the meeting with a resolution so, sensing the business’s urgency, I decided to ask the team if they would prefer to stay another hour or go home. That way I was not forcing anyone to blow our time box and I was not bowing to the business. The team, though frustrated, were keen to get planning out of the way and decided to stay another hour. This was going to take us to 18:30. Since company hours ended at 17:30, we would have worked one hour past the official end of the working day.

Anxiety grew in the room as we discussed each and every story. Some of us, including me, used humour to soften the mood. Orbury and the team looked the most frustrated, even though they were the people who were least aware of senior management’s desire for estimates. The next hour passed and we still had not completed the backlog.

“Ok guys!” I said. “Let’s call it a day.” The team got their stuff together quicker than I could say, “Thanks for your time, everyone!”

I reflected on the meeting while I prepared to leave. Scrum rules said that meetings always should be strictly time-boxed. We really should have stopped for the day as planned, but I put this to a vote. My intention was to alleviate the frustration of another meeting if people preferred to carry on. However, the outcome of the long meeting was even more frustrating. Sitting in a room with lots of debate and uncertainty for hours deserves a long rest. If I had just stuck to the strict time box, we would still have lived to see another day. On top of that, we still had not accomplished our goal of a fully estimated release backlog anyway.

**Lessons learned:**

- Avoid the appearance of multiple product owners: Even though we had made it clear to the team that Martha was the product owner, the presence of three product managers in the room created more debate than necessary and amplified the frustration. I recommend having a quiet chat with the real product owner to emphasise that only one product owner should be at the meeting. If there is an extremely good case for two, then only one should speak at the meeting.
- Time-box meetings strictly: *The Scrum Guide* is very clear that meetings should be strictly time-boxed. On top of that, working after hours reduces energy and enthusiasm since everyone is looking forward to going home. Sometimes, the team will agree to work later but inside they are frustrated. I recommend leaving any unfinished work for another day when the whole team is fresh – or take note of my next point.
- Experiment with splitting long meetings into multiple smaller meetings: In the release-planning meeting, we used the first hour for a discussion of the creative

vision. This could have been scheduled ahead of the release-planning meeting. The other three hours of planning could also have been scheduled as three separate estimation meetings. I use a weekly product-backlog-grooming meeting for the release estimation and discussion (among other things) to ensure that the team are always on top of their game. This prevents the need for long meetings. However, there must be some forward thinking to ensure that this happens before the date set for release planning.

- Avoid lengthy coaching within meetings: There are often situations within meetings when people need a process explained. Once the conversation goes beyond explanation and becomes a lengthy debate or even an argument, there needs to be another solution. In this case, suggest taking the matter offline, into another meeting or coaching session. This gives everyone some breathing space to understand your point, usually avoids rushed conversation in a time-boxed environment, and avoids delaying the meeting.

For basic and advanced training in a book and a complete overview of scrum, see the Scrum Mega Pack:

http://www.pashunconsulting.co.uk/scrum_mega_pack.html

## Confession 3: Intro to Scrum Gone Bad

I had recently become the scrum master for a new team within the company. Most of the team members were new to scrum and needed an overview, so I decided to give them one within our kick-off meeting. As part of this overview, I decided to explain some of the concepts of sprint planning to them. The kick-off meeting was not only a good way to give the newest recruits an overview but also a way to get some of the more experienced members to speak up and possibly advise the newbies.

I began the meeting with an overview of the scrum skeleton. I usually found that this was a simple means of explaining the whole scrum process in a nutshell. My experience had shown me that a picture or diagram can explain scrum almost as a process (rather than as the framework it is) and that, as a result, people often understood the foundational theory very quickly. Once I had talked through this, the teams grasped the key concepts with ease and there were only a few straightforward questions.

Next, I moved on to the concept of user stories. This is where the fun started.

“A user story,” I said, “is the description of a feature from a user’s perspective. It is usually written on a card. On the front of the user story is the description, in the following form: As a <role> I want <requirement>, so that <reason>. On the back of the card is the acceptance criterion." I went on to explain how stories were written with the team, but owned by the product owner.

I then followed up with the concept of sprints. I explained how the user stories committed to in the sprint could not be changed unless the team were happy to accommodate the change. I emphasised how this was meant to help the team to focus. I expected the team to be ecstatic at this news: finally, a process that would support them, give them room to breathe, and allow them to focus on quality.

However, I was dealing with a different breed of team. These were highly analytical, at times "glass half empty" people. They were also very much novices, which made it more difficult for them to understand how the benefits would positively affect them. Not knowing this, I decided to ask one team member's opinion.

"So what do you think, Claire? How about you?"

Claire looked around at the rest of the team. "Well," she said, "I think it sounds like the most over-complicated, convoluted process I have ever heard of! I thought this was meant to be a simple framework."

"It is meant to be," Jon added, suggesting that the process was actually being made complex.

Having only just met the team, I was surprised at the reaction, as I had been trying to explain the benefits, which had been very well received by other teams.

“So what do you think is complex about it Claire?” I asked.

“Well, the whole thing! Why shouldn’t the product owner be able to change the stories? And why should she have to go to all that trouble of writing acceptance criteria for every single story? What a waste of time!” she exclaimed.

I attempted to clarify. “I see why you may think that, Claire. It is not that the product owner cannot change any stories at all. It is left up to the team to confirm if they can accept the changes and still meet their commitment. This helps them to focus. Also, writing acceptance criteria is really no more effort than writing a functional specification, which you are used to. In fact, it is written in simpler terms, and therefore should be far simpler than a functional spec. It is written once and simply refined each sprint.” I felt happy that my reasoning was on point but listened for a reaction.

“No, I am not convinced. I still think it’s complex!” Claire replied angrily. The other team members all pitched in their thoughts.

I could see that the comments were causing the newer team members to think negatively of the process. The older team members (Judy and Sammy) were quiet. Although they had the experience to explain how scrum worked in real life, they were not as outspoken as the other team members. They looked on while the others discussed.

I had spent most of the meeting presenting and had left 10 minutes for questions. We also had another meeting directly after this kick-off meeting. I could see that we would need more than 10 minutes and therefore decided to let them run over but not for more than an extra five minutes.

The meeting was getting heated and the comments continued. Claire continued to highlight the complexity of the process. Jon continued to point out that not everything explained was necessary. Seb pointed out that he had worked on many other projects without scrum and did not see the point, etc. Hopefully, you get the picture.

Throughout all of this, I continued to explain and coach the team on the reasons for the process. The meeting was starting to become less of a kick-off and more of a scrum master justifying the need for scrum. I was not happy with the direction that the meeting was going.

The amount of necessary explanation took the meeting more than five minutes over the allotted time. My time box had been blown. Realising this and knowing that the team were less than impressed with the theories I had described, I decided not to spend any more time on it. I wrapped up the meeting and the team began to leave for the next meeting. We were now nearly 10 minutes late.

Later that day, I sat back and reflected on the meeting. I had done my best to be sensitive to other peoples' points of view while being true to my role. However, I could see that the characters in the team were, frankly, pretty negative. I needed to handle this well to avoid seeming like the bad guy. I was not happy with the way the meeting ended but I did feel that I had effectively explained the benefits of scrum.

In the end, the team became advocates of scrum, which by far ran counter to their views in the meeting, but I always felt that some reorganisation within the meeting could have got the team off to a better start and still kept us on time.

**Lessons learned:**

- Separate the kick-off from the overview: I could have set up an overview separately to allow plenty of time for debate and avoid a rush towards the end. Kick-off meetings are usually more focussed on the specifics of a project than the specifics of scrum.
- Ask veteran scrum practitioners for their experiences: We should not always feel like we need to be the only ambassadors of scrum. There were times in the

meeting when the more experienced team members could have participated. Sometimes, this can be prompted with a simple prompt like “Judy, you have some experience in this, don’t you? Would you mind sharing your experiences with the team?”

- Find ways to wrap up meetings during heated debate: The heat of a debate should not cause meetings to overrun. There are various techniques to avoid this. I recommend aiming to wrap up five minutes early. Also, in the last 10 minutes we can either draw the point to a conclusion or find the right point in the conversation and say something like “Okay, guys, I know this is obviously a hot topic, but we only have five minutes left. Why don’t we run a sprint and then discuss this in the retro?” You can then use the last five minutes to summarise the meeting.
- Always end on time: As with the point above, the meetings must be strictly time-boxed. The technique mentioned above has proved very handy when a meeting is getting heated. I recommend that you never deliberately run a meeting past its allotted time. You can always discuss urgent matters after the meeting. People will always respect you for ending meetings on time and this will also enhance your reputation as an excellent chairperson.

## Confession 4: Stand-Up vs. Sprint Review

Soon enough, the team had gone through all the scrum meetings. We were now three weeks into practicing scrum. The team were starting to feel anxious about the sprint review meeting. The main cause of anxiety and tension was the demo aspect of the meeting. The team were consistently worried that the senior stakeholders would be misled into thinking that the product was more complete than it actually was. Let me give you some background.

The purpose of the project was to build a revolutionary news Web site that made news more engaging and multimedia-rich than ever before. The Web site needed to be responsive (i.e. it should be built once and adapt across some core mobile devices as well as desktop PCs and Macs). The issue was that much of the live data from the back-end systems had not yet been created. The team were using test data to get ahead of the game and test the product. The team were concerned about the risk that stakeholders would leave the demo with a false sense of completion.

Of even more concern than this was the process of actually preparing for the demo. This was not actually an issue until Jon highlighted the idea. He would say things like “Guys! When we’re in that room and everyone is staring at us, and the site looks like crap, it is going to be a horrible feeling.”

Jon had a leadership role within the team because of his experience at the company. When he spoke, the team listened. He would usually give the team a speech or pep talk when he was trying to get them to do something he believed in. In this case, he was trying to get them to deliver high quality by the time of the sprint review. He was also quite fearful (internally) and at times resentful of management. This led to worry that the team would be ridiculed.

All of the positive things that had been instilled in the team were being challenged by this kind of feeling. The fact of the matter was that the management team were very diplomatic when dealing with teams. I spent time reassuring the team that they had done a great job thus far and that the stakeholders were used to seeing an incomplete product. However, the team were not quite convinced of this and the tension remained under the surface.

On the day of the sprint review, the team came in early. They worked to get the demo ready. I left a gap in time after the daily scrum in case the team wanted to further prepare and, of course, I intended to host the meeting as usual. It was important to keep the cycle going.

“Okay guys, let’s do stand-up!” I said in my usual positive tone. The team focussed on their screens, pretending not to hear. “Guys!” I reiterated, knowing that I was about to meet with some resistance.

“Oh! Why can’t we miss stand-up today! We’ve got a demo to prepare for!” said Seb.

“Yeah! We can do the stand-up anytime!” added Gordon.

Now, although I was confident and understood the value of the stand-up, this was a tricky situation. The team were preparing for a demo and I did not want them to do badly. I could also see tension building. I did not want the team to get any more frustrated, especially as I was their scrum master. It was tempting to abandon the stand-up for the sake of an easy life. On the other hand, how long would life be easy for? There would always be another challenge. On top of that, I knew that the stand-up was just as important as any other meeting. I had to make sure that the framework was carried out in full to prevent it from falling apart.

I deepened my tone and looked at them confidently. “Guys, I know you need to prepare and you will still have 45 minutes after the stand-up. We just need to make sure we know what each other is doing, and that we can get rid of any blockers, so, come on – let’s make it short and sharp and then you can get back to preparing for the sprint review."

They got up slowly, reluctantly, and while talking under their breath. The stand-up was carried out, but not happily. However, I knew this action would keep the habit going and pay off in the long run.

After the stand-up, the team went back to preparing for the sprint review. I waited for a few minutes and then went over to Gordon to see how the preparation was going.

"How are we getting on, Gordon? Feeling ready?" I asked in a warm friendly tone. Gordon looked up and smiled, appreciative of my concern. Before he could get any words out, Claire interjected harshly, walking over with some notes for the meeting.

"No we are not!" she said in a dismissive voice. She immediately began talking to Gordon about the sprint review. I thought about pulling her aside for a word, but decided that the timing was not right. The team were probably frustrated and another time would work better.

Another 30 minutes passed and the sprint review began. I kept the stakeholders occupied for the first two minutes while the team had a brief discussion. Once they were finally organised, the team came in and presented the product, and did so well. They were always very capable, despite their lack of faith. When the meeting ended, there was a sense of relief on the team’s faces. There was no animosity between us but I could tell that they had not all appreciated my making sure that stand-up went ahead.

That day, I reflected on how things had gone. I had received many negative looks and responses from the team. The enthusiasm for scrum had waned over the course of that week and I felt disappointed in the situation. Constant negative feedback makes it more difficult even for a scrum master to stay upbeat. However, in my usual spirit, I took it as a learning experience and decided to focus on how to improve.

**Lessons learned:**

In the example in this chapter, I came away feeling that most of the decisions I made were actually correct. However, being right or wrong was not the issue. The question is, how to reduce tension with the team and still uphold the scrum rules.

- Ask before the sprint review if code is shippable: I found out that the team were finishing pieces of code before the demo. This was part of the reason for the frustration. I learned I needed to spend more time coaching them on how to create a potentially shippable product. I also had to remind them that the day of the sprint review is actually the first day of the new sprint; the product should have been complete the day before. I have some tips on meeting deadlines in the book *How to Meet a Project Deadline with Scrum in 7 Simple Steps*.

- Add a buffer for sprint-review preparation: I could have suggested this in one of the retrospectives. We add a time buffer for other events such as server downtime and meetings. If the team need time to prepare or discuss the sprint review, a small amount of time set aside and a reminder to work out who is presenting and in what order will help. It can be added to the diary if necessary and will keep stand-up from becoming a contentious meeting devoted to organizing the sprint review.
- Select stand-up time wisely: If teams are not comfortable with stand-up preceding the sprint review too closely, the time can be adjusted to suit their need. Unfortunately, this can often conflict with the stakeholders' schedules, which may dictate when sprint review (and hence the stand-up meeting) can be held.
- Try and use live or "as live" test data: One of the team's concerns was that stakeholders would be misled by the test data in the site. I cannot remember this ever being an issue, but I could have emphasised the need for "as live" data in testing. For

various reasons (such as dependency on other teams), this is not always straightforward. However, it should be something the team should strive for and the scrum master should assist with.

- Discuss sprint review at retrospective: As with all things that occur in the sprint, we could have spent more time discussing the sprint review in our retrospectives. This way, any concerns or tensions would have been exposed before the event. It would have also prevented a build-up of ill feeling such as that exhibited by Claire.
- One-to-ones with team members: We saw at least one example where I felt that the team had not handled me respectfully. Although being a scrum master is not a command-and-control position, there is never an excuse for the team to speak disrespectfully to a scrum master (or to anyone). I could have held a personal meeting with Claire or anyone else whom I felt was not treating me correctly. This would bring the behaviour to an individual's attention without causing an uncomfortable public situation. In my experience, this has worked very well. If the person does not change their behaviour, you can discuss it with the person's line manager and decide what action is appropriate.

# Confession 5: Taking the Team to Task

At my job interview, I was asked my view on the importance of breaking down stories into tasks. I believe the answer I gave was one of the main differentiators between myself and other candidates. The head of department was keen to employ someone who understood the importance of planning from creation of the product backlog all the way to helping the team to create a detailed plan in tasks. In this chapter, you will see how this may sound fine in words but can prove very challenging in the work environment.

It was sprint-planning day and the meeting kicked off as usual. We discussed the first stories and as soon as the team reached an understanding, I asked them to begin breaking down into tasks. Jon, the longest-standing member of the team and of the department, sat up in his chair. I could see that he had something on his mind. I could also see that he was very serious about it.

“I don’t see why we need tasks at all!” he started. I began to explain how the tasks were the basis of a plan for the team, how they were necessary to ensure that there was a technical plan to complete the story. “Yes, yes I know all that! But we commit to a body of work. It doesn’t matter how we do it as long as we do it, as long as we deliver it.” Jon continued.

I replied, “That is true, Jon. You can build the product however you prefer. I trust you all in that. But before you build it, we need to build a sprint backlog. What I am coaching you on is how to build that backlog.”

Jon turned his nose up at the idea. The team looked equally frustrated. They had clearly decided that Jon was right. I was starting to look like the bad guy. I decided to continue in a positive manner.

“Okay, team. Please continue to discuss the plan and start writing tasks just like I showed you,” I went on. At first, the team reluctantly discussed the tasks, but once they had forgotten Jon’s interjection, they got into the flow and almost happily wrote their tasks down on paper.

“Okay. Cool. Now, remember to write your estimated time on each task. The guidance is that each task should be a day or less in length. That will let you show progress daily,” I said.

This triggered Jon’s attention. He kicked off again. “Look, this makes no sense! Why should we have to put time on the tasks? It doesn’t matter how long we think it will take. It’s up to us!”

Seb pitched in. “Yeah, putting time on is just another stick to beat us with, if you ask me!” I could sense another storm brewing. The other two developers looked disgruntled.

“That’s not it at all guys,” I explained. “We work as a scrum team and our relationship is based on trust. The time on the tasks will help you to track your own time. All we should be looking at daily is the amount of time remaining, so that we can make sure we are progressing towards our goal. We will use a burn-down chart to help to show progress and that you are working at a steady pace. Putting time on your tasks is one of the best ways to track this. I am just trying to use some of my experience to help coach you guys. That’s all.”

I should not have used the “e” word. Claire looked at me, taking my statement in the most negative form. “Well, we’re experienced too! I’ve been doing this for years. I’ve made many plans. So what about our experience?” I explained how we were all experienced but how we needed to respect each other’s expertise: mine as a scrum master and theirs as a team. The meeting continued in a tense atmosphere and eventually the sprint backlog was complete.

The next day, after stand-up, Gordon came to my desk and asked to talk to me for a second. He said, “Well, we have been discussing these tasks. I don’t think they work. Every time we do work, we are meant to put down time spent and work out time remaining. That means that every time I finish a task, I spend time working it out. I’ve just wasted five minutes working out how long it took me to do the work!”

Now, Gordon was different than the others. He was not as negative in personality but the team was influencing him. I had observed that the team were often all for an idea until someone put a negative slant on it. Then, they began to think differently and back up their team member. Gordon’s discussions with his colleagues had convinced him that working out time remaining was a waste of time.

I explained the benefits of calculating it (as I had done the day before) and passionately reminded him that without this kind of planning, many teams end up without a clear idea of how much work is remaining. Gordon also had a quick temper and got worked up easily if he had been led to believe something was wrong. I could see he was holding it in because he respected my knowledge but also because he wanted to align with the team.

“I might have a word with my line manager then because I don’t know. I am just not sure this is working,” he said.

I replied, “Gordon, if you feel you want to bounce this off your line manager, feel free to do that. I am coaching you all on this because I have seen what happens when teams don’t do this. I have seen many teams end up in chaos without a full plan. I really want you to understand that for the good of the team.” He nodded and left. I had a great relationship with his line manager. Regardless, my gut feeling told me he was not going to mention it once he had thought about it.

The team started writing, estimating, and amending tasks after sprints as a matter of course, with no complaining or negative feedback. A difficult and repetitive set of conversations had helped us jump the hurdles. I knew that not everything would be easy, but I still needed to invest some serious time thinking about how I could have made the experience more positive.

**Lessons learned:**

The above is more of a lesson in itself. I came away from it feeling that there was not much that could have been done to pre-empt the heated discussion. In fact, it was a healthy discussion that forced me to explain the value in carrying out a task breakdown. The important points here are:

- Always face opposition: We will always meet teams and other people who either do not understand, do not agree with, or are plain opposed to our suggestions. This is healthy and this is life. We just need to make sure we ourselves understand the reasons for our suggestions and to explain them clearly.

- Task breakdown is actually not a scrum rule: It is, however, a well-known agile practice to help increase the probability of a shippable product increment at the end of a sprint. Therefore, it is completely valid for a scrum master to suggest this.
- Explain: I suggest you always meet hostility (not that this team were overly hostile, but still…) with a clear explanation and reasoning. Team members are usually hostile or frustrated because they do not understand your reasoning. You may also be wrong about something. The only way to realise this is to talk through the explanation. If it is wrong, none of us should be too big to admit it. The team will have plenty of respect for someone who admits, "I was wrong." However, if you are not wrong, then the above point will help clarify for all.

## Confession 6: Retrospective Regret

The sprint retrospective began as usual. During the sprint, the team had many discussions regarding our processes and practices. For this reason, I knew that there would be some passionate points brought to the table. The process we ran was to ask two simple questions: “What worked this sprint?” and “What could be improved next sprint?” Each team member would then have a chance to speak (just like in the stand-up meeting) while others would remain silent or on occasion chip in very briefly. Giving each team member space to talk ensured that we all had a chance to speak. The team usually found it difficult to avoid talking over each other and keep to the time box so I would often need to step in to politely get each member to wrap up their point.

“Okay. Who would like to start?” I asked.

Jon looked ready to go. “I’ve got a few things to say,” he said.

I replied, “Okay, Jon. Go for it.” I had an idea of what to expect from him.

“Firstly, planning!” he began. “Let’s cast our minds back to the past few sprints. We always discuss use of that software tool. We always say that it doesn’t work. Then, in each retrospective, we forget to mention it. I think we should stop using it. It wastes our time. It drives me nuts that we have to wait for it to load the stories. It’s quicker to write on paper. That’s why when people take notes during a meeting they jot down notes in shorthand. They don’t bring a computer into the room. We have got to ban that tool!”

Jon made passionate speeches that the team could relate to. This, coupled with the fact that he had been at the company for years, earned him a lot of trust from the team, who had been together under a month and knew nothing about scrum before they began at the company. They nodded their heads, looking to see my reaction.

I agreed with many things Jon said, but not all. I felt passionate. I was looking at things from many perspectives, not just mine or his. I fully respected the team but I knew that product owners at the company had a habit of, frankly, not doing their jobs. This may sound harsh and, believe me, I think product owners are often undervalued in scrum. However, many in this business were often anti-scrum and resistant to change. They were also usually very direct and tough characters (to balance that, I have worked with dozens of pleasant ones who embrace change).

The planning tool made it easier for the product owner to update the backlog from anywhere. However, no product owner was going to admit that their productivity would go down if the tool were taken away with Martha (our product owner) at the retrospective. Given that product owners were often absent, this would have introduced serious overhead on a critical project. The business would inevitably blame the technology department (as was usual on other projects). On top of this, at least one member of the team was permanently offsite. The tool allowed him to fully collaborate efficiently and to see the sprint board at all times.

After considering Jon’s point, I replied, “Okay, Jon, I do see what you mean. We can do without the tool. I started off in scrum without any software tools so I definitely know it can work. Just bear in mind that the tool does have some value. With it, the backlog can be updated from anywhere. Some of your colleagues who work from home can update it without asking you. It makes sure that we never lose our tasks. It automatically generates a burn-down. It also has a searchable history of all bugs, issues, tasks, and stories created since the beginning of the project. So, we can get rid of the tool but I want us to be mindful of the pros and cons.”

Claire spoke up. “Well, that’s not the point is it? We think it’s wasting time.” Claire’s response disappointed me. I had listened to the team and taken on board their points. I could see that some of the team members either did not think I was listening or were not listening to me, or both.

Within minutes, everyone was talking and expressing views out of turn. Only Judy and Sammy (the testers) were not talking. They had been the most open to change and had very different personalities than the others. They always used reason and were rarely angry or worked up unless someone went out of their way to annoy them. They also rarely openly disagreed with the developers about issues of process. Even when they disagreed, they would rather come talk to me rather than voice their opinions in the retrospective. The only time they did strongly disagree was if there was an issue of product quality (since it was their job to be vocal about that). Even in this case, they were polite at all times, which I respected.

The two testers watched quietly while everyone raised their voices, one after the other.

“Everyone, one at a time please! Gordon, it’s your turn,” I said loudly to get the room under some kind of order.

Gordon spoke. “Look, obviously the tool is a bottleneck. No one wants to see one person typing. Why don’t we just ban the tool from meetings? We can use it outside of them.” People reluctantly nodded.

“Good idea,” I said. “Do bear in mind, though, that we will have both a tool and paper. This will mean we will have to print the cards out to keep them in synch. My gut feeling tells me this will slow us down in the meeting if cards are not prepared, since we already have trouble starting on time as it is.”

Everyone started speaking over each other again. Martha, the product owner, got involved in the discussion.

“Oh, come on! It will be easy,” she said. “Look, I will do it. I will print out the cards and cut them out for the meeting!” The thing to realise about Martha was that she resented the times when I had constantly reminded her to maintain the backlog. Martha was unfortunately not well-liked at the company due to her short temper. I was clearly not her favourite person because I often chased her to make sure that she was doing her job as a product owner. Many other scrum masters did not spend the time to do this and it frustrated her. Because of this, she would rarely agree with me.

I could see the room was about to erupt again. I gracefully allowed the team to do things their way. I thought that if it were a bad decision, the results would make that apparent.

The meeting was about to end and only three people had spoken. The others made very quick points since they were pressed for time. We left the room with an agreement to stop using the tool in the planning meetings. This practice was to begin the very same day.

I started the planning meeting on time but the team and I had to wait for Martha to arrive. She was 10 minutes late, which was the usual case. I was aware that she often had a busy schedule so I never complained or embarrassed her about this. However, we could not start the meeting without her since she had the stories.

She arrived with printouts of the stories on A4 paper. As she had not found the time to cut the paper into individual stories, we started the meeting with doing that. This lasted 20 minutes. Martha then discovered that she had forgotten some of the printouts. Printing them took about 10 more minutes. Sprint planning that day finished without resolution due to a lack of time.

The team agreed that planning had been far too slow that day. For the next retrospective, we all agreed to print the story cards on an A4 sheet and not to cut them out. This worked out much better going forward.

**Lessons learned:**

The retrospective is a great opportunity for the team to drive their own improvements. As a facilitator, my approach is to ask the team what worked and what could be improved and leave it to them to come up with ideas. I usually give my coaching and suggestions at the end, answering any queries from the team. This story ties into the issue certain members of the team had with using a tool, but primarily it shows how retrospectives can be ruined or side-tracked when we become passionate and lose focus on facilitating the team.

- Allow the team to learn from their own mistakes and solve problems: In the story, I give the team good reasons why I think the tool is the right way to go but they are not convinced. In this situation, sometimes the best course of action is to let the team try their own solution. This is often the only way to give them peace of mind. Also, in the end they adapted and came up with a completely different solution, which also worked for them. I should have remembered that the team are a group of experts and problem solvers. Presenting the use of tool versus the use of paper as a problem to solve would likely get a much better response from them.
- Time-box the meeting strictly and suggest a new meeting for important points: This repeats a previous point within the book, but a very important one. Once we get in the habit of letting meetings run long, the process starts to become disorganised and loses its efficiency. In this story, only three people had adequate time to air their views. The others may have left harbouring frustrations that would carry into the next sprint. I should have either time-boxed the

meeting more strictly and booked a separate meeting, or asked the team if they were willing to sacrifice more time for the discussion. Getting them to decide and obtaining their buy-in would ensure that they leave the meeting without frustration.

- Transparency, or discuss any issues with your product owner: This ties in to my point about having a good relationship with the product owner. It is especially important for a scrum master (but the same goes for the team and stakeholders) to invest a great deal of time in the product owner. When we meet people with personalities we find difficult, it is easy to overlook their point of view. However, you may find that a friendly one-to-one discussion will bring you together as a team. You will notice in this story that I did not hold any separate discussions with Martha. However, Martha owns the backlog. If I had spoken to her and found a way to voice my concerns, she would have likely come up with her own helpful ideas and been more positive in the retrospective.

## Confession 7: The Bloated Bug Backlog

I had spent three months away from the company around Christmas and returned to work with a new team working on a new product. My programme manager gave me an overview of the task at hand. I would be taking over from another scrum master who was moving to a new project.

The departing scrum master gave me a complete overview of what had been done so far. He began to explain how we had an excellent development team who were hard-working, happy to be coached, and, in general, a pleasure to work with. The product owner had good ideas, made good calls on priority, and bought into the process for the most part, although he ran short on patience.

The backlog, however, was not in a good state. Various business stakeholders had pressured the scrum master to get the team to estimate parameters based on what they knew. The designs were not finalised at the time and the stories in the backlog did not relate to each other because there were a number of concepts not yet decided upon. The scrum master admitted that it would be worth facilitating a rework of each story or even starting from scratch. I admired and appreciated his honesty.

The very first task on my agenda was to meet the team. I did this the next day. As discussed, they were all friendly, if a little quiet. I then met the product owner and set up a meeting to make introductions and get his vision, straight from the horse’s mouth. We had a good discussion and I left with an understanding of the project’s history.

My third task was to start facilitating the rebuild of the product backlog. I kept any stories that were perfectly clear and removed those that I had been advised were out of date or completely incorrect. I gathered the key stakeholder and product owner to set them in the right direction and discuss the format of stories. I coached them on the format of story description: “As a <role>, I want <requirement>, so that <reason>.” I presented the acceptance criteria as: “A user should be able to <bullet points for acceptance criteria>.” These are points I have spoken about in my blog (http://www.pashunconsulting.co.uk/blog.html).

Within three weeks, the backlog had all the epics in place and the team were in a good position to estimate. During this time period, I was also coaching the team on scrum. This went smoothly and the team were receptive. I was very much enjoying the process. However, as you have probably guessed, the real fun had not yet begun.

After a number of sprints of coaching, the team were churning out what were meant to be increments of the product each sprint. However, we had one big issue. These increments were not potentially shippable. The application would crash sporadically, small parts of the UI would disappear or overlap, and the app felt generally unstable.

I discussed this with the most experienced QA tester. He assured me that he had discussed the issues with the developer, who advised him not to raise bugs yet, as the app was in its early stages. The developer said that it did not make sense to raise the bugs until the user experience was further nailed down. Fixing bugs at this stage would be wasted effort and difficult.

This went against my understanding but I trusted the expert and decided to monitor for a sprint. I also made the product owner aware that he needed to make sure that any bugs raised after the sprint for previously built features were addressed as soon as possible. They needed to be brought into the backlog and prioritised along with everything else.

After another sprint, I could see two main problems. The first was that the team would have benefited from a stronger approach to quality. They were not building discrete features but whole pages with bugs, and not managing to get on top of the problem. The second problem was that any bugs raised after features were complete were logged in a bug backlog (good) but were not reprioritised in any future sprint (bad). The product owner, although asked to do this, was keener on building new features into the application. The team was expected to fix bugs without prioritising the bugs into the backlog so that the team would have time to work on them.

I thought for some time about how to solve the team’s problems of quality. In terms attention to fixing bugs, I called the team together for a meeting.

“As you may have noticed, our application has shown a few recurring bugs for the past few sprints and I wanted us to discuss and decide how we can improve the quality of our product,” I said. The team had a mixed reaction. Some felt eager to highlight that not every bug could be caught. Others were more concerned with the lack of automated testing. Still others did not think there was a problem.

By the halfway point of the meeting, we all agreed that the team could improve their practices around automation and test-driven development, and that bugs could and should be found earlier. We decided to continue the sprint and pay more attention going forward. This would hopefully help solve the quality issues. There were now, however, dozens of bugs in the bug backlog.

“But when are we going to get the bugs that are already there fixed?” asked Sammy.

“Well, Sammy,” I said, “the product owner is responsible for making priority calls so Orbury will need to work out which bugs are the priorities and get them into the next sprint. I will have a chat with him.” I wrapped up the meeting.

The following day, I booked a short meeting with Orbury. Orbury knew a lot about scrum and was far more helpful than Martha. However, he was not very experienced in dealing with bugs. He had not been prioritising the bugs into sprints as I had asked. I discussed the idea once more and emphasised the importance of dealing with bugs as they arose. He agreed that this was important and that he would prioritise them each sprint.

Three sprints passed and issues were still being found with older pieces of functionality. The team were not fully automated in terms of testing so this was an expected problem with such a complex application.

Of more concern was the fact that Orbury was still not prioritising the bugs into the backlog. He gave all sorts of reasons. His key reasons were not having time to look into them and thinking we could fix them in the final sprint. Despite my meetings with him, I had not been able to convince him of the value of prioritising bugs in sprints.

I thought long and hard about what to do. On the one hand, I did not want the team or product owner to deliver a poor-quality product. On the other hand, I was a scrum master, not a command-and-control project manager. If the product owner did not want to manage the bugs, then these would simply present an impediment to the quality of the product. This would cause him to learn from the mistake. After days of thinking this through, I decided to keep reminding him that we would end up with a poor-quality product, but not to confront him with anything else. I also decided that I wanted him to take responsibility for the backlog. I did this because I thought that the results of his own actions would be the best lesson.

Five sprints later, we had reached the final sprint. The team were concentrating on final testing and fixing showstopper issues. There would be a “go or no-go” meeting at which the senior stakeholders would decide whether to make the application live or not. The team had done an excellent job of fixing as many bugs as possible. They had also built all of the priority features, so the business were happy.

Gordon pulled me aside to tell me that he felt the application was not in the best state to go live. The bugs were too many and too serious. I agreed with him and agreed to impress that upon the stakeholders. However, as they had done with previous products, the stakeholders ignored the issues and decided to press ahead with the live application.

The application’s bug backlog contained a large number of bugs and issues but the team simply did not have enough time to fix them all and build new features in each sprint.

The priority decisions were slanted towards features and automated testing had not yet covered all previously written code. This meant that although everyone had worked hard and made the business happy, the application was by no means at the pinnacle of quality.

On launch day, the team and I reflected. We were happy with the effort we made to get the product to where it was, but we all collectively wished that the quality of the product was where we wanted it to be. As a scrum master, I felt that there must have been something that I could have done to improve the quality of the product.

**Lessons learned:**

In this story, I had to strike a balance between getting people to fulfil their role to ensure that the team maintained quality and avoiding doing other people's roles for them. I had to make some decisions. Scrum is a framework (as opposed to a complete process), which means that although it solves many problems, it also leaves it up to the scrum team to solve problems themselves. The following are decisions that I feel would have addressed our quality issues.

- Automation in the definition of done: Every team should agree on a definition of done up front. If the points in the definition are not fulfilled, then the feature is not done. I could have suggested and facilitated such a discussion around this in an early meeting. We could have agreed that test automation is part of the definition of done. Then we could have reflected on progress each sprint and made adjustments to ensure that no feature was passed until the definition was met.

- Bug review meeting: Following the experience above, I facilitated a bug review meeting within each sprint. A few key team members and the product owner would attend. The product owner would decide whether a bug should be tackled in the next sprint, left in the bug backlog for another sprint, or closed. The testers would already have assessed the bug by this point. This made the quality of the product visible to the product owner and forced a decision each sprint. It also gave him a scheduled time for his calendar. It vastly improved the quality of the product and the whole scrum team felt confident that they knew the state of the product.
- Design review meeting: In the story, one of the key reasons for holding off on raising bugs was the lack of a decision on design concept. These decisions are not meant to conflict with the agile design of the product. Rather than a big design at the start, the team needed a general concept. Stakeholder

decisions were holding this up because there were so many cooks involved. Therefore, I arranged a design review meeting with the key stakeholders, product owner, and select members of the team to ensure that decisions were made in time for the next sprint (if a related feature was in that sprint). This led to fewer unanswered questions and the team felt certain of what they were building and testing within a sprint.

- Product backlog-grooming meeting: Within scrum, it is well known that a product backlog-grooming meeting should be held regularly to ensure that the team are aware of new features, to allow the team to suggest new features, and to make sure that stories are split or amalgamated as needed. This is an excellent place for the team to highlight or suggest any quality-related stories. I have found that it vastly improves the quality of the backlog and hence the product.

Congratulations, you now have a set of confessions that are really retrospective stories and suggestions for improvement. They will surely help you to complete scrum projects on in the real world!

**Recommend on Twitter**
Click here to recommend this author on Twitter.

Recommend this author