Chapter 17

Scrum of Scrums as a Scaling Model

One of the greatest gaps in the guidance that Scrum creators and community pundits have left open relates to scalability. That is to say, scalability at multiple levels: across products, projects, large-scale and distributed teams, release planning, and product road-maps. There *is* the nearly infamous Scrum of Scrums that has been defined. But it provides guidance in very high level and general terms, while providing little tactical, day-to-day help in scaling your Scrum instance.

It's basically a hierarchical view towards Scrum dynamics. Ken Schwaber, Mike Cohn, and Jeff Sutherland are some of the Scrum thought leaders that have referenced it but, beyond a simple diagram[75], as seen in Figure 19, and some hand-waving surrounding the concept, Scrum teams are pretty much left on their own when it comes to scaling. I thought it important to at least try and put some additional guidance around the concept based on details from my own implementations of the Scrum of Scrums.

You might ask why does a Product Owner need to understand this; aren't they focused solely on their team and backlog? Well, yes and no. If you're only working in a single team or in a very small organization, then yes, you can probably skip this chapter. But for those of you working as a Product Owner in an at-scale Scrum instance, coordinating your product releases with other Product Owners, then this model can be quite helpful in fostering cross-team collaboration and planning.

---

[75] I got this diagram from Mike Cohn's website. It's been used for years as a graphic that quite simply illustrates the Scrum of Scrums concept. Here you see 3 levels from my perspective. You can find it here - http://www.mountaingoatsoftware.com/scrum/team/

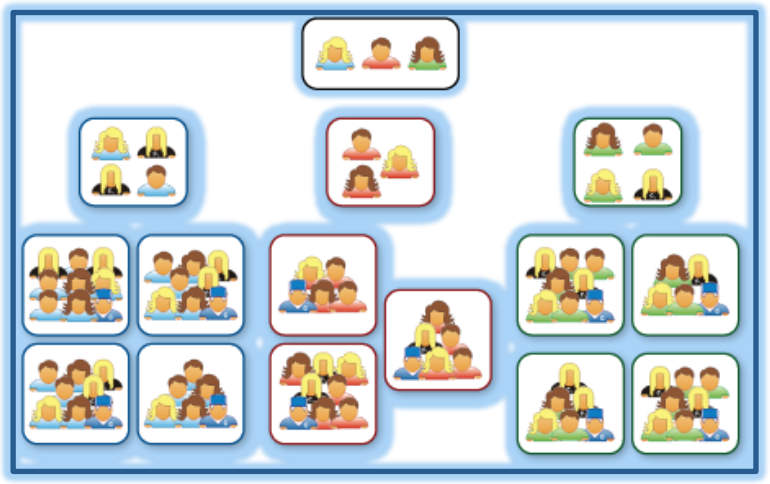

Figure 19, Example of 3-tier Generic Scrum of Scrums [76]structures

Scrum of Scrums implies that you run Scrum at *multiple levels* in organizations where you have larger scale development. If you have multiple Scrum teams, then the Scrum Masters of those teams meet frequently, using much the same Scrum individual team dynamics to coordinate across their respective teams.

You create a higher level, Meta-product backlog for the Scrum of Scrums that drives work across the teams. From a roles perspective, there will be a Product Owner of Product Owners, call him/her the *Chief Product Owner* for that backlog. There will also be a Scrum Master of Scrum Masters, call him/her the *Chief Scrum Master*[77]. They will lead periodic stand-meetings

[76] This picture is from Mountain Goat Software and Mike Cohn. It's referenced from the Scrum Alliance site in the following article: http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting

[77] To the best of my knowledge, there isn't agreement within the Scrum community as to what to call this second tier and beyond Product Owners and Scrum Masters. I've heard 'Chief' used as well as 'Meta' in some contexts. I'm sure there are others.

In my experience, these roles are often assumed by functional leaders, senior managers, directors or above. Or they are assumed by project managers. Often these roles are the 'glue' between traditional project tracking mechanism and the agile teams.

with, perhaps, slightly different questions and, on the whole, it will essentially be all of Scrum – just *UP a level*.

There will typically not be sprint reviews or retrospectives that represent these aggregated team results. Instead, the individual team reviews need to mention x-team interactions and dependencies and progress towards everyone's overarching release goals.

In my experience, the Scrum of Scrums stand-ups don't necessarily occur on a daily basis. Instead, I often see them occurring once or twice a week. Yes, there is an effort to keep them to the obligatory 15 minutes, but if they run over, that's usually fine. You want to ensure you're surfacing global release state across all of your teams; so take as much time as you need to fully collaborate around your releases.

iContact Scrum of Scrums Tempo Story

At iContact, we would oscillate our Scrum of Scrums quite a bit. It related to our release planning and our release cadence. Around the time we were planning the next release, we would scale back on our Scrum of Scrums; normally to once a week. The meeting was then typically focused on release planning and release readiness discussions.

Once we had a committed release plan and started sprinting, we would move to our normal twice a week cadence. We chose Tuesday and Thursday, late in the day times. Since we were a *SaaS*[78] business, our Scrum of Scrums would focus on our release and on our next targeted "hot fix" package. So, members of our Operations team would normally be in attendance.

As we approached our release target, the Scrum of Scrums attendance would increase, quite often doubling in size with interested parties trying to understand and support the upcoming release. We would also change the frequency of the meeting to a daily cadence somewhere between 1-2 weeks before our release. This made a lot of sense as our release content and preparation decision-making increased dramatically.

[78] Software as a Service

We would continue the daily Scrum of Scrums post release until the release stabilized; usually around a week or so. Then we would repeat the process. In essence, the Scrum of Scrums was our "one place" to coordinate our release efforts and our hot fix release stream. It was informative, incredibly well attended, and the best place to understand our progress, impediments, and next step action plans.

Mike Cohn has written *a nice article* [79] on the Scrum Alliance site that explains the Scrum of Scrums dynamic. In it, he changes and extends the basic daily Scrum questions for use in the Scrum of Scrums. Here are the questions he suggests:

1. What has your team done since we last met?
2. What will your team do before we meet again?
3. Is anything slowing your team down or getting in their way?
4. Are you about to put something in another team's way?

These questions certainly speak to broader, cross-team information sharing beyond the first level Scrum. Therefore, you can theoretically scale it upwards for any sized organization. While conceptually this sounds easy; it is, of course, the way you would scale any larger scale software work. The devil, as they say, is in the details of how to implement it within *your* organizational context.

This is where most of the traditional Scrum guidance stops; leaving implementation to each organization. The point is that every organization, beyond the small team agile sweet spot, has some need for Scrum of Scrum dynamics. They need to define, create, and then, refine it for themselves.

From a Product Owner perspective, I've seen multiple levels of defined Scrum of Scrums that you most likely want to participate in. Figure 19, is an example of the layering that I've found to be typical in larger scale Scrum instances.

[79] http://www.scrumalliance.org/articles/46-advice-on-conducting-the-scrum-of-scrums-meeting

More often than not, the layers and the focus points are strongly influenced by several key factors:

- The overall size of the Technology or Development organization.
- The number of Enterprise-level products being deployed, as well as, product lines being supported.
- The number of individual Scrum teams contributing (in parallel) to each of the primary product lines.
- The integration requirements across the separate products and product lines. For example: the look and feel, data consistency and sharing, regulatory guidelines, security, and consistent functional compliance testing.
- The existence of a Project Management Office (PMO) function and the negotiated requirements they have for initiating, tracking, and accounting for projects across the enterprise.

It's Not a Formal Organizational Structure!

Don't think of these layers in the same way you might an organizational structure. That would be a mistake. Instead, I like to emphasize their need by illuminating the *conversations* that naturally happen within larger scale Scrum instances. These conversations occur at *different levels*. For example, Scrum of Scrum (or S^2) conversations usually surround integration and dependency collaboration amongst multiple teams working together on an individual product or project.

Scrum of Scrum of Scrums (or S^3) conversations are at the next higher level of communication. Again, multiple teams are implied, and the conversations surround resource sharing, dependencies, code and component integration and system testing, along with product deployment. Often release planning is a part of the conversation, as is project budgeting and metrics.

Just as an example, I'm basing the structure in Figures 18 - 19 and 21 - 22 on a hypothetical organization where there are three separate products integrated into a *SaaS*[80] framework, with each succinct product having multiple (3-6) Scrum teams contributing application code to each. Given these characteristics, then a Scrum of Scrums framework along these lines *might be* a useful way to integrate Scrum deliverables and target them towards a customer release train model.

[80] Software as a Service

S^1 – The Scrum Team

At this level the conversations surround the individual Scrum team. The team, the Scrum Master and the Product Owner are the primary participants. Sure the Chief Product Owner can and should attend the daily Scrums and other ceremonies, but it's truly about the individual teams.

The focus is on sprint execution surrounding each team's product backlog. This is the place where most of the Scrum guidance is focused, towards the individual team, so I won't belabor the activities here. There is also an important focus on backlog look-ahead via regular refinement meetings.

From a Scrum of Scrums perspective, the team does need to 'care' about externals. For example, dependencies from this team to other teams and impediments they may be causing for other teams. They also need to be aware of the overall release plan and how they fit into it.

But at the end of the day, the team is "heads down" focused on delivering to their commitments.

General Focus of Activities	Product Owner Engagement
1'st level tier, focused towards product development; driving production of high value software components / projects from the individual teams. • **VISUAL: The team Sprint Board** • **VISUAL: The Product Backlog**	Individual Product Owner for each team; defining and driving a specific backlog of work. Demonstrating that work when completed.

Figure 20, Example of S1 level Responsibilities

S^2 – The Scrum of Scrums

At this level the driver is a release or other similar external delivery of software produced from multiple Scrum teams. The primary participants are the Scrum Masters and Product Owners from each team. And of course the Chief Scrum Master facilitates the Scrum of Scrums and the Chief Product Owner is a must-have participant as well.

All of the discussions surround each team's progress in supporting the current release goal and plan. Each *pair of team members* [81] speaks to their support of the goal; are they on track or off relative to their release plans and commitments? Do they have any distinct impediments or dependencies; either in their path or that they anticipate throwing in another team's path?

As the team approaches the release date, an important set of discussions should surround testing, mostly integration and regression, but all testing that is part of the overall release criteria. I would include in this transition talking about operational readiness and deployment activities.

General Focus of Activities	(Chief) and Product Owner Engagement
2'nd level tier, focused towards cross-team collaboration, sprint results integration, and coordination towards the next product release point. Close to the same dynamics as the Scrum teams; but driven by the overall release plan supporting the roadmap or meta backlog. Often collaborating with cross-team testers during full coverage integration, regression and other forms of *holistic testing.* [82] • **VISUAL: The Release Plan and the Release Burndown Chart**	Setup release plan so that the individual teams understand dependencies and expectations surrounding all work (integration, resource sharing, testing, deployment) leading towards the release. Helping teams to manage cross-backlog dependencies and deliverables. Resolving conflicts and issues. Tracking / resolving cross-team impediments. Connecting team 'state' to the Scrum of Scrums. Communicating progress to release goal 'outward' across the

[81] I think it's a very healthy sign when both the Scrum Master and Product Owner have the same views towards where the team stands. They can both report on "all aspects" of the teams' progress, impediments, challenges, needs, etc. And there is very little "disagreement" between them. It implies that they're working hard in their partnership and in collaborating with their team.

[82] Jeff Patton once described an Information Radiator model that indicated a Happy-Neutral-Sad Face for each component of a product from a testing perspective. Each day the testing team would update the component areas to the real-time state 'face' that represented their feelings. It was a wonderful way to drive cross-team conversation AND keep everyone's eye on the ball for the release. http://www.stickyminds.com/s.asp?F=S14400_COL_2

• **VISUAL: Dependency tracking list** • **VISUAL: Impediment tracking list**	organization. Ensuring the "release board" is accurate; includes all dependencies.

Figure 21, Example of S2 level Responsibilities

A Release Plan: Central to the Scrum of Scrums

An important distinction to make is that the S^2 usually relates to a planned project release. For example the following is already in place:

- An agile project charter
- A fully vetted and committed Meta-backlog the contains all of the work for a set of teams to complete
- The set of Scrum teams, fully formed to deliver on the work on their backlogs
- A release plan constructed with the team; where they have committed to a realistic "body of work" for a release date

The stand-ups are usually in front of your release plan. Discussions are relative to release goals, plans, dependencies, and assumptions. Progress is tracked across teams, so there is usually a release-level burnup (or burndown) chart that reflects feature level progress for the overall release. Typically the 'units' on this burnup chart are customer facing, committed features and not hours or stories, which are simply too much detail to effectively track at this level.

You're paying attention to cross-team progress, on a sprint by sprint basis, as it relates to your overall release plans versus expectations.

For example, it is not uncommon for the teams to fall behind. Quite often then, the discussion in the Scrum of Scrums focuses on whether other teams can pick up work so that the overall release goal is held. Having a well-planned release map can visually help make these sorts of adjustment decisions and/or guide the team towards scope adjustments.

Transparent & Visual

I personally like the model where the S^2 is a well-publicized event in a consistent location that can accommodate a large group. As I walk through an organization and get the inevitable questions around "How's it going?" I'll invite everyone to attend the Scrum of Scrums as a means of communicating release status information.

I also encourage the attendees to "speak the truth" at the Scrum of Scrums. You always want to be fully transparent with progress so that the entire organization can react to both good and bad news.
Another key is to keep the information on an information radiator in the common place. Figure 22 is an example of a *visual release board* [83] that would be used to track a release. You get an instant visual indication of release state and where challenges might be occurring.

Figure 22, Example of Visual Scrum of Scrums Board

[83] There's a wonderful website where I pulled this photo from. It is entitled the Visual Management Blog and provides some wonderful examples of powerful, low-fidelity visual boards for agile teams. Here's a link to a post on S2 visual boards: http://www.xqa.com.ar/visualmanagement/2009/08/scrum-of-scrums-making-it-visual/

George Dinwiddie has a wonderful example of a Multi-Release burndown example of an information radiator here: http://blog.gdinwiddie.com/2012/12/04/multi-release-burnup/

S³ – Project & Portfolio Planning

Quite a few years ago, I think it was in the 2007-8 timeframe, I was fortunate enough to coach a rather large client instance of Scrum. They had over 100 Scrum teams aligned around three primary product lines. It was virtually identical to the picture in Figure 19, but around 4x the number of teams.

This client had instantiated two additional 'layers' to the Scrum of Scrums model. They called them an S^3 and an S^4, again, abbreviating the repetition of Scrum.

The S^3 was focused beyond the current project. I likened it to be a portfolio management construct but it was more than that. First of all, the S^3 was a place for the Product-side and the Technology-side of the organization to collaborate on future plans. The product folks brought product roadmaps, ideas, and fairly aggressive forecasts to the table. The technology folks brought their team's skill sets, capacity, and capabilities to the table. They would also bring future hiring plans and budgets into play.

Quite often, cross-cutting concerns were a primary part of these discussions, for example, exploring how to effectively integrate UX, Enterprise Architecture, Shared QA, and Shared Deployment or DevOps within the Scrum teams.

Then, at a high level, the two groups would forecast out roadmaps and team plans (team composition, backlogs, team-to-product mappings, identity, skill-set alignment, architecture alignment, UX alignment, etc.) that aligned organizational teams towards product execution.

S^3 Logistics

The S^3 rarely meets on a regular basis. Instead, a meeting is normally called only when needed. For example, if you are halfway through the current release and getting ready to groom the backlog for a high-level plan for the next release, you might want to have an S^3 level meeting to discuss preparation for planning, staging, and kicking off work.

Another focus of the S^3 in this case might be instantiating the project in your phase-gate model [84]in gaining organizational leadership approval. Your S^3

[84] These models are connected to the phases of Waterfall process. For example, you might have an "approval phase or gate" between the requirements complete phases

team would then meet as appropriate and pull together the charter, business cases, and high-level estimates as appropriate for approvals.

Sometimes you don't need a meeting at all. I remember at iContact we would sometimes shift people from team to team based on our growth and attrition. In these cases, we would send out an email with the proposed changes to our S^3 list and wait a few days for feedback. Then, we'd simply make adjustments as needed; then make the change.

We tried very hard to collaborate between Technology and Product on all aspects related to our teams executing our release backlogs. You'll see more on this in the next chapter under the Triad discussion.

To the question of who engages with the S^3, the short answer is "it depends". Clearly this is a "management-level" meeting. So your product and technical leadership teams should be a part of this. I usually do not include the Scrum Masters and Product Owners directly in the discussions unless we have some point questions for them or want to get their advice on a nuanced change or adjustment.

Clearly the first folks who are apprised of S^3-level discussions, decisions and straw-man plans are the relevant Scrum Masters and Product Owners. In fact, they usually are the ones tasked to guide the changes within the teams.

General Focus of Activities	Chief Product Owner Engagement
3'rd level tier, intended to focus on staffing, resource sharing, product portfolio management, ROI, etc. from the various projects. Can also drive cross product line consistency, for example, look and feel, quality levels, and regulatory and security requirements. Often drives budgeting and overall product portfolio planning across a particular larger scale or enterprise-	Business case development, phase-gate control for beginning projects. Portfolio planning leading towards effective people & resource utilization and balancing. Defining overall product/project consistency goals: usability, quality, security, regulatory / certification, etc. Running planned Alpha/Beta

and software construction phase. Usually specific artifacts are defined and a meeting is held to review your readiness to "pass the gate".

level product(s) or product family(s). • **VISUAL: Usually a longer term road-map that spans multiple products (Portfolio)** • **VISUAL: Usually some sort of spreadsheet capturing teams, skills, capacities, and focus**	programs. Deployment strategies. Overall committed release plans and external coordination within cross-functional teams. External partner and 3'rd party integration.

Figure 23, Example of S3 level Responsibilities

S^4 – Agile Steering

The fourth level of the Scrum of Scrums model that I was exposed to at my client and have adopted in a wide variety of my own coaching experience, has very little to do with projects. The S^4 is effectively all about steering your agile methods. It focuses on defining and adhering to your practices; having baseline guidelines for your agile methods, your roles, and your tactics. It also revolves around your agile capabilities and establishing the training and coaching your teams might need to improve their execution performance.

There are many ways to steer agile adoption and transformation. So the S^4, while it tries to adhere to basic Scrum practices and behaviors, can take on three common flavors. I'll explore the Scrum variant next and the other two in the next chapter.

S^4 as a Scrum Team

In this instance, the S^4 operates as a Scrum team. You find a Scrum Master and form a team that will be focused on guiding organizational agile adoption and transformation. Clearly the team will need a Product Owner and a Product Backlog. Quite often, the members of senior leadership who are championing the adoption will serve in this role.

Next the team brainstorms the backlog of all activities required to instantiate, grow, and evolve Scrum. Then it's simply a matter of sprinting towards accomplishing these items. Often all of the ceremonies are applied; with the sprint review and retrospective being particularly useful in making the adoption progress and plans transparent across the organization.

Since agile adoption can be complex, often the backlog is sliced into initiatives, which is synonymous with releases. Goals are established so everyone has something to "shoot for". And of course, as in any project, there are risks, dependencies, and adjustments that are required.

I've often seen a quarterly tempo used for work at an S^4 level. This usually aligns nicely with typical organizational transformations and executive reporting. It also allows some time to make significant progress between releases. There can also be the need for S^2 like meetings for making tactical progress and cross-team impediments more visible.

General Focus of Activities	Chief Product Owner Engagement
4'th level tier, in my experience tending to focus towards Scrum and Agile adoption strategies and training across the entire organization. Leading in defining 'standards' or guard rails for cross-team Scrum utilization. Perhaps including focus groups for Scrum Masters and Product Owners.	Chief Product Owner is fully engaged here at a leadership level focusing on Enterprise-level goal setting, defining long term agile evolutionary road-maps. Participating in or delegating individuals to help 'steer' agile adoption Effective staffing and training strategies of Product Owners are also defined here.

Figure 24, Example of S4 Level Responsibilities

A Final "Scrum of Scrums" Story

A short time ago, I had a development organization in St. Petersburg, Russia that was split up in two scrum teams to work on a couple of products. These products were very different in functionality, but shared a common UI, data storage, and communication layer. The teams shared a Test manager, a Research and Development manager / Scrum Master, a Business Analyst, and the Product Owner. The majority of the other team members were assigned to one of the two teams.

A few of them were assigned to a team on a release basis (4 sprints) or a sprint basis depending on the technology affected by the release or the focus of the sprint. For example, sprint #4 was typically focused on getting

the product ready for release to one (or multiple) customer(s). This sprint, therefore, always had more testers than other sprints.

We would skew the releases and sprints so that we would have the maximum amount of time between sprint end/beginning and the same for the releases. This way, we could shift people for specific sprints with minimal impact on the other project. We were also able to focus better on the needs of a specific team. The Product Owner, the Scrum Master, and the business analyst would participate in product demos, sprint reviews, and sprint planning for both products.

We had daily stand- up meetings where the entire team would participate, as well as, weekly meetings to discuss the integration of technology and plans with the Scrum Master, test manager, and architects. In this meeting, we would make decisions regarding higher level architectural or strategic directions. In general, these were things that would either not affect the team in the next couple of sprints, or, they would affect both teams equally.

We (the Product Owner, Scrum Master, and sometimes the business analyst or architect) additionally had regular meetings where we would discuss process optimizations, technology needs, and/or lessons learned. These meetings were held on a regular basis, but without a scheduled time. Most meetings were scheduled whenever there was a need – usually, every 2-4 weeks. Follow up meetings were planned to work out details and tracking changes to ensure that all changes were a step forward, rather than a step backward.

Furthermore, we had quarterly planning meetings with representatives of Sales and Marketing, as well as the executive team to track our progress, refine plans and to adapt to changed business needs. This was the time to get a formal “thumbs up” stating that we were still on the right track (or, if not, to change accordingly).

Having thought about this more, I realize I was having four levels of Scrum of Scrums :-). It just never felt as structured or rigid as it seemed on paper…

—Michael Faisst

I like Michael's story because it emphasizes the collaborative looseness and application of common sense that needs to be an integral part of your Scrum of Scrums definition. It also helps illustrate the different conversations that are essential for effective collaboration.

Wrapping Up

Is the Scrum of Scrums always the answer for Agile & Scrum scaling? I don't think so. But it can be effective in many contexts.

I remember talking to a fellow coach from a large company that was a competitor of ours at iContact. One of the reasons that they chose to move to Kanban was that their Scrum of Scrums were basically dysfunctional. It was not helping them with cross-team collaboration nor effectively integrating to their Operations team for *product deployment*[85].

As we discussed the dynamics of what went wrong, it became clear to me that the Scrum of Scrums hadn't failed. It was his organizations culture and their implementation of S^2 activities and behaviors surrounding it that were flawed.

For example, key contributors didn't bother showing up at the Scrum of Scrums. And they didn't lead it with holistic release planning. There were other problems, but in general it wasn't an S^2 problem, it was a planning, collaboration, and commitment problem. We realized together that their move to Kanban wouldn't fix these problems, but it might shed a different light or view to them; so the organization could improve.

Point being – implementing an effective Scrum of Scrums hierarchy can be incredibly hard. However, it can also be the difference maker for project results at scale.

[85] The product in this case was a SaaS