SCRUM HANDBOOK

MULTI-TEAM SCRUM (MTS)

Example of Scaling

Structural Patterns

Governance Patterns

Managing Transformations

Dan Rawsthorne and Doug Shimp

> *"We make sense of the world through mental models, not just accumulating facts. Mental models help us recognize patterns. A good model allows us to see different things every time we look at it! The best mental models are simple, flexible, and open enough to capture complex situations and encourage us to see more, and ask better questions."*

- Marc Applebaum, PhD

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where these designations appear in this book and the authors were aware of a trademark claim, the designations have been printed in initial caps, all caps, or with appropriate registration symbols.

The authors have taken care in the preparation of this document, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained herein.

'Get To Done' is a registered Trademark of Get To Done, LLC

'Scrum Dictionary' is a registered Trademark of ScrumZone.org, which is an organization focused on Organizational Scrum: Single-Team and Multi-Team

Book Design by Tuna Traffic, LLC

This Scrum Handbook ©2018, 2019, 2024, 3Back LLC,

Offered for license under the Attribution Share-Alike license of Creative Commons, accessible at http://creativecommons.org/licenses/by-sa/4.0/legalcode and also described in summary form at http://creativecommons.org/licenses/by-sa/4.0/. By utilizing this Scrum Guidebook you acknowledge and agree that you have read and agree to be bound by the terms of the Attribution Share-Alike license of Creative Commons.

First Edition publication date: June 7, 2018
6th Version publication date: August 1, 2024

ISBN-10: 1720780557
ISBN-13: 9781720780557

SCRUM HANDBOOK

MULTI-TEAM SCRUM (MTS)

Dan Rawsthorne, PhD, CST
Chief Scientist, 3Back LLC
ScrumZone.org

Doug Shimp, CST
President, 3Back LLC
ScrumZone.org

Example of Scaling

Structural Patterns

Governance Patterns

Managing Transformations

Table of Contents

Introduction....1
Example of Scaling....3
Growing an Organization....4
Absorbing another Organization....10
Sharing Common 'Best Practices'....11
Exploiting Economies of Scale....12
Decentralizing some Client Contact....13
What we learn from Sam's Organization....14
Scaling is context-driven....14
Both Structure and Governance have to Change....14
The Result of Scaling is often a Team-of-Teams....15
There are Five Basic Scaling Problems to Solve....15
Summary....17
Structural Patterns....19
Purpose of the Structural Patterns....19
Building Blocks....20
Scrum Team....20
Pod....25
Group....28
Small Pod Structural Patterns....30
Pattern: The Pod's Flow Management Team....31
Pattern: Pod Scrum Master Team....44
Pattern: Scrum-of-Scrums (SoS)....47
Pattern: Test Lab Team....49
Pattern: Collaborative Work within a Pod....51
Pattern: Cross-Cutting Workgroups....53
Pattern: Community of Practice (COP)....58
Discussion of the Small Pod....60
Large Pod Structural Patterns....62
Notation for Pods....62
Pattern: Multiple Flow Management Teams....62
Pattern: Pod Scrum Master Team....67
Pattern: Test Lab Team....68
Pattern: Cross-Cutting Workgroups....69
Discussion of the Large Pod....70
Group Structural Patterns....72

Pattern: Leadership Team 72
Pattern: Group Scrum Master Team 75
Pattern: Cross-Cutting Workgroups 76
Discussion of the Group 78
Governance Patterns 79
Purpose of the Governance Patterns 79
Pattern: Use GOs to Drive 80
Pattern: Synchronized Daily Scrums 82
Discussion of the Scrum Ceremonies 84
Managing Transformations 89
Transformations are Chaotic 90
Pattern: Internal Transformation 91
Pattern: Agile Transformation Team 94
Discussion of Agile Transformations 95
MTS Glossary 97
Acknowledgements 101

This Page Intentionally Left Blank

Introduction

The purpose of this handbook is to describe Multi-Team Scrum (MTS) – what it is and how it works.

Single-Team Scrum is about a single Scrum Team producing results in an incremental, iterative, and agile way, in order to maximize the value it can provide for its Stakeholders… even though the Stakeholders are changing their minds, and disagreeing with each other, about what they want.

Many people. in Organizations bigger than a single team, have seen this happen, and have asked: *"How can I use Scrum to increase my Organization's capacity to provide value?"*

We call this the **Scaling Problem**.

Now, Scaling, in general, is defined as *"an Organization's response to a need to change capacity,"* which can either be an increase (scaling up) or a decrease (scaling down). In this Handbook we will be discussing primarily 'scaling up' or, simply, scaling. Even though capacity can be improved with better techniques and tools, we will consider that to be 'continuous improvement' and not scaling.

There are two versions of the Scaling Problem: 1) either the Organization is small and wants to grow, or 2) the Organization is large and wants to transform itself. In either case, you need to know what a large, successfully scaled, Organization could look like.

We have some beliefs, based on decades of experience:

1. We believe that a Scrum Team (a small, self-organized, self-contained, value-driven, and empowered group of people) is the best tool to use to solve hard problems;
2. We believe that the best way to organize a large group of people (more than one team's worth) is to create a team-of-teams along with empowered, accountable leadership; and
3. We believe that incremental, iterative, lean, and agile processes, used with knowledgeable stakeholders to provide feedback, produce the best results.

So, here's what we're going to do here:

1. First, we're going to go through an example of Scaling, watching "Sam's Web Services" grow from a one-person shop to an Organization of 50. Along the way we'll explore the forces on Sam, and the decisions he made along the way, in order to grow his company; and
2. Second, we'll identify the Patterns typically used in a Scaled (or Scaling) Organization, and note which forces are being resolved by the Patterns as we go.

In this Handbook we will rely on the description of Single-Team Scrum that is found in the *Scrum Handbook: Single-Team Scrum (STS)*; and we implicitly and explicitly rely on that description as our jumping-off point.

Ok, let's go…

Example of Scaling

Scaling is an Organization's response to the need to provide more capacity. Understanding the scaling problem is important when looking at an Organization that is becoming, or has become, big. The best way to understand it is to go through it, but hearing a story is a close second…

Purpose of the Example

We are telling you a story about a small Organization 'scaling up' for a number of reasons:

- You need to understand that scaling is context-driven; that scaling is a series of 'moves', each of which has a reason.
- When you understand the reasons for scaling, you are more likely to be able to scale your Organization – it won't look like magic to you.
- Perhaps most importantly, you will realize that scaling is not simply applying a large, structured, process diagram as a template for your Organization.

Growing an Organization

Let's say that Sam is a One-Person Shop providing web services to Clients as "Sam's Web Services". Sam's job is reasonably simple. Sam wears a jaunty hat, talks to Clients, figures out what they want, and gives it to them. Sam also 'keeps the books' and manages the money. Sam is in complete control, and life is good. Easy peasy…

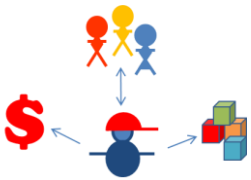

Figure 1: "Sam's Web Services" – a One-Person Shop

But all is not perfect. Maybe there are new Clients Sam wants to work with. Maybe Sam wants to deliver Results faster for existing Clients. Maybe Sam is ambitious and wants to grow and make a lot of money. Maybe Sam is just getting tired and worn out and needs to offload some of the work. Whatever… In any case, Sam decides to hire people to help – the One-Person Shop is going to become a Team.

So, Sam hires Josie to help. Sam is still talking to the Clients to figure out what they want, keeping the books and managing the money – now including payroll. But Sam has Josie to help deliver the web services to Clients, and Sam is teaching her all his cool tricks. Sam and Josie, together, can now deliver more Results faster.

This is great! So, Sam decides to hire some more workers and become an even bigger Team.

As each new worker comes on board, Sam's team adds more, and different, capabilities that allow it to deliver even more Results even faster. But it comes at a cost. Since Sam is the *only* manager, Sam will need to manage and train the workers (and he has lots of cool tricks to teach them); and this takes time. Sam will need to make sure that everybody on the Team is delivering the right Results the right way, and this takes time. Sam is not only doing work, but he is busy mentoring and coaching to get things running smoothly; and this takes time.

So, Sam decides to move to Scrum. His hope is that the self-organization he will get with Scrum will take some of the load off of him. He appoints himself as the Team's Product Owner, and appoints Josie to be the Scrum Master. At this point, Sam's Organization looks like this.

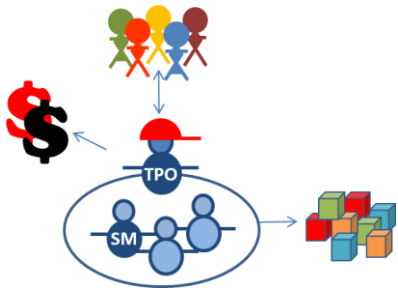

Figure 2: Adding Workers and becoming a Scrum Team

Now that things are running smoother, a new problem emerges. Sam's team is producing Results so fast that it will need new Clients to do work for; it has increased its capacity significantly. Since Sam is the Client interface, this will take even more of Sam's time. Sam is getting overloaded, and needs to offload some of the work… again…

Sam likes working with the Clients, and likes working with the Team, so Sam decides to outsource the Financials. Sam hires Fred to keep the books, do payroll, and manage the money. Sam's Organization has *scaled*; it has adapted to growth. Sam doesn't have to spend as much time as he used to managing the Financials; he just checks in with Fred for a few minutes each day.

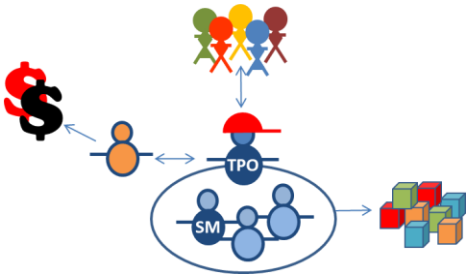

Figure 3: Outsourcing the Financials to Fred

Things keep going well, so Sam keeps hiring more workers – and can handle the growth for a while because of offloading the Financials to Fred. However, Sam eventually gets overloaded trying to manage both the Clients and the Team. Luckily, his original hire, Josie, has got so good, and learned all Sam's tricks so well, that Sam makes her the Team's Product Owner. This requires another one of the Team Members to become the Scrum Master. We see this new structure in Figure 4, on the next page.

This allows Sam to focus only on Clients – which Sam really likes to do. Of course, Sam is still mentoring Josie, but most of the conversations he used to have with the other Team Members are now done by Josie. Josie takes care of Team, and Sam just checks in with her for 15 minutes a day to make sure they are on the same page about what should be done, and how to do it.

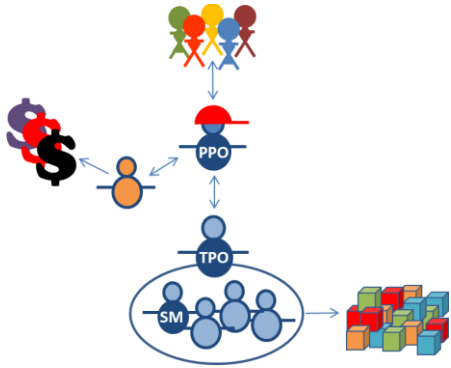

Figure 4: Sam has made Josie the Team's Product Owner

Business is still booming! So, Sam hires even more people to be on the Team. As this continues, Josie is eventually overwhelmed with workers, and the Team needs to be split in two – there will now be two Scrum Teams. Josie's been training them, so she identifies two people, George and Paul, to become the Teams' Product Owners, and the three of them pick the new Scrum Masters. Josie becomes the Development Lead, and sets up a Dev Mgmt Team with George and Paul. She works with them for an hour or so a day to make sure everybody is on the same page about what's being done, and how.

She also talks to Sam for 15 minutes a day, so they're on the same page, as well. She spends most of her time working with the Teams so she can pass on Sam's tricks, as well as a few of her own; she knows that the biggest risk will be dilution of quality…

Note: Josie is worried about Quality because the push for 'more' often becomes a push for 'faster'; and 'faster' often becomes a push to take shortcuts that usually cause a drop in quality. Dropping quality to increase throughput is a short-lived and dangerous game, as you always wind up paying for this lack of quality sooner or later.

We often see the arguments for dropping quality indirectly presented as a need for 'efficiency'. We hear managers say things like: "We need to be more efficient in coordinating customer needs," "You're spending too much time on testing," "We don't have time for training," and so on. Managers who say these things are ignoring the long-range impact of the short-range view, and this drives up total cost of ownership.

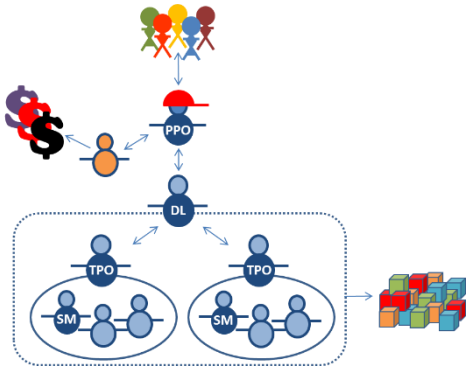

Figure 5: DevTeam splits because of overloaded Team Leader

What Sam's Organization has done so far is typical; growth keeps putting pressure on individual manager-types (Product Owners and others) until they are stretched so far that something has to give. This pressure forces the Organization to change, usually by splitting things up and creating some sort of *structure* of people and nested and inter-connected teams.

Sam's structure *looks like* a top-down hierarchy of manager-type people. Such hierarchies are usually used to project power and control from top to bottom, and we often refer to them as 'power' hierarchies.

This is not that.

Each of these nodes represents a collaborative team, and the overall structure actually represents a collection of interlocking collaborative teams. This is a 'purpose' hierarchy, designed to manage and provide value, and the interlocking teams provide distributed management and decision-making.

This distributed management and decision-making, along with the associated communications, is called the Organization's *governance* mechanism. In this case the governance mechanism consists of the conversations Sam is having with Fred and Josie, the management meeting Josie is having with George and Paul, and the self-organization that happens within the DevTeams.

An Organization's governance mechanism has two primary purposes:

1. Assure that everyone is *on the same page* about what to do, and who is doing it; and
2. Assure that the people who are doing the work are using a shared Standard of Care and *working together harmoniously* while managing dependencies and common issues.

When we talk about Scaling, we are referring to the changes in Structure and Governance that enable successful growth. Thus far in our story about Sam's Organization, the Scaling has been forced by the overloading of managers caused by hiring more workers.

This is a common reason for Scaling, but not the only reason for Scaling. Let's look at some other, representative, examples.

Absorbing another Organization

Work keeps coming in, and Sam tries to figure out what to do to handle it. Luckily, Sam has a friend, Sri, who also wears a jaunty hat and has a small team that supplies web services. Sri's Team has just lost three large clients, and needs some work to do. So, Sam makes a deal with Sri to have Sam's Organization absorb Sri's Team as a complete entity; Sri's Team still manages its existing Clients, does its own Financials, and so on, but Sam can offload some work to Sri. Everybody is happy for now…

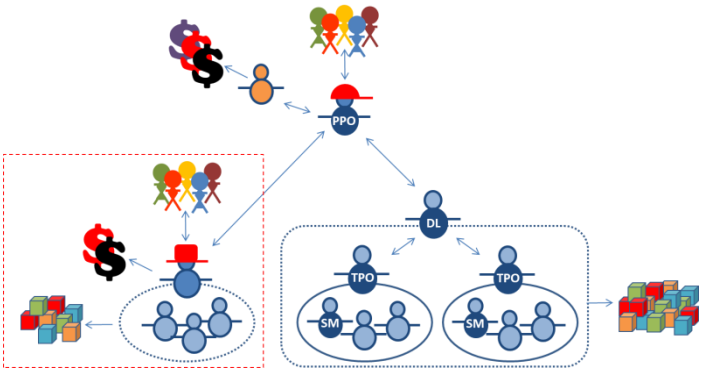

Figure 6: Absorbing another Organization

Sharing Common 'Best Practices'

Now, Sam and Sri do things in slightly different ways: Sam has some tricks he teaches his people, and Sri has some tricks she teaches to hers. In order to make sure they can get the best out of the entire organization, they set up a cross-cutting virtual team that will 'own' commonalities, standardization, and the Patterns they will all use. This virtual team consists of a member of each of the three development teams; they work together to:

- discover, gather, and agree on things that should be commonly known,
- document and assemble a library of good Patterns and standards to follow, and
- train everybody on the commonalities.

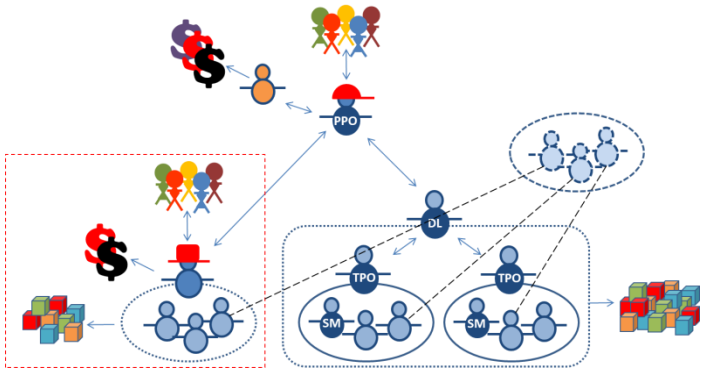

Figure 7: A Virtual Cross-Cutting Workgroup for 'Best Practices'

Exploiting Economies of Scale

Eventually, after Sri's Team has become comfortable, Sam and Sri jointly realize that it is inefficient for each of them to be doing Financials – the Financials should be centralized. This combining of their Financials is an example of exploiting the Economies of Scale by centralizing common work and functions. Organizations often exploit the economies of scale by centralizing things like HR, Legal, Contracts, IT, and so on… the resulting consistency often provides an additional, unexpected, benefit – better customer service.

Sri is also uncomfortable working with Clients (maybe that's why Sri lost some of them… just sayin'…) and wants to concentrate on the technical stuff. So, Sri's whole Team is going to move in with the other DevTeams, with Sri serving as its Team Leader, and Sam will handle *all* the Clients.

The final result of these changes is the following structure.

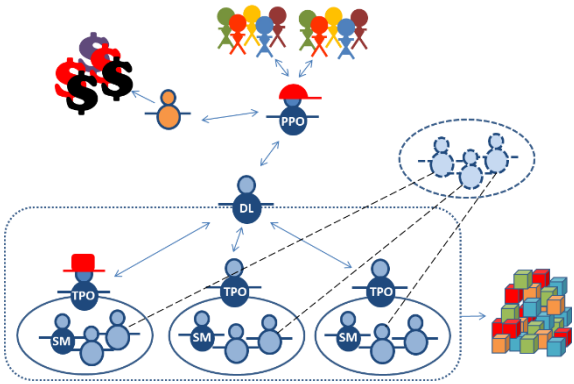

Figure 8: Refactoring and taking advantage of Economy of Scale

Decentralizing some Client Contact

It turns out that the Organization went a little bit too far in its centralization, though. Some of the Clients are complaining about reaction time and the distortion of information that happens by relaying too much information through Sam; these Clients are now too far removed from the DevTeams they should be working with. The Organization adapts by de-centralizing a little bit: Sam will focus on future/prospective Clients, and the Scrum Teams will work with their current Clients directly.

And here's the final result…

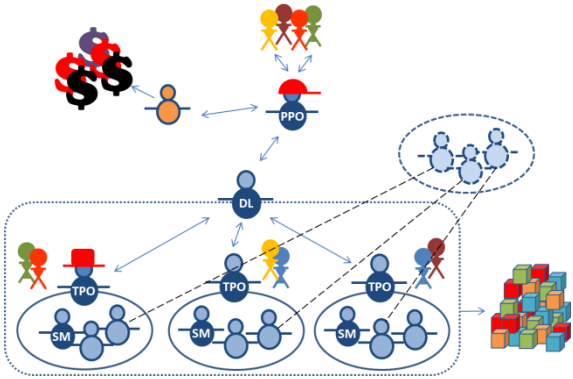

Figure 9: De-Centralizing Client Contact to improve Reaction Time

Wow! How Sam's Organization has grown!

What we learn from Sam's Organization

There are four main things we can learn about scaling from this Story about how Sam's Organization grew:

1. Scaling is context-driven;
2. Both Structure and Governance have to Change;
3. The Result of Scaling is often a Team-of-Teams; and
4. There are Five Basic Scaling Problems to Solve.

Let me discuss each of these learnings…

Scaling is context-driven

As an Organization scales, the *actual* actions it takes are highly context-dependent and influenced by the 'forces' within and surrounding the Organization – there is no recipe for scaling, only guidance and Patterns to use.

For example, when Sam was overloaded, and needed to offload some responsibilities, he decided to stay with the Customers, and let Josie take over Development. Sam could just as easily have stayed with the Developers and hired somebody else to manage the Customers, and this would have led to a completely different Organization as the Scaling continued…

Both Structure and Governance have to Change

As Sam's Organization grew, both its structure and governance had to change.

- The organization's structure changed in order to enable coordination, cooperation and collaboration – and not just between workers, but between managers, as well.

- The organization's governance mechanisms changed in order to make sure the 'right' collaborations take place. Guidelines and rules were established so that:
  - Team Members collaborate with each other,
  - Managers collaborate with their teams, and
  - Managers collaborate with each other.

The Result of Scaling is often a Team-of-Teams

In order to enable the conversations, coordination, cooperation, and collaborations mentioned above, the Organization often grows to become a Team-of-Teams; this is because small Teams are natural collaboration units, especially when they are self-organized.

Sam's scaled Organization is a Team-of-Teams; in fact, it has three pieces (I'll call them Pods) with Sam at the center of all of them: a Finance Pod, a Customer Service Pod, and a Development Pod made up of several teams. In general, we expect a Scaled Scrum Organization to have the general shape of a Team-of-Teams – but we must realize that a single person can be sometimes seen as a standalone Team…

There are Five Basic Scaling Problems to Solve

As mentioned before, the Scaling Problem is to *"increase an Organization's capacity to provide value,"* and Sam did this successfully with his Organization. In order to do this, Sam and his Organization had to solve the following five basic scaling problems:

- **Aligning the Organization on What** should be done, and who should do it. Sam accomplished this alignment with a hierarchy of manager-types, who worked collaboratively to determine which teams should work with which Clients.

- **Aligning the Organization on How** to do its work, so that there is consistency and compatibility throughout the Organization. This can be done in two ways:
  - Sam established a virtual 'Best Practices' Team to discover, document, and train the Organization, and
  - Even though Sam's Organization didn't do it, it is common to institute Practice Ownership that enables people in the organization to mentor others in their areas of expertise.
- **Leveraging the Economy of Scale** to remove inefficiencies and centralize common functionality. Sam centralized both financials and client management.
- **Decentralizing for Effectiveness** when centralizing goes too far. Sam had to decentralize some client management to the Teams; the Clients in active engagement with a Team needed to work directly with that Team.
- **Managing the Changes** as the Organization grew and morphed. Sam's Organization was small enough that Sam could own and manage the growth himself. In order to not overload himself, Sam empowered both the Teams and the Team Members to take ownership of local problems so that not all problems had to be solved by Sam.

Summary

The Scaling Patterns in this Handbook illustrate these learnings. Most of the Patterns are designed to help an Organization solve the Five Basic Scaling Problems, and suggest changes in both Structure and Governance.

The majority of the Patterns are Structural, showing how to create appropriate Teams-of-Teams to help solve the problems an Organization faces. The Structural Patterns are Team-Based, with particular emphasis on creating Scrum Teams to solve particular types of problems.

The few Governance Patterns provide guidance on how to manage the flow of work and lead the people within a Scaled Organization. The Governance Patterns stress People over Process, leverage self-organization, and focus on accountability, empowerment, and collaboration.

Because these Scaling Patterns re-use what we already know about Scrum, we call the overall concept **Scaling Scrum with Scrum®** (SSwS). In order to understand the particular 'flavor' of Scrum we are leveraging, see the *Scrum Handbook: Single-Team Scrum (STS)*.

Of course, the application of the Patterns is context-driven and influenced by the 'forces' within and surrounding the Organization. The Patterns must be properly applied and honestly used in order for them to be effective. Like Scrum itself, Scaling Scrum with Scrum will not work unless the people make it work; using Scrum correctly almost always requires Organizational Culture Change.

So, on with the Patterns!

This Page Intentionally Left Blank

Structural Patterns

(with Commentary)

Structural Scaling Patterns determine the 'shape' of a multi-team Organization. We will view Organizations as "Teams-of-Teams," and the Organization's shape is crucial because it determines the primary pieces of the Organization and the lines of communication between them.

Commentary: Throughout this section you will see boxes like this one. They are not part of the 'official' description, but contain advice, explanations, comments, and the like.

Purpose of the Structural Patterns

These Structural Patterns are used in two ways:

1. To help an Organization 'figure out' how to grow or, more commonly,
2. As guidance about how to re-structure an existing (large) Organization to make it more effective.

In either case, the result will be a Scaled Organization that was formed (or look like it was formed) based on specific Forces and Context. Ideally, a Scaled Organization would 'look like' it was formed based on the Forces and Context that currently exist. Because the Forces and Context are constantly changing, this implies that a Scaled Organization must be constantly changing to 'keep up'.

So, here we go…

Building Blocks

The multi-team Patterns in this Handbook are based on the principle that hard, complex, problems should be solved with teams, not heroes. This principle says that when you have a hard problem, you should: 1) choose somebody to be accountable for solving the problem, 2) find out who they need to help them solve the problem, and 3) put them all together on a Team with the accountable person as the Team Captain.

Scrum formalizes this principle when you want to solve the problem in an iterative, agile, way. Scrum says: 1) put such a Team together, 2) add a facilitator and a few feedback loops, and 3) you have a Scrum Team. Our structural Patterns largely consist of creating appropriate Scrum Teams, and assembling these Scrum Teams into Teams-of-Teams.

Scrum Team

The basic building block for all these Patterns is a Scrum Team, as we see here:

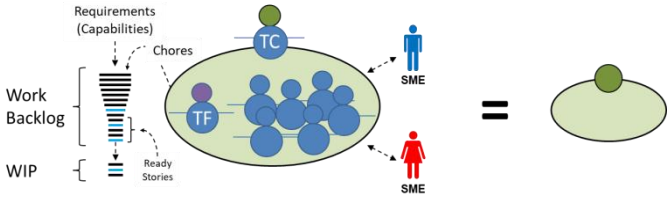

Figure 10: A Scrum Team

In order to understand the Scrum Team in its totality, you need to read the *Scrum Handbook: Single-Team Scrum*. However, the main things we need to know about a **Scrum Team** in order to understand the multi-team Patterns are:

- The Scrum Team does work based on **Requirements** coming from a **Business Owner**. This Business Owner could be working with Stakeholders representing a single product, in which case he or she is also a **Product Champion**; or the Business Owner could represent one or more Product Champions.

- A Scrum Team has a **Team Captain** (TC), who is accountable to the Business Owner for maximizing the value of the Team's work; owning the 'play calling' for the Team. This requires prioritizing between the **Requirements** (or **Capabilities**) coming 'down' to the Work Backlog, and **Chores** that are added by the Team as part of **Refinement**.

> **Commentary**: Once a Requirement drops into the Team's Work Backlog, it's something a Business Owner really wants Done. Therefore, this prioritizing isn't about 'if' a Requirement will be Done; it is only about 'when' it will be Done. The Team (through its Team Captain) owns what Chores they need to do, and when they need to do them, in order to support getting the BO's Requirements to Done. Of course, there is a constant conversation going on between the Business Owner and the Team Captain about this… this is one of the fundamental 'accountabilities' the Team Captain has.

- A **Scrum Team**, augmented by technical SMEs from the Organization and business SMEs supplied by the appropriate Product Champions, is capable of refining its **Work Backlog** and getting its Stories to **Done** – a definition that 'guarantees' that the work has sufficient quality.

- A Scrum Team has a **Team Facilitator** (TF), who facilitates the Scrum Team's self-organization in order to 1) help them do their work, 2) deal with their **impediments**, and 3) improve their practices and teamwork.
- A Scrum Team has a **Daily Scrum** (or Sync-up meeting), where it 1) discusses its current state, 2) identifies any impediments it is running into, and 3) plans what it hopes to do today.
- A Scrum Team incrementally produces **Results** by getting Stories to Done. Appropriate Stakeholders Review these Results (**Increments**) every **Sprint** in order to provide meaningful feedback about acceptability, what should be done next, and so on.
- By utilizing this feedback, the Scrum Team, with help from their Business Owner and Product Champions, improves their Results from Sprint-to-Sprint, with the objective of producing **Deliverable Results**, and then delivering them. *(Note that, in a large Organization, the Team's Deliverable Results may be delivered to other Teams, not the ultimate users of the Product.)*

All of these concepts, and more (see the *Scrum Handbook: Single-Team Scrum (STS)*) are encapsulated in the symbol we see at the right of Figure 10, with different colors representing different 'kinds' of Teams and Team Captains. The Scrum Team's colors in Figure 10 mean that we're looking at a Production Scrum Team (light green oval) with a 'technical' Team Captain (green head or circle).

Commentary:

Without a solid understanding of Single-Team Scrum as described in the "Scrum Handbook: Single-Team Scrum (STS)",

the rest of this Multi-Team Scrum Handbook may be unintelligible to readers, even experienced ones. It is strongly advised that you review your understanding of Single-Team Scrum (STS) before continuing.

One of the most important things you need to understand is the difference between the Product Ownership roles of Team Captain, Business Owner, and Product Champion. We hope the following picture helps explain the difference.

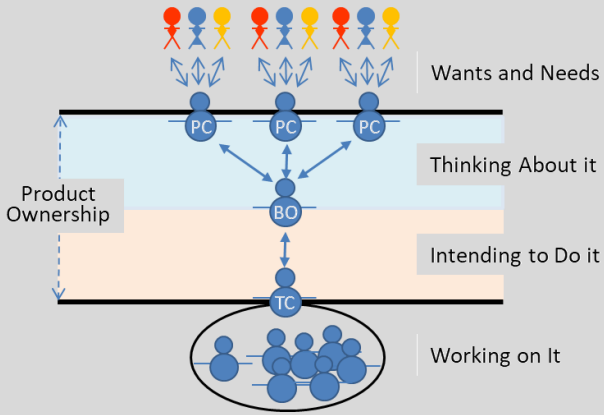

Figure 11: States of the Work

Product Ownership is "the collection of Responsibilities and Accountabilities that sit between the Stakeholders and the Developers," and has the following primary roles:

- **Product Champions** know what Results Stakeholders 'Want and Need', and the order to provide them. A Product Champion says "This is what we should be delivering next!" **A Product Champion speaks for (a set of) Stakeholders**, and maximizes the value of the Results being delivered to them.

- **Business Owners** 'think about' what the Stakeholders 'Want and Need' and decide which Results will actually be produced next – what we're actually 'Expecting to Do'. It is the Business Owner's job to do what is best for the Business (not any particular set of Stakeholders). **The Business Owner speaks for the Business**, and maximizes the value (to the Business) of the Results being produced.
- A **Team Captain** owns the decision of what his or her Team will work on next, which includes the Requirements, Results, or Capabilities the Business Owner wants, as well as what Chores need to be done. **The Team Captain speaks for the Team**, and maximizes the value of the Team's Work. There are different types of Team Captain, including:
  - **Squad Leaders**, who are also Developers on the Scrum Team,
  - **Crew Chiefs**, who are also Developers and are Scrum Mastering on the Team, and
  - **Team Representatives**, who are virtual Team Captains who are produced 'on demand' by the self-organizing Scrum Team.

If you don't have a thorough understanding of these three roles, you are unlikely to obtain a good understanding of the Patterns in this Handbook.

A single Scrum Team can be the core of a small-ish Organization of 15-20 people, as we see in Figure 4 about Sam's Web Services Company. However, as an Organization gets bigger than that, we need to have multiple Scrum Teams, organized as we find in the multi-team Patterns.

There are many types of Scrum Teams involved in our multi-team Patterns. The most important are 'Production Teams', those teams whose Deliverable Results are (at least part of) the Deliverable Results that Product Champions want for their Stakeholders, which we typically call Products.

In addition to Production Teams, we will discuss Flow Mgmt Teams, Leadership Teams, Scrum Master Teams, and so on; Teams whose Results are something besides Deliverable Products. Whenever you see the word 'Team' in this handbook, it will be a Scrum Team (which could consist of a single person), unless *explicitly* stated otherwise.

Pod

The smallest multi-team structure is a Pod, which is a Team of Scrum Teams. Each Production Team is capable of producing proposed or partial Deliverable Results by itself (sometimes called a Feature Team), and a Pod is a Team-of-Teams that produces and delivers Results for (possibly many) Product Champions (PCs) who require similar support or production capabilities.

A Pod exists in order to manage work flow between (possibly many) Product Champions and (possibly many) Production Teams who will provide proposed and partial Deliverable Results for the Champions' Stakeholders.

A Pod usually focuses on a specific functional area or type of problem to work on. Not all Pods are about building Products (remember Sam's Financial Pod), but we refer to their primary Scrum Teams as Production Teams, anyway.

It is possible that a single Production Team could be a Pod, but it is more common that a Pod contains several Production Teams, either because there is a need for 1) more throughput than a

single Team can produce, or 2) more skills than a single Team provides. In essence, a Pod is a Team-of-Teams that acts like a BIG Scrum Team, with each of its Production Teams acting like a Team Member with a specific set of knowledge and skills. Together, the Pod's Teams incrementally produce Done Results for their Product Champions to deliver to their Stakeholders.

A Pod does three things: 1) it manages the flow of work from the Product Champions to the Production Teams, 2) it aggregates the Done Results from the Production Teams into (possibly bigger) Done Results for the Product Champions, and 3) it iterates this process to improve the Done Results until they are Deliverable/Delivered.

A Pod has a **Pod Owner** (a kind of Business Owner) who is accountable for maximizing the value of the Pod's Work and Results. Here is a picture of a Pod along with arrows showing the work flow problem we are trying to solve:

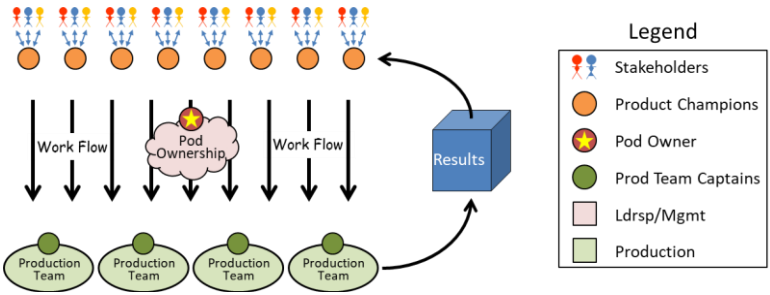

Figure 12: the Overall Concept of a Pod

Commentary:

- Pod Ownership involves managing (and improving) the flow of work within a Pod and leading the people within the Pod – Pod Ownership involves both management and leadership. Therefore, we combine the two concepts in

the Legend, in order to separate a Team that Leads/Manages from a Team that Produces Product.

- The flow of work should be seen as a 'pull' process. The Product Champions own an Inventory of Work that is pulled down through the Pod as the Production Teams get Work to Done. If the Product Champions try to 'push' the work through the system too fast, this causes bottle-necks and log-jams of Inventory that gets 'stuck' or 'bogged down' inside the Pod.
- It is the Pod Owner's job to make sure that the Inventory of Backlog Items **stays with** the Product Champions until the 'work side' of the Pod can consume them, which has two major benefits: 1) it keeps the flow inside the Pod smooth and 2) it keeps the Inventory visible to the Stakeholders.
- The visible Inventory of Backlog Items maintained by the Product Champions is often referred to as a 'Wish List' and represents Pent-Up Demand for the services of the Production Teams.

We must have some way to indicating how big a Pod is, and we will keep it simple. The Pod in Figure 12 is a [4:8]Pod; it has four Production Teams and eight Product Champions. Pods come in all shapes and sizes. For example, we have seen Pods ranging from a [1:30]Pod (a single Production Team supporting 30 Product Champions) to a [700:1]Pod (700 Production Teams working for a single Product Champion), and almost everything in between. Our primary example in this Handbook will be a [3:4]Pod (as seen in Figure 14); largely because it is 1) large enough to be interesting, and 2) small enough to display in

detail). After going into detail with this 'smallish' Pod, we will look at larger ones.

In any case, when looking at the examples of Pods in this Handbook, it is important that you realize they are there only to *explain* the Patterns; they are not to be seen as prescriptive recipes to follow. It is up to you to use the Patterns in order to design and develop your own Scaled Organization based on your own context and 'forces'. In other words, when you use the Patterns contained in this Handbook, your organizational design will be unique to your situation, and the descriptions we offer here are for understanding; not direction.

Group

While the purpose of a Pod is to satisfy the needs of several Product Champions (PCs) who require similar support or production capabilities; a Group is a collection of Pods (and sub-Groups) that each provide (possibly) different capabilities. Sometimes the same Product Champion will require services from more than one Pod within a Group, so they are not required to be independent, just dissimilar.

For example, when working on a large-ish e-commerce website, there may be a Development Group that consists of a Product Pod, a Checkout Pod, a Client Relationship Pod, Shipping and Returns Pod, and so on… and the Development Group is part of a Business Group that also contains an HR Pod, a Finance Pod, a Facilities Pod, and so on. This large-ish Organization is seen in Figure 13.

Each Group has a Group Owner who is accountable for maximizing the value the Group produces. As before, this involves both management and leadership, so we label it that way in the legend. The lines indicate the main communications pathways between the Ownership and elements of the Group.

Commentary:

We treat most of the 'business-side' Pods as Production Pods; their products may not be the Deliverable Results the Business delivers, but they do provide Deliverable Results for use inside the Business.

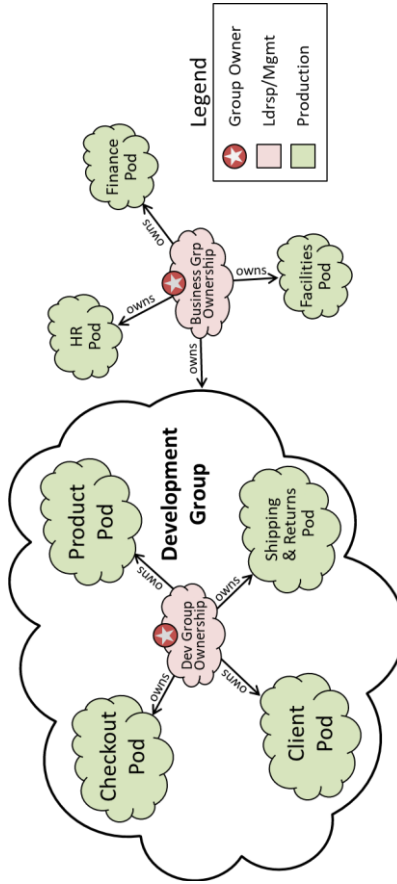

Figure 13: Business Group for a Large-ish e-Commerce Website Company

Small Pod Structural Patterns

We will talk about small Pods first; these Pods have only one Team involved in Pod Ownership, sitting between the Product Champions and the Production Teams. We call this type of Pod a single-layer Pod, since it has only a single layer of Pod Ownership. Once we have identified the main Structural Patterns involved in the single-layer Pod, we'll go on to bigger Pods.

We're discussing Structural Patterns in this section; and most of them are about establishing virtual Scrum Teams, which are Scrum Teams made up of people who are already on other Scrum Teams. These people spend a significant amount of time with each of their teams, and must not prioritize one over the other. A person's priority is based on which Team he/she is working on *right now*, and that is determined by the Pod's management.

This network of Teams enables collaborative conversations that span multiple Teams within the Pod. These conversations are the primary enablers of Governance, which is the distributed management and decision-making within the Pod.

So, when you are reading about these Structural Patterns, think about which cross-team conversations are being enabled by the Pattern. Think about what Organizational problems are 'automatically' being solved because the right people are in the same place at the same time – and thus able to collaborate to solve the problems. Think about how the virtual Teams' self-organization becomes virtual self-organization across all the Teams.

Pattern: The Pod's Flow Management Team

In a small single-layer Pod, the Pod Ownership is done by a single Scrum Team, the Pod's Flow Mgmt Team. The primary Mission of the Pod's Flow Mgmt Team is to manage the work flow within a Pod in order to maximize the value of the Pod's Results; the Pod's Flow Mgmt Team is **Aligning the Organization on what** should be done, and who should do it.

The Flow Mgmt Team within this single-layer Pod does five things: 1) it develops and manages the Pod's Results Backlog, 2) it distributes Stories to Production Teams to work on, 3) it provides Guidance and Objectives (GOs) to the Team Captains and Product Champions, 4) it maintains necessary Delivery Forecasts, and 5) it improves the functioning of the Pod. Additionally, the Pod Owner (the Pod's Flow Mgmt Team's Team Captain) is the person accountable to the Business for maximizing the value of the Pod's Work/Results.

Even though the Pod Owner is a leader, owning the Guidance and Objectives (GOs) that are provided to the Pod's Team Captains and Product Champions, we refer to this team as a 'Management' Team. This is because the *main* thrust of the Team is not the Pod Owner's leadership role, but the fact that the Team *manages* the work flow within the Pod *(note: things are managed, while people are led)*.

Structure

Since a picture is worth a thousand words, here is a diagram showing a typical Flow Mgmt Team for a [3:4]Pod (three Production Teams, four Product Champions).

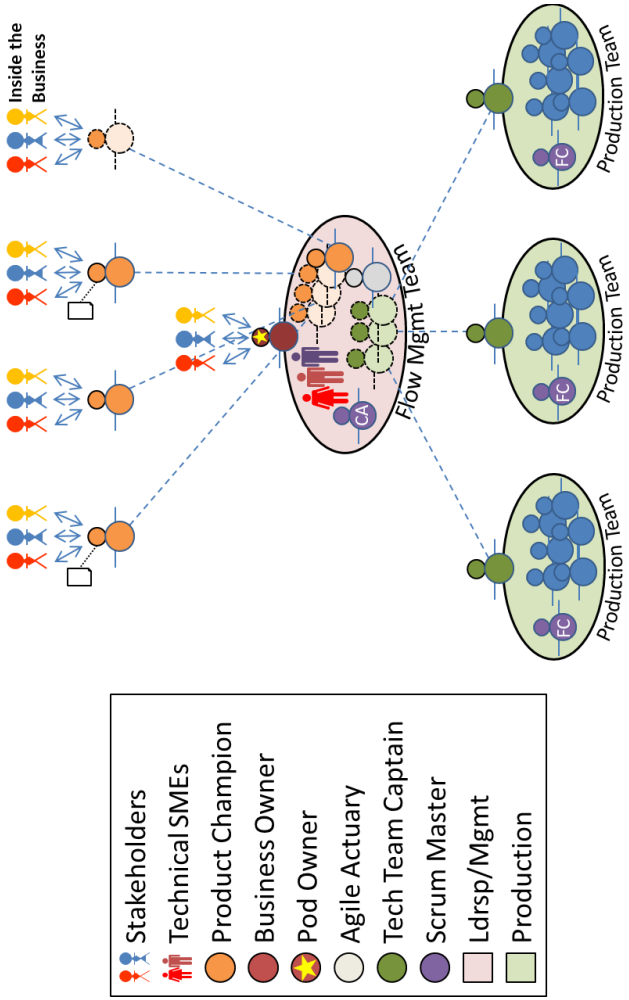

Figure 14: A [3:4]Pod with its Flow Mgmt Team

As you can see, the Pod's Flow Mgmt Team's Captain is a Business Owner who is also the Pod Owner (as notated with the 'star' in the diagram). The only other required members of the Pod's Flow Mgmt Team are the Product Champions who are asking for work

to be done, and the Team Captains of the Production Teams who are looking for work to do.

This particular Management Team has more than the bare minimum, though:

- This Pod's Flow Mgmt Team contains 'technical' SMEs (e.g., Chief Architect, UX Expert, Senior Business Analyst, Tech Writer …) who are shared across the Teams in the Pod as necessary.
- Some of the Product Champions in this Pod are required to maintain Delivery Forecasts (indicated by the icons of documents they are holding), so the Pod's Flow Mgmt Team has an Agile Actuary onboard to help them.
- One of the Product Champions (the one on the right) 'lives' on the Pod's Flow Mgmt Team and 'visits' his/her Stakeholders (who happen to be *inside* the Business, not outside Clients), while the other Product Champions 'live' with their Stakeholders and 'visit' (are virtual members of) the Pod's Flow Mgmt Team.

Commentary:

- This notion of 'living' on one Team, and 'visiting' another, is common throughout these Patterns. A basic 'rule' is that nobody should be in more than two places – each person should have one primary Team, and some people also have a secondary Team. Symbolically, this is shown with 'ghosting' on the 'second' Team.
- The SMEs we see on the Flow Mgmt Team are the only exceptions to this rule; their primary Team is the Pod's Flow Mgmt Team; and they usually help many Teams throughout the Pod.

- A person working on multiple Teams **must not** prioritize one over the other. Each person's allegiance is to the Team he or she is on **at that moment**; any question about which Team that is should be answered by the Pod Flow Mgmt Team.

- This Pod Owner has his/her own Stakeholders, who are usually high-level people in the Business who want to 'sneak something in' through the Pod Owner. Since this almost always happens, we acknowledge it here, and the Pod Owner becomes the Champion for their wants and needs. Would it be better if they used the four 'official' Product Champions? Yes, of course, but reality wins, and we'll reflect this reality in our example. *(Note: the Pod Owner could be an 'official' Product Champion, but this is risky and could be a conflict of interest for the Pod Owner unless he/she is the **only** Product Champion.)*

- Each of the Production Teams has a Facilitator/Coach, not just a Team Facilitator, and the Team Facilitator (Scrum Master) for the Pod's Flow Mgmt Team is also the Organizational Change Agent for the Pod.

Commentary: Remember that this Pod's Flow Management Team is just an example; it is not a recipe or prescription! We will provide several variations in a Commentary at the end of this section.

The Main Thing it does is Manage Flow

The main Mission a Pod's Flow Mgmt Team has is to manage the flow of Work in the Pod, moving Work Items from 'above' the Flow Mgmt Team to 'below' the Flow Mgmt Team, using the Flow Mgmt Team's Results Backlog as a 'holding pen' and work area.

There are many facets to this, and the Pod's Flow Mgmt Team has the 'right people' on the Team who will collaborate to do them, as we see in the following diagram.

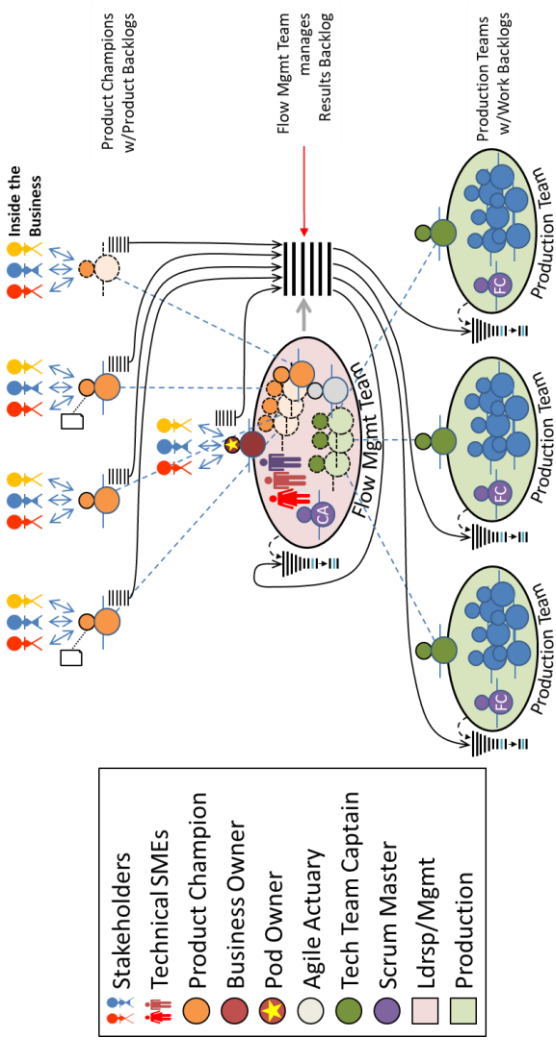

Figure 15: A Pod's Flow Mgmt Team Manages the Flow of Work

Now, the Pod Owner is the Team Captain of the Flow Mgmt Team, and many of the other members of the Flow Mgmt Team are empowered, accountable decision-makers; there are Team Captains and Product Champions who are empowered to make significant decisions they are accountable for every day. They make these decisions based on Guidance and Objectives (GOs) determined by the Flow Mgmt Team (which they are members of) and owned by the Pod Owner. These GOs underlie and influence all of the work done within the Pod.

Here's what's going on every day in Figure 15; this is the bulk of what the Pod's Flow Mgmt Team does:

- Items are brought in from *above* the Pod's Flow Mgmt Team and are added to the Results Backlog in the 'right' order. *(We call this **consolidation**, as many backlogs are consoledated to a single Backlog.)* This is mostly an ordering/ prioritization issue *(whose Product is next?)* and could be contentious, so the Product Champions and the Pod Owner collaborate on this issue as part of the Flow Mgmt Team's self-organization.

  Commentary: If there is only one Product Champion, prioritization is automatic, but the Product Champion **is a member** of the Pod's Flow Mgmt Team (and could be the Pod Owner) to help with further Refinement and re-ordering.

- Items on the Results Backlog are Refined so they are Ready to move to Production, which includes: 1) analyzing, splitting, combining, and converting them into (other, new) Items, 2) adding necessary Chores to the Results Backlog, and 3) sizing them as necessary. The Team Captains, Technical SMEs, and Business SMEs (supplied by the Product Champions) collaborate on this.

Notes:

1. Refinement is not always necessary; sometimes the Items are just 'passed through' with no Refinement. This could happen if the work consists of simple bug fixes, for example.
2. In any case, the goal of this Refinement is to have Items in the Results Backlog that:
    - Represent what maximizes ROI for the Business, not just provides Value to the Stakeholders. (This may require using techniques like Commonality and Variability Analysis, Root-Cause Analysis, Five-Times-Why, and so on…), and
    - Be an indecomposable 'prioritization unit'; that is, the Item should represent some sort of 'single whole' whose Done requires that **all** parts of it need to be Done.

- The Results Backlog is re-ordered as necessary. Everybody collaborates on this in order to maximize overall value, and the Business/Pod Owner (the Flow Mgmt Team's Team Captain) owns the decisions. The goal is that the Results Backlog *always* represents the most important work that needs to be done; the Results Backlog should *always be current*.

Note: 'Important' has many facets based on both Business Value and Technical Dependencies. Technical Dependencies should be resolved by doing things in the 'right order', and Business Value is often determined using some sort of prioritization method, such as the Kano Method, Quality Functional Deployment (QFD), the Eisenhower method, or MoSCoW.

- The Refined Items are assigned (moved down) to the Production Teams to be further Refined and worked on. *(We call this **distribution**, as a single backlog is distributed to many backlogs.)* This could be contentious, and the Team Captains and Business/Pod Owner collaborate on this.

  > **Note**: if there is only one Production Team, this assignment is automatic, and the lone Team Captain **does not need to be a member** of the Pod's Flow Mgmt Team. Conversely, the Pod's Flow Mgmt Team might become more strategic by allowing the Team Captain to provide a flow of value directly to the stakeholders.

- As Stories get further Refined and worked on by the Production Teams, the appropriate Product Champions are accountable to supply Business SMEs to help them, as necessary.

  > **Note**: in general, the Product Champions need to know where their Stories are, and what's happening to them, at all times.

- The Pod's Flow Mgmt Team should be constantly improving, and smoothing, the flow of work within the Pod.

  > **Note**: The Scrum Master of the Pod's Flow Mgmt Team is accountable for suggesting Improvements to the Pod Owner about how to smooth, and improve, the flow of work.

The Pod's Flow Mgmt Team Schedules Reviews

A Pod is likely to be working on several Products at once, so there will probably need to be multiple Product and Progress Reviews every Sprint. Each Review may require many attendees (Pod Members, Stakeholders, and Product Champions) from 'all around' the Pod, and since each Product needs to be Reviewed

during – or immediately after – every Sprint it's worked on, it is up to the Pod's Flow Mgmt Team to coordinate the Reviews so that each Product can be properly Reviewed.

Because there could be multiple Reviews, with multiple required attendees, there may not be enough time 'between' the Sprints to have all of them done then. Some of them will have to be done as Stories 'inside' the current (or next) Sprint, as part of the more-or-less continuous refinement/planning that happens in Scrum.

The Workflow is a 'Pull', not a 'Push'

The Pod's Flow Mgmt Team needs to balance the overall Production Capacity of the Pod with the wants and needs of the Product Champions.

The Pod's Flow Mgmt Team should see the work flow as a 'pull' system, where the work getting Done by the Production Teams (the Results) is a 'pull' for new work to be placed in their WIPs. This, in turn, provides a 'pull' to move Work from the Results Backlog to the individual team's Work Backlogs, and this creates a pull to move work from individual Product Backlogs to the Results Backlog.

Of course, the Product Backlogs (which are actually only 'Wish Lists') can be as long as they need to be; work items are NOT Inventory until they enter the work flow inside the Pod – until they 'enter the system'. As this Inventory (held by the Product Champions) increases, it represents Pent-Up Demand from the Stakeholders.

It is likely that some Product Champions will want to 'push' work through the System, and the Pod Owner must resist this and make sure the flow's driving force is the 'pull' from the Production Teams, not the 'push' from the Product Champions.

Other Things the Pod's Flow Mgmt Team Might Do

Depending on circumstances, the Pod's Flow Mgmt Team may do many other things. Even though some of these things may help produce Product, the Pod's Flow Mgmt Team is NOT considered to be a Production Team – this Pod is still a [3:4]Pod. Here is a short list of such things.

Introduce New Work into the Work Flow

The Pod's Management Team may be adding Work to the Pod's Results Backlog that does NOT originate from 'external' Stakeholders; for example, initiatives that come from the Business like *"Swap out the Database"* or *"Add the New Feature the CEO wants"* or something.

This can be thought of as either:

- extending the concept of Refinement to include adding new Capabilities, and not just Chores;
- having a Product Champion from inside the Business, or
- having the Business's Product Champion on the Pod's Flow Mgmt Team.

In Figure 15, the Product Champion at the upper-right is one of the latter kind.

Create Delivery Forecasts

Often, Product Champions have a Delivery Forecast they share with their Stakeholders. Keeping a Delivery Forecast 'accurate' through time is a tough job, so the Pod's Flow Mgmt Team may have an Agile Actuary to help.

In these cases, updating the Delivery Forecasts (by comparing estimated Sizes to actual Effort) is work that lives on the Pod's

Flow Mgmt Team's Work Backlog. The Pod Owner is accountable for assuring that the Delivery Forecasts are (collectively) justified by the realities of development.

Decide How technical SMEs are Used

The Product Teams often need to share technical SMEs (who are often members of the Flow Mgmt Team) when they do their work. Ideally, the Product Teams would collaborate about how to share, but the Pod Owner can make/facilitate sharing decisions, if necessary. This should be easy to do because the 'sharing' issue could/should be raised at the Pod's Flow Mgmt Team's Daily Scrum.

Create an Integrated Product

If the whole Pod is working on only one Product, it is common, in order to **Leverage the Economy of Scale**, for the Pod's Flow Mgmt Team to do the Integration of the Results produced by the Production Teams.

After all, the Pod's Flow Mgmt Team decomposed the work that was passed down to the Production Teams, so the Flow Mgmt Team should know how to put the Results back together.

Using the Flow Mgmt Team in this way allows for a central location for Reviewing and Releasing the Product, which makes a lot of sense. Unfortunately, it works best when the Pod is working on a single Product, and the most common scaling problem is that there are multiple Products involved… just sayin'…

Be a Test Lab

In the same way that the Pod's Flow Mgmt Team can be an Integration Team if there is only one Product, it makes sense that the Pod's Flow Mgmt Team can be a Test Lab when there are

common testing issues or environments, in order to **Leverage the Economy of Scale**.

The testing done in this Lab is NOT the verification testing to make sure Items are Done – verification testing is still 'owned' by the Production Teams – it is validation testing such as Usability Testing, Performance Testing, Exploratory Testing, and so on, that requires a specialized Test Lab. When the Pod's Flow Mgmt Team is acting as a Test Lab, it often creates new Stories (often called 'bugs' or 'defects') that will be added to the Results Backlog to be worked on.

How big can a single Pod's Flow Mgmt Team be?

A single-level Pod has a single Mgmt Scrum Team that has all the Product Champions and Team Captains on it. Since a Scrum Team is limited in size, this limits the size of a single-level Pod. Scrum Teams usually have fewer than ten Team Members (with five being the sweet spot), but the limit is different for this Team.

The reason for this is that the whole Team is seldom collaborating on the same Story. At any given time, the Flow Mgmt Team breaks into sub-teams that should (of course) be small. For example, it is typical that the Pod's Flow Mgmt Team divides into separate sub-teams to do Consolidation, Distribution, and manage technical SMEs…

At any given time, it's as if there are multiple Scrum Teams that all have the same Team Captain. Because of the pictures we draw, we refer to this as the 'Multiple yolks, one egg' Pattern, as we see in Figure 16, below. In this particular Flow Mgmt Team (for a [6:6]Pod) you can see three sub-Teams – this leads us to believe that this Flow Mgmt Team can contain up to 20 people.

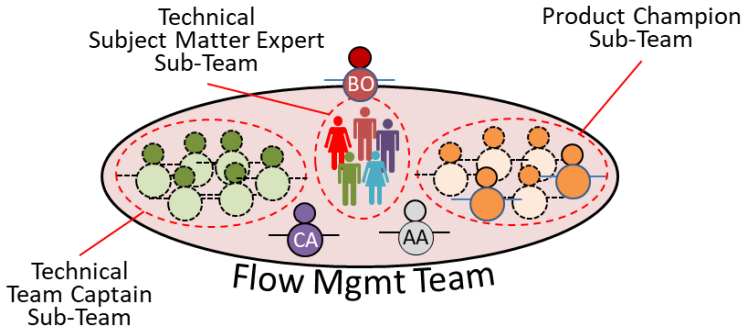

Figure 16: The Flow Flow Mgmt Team for a Typical [6:6]Pod

Pattern: Pod Scrum Master Team

Every Pod needs Scrum Masters who work as Team Facilitators, Agile Coaches, and Change Agents. Each Scrum Team has its own Facilitator, and the Pod's Scrum Master Team (SM Team) assures that the Pod has all the additional Scrum Mastering that it needs.

The Pod's Scrum Master Team will be the 'center' of Scrum Mastering for the Pod. The SM Team's Team Members help each other improve their own Scrum Mastering, discuss Kaizens for their Teams, provide Coaching, Evaluations, and Agile Training for the Pod's Production Teams, and so on.

Structure of the Pod's Scrum Master Team

As we see in Figure 17, the Scrum Master Team is a virtual Scrum Team (notated by the dashed lines) that consists of each of the Team Facilitators from each of the Production Teams and the Team Facilitator from the Flow Mgmt Team. It may have other members, as well, such as Agile Coaches.

The Flow Mgmt Team's Facilitator serves as the Scrum Master Team's Team Captain and the Pod's Change Agent, and is also referred to as the Chief Scrum Master (CSM) for the Pod.

We expect the members of the Scrum Master Team to supply the Pod's Scrum Mastering. This probably includes the Agile Coaching, Evaluations, and Training for the Pod. in our example, the Product Teams' Facilitators are also Agile Coaches who are members of the Scrum Master Team.

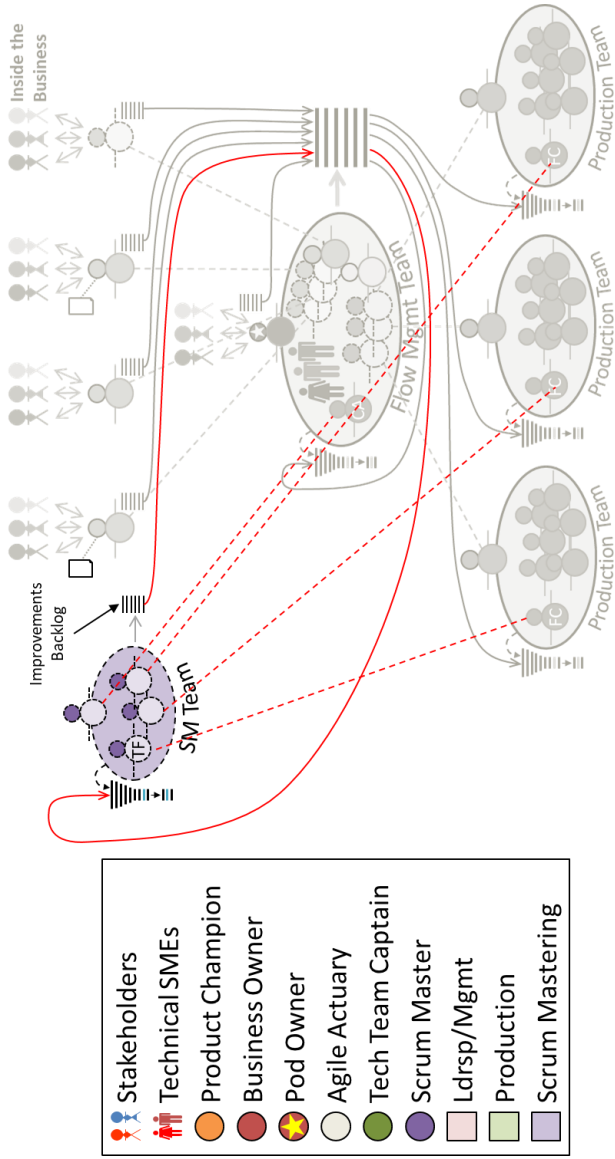

Figure 17: Adding the Scrum Master Team (SM Team)

What a Pod's Scrum Master Team Does

Normally, in a Pod like this, the Production Teams will have their Daily Scrums, and then their Facilitators will go to the Scrum Master Team's Daily Scrum. In this way, all impediments and other Scrum Master issues will be discussed by the Scrum Master Team every day. This allows the Pod's Change Agent to discuss these issues (as necessary) with the Pod's Flow Mgmt Team during its Daily Scrum.

The Scrum Master Team's primary Product is the Pod's Improvement Backlog, which is a list of potential improvements of processes, practices, and organizational design – all in order to improve the Flow of high-quality work within the Pod. The Pod may need to Scale Up, it may need to Scale Down, it may need to improve technical practices, or it may need to change in some other way in order to improve this Flow. The SM Team's Team Captain (the Pod's Change Agent) is the Champion for the Improvement Backlog on the Flow Mgmt Team.

The Improvement Backlog includes small Improvement Stories, and large Improvement Epics. The Change Agent's primary objective is to improve the Pod to make it more Agile, more conducive to Scrum, and as a result, more successful.

As a member of the Flow Mgmt Team, the Change Agent is the Champion for this change; trying to get the Pod Owner to agree to implement improvements that will then show up as Items on the Pod's Results Backlog.

Pattern: Scrum-of-Scrums (SoS)

At a purely tactical level, the Production Teams need to keep 'in sync' with each other, and cooperate, coordinate, and collaborate with each other when necessary. The Scrum of Scrums (SoS) is a good way to facilitate this.

Structure of the Scrum-of-Scrums

As we see in Figure 18, the Scrum-of-Scrums (SoS) is a non-Scrum Team that includes one Team Member from each Production Team, and this Team Member may be neither the Team Captain nor Team Facilitator. The decisions of who to send to the SoS belong to the Production Scrum Teams. The SoS's Team Members will probably change over time, as each Production Scrum Team should choose its representative to the SoS based on who will be in the best position to understand, and comment on, the issues most likely to arise at that time.

What the Scrum-of-Scrums Does

The mission of the Scrum-of-Scrums is almost totally open-ended; although it's clearly an informal Leadership Team. It decides when it's going to meet; and it is common for the SoS to meet right after the Product Teams' Daily Scrums (concurrent with the Scrum Master Team's Daily Scrum). It's interested in only one question: *"how do our Teams need to interact (cooperate, coordinate, or collaborate) right now?"* and their conversations often concern common dependencies, common impediments, sharing SMEs, and so on.

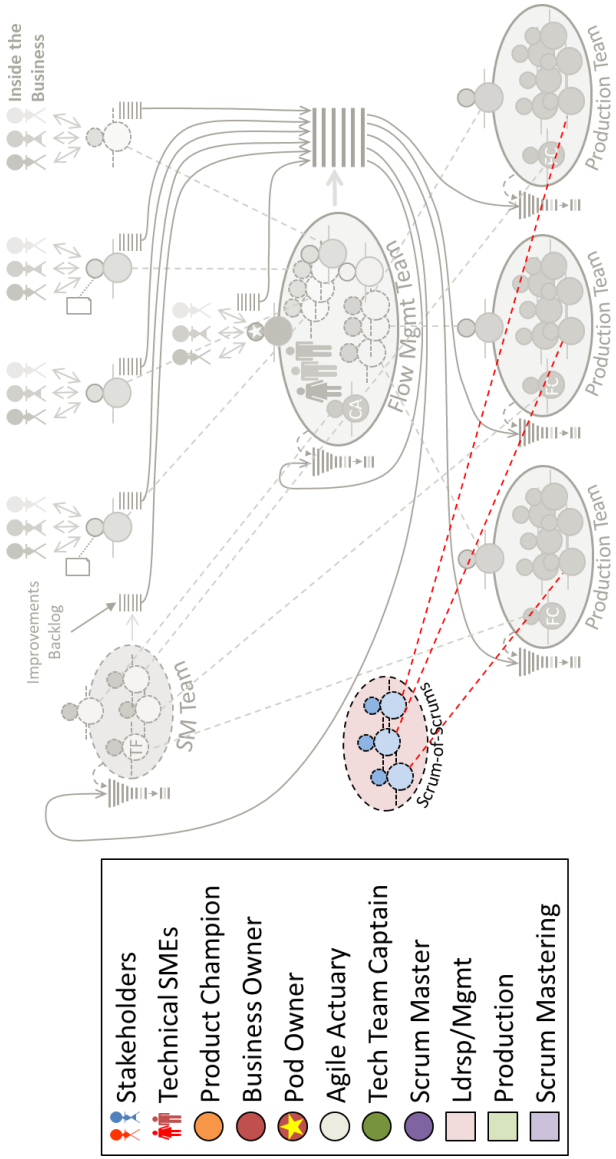

Figure 18: Adding the Scrum-of-Scrums (SoS)

Pattern: Test Lab Team

One common **Leveraging the Economy of Scale** issue that arises in development is the need for centralized, specialized, testing. The testing is NOT the verification testing to make sure Items are Done – that testing is 'owned' by the Production Teams – it is validation testing like Usability Testing, Performance Testing, Exploratory Testing, and so on, that often requires a specialized Test Lab.

Structure of the Test Lab Team

The Test Lab Team is a Scrum Team whose Team Members are either full-time members of the Test Lab Team or borrowed from other Teams. In our example (see Figure 19), it is a separate, full-time, Team.

What the Test Lab Team Does

The Test Lab Team does Validation Testing for the Pod. Its primary Product is a "Bug" List that feeds back to the Pod's Results Backlog to be dealt with by the Flow Mgmt Team.

The Test Lab Team is treated like a Production Team in several ways: 1) its Team Captain is a member of the Pod's Flow Mgmt Team (and acts as the Champion for the Bug List), 2) its Facilitator is a member of the Pod's Scrum Master Team, and 3) it has one of its Team Members on the Scrum-of-Scrums. However, since the Test Lab Team's Results are not actual Product to be delivered to Stakeholders, it is NOT a Production Team. This Pod is still a [3:4]Pod, even when it includes a Test Lab Team.

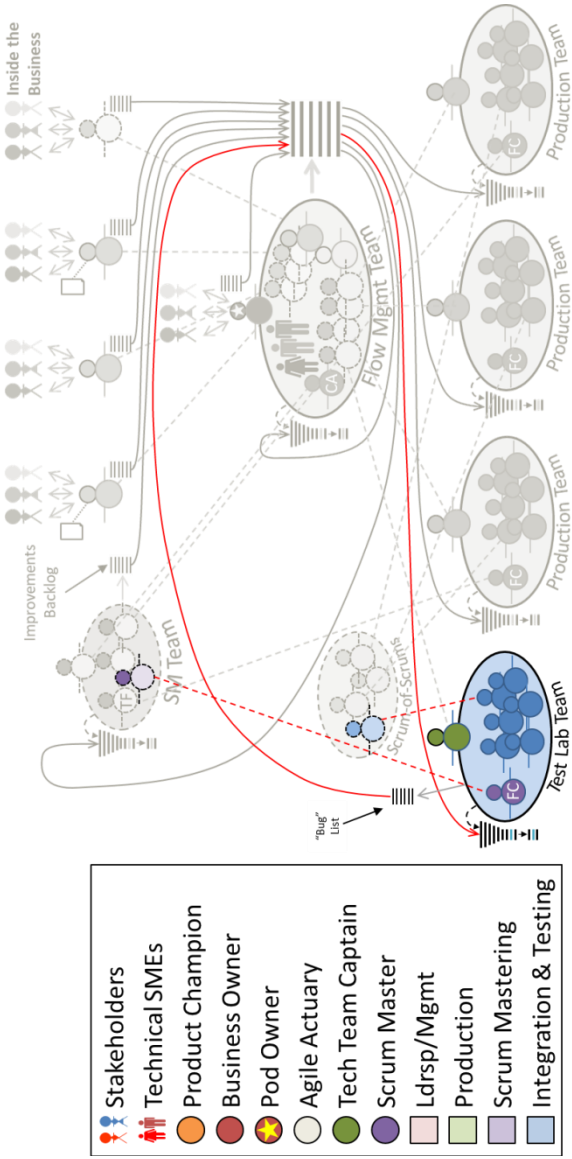

Figure 19: Adding a Test Lab Team

Pattern: Collaborative Work within a Pod

This Pattern enables cross-Team collaboration within a Pod in order to manage dependencies and common issues.

Structure of the Production Train

The Production Teams of a Single-Level Pod are often called a **Production Train** because 1) all the Production Teams are like cars on the same Track, 2) they're all being pulled by the same engine (the Pod's Flow Mgmt Team), and 3) we expect Team Members to be able to wander from car to car when they need to, in order to coordinate, cooperate, and collaborate.

Figure 20 shows this kind of work. We have added the Test Lab Team to the Production Train even though it does not count as a Production Team when sizing the Pod; this is because the Test Lab Team Members need to interact with the people on the Production Teams in order to do their work. Also note that there are both technical and business SMEs participating in these discussions on an as-needed basis.

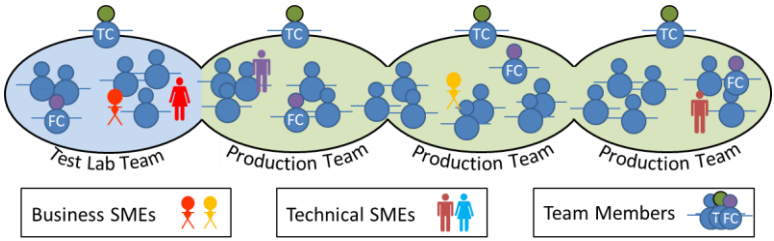

Figure 20: Collaboration within the Pod's Train

The Production Train enables Collaboration

Figure 20 shows the Team Members, Facilitators, and SMEs (both technical and business) coordinating, cooperating, and

collaborating as they do work, while their Team Captains 'guard the door' – making it safe to self-organize to do the work.

This is the way work *should* happen with agile teams. In this Pod we have described, there is information coming out of the Pod's Flow Mgmt Team, the Pod's Scrum Master Team, and the Pod's Scrum-of-Scrums that indicates what they should be collaborating on. What we see in Figure 20 is a Team-of-Teams working as a single 'big' team (when it needs to) and acting the way Scrum says it should act.

Pattern: Cross-Cutting Workgroups

The Scrum-of-Scrums Pattern allows Production Teams doing work to keep 'in sync' constantly, and cooperate, coordinate, and collaborate with each other when necessary. Sometimes, this is not enough; sometimes the Production Teams need to work together in order to resolve dependencies, remove common impediments, share learnings and expertise, develop common solutions to common problems, and so on. A good way to do this is to use Cross-Cutting Workgroups that are **Aligning the Organization on how** to do its work.

Structure of Cross-Cutting Workgroups

A Cross-Cutting Workgroup is a Scrum Team, with a well-defined mission, whose Team Members come from the Pod and surrounding Stakeholders. Typically, the Team Members are not already Team Captains or Team Facilitators of other, existing, Teams.

It is common for the Technical SMEs that belong to the Leadership Team to become Team Captains of Cross-Cutting Workgroups that work on problems in their area of expertise – this makes the Workgroup's Team Captains *automatically* members of the Pod's Flow Mgmt Team. This is true for both of the Workgroups in Figure 21. In this example, there are two Workgroups:

- The Architecture Team is a cross-cutting workgroup whose Team Captain is the Pod's Chief Architect (who belongs to the Pod's Flow Mgmt Team), one Team Member (who has, or wants to have, architecture expertise) from each of the Production Teams, and an external Stakeholder (possibly a Regulator with expertise in Security); and

- The Usability Team is a cross-cutting workgroup whose Team Captain is the Pod's senior Usability Experience (UX) Designer, and whose Team Members are people (who have, or want to have, UX expertise) from the Production Teams.

If the Workgroup's Team Captain is *not* a member of the Pod's Flow Mgmt Team *automatically*, somebody on the Pod's Flow Mgmt Team *must* Champion the work the Workgroup does (or needs to do), and the easiest way to do this is to have the Workgroup's Team Captain become a member of the Pod's Flow Mgmt Team.

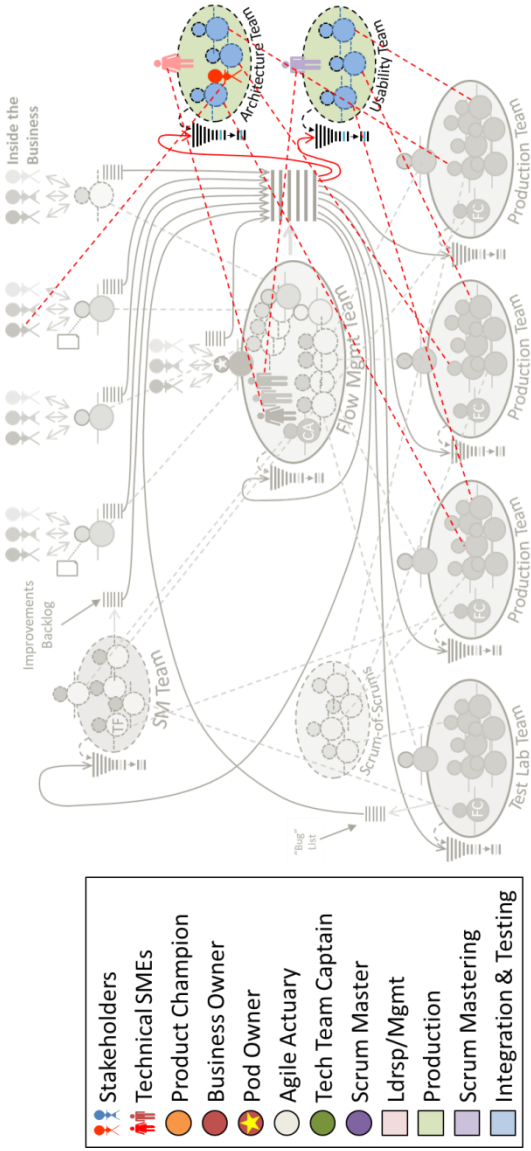

Figure 21: Cross-Cutting Workgroups

What Cross-Cutting Workgroups Do

Since Cross-Cutting Workgroups are Scrum Teams, they work from a Work Backlog, and they have Daily Scrums… just like other Scrum Teams. Cross-cutting Workgroups exist for the good of the business, and the primary inputs to their Work Backlogs are determined and prioritized by the Pod's Flow Mgmt Team.

Usually, the Workgroup's Daily Scrum is done immediately after the Product Teams' Daily Scrums, which is then followed by the Pod's Flow Mgmt Team's Daily Scrum. Because of this ordering, the Production Teams' Team Captains, the Cross-Cutting Workgroups' Champions, and the Pod Owner, will be able to have collaborative discussions (every day) about where the individual Team Members are, or will be, spending their time.

Resolving the issue of where people 'should be' working is a big part of the work of the Pod's Flow Mgmt Team, and one of the main reasons we need all those people in the same conversation at the same time every day.

Many Cross-Cutting Workgroups do not have work to do every day – they do their work on an as-needed basis. Sometimes, when you need a Workgroup to solve a one-off problem, it makes sense to instantiate a Cross-Cutting Workgroup with the people you need, when you need it, and then disband it after the problem is solved.

Discussion of Cross-Cutting Workgroups

Let me make a few comments:

1. The Test Lab Team we discussed earlier could also take the form of a Cross-Cutting Workgroup if its Team Members came from the Production Teams. This is common when the Test Lead 'lives' on the Pod's Flow Mgmt Team as a Technical

SME and also serves as the Team Captain of the Test Lab Team.

2. Cross-Cutting Workgroups can have both permanent members and part-time members that come from elsewhere in the Pod or its surroundings. For example – again – the Test Lab could have a permanent Team Captain (who is also a virtual member of the Pod's Flow Mgmt Team), a few testing specialists as permanent members, and part-time members from the Production Teams.

3. Sometimes the Team Captain of a Workgroup also plays a 'line manager' role within the Pod, responsible for developing people and setting salaries (see discussions of Spotify's Chapters, which you can find elsewhere online). This usually happens when the Workgroups are skill-based; for example, having a Testing Workgroup, an Architecture Workgroup, a Business Analysis Workgroup, a Back Office Developer Workgroup, and so on.

4. Workgroups are often confused with Communities of Practice (CoP), because they have similar forms. However, they have completely different functions. Cross-Cutting Workgroups exist to help the Pod solve problems it is finding with its Product Development – what a Workgroup does is part of the Pod's Results Backlog. A Community of Practice, on the other hand, is a group of people who share a common interest and interact regularly to learn more about it and get better at it – what they do and how they do it is determined by them.

Pattern: Community of Practice (COP)

Professionals want to be good at what they do; this is one of the primary things that makes them 'professionals' and not merely 'employees'. One way they can get good at what they do is be a member of a Community of Practice (CoP), which is a group of people who share a concern or a passion for what they do and get together (outside of work, usually) in order to learn how to do it better as they interact.

Structure of a COP

In general, there is no form for a CoP; they are just a group of people that interact regularly. There are many versions, from a High School's Chess Club to a 'Thursday night hack-a-thon' at Starbucks.

However, as a Pattern for an Organization, we recommend that Functional Managers foster, support, encourage, and sponsor Communities of Practice amongst their people. The Functional Manager could provide a location, equipment, food, and (possibly) money for invited guests. Of course, the Organization may NOT require people to attend, or belong to, a CoP.

These CoPs could be defined by *any* shared domain of interest; for example, in the software domain there could be CoPs on web technology, test automation, agile coaching, Clean Coding, and so on. There can also be, of course, CoPs whose domain is outside software, like a company's Chess Club.

What a Functional Manager does NOT do is 'own' the CoP; the Community of Practice belongs to the community, not the manager, not the Organization – a Community of Practice exists for the good of its members.

What a COP Does

Members of a Community of Practice develop a shared repertoire of resources: experiences, stories, tools, ways of addressing recurring problems – in short, a shared practice. This takes time and sustained interaction.

A Chess Club could be a CoP, but a Chess Team is not, even though the Team members may be part of the Club, and Club members may aspire to be part of the Team. The Club is informal, and the Team is formal. The goal of the Club Members is to have a good time and get better at Chess; but the mission of the Team is to compete and win.

Communities of Practice are a general Pattern; they are not limited to Pods, or even Organizations. We are discussing it here because sponsoring and encouraging CoPs is an easy, and inexpensive, way for an Organization to help its people improve themselves.

Discussion of the Small Pod

The small Pod, with a single Leadership Team, is a very powerful thing. It leverages the power of multiple Production Scrum Teams, and can provide Results for several Product Champions simultaneously. The Pod Owner, who is the Team Captain of the Pod's Flow Mgmt Team, is accountable for maximizing the Business Value of the Pod's Results. The single-layer Pod clearly Illustrates the fundamental differences between the Product Ownership Roles (Team Captain, Business Owner, Product Champion), and shows why each is important.

Every Structural Pattern that we have introduced either:

- helps the Pod increase its Capacity (Scrum-of-Scrums, Cross-Cutting Workgroups, Collaborative Work, Community of Practice),
- gives more information to the Pod Owner about what provides value (Flow Mgmt Team, Test Lab Team), or
- both (Scum Master Team).

The Pod we have so far is a single-level Pod with all the 'bells and whistles'. However, it is not a recipe or a framework. It is simply a Pod that is using ALL the Patterns; you may not need all of them. But, for reference, Figure 22 shows the whole thing.

The single-layer Pod is very powerful, but sometimes we need more. In the next section we will discuss an example of a bigger, multi-layer Pod.

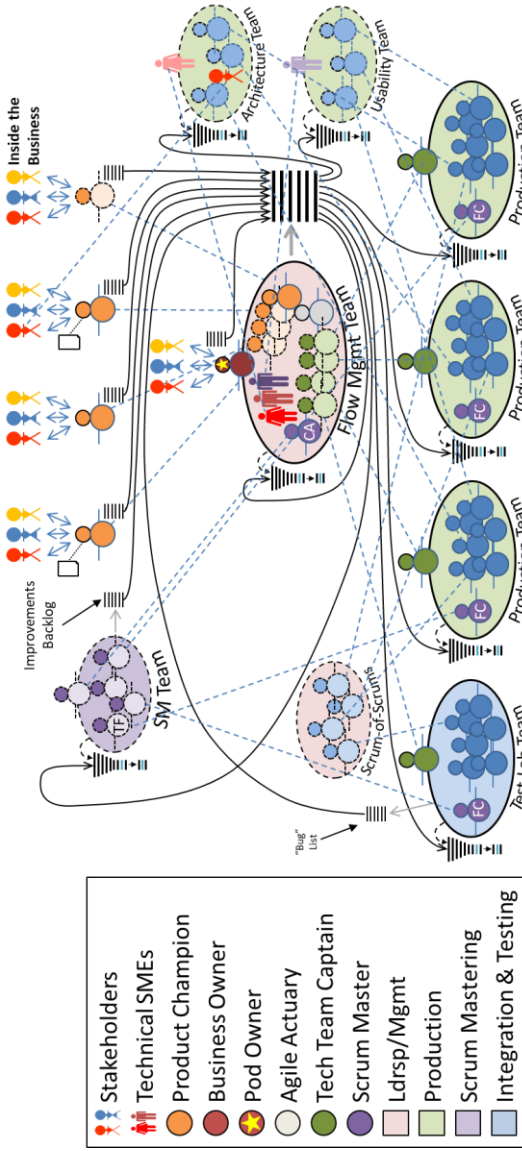

Figure 22: The [3:4]Pod with all the 'Bells and Whistles'

Large Pod Structural Patterns

So, we've got a collection of Patterns that, together, create a single-level Pod, as in Figure 22. We're going to be making bigger Pods now, and need a simpler notation to do so.

Notation for Pods

So, this next diagram represents exactly the same [3:4]Pod we've been working with. *(Note that the work-flow arrows have been removed from the Cross-Cutting Workgroups for simplification.)*

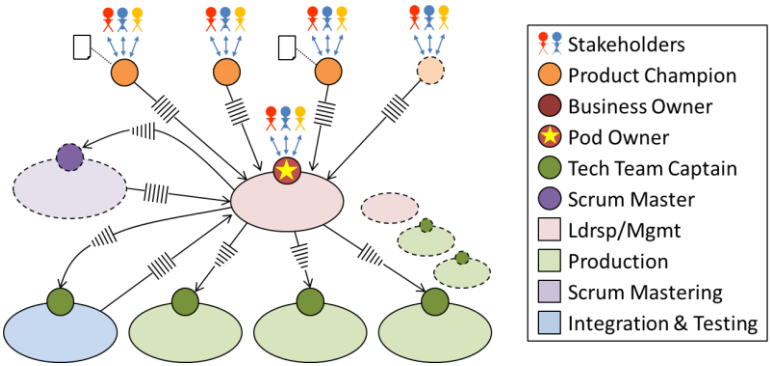

Figure 23: A Simpler Visual Notation for the [3:4]Pod

We will be using this basic notation to make multi-layer Pods, so you need to understand it. The colors are crucial; when we make smaller versions of the Pod the colors will be the major clue as to what you're looking at. So, take a few minutes to compare Figure 22 and 23, and make sure you understand them.

Pattern: Multiple Flow Management Teams

We will exploit the fractal nature of these Patterns to make a multi-level Pod, as we see in Figure 24, which basically contains

nine Flow Mgmt Teams with their own Business Owners (Team Captains), but only one of them is the Pod Owner. Each of these nine Flow Mgmt Teams is analyzing, refining, and prioritizing its part of the overall Work Flow in the Pod.

This is a (15:18) Pod: there are 15 Production Teams, and 18 Product Champions *(remember that Test Lab Teams aren't considered Production Teams, even though they are highly technical…)*.

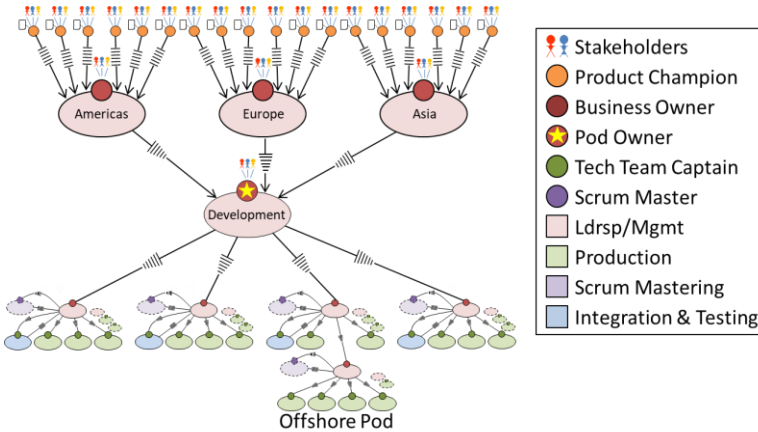

Figure 24: A Bigger, multi-layer, [15:18]Pod

The Context and Forces on this Pod

Way back when, we told you that a Scaled Organization should 'look like' it was formed based on the Forces and Context that currently exist – that it should be appropriate for its current context. So, let me tell you a story about this Pod: its context and why it looks like it does:

1) This Organization develops a retail software product consisting of core technology with individual tailoring for Clients. There are thousands of installations all over the

world, with hundreds of requests for new features and/or tailoring coming in every month.

2) There are three Business Offices, one for the Americas, one for Europe, and one for Asia. Each consists of a Business Manager, some Business Analysts, an Agile Actuary, and six Client Representatives. Each Business Office collects, analyzes, prioritizes, and manages Change Requests from Clients in their part of the world. And, of course, each Business Manager deals with some of the Stakeholders directly.

3) All Development work (both new features and tailoring) is done at a single location in the US, by a single Development Organization that contains a Development Management Team and four co-located sub-Pods that are all technically comparable. One of these sub-Pods is using an Offshore Pod as sub-contractors that increases its capacity, but does not add to its capabilities.

Different Kinds of Management Teams

In the small, single-layer, Pod there was a single Management Team, and it had five primary responsibilities: 1) developing and managing the Pod's Results Backlog, 2) distributing Stories to Production Teams to work on, 3) providing Guidance and Objectives (GOs) to the Team Captains and Product Champions, 4) maintaining necessary Delivery Forecasts, and 5) improving the functioning of the Pod.

In this larger, multi-layer, Pod, there are multiple Management Teams that are of three types:

1) There are (possibly) *many* Management Teams that receive several prioritized input Work Streams and produce a single prioritized output Work Stream. We call these Teams **Consolidation Teams**; they *organize and prioritize* the work.

2) There are (possibly) *many* Management Teams that receive a single input Work Stream and produce several output Work Streams. We call these Teams **Distribution Teams**; they *decompose and assign* the work.

3) This is a *single* Pod Management Team (containing the Pod Owner as its Team Captain) that receives several input Work Streams and produces several output Work Streams, using the Pod's Results Backlog as a 'holding pen'. This Pod Management Team acts as both a Consolidation Team and a Distribution Team.

As in our simple model, there can be a layering of Consolidation Teams, and a layering of Distribution Teams, but there will always be only one Pod Management Team 'in the middle'. Essentially, the Pod's Management Team connects one or more Consolidation sub-Pods with one or more Distribution sub-Pods, each with its own sub-Pod Owner.

In Figure 24 there are three Consolidation sub-Pods (above the Pod Management Team) and four Distribution sub-Pods (below the Pod Management Team).

From a Pattern perspective, the Pod Management Team treats the Consolidation sub-Pod Owners as if they were Product Champions and treats the Distribution sub-Pod Owners as if they were Production Team Captains.

With these conventions, the Responsibilities of the Pod Management Team are the same four as they were for the Small Pod (see above), with the following caveats:

- the refinement inherent in *'developing and managing the Pod's Results Backlog'* may be shared with the Consolidation sub-Pods, and

- the refinement inherent in *'distributing Stories to Production Teams to work on'* may be shared with the Distribution sub-Pods.

How this sharing of refinement is done is determined by the Pod, and is dependent on the type of work being done and the capabilities of the people and Teams.

The Fractal Nature of the Pod

In our Large Pod (see Figure 24), each Business Office is a [1:6]Pod with its Business Manager as sub-Pod Owner, and its six Client Reps as Product Champions. Each Business Pod collects, analyzes, refines, and prioritizes Change Requests from its part of the world that are then sent to the Development Pod's Management Team.

The Development Pod's Management Team contains the Pod Owners from the Consolidation and Distribution sub-Pods, the overall Pod Owner, and possibly many SMEs. It receives the Change Requests (both Epics and Stories), and further refines, prioritizes, and distributes them to the four Development sub-Pods (three [3:1]Pods and one [6:1]Pod) who will get the work to Done.

While the work flows through the system, the Product Champions are accountable for identifying and/or providing Business SMEs to the Management and Production Teams in order to help with Refinement and Production.

Now, as all this is going on, each Team Captain and Product Champion is empowered to make decisions, taking into account the Guidance and Objectives (GOs) that apply to them. These GOs start at the top, with the overall Pod Owner, and flow down. We will discuss this issue later, when discussing the Governance Patterns.

As you can see, this is a fractal application of the Management Team Pattern; we've connected three Business Pods and four Development Pods with a Pod Management Team to create a bigger, multi-level, Pod.

Pattern: Pod Scrum Master Team

Not surprisingly, there should be a Scrum Master Team for this bigger Pod, and this is in addition to the Scrum Master Teams for each of the sub-Pods. The Pod's Scrum Master Team includes the Pod's Chief Scrum Master (CSM) as well as the CSMs from each of the Development and Business sub-Pods. Each of the Scrum Master Teams in the Pod has its own Improvement Backlog and acts like the Pod Scrum Master Team described in the 'Small Pod' section.

Note: The Business sub-Pods are only one Team deep – so their CSM is actually only a Team Facilitator – but if they were deeper they'd have a 'proper' Chief Scrum Master…

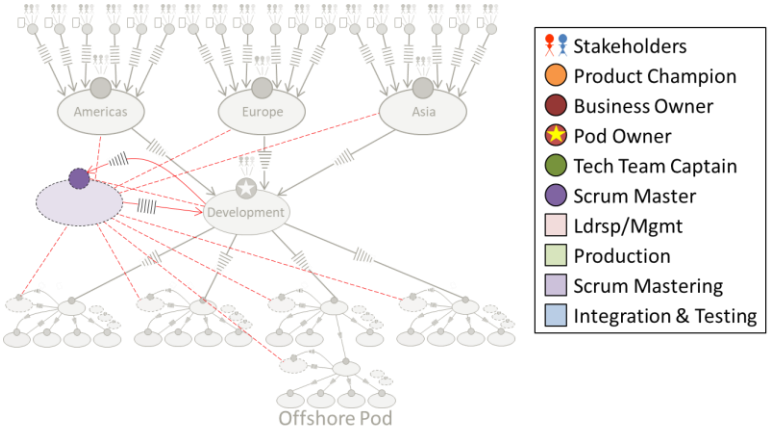

Figure 25: Adding a Scrum Master Team

We have added the Offshore Pod's CSM to the overall Pod's SM Team in this diagram. We could just as easily have the Offshore Pod's CSM added to the SM Team of the sub-Pod it belongs to – this is a self-organization issue.

Pattern: Test Lab Team

Let's add a Test Lab to the big Pod to replace Test Labs on the lower-level Pods (The Production Trains). This is an example of **Leveraging the Economies of Scale** in order to be efficient. As you can also see, one of the Production Trains still has its own Test Lab, which is an example of **De-Centralizing for Effectiveness** (it probably needs it to support its Offshore Pod). Basically, we can imagine that the three other Production Trains combined and streamlined their Test Labs into one 'bigger' one at the next level up...

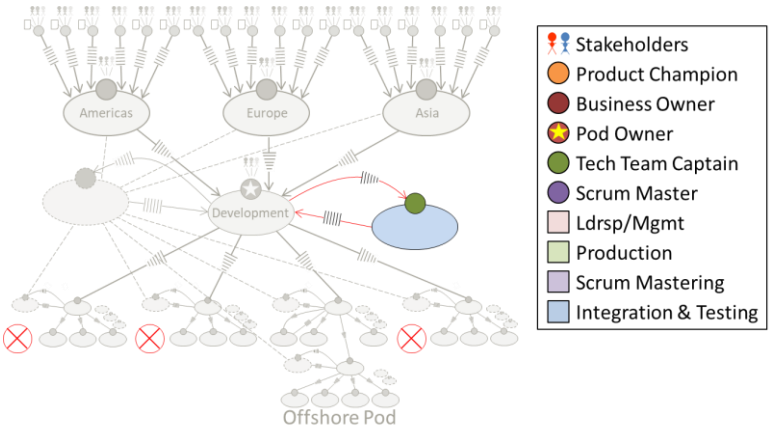

Figure 26: Adding a Test Lab

It is common to have many different Test Labs in a large Pod (or Group), especially if the Organization is working on many

different Products. Multiple Test Labs allow for extensive User Acceptance Testing and Reviews, which is always a good thing.

Pattern: Cross-Cutting Workgroups

Teams (Production, Management, and otherwise) often need to work together (even across Pod boundaries) in order to resolve dependencies, remove common impediments, share learnings and expertise, develop solutions to cross-cutting problems, and so on. One way to do this is to use Cross-Cutting Workgroups that help **Align the Organization on** both **what work should be done** and **how to do** its work.

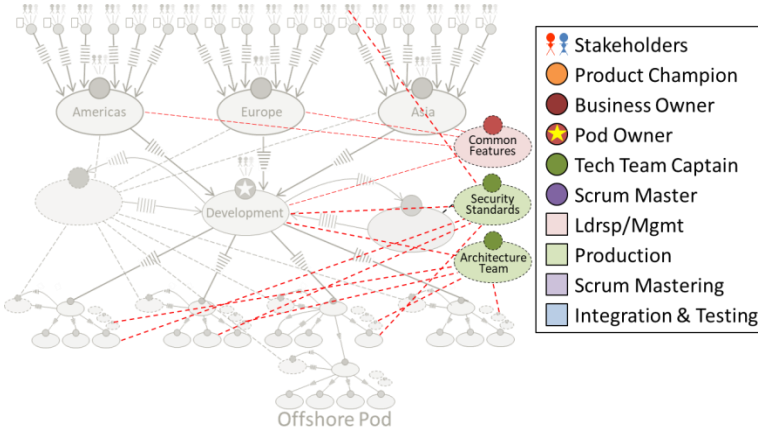

Figure 27: Examples of Cross-Cutting Workgroups

In this case, we see three Cross-Cutting Workgroups:

- The "Common Features" Workgroup includes Business Analysts across the Business Pods who are analyzing the incoming Change Requests in order to find common features to add to the common core of the system. Its

Team Captain is the Senior Business Analyst, who is a technical SME on the Pod Management Team.

- The “Architecture Team” is a Workgroup that gets its members from the Development Pod’s Architecture Teams. This team is solving organization-wide problems and finding common architectural and design Patterns. Its Team Captain is the Pod’s Chief Architect, who is also a technical SME on the Pod’s Management Team.
- The “Security Standards” workgroup is acting as a Center of Excellence on Security Issues, and includes a Stakeholder from outside the Organization (possibly a Regulator). The “Security Standards” Team Captain is the Pod’s Security Expert, who is also a technical SME on the Pod’s Management Team.

Discussion of the Large Pod

The Large Pod we just discussed was only an example, not a template or recipe. However, it did illustrate several Patterns that can be used to help the pieces of a large Organization work together as a Team-of-Teams.

There are other Patterns that can be used as well, including the Community of Practice we discussed before. Here is a diagram of the complete [15:18]Pod we’ve been developing.

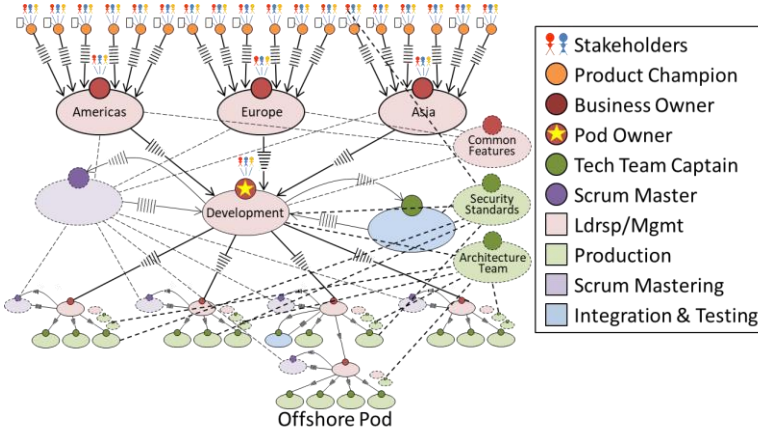

Figure 28: The Complete [15:18]Pod

Basically, building a large Pod is a purely fractal thing. Put the Product Champions at one end, the Production Teams at the other, and connect them with enough Management Teams that no one of the sub-Pods is too large. There are many different ways to do it, and the one you wind up with depends on the Context and Forces that are/were in play.

Group Structural Patterns

So far, we've been talking about Pods, now it's time to talk about Groups. Remember that Pods exist to manage work flow from Stakeholders to Scrum Production Teams. A Pod extends the concept of a single Scrum Team producing a single Product for a single set of Stakeholders to a Team-of-Teams producing the same, or similar, Products for (groups of) Stakeholders.

A Pod is all about the Work Flow… and it *should* be understood as a *single* Work Flow, with the possibility that *any* of the Pod's Production Teams could produce Results for *any* of the Pod's Stakeholders (represented by a Product Champion).

A Group is different. A Group is a collection of Pods (and smaller Groups) that produce different kinds of Results for their Stakeholders. A Pod is about a single, complicated, Work Flow; a Group is about providing value across multiple Work Flows.

A Group Owner does not manage a Work Flow; a Group Owner leads the Group by providing Guidance and Objectives (GOs) to Pod Owners (and sub-Group Owners) that maximize the overall value the Group produces. Each subordinate Owner (either Pod or sub-Group) then manages/leads based on these GOs, and is accountable to the Group Owner to do so.

Most of the Patterns that apply to a Large Pod will also apply to a Group, so we'll just show a couple of them. As our example we'll use a Group that is very similar to the large Pod we just used, but the forces on the Organization were different as it scaled.

Pattern: Leadership Team

The Group we show next has the same basic capability as the Large Pod we studied in the last sections. It is very similar to that Pod: it has 15 Production Teams, 18 Product Champions, and

develops and maintains a retail software product with a world-wide Client Base. However, it didn't scale based on the same forces, or in the same context.

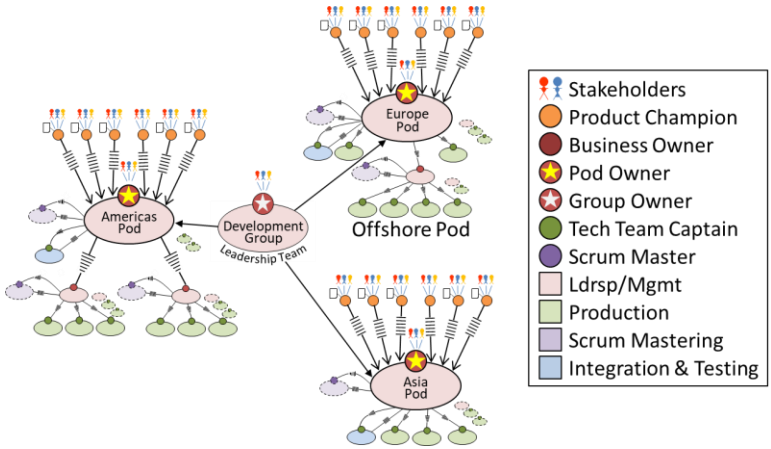

Figure 29: A Development Group

The Context and forces on this Group

So, let me tell you a story about this Group: its context and why it looks like it does:

1) This Organization develops a retail software product consisting of core technology with individual tailoring for Clients. There are thousands of installations all over the world, with hundreds of requests for new features and/or tailoring coming in every month. *(this part is the same…)*

2) There are three Business Offices, one for the Americas, one for Europe, and one for Asia. Each consists of A Business Manager, some Business Analysts, an Agile Actuary, and six Client Representatives. Each Business Office collects,

analyzes, prioritizes, and manages Change Requests from Clients in their part of the world. And, of course, each Business Manager deals with some of the Stakeholders directly. *(this part is the same…)*

3) Because of Language and Tax differences, each Business Area has its own Development Pod that does tailoring work for the Clients in its part of the world. *(this is different…)*

   - The European Pod is a [6:1]Pod that has incorporated an "Offshore Pod" in India, and added its own Scrum Master Team, Test Lab, and Cross-Cutting Workgroups.
   - The Asian Pod is a [3:6]Pod that has added its own Scrum Master Team, Test Lab, and Cross-Cutting Workgroups.
   - The Americas Pod is a 2-layer [6:6]Pod that has added Scrum Master Teams on both layers, a single Test Lab Team, and Cross-Cutting Workgroups on each layer.

4) The Development Pod for the "Americas" Pod is in the US, and does all the development of new features, as well as the tailoring for the Americas. *(this is different…)*

As you can see, this Group has the same "size" as the Large Pod we were working with, but it has a completely different personality. It is not cohesive from a Development Point of View, but it must be viewed as a cohesive whole from a Business Point of View. This is definitely a Group, not a Pod.

Leadership Team

What holds a Group together is a Leadership Team, as we see in Figure 29. The Group's Leadership Team (Ldrsp Team) contains the Group Leader as its Team Captain, and the Pod Owners (and any sub-Group Owners) as its primary Team Members. And, of course, there is a Team Facilitator (who is probably the Chief

Scrum Master for the Group, and there may be Technical SMEs, as well.

The purpose of the Group's Ldrsp Team is to maximize the value the Group produces. It does this by providing Direction for the Group. This direction takes the form of Guidance and Objectives (GOs) issued to the subordinate Pod and sub-Group Owners.

These GOs are the primary product of the Group's Ldrsp Team, which is a Scrum Team like any other. It should be small, self-organized, self-contained, value-driven, and so on… the members of the Group's Ldrsp Team have collaborative conversations about what the Group should be doing, and the GOs are the Work Results of those conversations.

The Group's Leadership Team is called a 'Leadership Team' because it is NOT managing a Backlog, it is leading people. This is a true Leadership Team; it is all about the people.

Pattern: Group Scrum Master Team

Not surprisingly, there should be a Scrum Master Team for a Group. The Group's Scrum Master Team includes the Group's Chief Scrum Master (CSM) as well as the CSMs from each of the Pods and sub-Groups. This Group's Scrum Master Team is the major force for change within the Group.

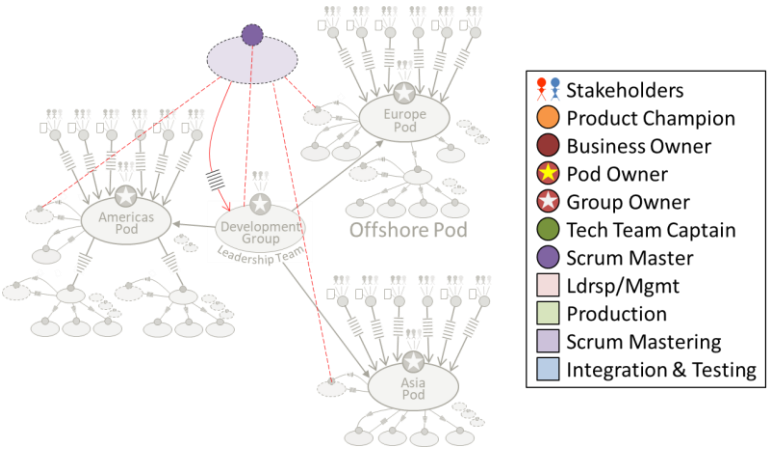

Figure 30: Adding a Group Scrum Master Team

The Group's Scrum Master Team makes recommendations to the Group Owner about how to improve the Group, and the Group's Chief Scrum Master is the Champion for the Group's Improvement Backlog. There are many ways to improve the Group, including 1) adding, splitting, combining sub-Groups, Pods, and Teams; 2) improving subordinate Groups and Pods (the members of the Group) so they can deliver high-quality Results; 3) making recommendations to subordinate Groups and Pods about how to improve, and so on.

Basically, the mission of the Group's Scrum Master Team is simple: convince the Group Owner to do whatever it takes to improve and remove the Group's impediments to delivering high-quality Results.

Pattern: Cross-Cutting Workgroups

Large Organizations (Pods and Groups) often need to work together in order to remove common impediments, share

learnings and expertise, and develop solutions to cross-cutting problems. One way to do this is with Cross-Cutting Workgroups.

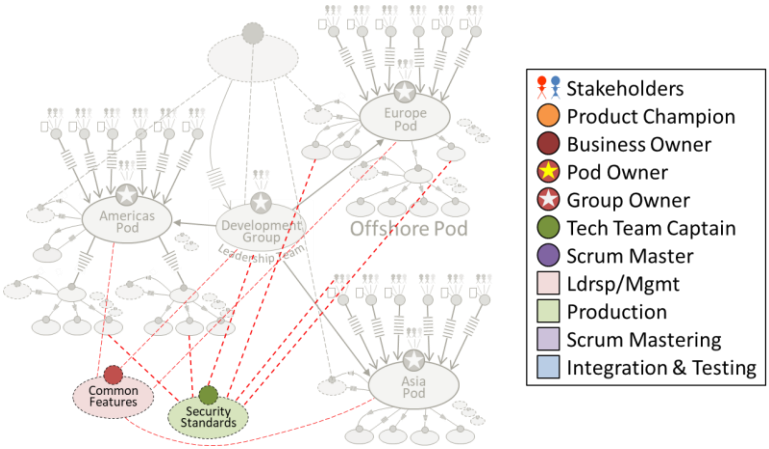

Figure 31: Adding Cross-Cutting Workgroups

In this case, we see two Cross-Cutting Workgroups:

- The “Common Features” Workgroup includes Business Analysts, across the Business Pods, who are analyzing the incoming Change Requests in order to find common features – to be added to the common core of the system. Its Team Captain is the Senior Business Analyst, and is also a Member of the Group’s Leadership Team.
- The “Security Standards” workgroup is acting as a Center of Excellence on Security Issues, and includes a Stakeholder from outside the Organization (possibly a Regulator). Its Team Captain is also a member of the Group’s Leadership Team.

Discussion of the Group

The Group we just discussed was only an example, not a template or recipe. However, it did illustrate several Patterns that can be used to help the pieces of a large Organization work together as a Team-of-Teams.

We kept it short and sweet, as many of the Patterns that pertain to Pods also pertain to Groups. For example, a Group should support Communities of Practice in order to help its people help themselves improve.

Basically, building a Group is a purely fractal thing. Assemble a bunch of Pods and sub-Groups, and put a Leadership Team (and other Teams) in the middle. It's not that hard to understand, but it is hard to do… and very powerful.

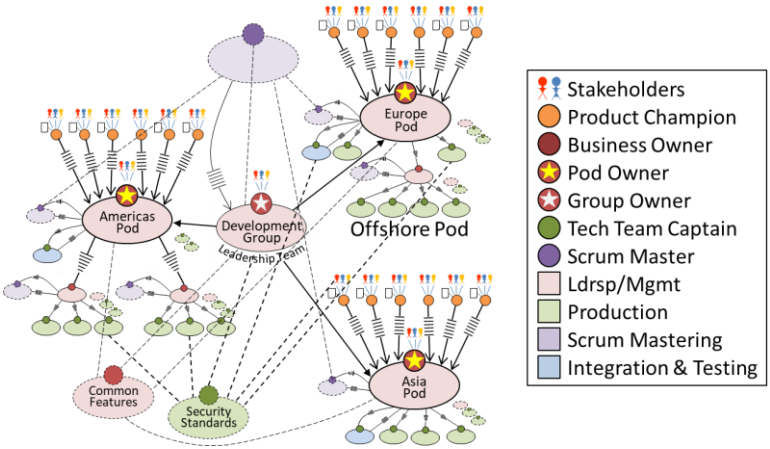

Figure 32: The Complete Group

Governance Patterns

(with Commentary)

Scaling Governance Patterns provide ways for Product Owners (of all types) to manage the flow of work and lead the people in the scaled Organization (Pod or Group). The purpose of Governance is to assure that 1) everyone is on the same page about what is being done and who is doing it, and 2) that people are using a shared Standard of Care and working together harmoniously, while managing dependencies and common issues.

Purpose of the Governance Patterns

In Multi-Team Scrum the individual Scrum Teams are all working together as a Team-of-Teams, with distributed management and decision-making. This distributed management and decision-making, along with the associated communications and the individual Teams' self-organization, is called the Organization's *governance* mechanism.

The Organization's governance is enabled by its Structure, which provides virtual Scrum Teams – often through a 'purpose' hierarchy. The Governance Patterns we describe here are mostly guidance about how to leverage the Structural Patterns in order to guide and manage the Organization.

The Governance Patterns describe how, and what, information should flow, and what decisions should be made, in order to maximize the overall value produced by the Organization. The Governance Patterns stress People over Process, focusing on Accountability, Empowerment, and Collaboration.

Pattern: Use GOs to Drive

Each Scrum Team has a Team Captain, who is accountable to the Business for maximizing the value of the Team's work. It is only fair that there is a Business Owner who gives the Team Captain some Guidance and Objectives (GOs) to help them know what will provide value – what is hoped/expected of them and their Teams.

In our Structural Patterns, Pods and Groups are 'held together' with Management (or Leadership) Teams. These Ldrsp/Mgmt Teams have a Business Owner as Team Captain and Team Members who *also* have other Product Ownership roles; they either work with Stakeholders or are Team Captains of other Scrum Teams.

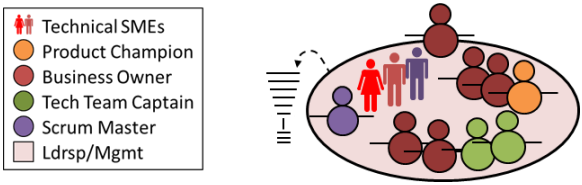

Figure 33: A 'Generalized' Leadership/Management Team

No matter what else this Ldrsp/Mgmt Team does, one of its most important jobs is to provide GOs to its Team Members that they can refer to when making decisions in their Product Ownership role. One of the main uses of GOs is prioritization guidance, and GOs are the *primary* way of providing alignment amongst the Teams in a Pod and the Pods in a Group.

It is important that the Ldrsp/Mgmt Team Members collaborate to produce these GOs so that they each have ownership of them. A Ldrsp/Mgmt Team Owner who simply provides the GOs without collaborative discussions with the Team Members is both disempowering and micro-managing.

Sample GOs

We use the term "GO" to indicate everything from a piece of Guidance to a specific Objective. The form is unimportant; a GO provides focus and intent; it provides a shared understanding about what we're doing. Anything from *"this is what I'm thinking…"* to *"make sure that…"* is a form of GO.

The easiest way to understand GOs is to see some examples, so here are a few:

- "Time to switch to **Trade Show Done**"
- "Prioritize Project ABC for now"
- "All of us need to Focus on Quality"
- "You two guys need to get your Teams talking and working together on Feature123… right now"
- "We must get a version of Product 123 out the door this week"
- "Ok, so now we have a plan for how our Teams are going to work together. Go get it done…"
- "Let's focus on fixing bugs in Feature XYZ"
- "Test Team: focus on the Demo for next week"

As you can see, GOs are whatever they need to be in order for the Organization's Teams to work together. It's a way to manage the Organization by focusing the cross-team self-organization, such as that seen in Figure 20.

Pattern: Synchronized Daily Scrums

A Pod/Group Owner is accountable for maximizing the value produced by the Pod/Group. This means that the Pod/Group Owner must have visibility into what is going on within the Pod/Group. This information is gathered by using synchronized Daily Scrums.

The Daily Scrum on a non-Production Team

On non-Production Scrum Teams, most of the Team Members represent another team (either a 'team' of Stakeholders or another Scrum Team), and the mission of a non-Production Team is to provide some service in support of those other teams.

Therefore, when a non-Production Team has its Daily Scrum, it needs to gather more information than a Production Team does. Typically, a Daily Scrum starts with each Team Member answering the following three questions:

1. What did you do for this Team yesterday?
2. What are you planning to do for this Team today? and
3. What's standing in your way?

But a non-Production Team also needs to ask something like:

4. What's going on with your *other* team that we need to know about?

Once the Team has this information, it plans its day and does its work. As a by-product of this planning, the Team Captain has the information he/she needs to take to his/her Ldrsp/Mgmt Team's Daily Scrum.

Ordering the Daily Scrums

In order for the Pod/Group Owner to get a complete picture of what is going on within the Pod/Group every day, the Daily

Scrums must occur in the right order. The purpose of a Daily Scrum is to provide information – they are for feedback, not dissemination of information or direction. Therefore, they must be synchronized correctly in order for current information to flow 'up' to the Pod/Group Owner.

The Daily Scrums start 'furthest away' from the Pod/Group Owner, and then progress towards the Pod/Group Owner. This will assure that everybody knows what needs to be known within their span of control. Here's a diagram showing the ordering of the Daily Scrums for the Small Pod we see in Figure 22.

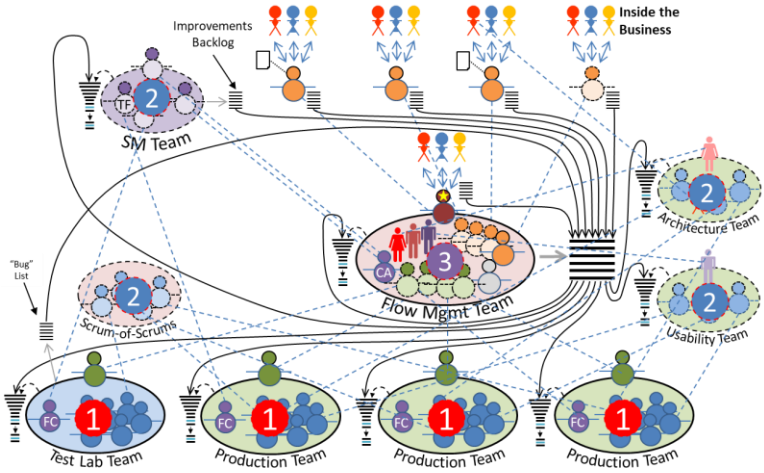

Figure 34: Ordering the Daily Scrums in a Small Pod

As you can see, this is pretty straightforward. Many Members of the Pod are attending two Daily Scrums – the one for their 'home' Team and the one for their 'away' Team – but nobody should be attending more than two. *(Note: we treated the coordination conversation the Scrum-of-Scrums has as if it were a Daily Scrum for scheduling purposes…)*

The conversations in these Daily Scrums, as well as the use of the **Collaborative Work within a Pod** Pattern, is what allows a small, single-layer, Pod to act as one big, self-organized, Team. Basically, the extra Daily Scrum (for many Team Members) is the overhead you pay if you want a Team-of-Teams to be, and act like, 'one big team'.

Once a Pod gets bigger, it gets more complicated, and the diagrams we draw get really ugly, believe me. So, rather than me drawing a picture here, draw one yourself for your Organization. Draw a diagram of your Teams, and think your way through it. Start the Daily Scrums furthest away from the Pod Owner, and do them in the right order so there's a roll-up of information to the Pod Owner. The goal is to have the Pod Owner know everything they need to know in just a few hours.

After each Ldrsp/Mgmt Team's Daily Scrum, it may have some work to do because of the information coming up from the sub-teams. The basic guidance is that any problem or issue should be handled by the 'lowest possible' Ldrsp/Mgmt Team that has all the necessary information. This works because we always have the 'right people in the room' to have the conversations that will solve the problems, and that's what makes the patterns work in the first place.

For a Group, it's the same idea, except the Daily Scrums roll-up towards the Group Owner. It's basically the same except, metaphorically speaking, except that the pieces of a Group are working in parallel, while the pieces of a Pod are working together.

Discussion of the Scrum Ceremonies

So far, we have barely discussed the Scrum Ceremonies that occur 'between the Sprints': Sprint Planning, Product Review, Progress Review, and the Team Retrospective.

This is intentional, because the Sprint has little or no impact on Multi-Team Scrum as we have described it.

I know this sounds crazy, but hear me out…

The Sprintly Ceremonies for Single-Team Scrum

Basic, Single-Team, Scrum describes how a single Team can develop a Product for Stakeholders who are represented by a Product Champion (see the *Scrum Handbook: Single-Team Scrum (STS)*). Its model prescribes that a Team handles a Continuous Flow of Work and has a Sprintly interrupt for Ceremonies (or Events) that allow for Feedback, Improvement, and Re-Planning. These Ceremonies are:

1. Product Review: The Team, the Business Owner, and the Product Champion Review the Product with appropriate Stakeholders in order to obtain Meaningful Feedback.
2. Progress Review: The Product Champion and Business Owner Review the Delivery Forecast with appropriate Stakeholders in order to evaluate risks and set expectations.
3. Team Retrospective: The Team has a Retrospective (facilitated by the Team Facilitator with possible support from an Agile Coach) to discuss ways they could improve their Practices and teamwork.
4. Sprint Planning: The Team: 1) Establishes Sprint End (when the next interrupt will take place), 2) Selects a Kaizen to accomplish, 3) Makes sure there are enough Stories Ready or 'In Progress' so the Team can get back to Work, and 4) Determines a Sprint Goal.

Discussion of the Ceremonies for Multi-Team Scrum

Multi-Team Scrum, as described in this Handbook, is about Pods (or Groups of Pods) – not just Teams – developing Products for Stakeholders, represented by Product Champions. How does that impact the Sprintly Ceremonies?

Well, for each Product, we still need Product and Progress Reviews:

1. Product Review: Pod Members who worked on the Product, the Pod Owner (or representative), and the Product Champion Review the Product with appropriate Stakeholders in order to obtain Meaningful Feedback.
2. Progress Review: The Product Champion and the Pod Owner (or representative) Review the Delivery Forecast with appropriate Stakeholders in order to evaluate risks and set expectations.

These Reviews should be held every Sprint for each Product that was worked on. Because the Pod can work on many Products for many different Stakeholders (Clients, Customers, and others), it is often impossible to do all the necessary Reviews in the day or two 'between the Sprints'.

Therefore, the Scrum Model *cannot* insist that they do.

> **Note**: For example, one of the Pods we work with is a single Scrum Team supporting thirty different Products, and there is no way they can get all the meaningful feedback they need, on all thirty products, in a single two-hour Review every two weeks.

We recommend that the Pod's Flow Mgmt Team own the scheduling of the Reviews. Because they sit right 'in the middle' of everything, they are in a perfect position to know (or to find out) which Reviews are needed, when they are needed, and who

should be there. (See the *'The Pod's Flow Mgmt Team Schedules Reviews'* sub-section of the Pod's Management Team Pattern for more discussion.)

As for the other two Ceremonies, we believe that every Team should have a Retrospective and do Sprint Planning. However, there are some special issues.

3. Team Retrospective: Every Scrum Team (not just Production Teams) has a Retrospective (facilitated by its Team Facilitator with possible support from an Agile Coach) to discuss ways they could improve their Practices and teamwork. Because there are likely more Teams than Agile Coaches, the Retrospectives may need to be synchronized in some way, which is probably handled by the senior Scrum Master Team.

4. Sprint Planning: Every Scrum Team needs to do its own Sprint Planning, where it: 1) Establishes Sprint End (when its next Retrospective will take place), 2) Selects a Kaizen to accomplish, 3) Makes sure there are enough Stories Ready or 'In Progress' so the Team can get back to Work, and 4) Determines a Sprint Goal.

There are still several unanswered questions:

Q1: Does Scrum require that the Pod/Group (as a whole) needs to have a Retrospective?

A: No. The Scrum Master Team is looking out for the Pod/Group, and could decide to have a Pod- or Group-wide Retrospective if they wished. It is not required by Scrum.

Q2: Does Scrum require that the Pod/Group (as a whole) needs to do Sprint Planning?

A: No. The Management Teams are continuously keeping the Results Backlog current and issuing GOs to their subordinate Business Owners and Product Champions. The Pod/Group is always in alignment (as a whole) about what to do, so there is no need for a specific Sprint Planning Event for the complete Pod/Group.

Q3: Do other (non-Production) Teams have Reviews?

A: Yes. Each non-Production Team's Captain needs to determine who its Stakeholders are and hold appropriate Reviews with them. This is true for all Scrum Teams, both physical and virtual.

Managing Transformation

(with Commentary)

Agile Organizations are constantly changing and improving. These changes can be small or large; they can be evolutionary or revolutionary. Revolutionary changes are often called transformations; they fundamentally change the way work gets done. An Organization in transformation is focusing on change, not product development.

When we are transforming an organization, it should transform in a way that it 'looks like' it grew because of the existing forces on it. This means that a 'large' organization must periodically re-transform itself since the forces are constantly changing. There is no 'static' large scaled organization… there can't be if it's to stay effective.

Transformations are Chaotic

Many Organizations want to become Agile, which requires improvements and transformations. The way we see it, a Transformation is much more than a simple collection of Improvements. Here are the definitions we use:

- Organizations **Improve** while doing their regular work; Improvement Stories are championed by the Organization's Scrum Masters and compete against normal Stories for prioritization.
- When Organizations **Transform**, however, the Transformation *is* their regular work; most of the other work being done aids in the Transformation – usually by validating that the Transformation is (or is not) having the expected results.

Because an Organization Improves as part of their regular work, we will not discuss Improvements here, we will only talk about Transformations. There are two types of Transformations:

1. Transformations that are *totally enclosed* within the agile part of the Organization. These Transformations often consist of single Scrum Teams or Pods going into 'training mode'.
2. Transformations that require changes in both the agile and non-agile part of the Organization.

The following figure shows this concept, where the 'red circles' depict where the Transformations are taking place. The 'red circle' labeled "1" is totally enclosed within the agile part of the Organization, while the one labeled "2" requires changes in both the agile and non-agile parts.

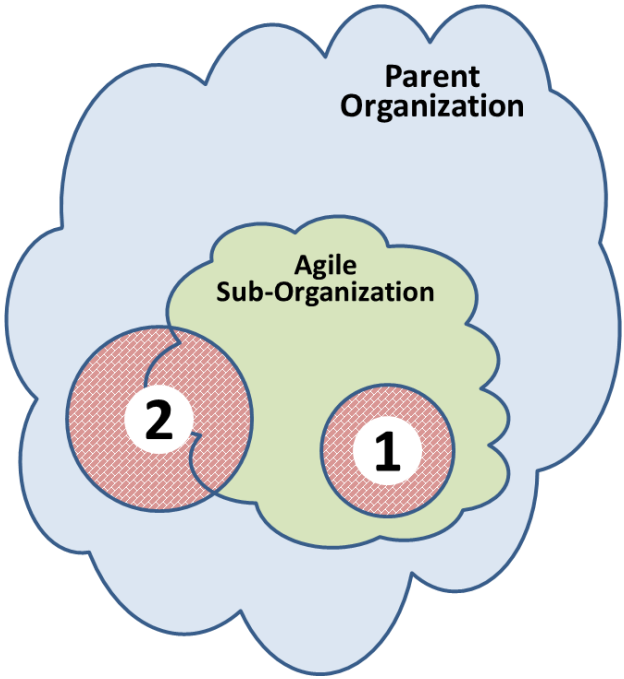

Figure 35: Two Types of Transformations

Let's look at each of these types of Transformation, as they are managed differently.

Pattern: Internal Transformation

A Transformation that is limited to changes inside an Agile Organization is managed inside that Organization. The idea is that the Organization is improving itself by Transforming one or more of its sub-Organizations. While the sub-Organizations are being Transformed, they are *not* doing their normal work – they are dedicated to being Transformed.

Note: A good analogy to this concept is improving one's home by remodeling the kitchen. While the kitchen is being remodeled, it is **not** being used as a kitchen. If it were, the changes would be merely Improvements, not a Transformation.

In order to ensure that the sub-Organization being Transformed is dedicated to being Transformed, we remove its Team Captain (which I'm using as a generic title) from the usual chain of command. The Organization's Team Captain is then reassigned to the Transformation Owner for the duration of the Transformation.

The best way to understand this is to take a look at Figure 36, which shows a single-layer Pod that is Transforming one of its Production Teams. The Pod's Change Agent is acting as the Transformation Owner, and the Pod's Scrum Master Team is 'in charge' of the effort.

Note: that the Scrum Master Team is also doing its normal job of Improving the Rest of the Organization. It is Improving the Organization while it Transforming the Production Team at the lower-left.

As you can see, the Scrum Master Team has been augmented with Agile Coaches who are assisting in the Transformation, and the Team Captain whose Production Team is undergoing the Transformation is moved onto the Scrum Master Team, as well. The Scrum Master Team is still doing its 'normal' work in the Pod, but is also *in charge* of the Transformation. The work of the transformation will show up in two places: as part of the Scrum Master Team's Backlog, and as part of the Team Backlog of the Team being Transformed. The management of the Transformation will be done within the augmented Scrum

Master Team that will be working closely with the Team being Transformed.

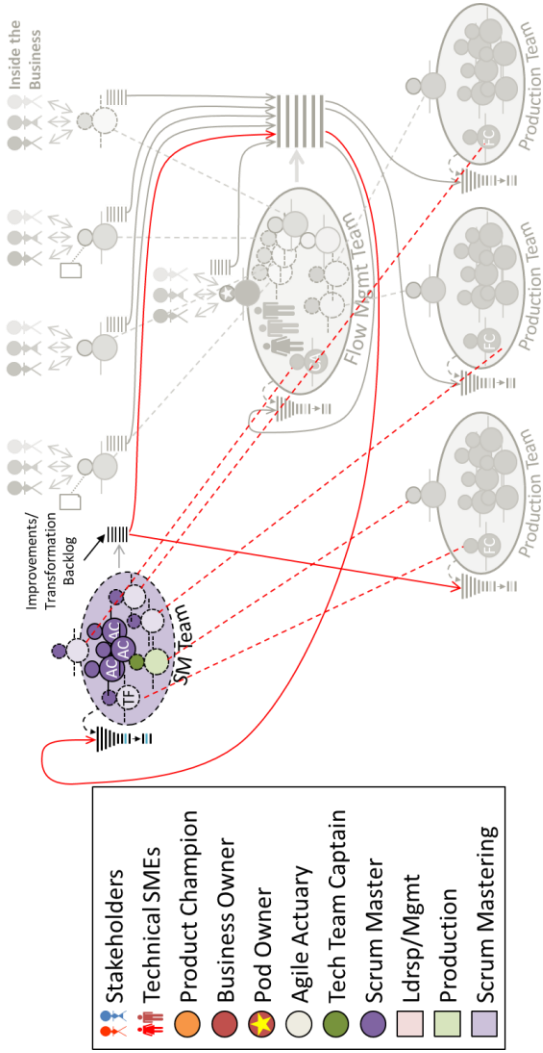

Figure 36: an Internal Transformation Team

Pattern: Agile Transformation Team

Many times, an Organization's Agile Transformation requires changes to both the Agile and non-Agile parts of the Organization. This requires that the Agile Transformation Team (ATT) be outside the Agile part of the Organization, and that it contain "People with Power" (PwPs) – people who can affect the necessary changes in the non-Agile part of the Organization.

Figure 37 shows a typical Agile Transformation Team (ATT).

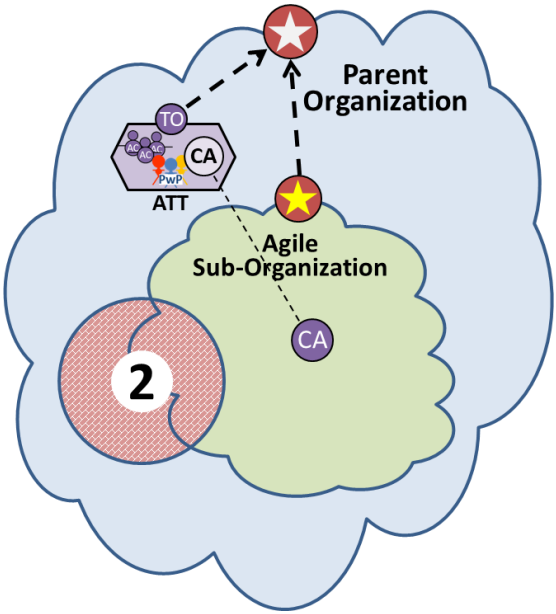

Figure 37: Agile Transformation Team (ATT)

As you can see, this Agile Transformation Team (ATT) contains:

- A Transformation Owner (TO) (a combination Business Owner and Scrum Master) who is accountable to the 'Big Boss' for implementing the Agile Transformation;

- "People with Power" who can *get things done* in the non-Agile part of the Organization;
- Agile Coaches who supply subject matter expertise on Agile Transformations; and
- The existing Agile sub-Organization's Change Agent, who is managing the part of the Agile Transformation that is inside the Agile sub-Organization, probably through using existing Scrum Master Teams as we saw in the preceding "Internal Transformation" Pattern.

Notes:

1. The Agile Transformation Team is not necessarily a Scrum Team, which is why it is shown as a hexagon rather than an oval.
2. The Agile sub-Organization's Team Captain is **not** a member of the Agile Transformation Team. He or she has value to produce, and shouldn't also be worried about the Agile Transformation that is happening within his or her sub-Organization.
3. The Agile Transformation Team goes by many different names in different contexts; two common ones are the "Organization Transformation Team" and the "Executive Action Team (EAT)". The name is not important, what it does is what's important.

Discussion of Agile Transformations

The first version of an Organization's Agile Transformation Team (ATT) is often the first 'Scrum' Team that exists. The ATT may shepherd the growth of the Agile Organization from a single Production Team to a Group.

As the Agile Organization grows, from one Scrum Team, to several Scrum Teams, to a Pod, then several Pods, and finally a full-fledged Group, the Agile Transformation Team may change, as well. It can take several forms, such as:

- It can consist of the Organization's Change Agent all by themself, working with the 'Big Boss' to create the first few Production Scrum Teams;
- Once there is a small Scrum Pod to work with, the ATT can look we see in Figure 37; and
- As the Agile Organization grows into a Group, the ATT could morph to become its own Pod, perhaps called something like the "Agile Center of Excellence."

When it comes to Transformations, there are no hard-and-fast rules. Put somebody powerful in charge, give them a good Team of Agile Coaches and 'People with Power' to work with, give this new Team the Mission to 'Transform this Organization to Scrum', and get out of their way and let them get to work.

Then watch what happens…

MTS Glossary

Scrum has confusing terms. We have listed many of them in the Glossary of the "Scrum Handbook: Single-Team Scrum (STS)," but there are some that pertain only to Multi-Team Scrum. This Glossary contains mostly multi-team-specific terms, but also has a few single-team-specific terms as well.

Agile Actuary | A professional who does qualitative and quantitative analysis to help Business Owners mitigate risk, support decision-making, evaluate options, and produce the Delivery Forecast.

Business Owner | A Product Ownership role that represents a person who is accountable to the Business for maximizing the value of Deliverable Results that could represent one, or many, Products. *(see Product Champion)*

Chief Scrum Master (CSM) | The Team Captain of a Scrum Master Team; we expect to see a CSM for every Pod, sub-Pod, Group and sub-Group. *(see Team Captain, Scrum Master Team, Pod, Group)*

CSM | (Chief Scrum Master)

Cross-Cutting Workgroup | In a Team-of-Teams, a Cross-Cutting Workgroup is a Virtual Scrum Team, consisting of Team Members from multiple Teams, that has a specific mission or problem to solve. *(see Virtual Team)*

Delivery Forecast | The Delivery Forecast predicts Delivery Dates and Costs, and it should be based on Data, not Hope. It is important to note that the Team telling you they will deliver on time is not Data. Telling the Team what it will deliver, and when, is not Data.

GO | Guidance and Objectives.

Governance | An Organization's Governance Mechanism is the method it uses to manage and make decisions.

Group | A Group is a collection of Pods and sub-Groups along with a Group Leadership Team. *(see Pod, Group Leadership Team)*

Group Leadership Team | A Virtual Scrum Team that consists of the Group Owner (as its Team Captain) and its subordinate Pod and sub-Group Owners. *(see Virtual Team, Team Captain, Group Owner, Pod Owner)*

Group Owner | A Business Owner who is accountable for maximizing the Value produced by a Group; the Group Owner is the Team Captain of the Group Leadership Team. *(see Group, Group Leadership Team)*

Improvement Team | another term for Pod/Group Scrum Master Team. The Improvement Team creates and manages the Pod/Group's Improvement Backlog.

Leadership Team | (Group Leadership Team)

Management Team | (Pod Management Team)

Pod | A Pod is a Team-of-Teams consisting of Production Teams, sub-Pods, a Pod Management Team, and other (physical and virtual) Teams, that produces similar Products for similar Product Champions. *(see Product Champion, Pod Management Team)*

Pod Owner | A Business Owner who is accountable for maximizing the Value of the Results produced by a Pod; the Pod Owner is the Team Captain of the Pod Management Team. *(see Pod, Team Captain, Pod Management Team)*

'Power' Hierarchy | A management hierarchy used to project power and manage people. *(see 'Purpose' Hierarchy)*

Pod Management Team | A Virtual Scrum Team that has the Pod Owner as its Team Captain, and its (immediate) subordinate Team Captains, Product Champions, and sub-Pod Owners as Team Members. *(see Virtual Team, Team Captain, Product Champion, Pod Owner)*

Product Champion (PC) | A Product Ownership role that represents the person who represents the interests of a cohesive set of Stakeholders focused on the same Product; *(see Business Owner)*

'Purpose' Hierarchy | A management hierarchy used for collaborative decision-making in order to manage and provide value. *(see 'Power' Hierarchy)*

Scaling Scrum with Scrum® (SSwS) | A Scaling method that leverages what we already know about Scrum rather than developing new concepts.

Scrum Master Team | The Scrum Master Team is a Virtual Scrum Team consisting of all the Scrum Masters of the subordinate Teams, Pods, and Groups; with the Scrum Master of the Ldrsp/Mgmt Team as its Team Captain. There should be a Scrum Master Team for each Pod and Group, even the subordinate ones. *(see Virtual Team)*

Structure | An Organization's Structure shows how the Teams, Pods, Groups, and Individuals are connected and related to each other.

Team Captain | A Product Ownership role that represents the Team Member who is accountable to the Business for maximizing the value of the Team's Work. *(see Business Owner)*

Transformation Owner | a combination Business Owner and Scrum Master who is accountable for implementing an Agile Transformation. The Transformation Owner is often the Team

Captain of an Agile Transformation Team. *(see Transformation Team)*

Transformation Team | a Scrum Team that shepherds an Organizations transformation to Agility. A Transformation Team contains Agile Coaches and 'People with Power' who can 'get things done', and is external to the Organization being transformed.

Virtual Team | A Team consisting of Team Members who 'live' on another Team.

Workgroup | (Cross-Cutting Workgroup)

Acknowledgements

We have both been involved with Scrum for over 20 years, and we'd like to thank Ken Schwaber and Jeff Sutherland for bringing Scrum to the software community. Without them, there would be no Scrum.

We would also like to thank all the teams we have worked with and observed, and all our students who have brought us news and information about their Scrum teams. We know that "*Scrum is what successful Scrum Teams do,"* and without this feedback about what successful Scrum Teams do, there would also be no Scrum.

Good luck, and happy scrumming!

About the Authors

Dan Rawsthorne

Dan has developed software in an agile way since 1983. He has worked in many different domains, from e-commerce to military avionics. He has a PhD in Mathematics (number theory), is a retired Army Officer, and is a Professional Bowler and Coach. Dan is very active in the Agile/Scrum community and speaks at conferences and seminars. He is a transformation agent, helping Organizations become more successful through agility. His non-software background helps him immeasurably: his mathematics back-ground tells him to look for underlying problems rather than focus on symptoms; his military career gave him experience in teamwork and empowerment; and his work with bowlers helps him understand that coaching is a two-way street.

Doug Shimp

Doug has worked in the technology field since 1992 and has played many key roles on software teams, including Coder, Tester, Analyst, Team Leader, Manager, Coach, and Consultant. Doug's passion is for team learning to improve product development, and he is a leader in the area of Agile/Scrum transitions and applied practices. He believes that the core basis for applied agility is that 'You must see the result for it to be real; otherwise it is all just theory…' Much of his experience with teamwork and agility comes from outside the software field, including an earlier career as an owner/manager of a painting company – which enables him to learn about small-team dynamics in a very hands-on way.

For More Information

To learn more about 3Back, LLC and our Scrum-related services contact us at info@3back.com. Follow @Scrum_Coach on Twitter, and to subscribe to our newsletter, visit: https://3back.com/blog/.

We Make Teams Better

We don't just talk Agile, we live Agile. Our 3Back Team is a well-formed, Agile team; applying Scrum$_H$ in our own workplace. From our hands-on support staff to our seasoned consultants and trainers, every member of the 3Back Team is, at a minimum, a CSM (Certified Scrum Master). Within every level of the 3Back Team, we bring a real-world appreciation and understanding of your team's needs.

For Help with Your Company's Scrum: info@3Back.com

For Training: training@3Back.com

Managing Work

At 3Back we are a fully distributed team. We actively build and manage Get To Done (gettodone.com), an online Scrum software development tool. Get To Done helps us train as a pedagogical tool, explore new ways of collaborating and focus our most precious resource – attention – on work being done.