Scaling Up - pushing Scrum out of its Comfort Zone

Richard Lyon & Marcus Evans
British Broadcasting Corporation
richard.lyon@bbc.co.uk

Abstract

Single team implementations are where Scrum is most comfortable. Having successfully adopted Scrum for single teams within our department we were now faced with the choice of dropping the use of Scrum on a large multi-team project or scaling Scrum up. We chose the latter, despite the fact that neither we nor anyone else seemed to know exactly what was involved. This is the story of how we dragged Scrum kicking and screaming out of its (and our) Comfort Zone.

1. Introduction

The Comfort Zone [1] is, in general, a place we like to hang out. Step outside of it and we're in the Learning Zone – this is a good place to be, if for no other reason then it should result in our Comfort Zone becoming bigger. When we're in the Learning Zone we may not feel completely at ease but we do feel as if we still have some control over our situation. Step too far out the Comfort Zone though and you're in the Panic Zone — not a great place to be because you'll sense a total loss of control and feel agitated and apprehensive.

Within our department, BBC Future Media & Technology, (FM&T), using Scrum [2] within single teams was in our Comfort Zone. We'd been using Scrum in our department for around two years. Of course, we were continually inspecting and adapting and trying to improve our application of Scrum but so far it had proved a success.

Whilst our teams worked on components that interacted with each other, the teams themselves worked independently. That was until a project that came along whose aim was to launch a highly strategic product for the organisation that would be backed by a sizeable marketing campaign. Nine teams would be building components for this product and could no longer work independently – they would have to work together closely under a single project structure in order to deliver a single product.

It didn't occur to us to stop using Scrum just because we were now in a multi-team project environment, nor did we have the time to consider other approaches. However, we weren't sure what we'd have to do differently now that we needed to scale up Scrum. We researched Agile and Scrum books, online resources and consulted domain experts but no coherent solution was forthcoming. The most concrete suggestion was the "Scrum of Scrums" but we were to find that this barely scratched the surface of what was needed. It almost felt as if Scrum on a multi-team project had never been used but Scrum wasn't owning up to it. So, just as our Comfort Zone with Scrum was using it within single teams we also felt that Scrum as process only had single team usage within its Comfort Zone.

2. Project Organisation

Our organisation, the BBC, is one of world's leading and most highly regarded and trusted broadcasters. Its funding is based on a licence fee model linked to television set ownership per household.

This project's aim was to make the majority of the BBC's content available on-demand and for free from a rolling seven-day window, via broadband internet. This product would be called "BBC iPlayer".

The project organisation was set up as follows:

- an external supplier providing the encoded audio video (AV) content, programme metadata and setting up and running the ongoing production processes to support these
- an external supplier providing the infrastructure and AV distribution capability
- an internal supplier (FM&T) providing the application
- a central team who acted as the client and co-coordinated the three main suppliers listed above.

978-0-7695-3321-6/08 $25.00 © 2008 IEEE
DOI 10.1109/Agile.2008.19

This experience report is from the perspective of the internal supplier, our department, FM&T. Within FM&T there were nine teams developing components for iPlayer. Most of these were existing teams who were well established with their use of Scrum. A couple of new component teams needed to be set up – but these were made of up of people with experience of Scrum. Some of the existing teams would continue to have clients outside of the iPlayer project. All teams were made up of a Project Manager who fulfilled the Scrum Master Role, a Producer[1], a Technical Lead and the team, (an additional 4 – 7 people).

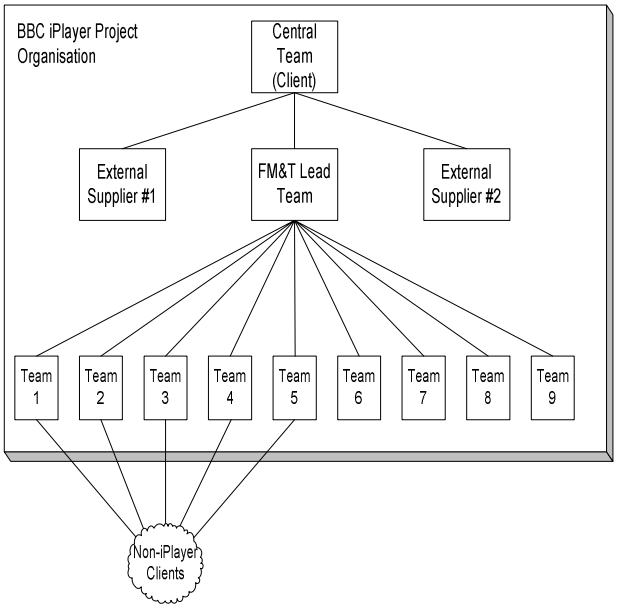

figure 1 : BBC iPlayer Project Organisation

3. Scaling, it can't be that difficult, can it ?

We were almost scaling Scrum by default and we didn't know how hard or easy it was going to be. But we assumed there must be right ways and wrong ways of scaling Scrum and some things that needed to be done – we just had to find out what these were.
We consulted online resources, the books [2] [3] and Agile experts but there was no coherent solution. The most concrete suggestion was to have a Scrum of Scrums which we duly started but this didn't feel like it was enough. We felt like there was more that was needed to pull the teams together and get them behind a single product. So what were our other Scaling measures ?

We aligned sprints. Some teams were running four-week sprints. Other teams ran two-week sprints and they were all overlapping. So we decided that all teams should run two week sprints concurrently that were all referenced the same way; Sprint 1a, Sprint 1b,Sprint 2a etc. This was done in order to give a common heart beat across the whole project, give us focus points for integration and help cross project communication, particularly around timescales, e.g "Sprint 4a" meant the same to everybody.

We created one Master Product Backlog that covered the entire project for our department, FM&T. We figured that as we were building a single product, we should have a single Product Backlog that was owned by a single Product Owner. So after a couple of sprints we combined all the teams' backlogs into a single backlog. This resulted in a huge Product Backlog.

We created an FM&T Lead Team. This oversaw and led the work of the nine component teams. It matched the make up of the component teams in that there was a Senior Technical Lead as well as a Senior Project Manager (a "Scrum of Scrums Master") and a Senior Producer (Richard Lyon and Marcus Evans respectively, the authors of this report). In time this team was supplemented by 2-3 extra people who ran the integration environments and supporting processes (integration testing, configuration management etc.).

We created a unified look and feel for the Scrum artifacts that the teams used to manage their work and report progress. This may seem like a trivial point but it was actually quite important. Although the department had been using Scrum successfully for some time, we didn't have a common toolset—each team created the tools they needed, generally in Excel. Therefore, we had all teams use the same tools and produce artifacts (burndown charts, sprint backlogs etc) that were formatted the same way. A lot more people were now taking an interest in these artifacts so we wanted them to be "branded" in the same way. It also helped create a common language between teams and emphasise that they were working on the same project.

4. Help! We're heading for the Panic Zone.

So we thought we must have Scaling Scrum covered with our initial scaling measures – after all we'd gone beyond what had been recommended (Scrum of

[1] An FM&T role that primarily covers product management but can also incorporate business analysis activities.

Scrums). So were we still in our Comfort Zone or at the least taking a short safe trip into the Learning Zone ? Well actually, some of the initial measures had us heading rapidly towards the Panic Zone.

4a) The Backlog

One of the key problems was the Product Backlog. It was the right decision to have a single product backlog but our problem was the level of detail in the backlog. By combining the individual teams' backlogs we had created an unwieldy monster. It was impossible for a single Product Owner to come to grips with or to use it to communicate to anyone the scope and priorities of our work. It was also rapidly ceasing to be of any use to the teams both in terms of a tool to direct their work and sprint planning and in the sheer mechanics of trying to share a single Product Backlog between nine teams.

4b) Product Ownership

Aligned to the Master Product Backlog issue was a problem with Product Ownership. The role of Product Owner had been thrust on to someone within the central (client) team without it really been explained to them or anyone else in the central team what Product Ownership meant. One of their principal tools, the Master Product Backlog, was ineffective and because there was only one of them, there just wasn't enough time in the day for them to give direction to the teams. [2]

4c) Breaking what we already had

We were breaking the Scrum engagement model for teams who had existing clients. The iPlayer project had taken over the teams' backlogs. But teams with ongoing clients outside of iPlayer still needed a Product Backlog with which to engage with these clients. So effectively we were forcing single teams to work from two separate Product Backlogs, one of which wasn't working effectively.

4d) Estimating

Our Agile estimation techniques [3] wouldn't scale. Teams were used to estimating Product Backlog items, usually using points. Teams had a known velocity, which after a few sprints was generally quite reliable. Under normal, single team, circumstances this allowed a Product Owner to prioritise work against Sprints and for the team to put together release plans. We've mentioned how the Product backlog was unwieldy due to the sheer volume of items in it, so adding estimation information to these items wasn't in any way useful.
We started to group together backlogs items under a feature to see if we could aggregate an estimate for a feature but this proved impossible. Why? Because teams were running at different velocities, without a common agreement of the value of an estimation point and some teams weren't using points as their unit of estimation, therefore adding together estimates under a particular feature would produce a meaningless number. If we'd succeeded in working out a common team velocity and a common understanding of the value of an estimation point , and found a reliable way to adjust team estimates to this common velocity in order to produce a aggregated estimate for a feature, would that have been any use ? No, because, the Product Owner may select a feature for a sprint based on its estimate and a common team velocity but the work for a particular feature is unlikely to spread evenly amongst all teams – therefore some teams could end up completely overloaded in a sprint and others have nothing to do. Perhaps there could be some way we could have found of weighting feature estimates according how the effort was spread across the teams – but this would have left us with a system so unfathomable with a ridiculous amount of manipulation needed on the original estimate provided by a team. So, the bottom line was that we were unable to provide estimates for the Master Product Backlog.

4e) Religious battles

As if we didn't have enough to deal with, we had to solve these problems while fighting religious battles. The central client team had attached to it a Programme Management Office (PMO) function. This central team and the PMO hadn't had any exposure to Agile techniques before. Interestingly, the rest of the Programme including the other two main suppliers adopted the two weekly sprint patterns we were using. However, whenever problems were hit as you would do on any major development project, rather than trying to collaboratively solve the problem, the first thing we'd usually have to re-direct our energy to was defending general accusations of "Agile doesn't work".

4f) Team-of-teams

The team ethos is difficult to scale. One of the main principles of Scrum is the team over the individual—that it's not individual success that counts but the success of the whole team. We tried to scale this up by

[2] For more detail on our problems with scaling Product Ownership, see Mike Lowery's and Marcus Evans's 2007 experience report "Scaling Product Ownership")

encouraging a team-of-teams ethos—that an individual team's success has no real value unless all teams were successful—and all teams would need to be successful in order for us to deliver a great product. One of the tools we thought would help with this was setting a joint goal for the sprint. Sprint goals always worked well with individual teams, helping them to maintain focus throughout the Sprint. But when it came to trying to set a joint sprint goal across nine teams, especially early on in the project, it proved difficult—two or more teams would often be working on completely separate areas of the product. As much as we tried to keep all teams focused on a common goal, teams become inward looking and all that matters is delivering what their team needs to deliver. Add to this the high-pressure environment that this project operated under, then a culture of blame between teams started. We kept trying to adjust this attitude across the teams but if we were honest with ourselves we realised that we demonstrated similar traits at a supplier level. The product wasn't going to deliver unless all suppliers were successful but what always mattered most to us was getting our deliverables out, not necessarily if the other suppliers did or not (unless we had a particular dependency on them). Most of the other problems described above could be addressed by inspecting-and-adapting in order to "fix" a particular process but this team-of-teams issue was down to human nature so was always going to be more challenging.

When we started to experience the problems with our initial scaling measures described above, it didn't mean that there wasn't productive work being done within the teams. But the longer these problems continued unchecked, the more chance we had of things completely unraveling.

5. Pulling back from Panic

The first thing we believed we needed to address was the Master Product Backlog as this was at the heart of the whole process along with the Product Ownership of the Backlog. It was obvious that we had to get this "right" even if it meant breaking Scrum rules that would normally apply in a single team application of Scrum.

We realised that the Master Product Backlog could no longer be an amalgamation of all the team backlog items. We generalised the backlog items together and grouped them under themes and also grouped by releases. These backlog items were no longer constrained by being delivered by a single team within a single sprint. We also realised, for reasons explained above, that this backlog could not contain estimates. The Senior Producer from our integration team became the Product Owner of this backlog, (the "Master" Product Owner). We stopped trying to pin the role of Product Owner on someone in the central (client) team—instead we asked them to express their priorities to us which we confirmed back to them in the Master Product Backlog. This proved effective, particularly when there were conflicting priorities coming out of the central team as it made it clear what we believed the priorities would be and what our teams were planning to.

Each team now had control back of their own team Product Backlog and a Product Owner within the team. The only project-specific rule was that every item in their team product backlog had to be linked back to a Master Product Backlog Item. To get around enforcing this rule for teams with clients outside of the project whilst allowing them to maintain a single team backlog, we added a single item to the Master Product Backlog entitled "non-iPlayer work". Besides that, the teams ran by the usual Scrum rules. The team Product Owner could set his or her own priorities within their team backlog. The Master Product Backlog served to inform them of what the wider project priorities were.

We still had to work out how to present estimation information. Teams would be asked to estimate all of their backlog items linked to Master Backlog Items that were tagged with a particular release number. They would then schedule these against upcoming sprints based on their team velocity. We then found that the most convenient unit of measure to express this combined estimation data from all of the teams was time (or number of sprints). This would be expressed as a graphical timeline, or dare I say it in a pro-Agile article—a Gantt chart view. It was the easiest way for all stakeholders to absorb this information in a way that they were familiar and comfortable with. But also for us within the Lead Team and even for communicating back to our own teams, this view of the overall project was the easiest to digest. When working within a single team using a simple list view, (i.e. the Product Backlog), does suffice in communicating a overall view of a project or release but we had realised, even after reducing the number of items in the Master Backlog, that the same wasn't true of a multi-team project. The key was that all the teams had their backlog items linked to a Master Backlog item. This allowed the graphical timeline view of the project to be generated relatively easily, which could then be rolled up by Theme, Release or Master Backlog Item. Including the "non-iPlayer" work in this

view also proved useful – it allowed the FM&T Lead Team Product Owner to see how non iPlayer work within the teams was affecting iPlayer timelines and then question and understand a team Product Owner's prioritisation.

We gave up trying to set a joint sprint goal across all teams. Instead we concentrated on making sure teams understood the priorities in the Master Product backlog. Towards the end of each sprint we'd have a joint planning meeting with the Product Owner, Scrum Master and Technical Lead from each team and the FM&T Lead Team. The aim of the meeting was to go through the priority items in the Master Product Backlog, confirm how the work was allocated between teams, identify team inter-dependencies, and if necessary allow the FM&T Lead Team Product Owner to change priorities. We originally thought this meeting would take up a whole morning, particularly to get to grips with the dependencies. What actually ended up being needed was a one hour meeting – and sometimes we'd even finish early.

Over time the "religious battles" of Agile vs a traditional approach died down as we and the PMO learned to live together—the production of Gannt type view into our planning helped. In retrospect spending more time at the start of the project educating the central team about Agile approaches and us understanding what their requirements were in terms of corporate reporting and governance may have helped. What we didn't foresee was that working on such a high profile and large-scale project would result in all levels of the organisation delving into and analysing in detail all aspects of the way we worked, including Agile. In future, on similar sized projects, we'd try and ensure that all debate around what approaches were being used were wrapped up and (formally) agreed at the start of the project so that when problems arose we could collaboratively solve them rather than replay unconstructive debates around Agile vs a traditional approach.

End of Sprint demos really began to work for us. In these show-and-tells all teams would come together for a couple of hours on a Friday afternoon and we'd have a walkthrough of the emerging application. As we went through the user journey the demo would move from team to team as each component was called into play. Almost as important as the demo was the chance to get everyone working on the project together so that they could get a sense of the wider project and grab a beer and socialise with people outside their immediate team. This helped everyone feel part of something exciting and went some way to addressing the team-of-teams issue.

6. Are we back in the zone ?

Our measures to pull back from the Panic Zone worked. They put us back in the Learning Zone and by the end of the project we had scaled up Scrum firmly in our Comfort Zone.

Towards the end of the project, (around 18 months after we'd started it), we also re-examined Scrum as a process to see if its Comfort Zone had grown beyond single teams. Whilst there was more in the Learning Zone in terms of experience reports, blogs and some new publications that dealt with scaling, it still felt like Scrum as a process was still only comfortable with single teams.

7. Conclusion

Scrum can scale successfully. We've found a way of scaling that works and that is a repeatable process for other multi-team projects we run.
In general, the larger something gets the less agile it becomes and the same goes for Scrum. You do lose some agility on a multi team Scrum project and you do need to break some Scrum rules on some of the scaled up elements of Scrum.
One of the advantages of Scrum is its realistic and effective approach to managing change. Scaled up Scrum also gives us an effective framework for change but the effect of drastic change is amplified much more and therefore more disruptive than it would be for a single team.

What's less certain is whether an organisation has to first take a trip to the Panic Zone in order to make a scaled up Scrum work for them. Ideally organisations shouldn't have to do this but that would mean extending Scrum as a defined process. There are at least two issues with this;

i) There may not be a one size fits all approach for scaling up Scrum – in fact there probably isn't. Our approach could be replicated in other organisations with a similar component team structure but there are most likely flavours of organisation or project where it wouldn't work.

ii) One of the key factors in the success of Scrum is the simplicity of its rules. Once you start adding more rules and becoming more prescriptive then this simplicity and therefore the continued success and adoption of Scrum

could be threatened. So if a simple straight forward solution to a challenge within Scrum adoption can't be found then maybe the best thing for Scrum is for it to ignore that challenge.

Even if these two issues can be resolved, then how is a scaling up Scrum approach made "official". So maybe for now we'll all just have to continue with the thrill of the ride into the Panic Zone.

8. References

[1] http://en.wikipedia.org/wiki/Comfort_zone
[2] Schwaber Ken and Beedle Mike, Agile Software Development with Scrum,
Prentice Hall, 2002
[3] Cohn Mike, Agile Estimating and Planning,
Prentice Hall PTR, 2006