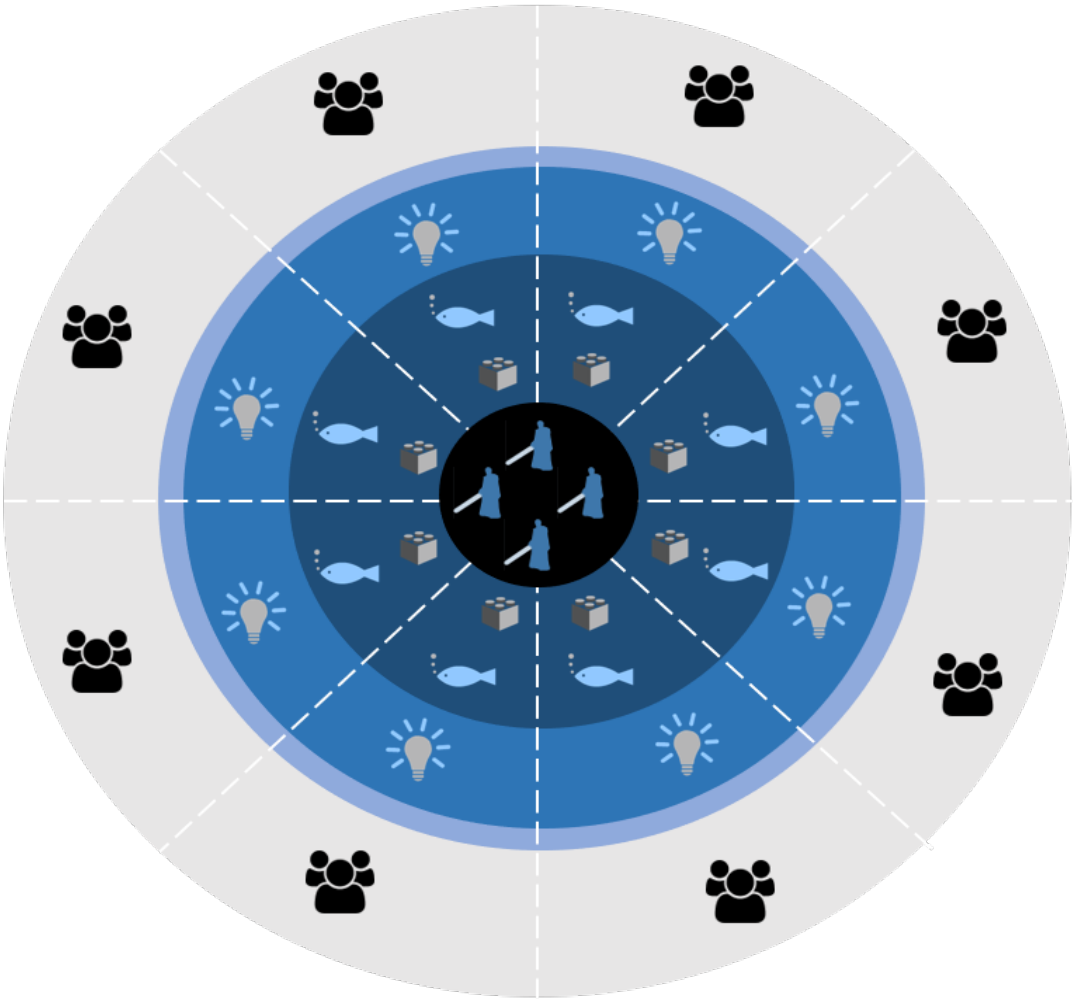
Agile Organizational Design

Growing Self Organizing Structure at Scale



By Jeff Anderson

Agile Organizational Design

Organizing towards innovation, agility and greater humanity to met the needs of the modern age

Jeff Anderson

This book is for sale at
http://leanpub.com/agileorganizationdesign

This version was published on 2020-11-03

Contents

Introduction **1**
Principles for Agile Organizational Design 2
So Why Am I Writing About Agile Organizational Design? 4
Sources of Inspiration: 5

From Industry to Uncertainty **7**
The Industrial Era 7
The Rise Of The Industrial Machine 9
Ch-Changes, Turn and Face The Strange 11
From People to Machine, back To People Again 14
Time For A Radically Different Approach 15
The Modern Organization 16

The Team is The Core Building Block **22**
A Mental Model For Your Organization 22
The Power Of Decentralization 23
The Goal of Self-Organization 24
Why Agile Teams Work 25
Agile Teams Deliver Value To The Market 28
Functional Departments Grow Capability 32

Other Building Blocks To Scale Agile Teams **35**
The Need To Build Upon Agile Teams 35
There Are No Independent Teams 38
Foundational Organizing Structures 41
The Market 43

The Identity 45
The Edge 47
The Core 50
The Majority Works In The Edge 51
The Core Travel to The Edge 53
Agility Does Not Come from Agile Teams 55
Organize Around Social Boundaries 59

Facilitating an Organization Mapping Exercise 63
Before we get started 63
1 - Identify Market Actors 66
2 - Define Identifying Goals 69
3 - Map Edge Capabilities and Teams 75
4 - Define Core Capabilities, and Teams 77
A Special note On Context Boundaries 79

Refining Your Organization Through Collaboration Patterns 81
Why are Patterns Important 81
Collaboration patterns 84
Dedicated Team Member 85
Traveler Pool Member 88
Service Provider 92
Enablers 97
Communities of Practice 102
Dynamic Team Member 104
Agile Ecosystem 107

Defining Jobs and Roles in an Agile Organization 112
Traditional jobs and roles suck 112
The Sports Team: A Metaphor for Thinking About Jobs . 115
People Play Many Different Roles 119
Example - A Typical Tech Product Organization 123
Role Personas 126

Reimagining The Hierarchy 131

The Industrial Hierarchy Is Broken 131
Seniority Equals Mastery Not Authority 132
The Advice Process 137

Scaling Agile Behavior For Continuous Organization . . 139
Truly Stable Teams Erode Agility 139
The Need For Continuous Evolution 143
Agile Enables Teams To Organize 144
Agile Behaviors of a Self-organizing Team 146
Scaling Agile Enable Larger Groups To Re-Organize . . 147
Agile Behavior For Re-Teaming 148

Facilitating Continuous Organization With Agile Practices 150
Forecasting Change With Agile Long Term Planning . . 150
Organizing Around The Work With Kanban 161
Balance Stability With Eliminating Hand Offs 172

Using Kanban to Enable Continuous Evolution of Organizing Structures 177

Introduction

The traditional approach to organizational design has got it wrong, completely wrong.

The way we structure organizations, along with the accompanying management system, is based on concepts that are over 100 years old, and is built to meet the needs of the industrial age. The goal back then was to drive down the price of production, and eliminate scarcity. Organizations did this through the principles of division, standardization and control.

The machine is broken…

But the world looks very different today. We are in a hyper competitive, global economy. Markets are both crowded, and quick. Users are fickle, and our customers are differentiated and exacting in their needs.

Volatility, Uncertainty, Complexity, Ambiguity, Oh My!

We are in an age of change, an age of uncertainty, an age of complexity. Organizational Design needs a new approach to help us organize in a way that helps meet and exceed the needs of our current era. For organizational design to meet the needs of today we need to take up new methods guided by a very different mindset than the one that help up design organization for the industrial age.

Principles for Agile Organizational Design

I believe this book will present a very different approach to organizational design than what you would see in most of the established texts. I'll provide pragmatic tools to help the reader make progress on the following **principled behaviors**:

1. **Organize for Cross Functionality**

Key to designing the new organization is the idea that team matter more than functional departments. Teams that are cross-functional. Comprised of people who are both experts in their respective fields,

but also able to pinch hit and swap roles with other team members. We need to design for teams that are capable and empowered to get the job done with a minimum of interference.

2. **Organize Through Markets**

When you want to scale the concept of teams across your enterprise, you don't do it by adding in a slew of mandated centralized services. You don't add in layers of mandated governance, coordination, and bureaucracy. You connect teams through markets. Firstly, you place the majority of people in teams that have direct contact with real customers. Secondly, you set up any enterprise services up as voluntary to use by market facing teams, in this way you being the benefit of market mechanics inside your organization; and avoid the cruft, senselessness, and even immorality that comes with command side decision making.

3. **Organize With Flexibility**

For teams to be effective we need the team roster to exhibit stability. But the real world is not always so accommodating. Change happens. Whatever team and org structure we come up will only only be correct for a short period of time. The market will change. And we will need to adapt. We want stability, but we also want to eliminate hand offs. These are forces that counteract each other. So we need to empower every knowledge workers with the insight and understanding required to form together into teams that create value without hand-offs. This constant re-shifting takes effort and will take time, but is critical if we want to avoid stagnating back into a world of dependencies and handoffs.

4. **Organize Around Social and Domain Boundaries**

Teams of 5 -8 is often the golden rule in most agile circles. But if we want to scale with agility we need to consider how to achieve social density are larger scales. We can use use *dunbar's number*

to group people according to different social outcomes. (5, 15, 35, 150) We can take a page from *Domain Drive Design* and align our organization and solution architecture according to domains that can be delivered and managed by independent, full stack teams. In this model it is critical that leaders Influence outcomes using social density not command and control.

As I go through how to incorporate these principle into your organization, I'll share a set of *organizing constraints* we can use when thinking about our organizational design. Ignoring any one of these constraints means we are likely to have less organizational agility a a result.

So Why Am I Writing About Agile Organizational Design?

The need for a better approach to getting organized, especially at scale is obvious; at least I hope it is.

My goal in writing this book is to provide a set of thinking tools and design skills to help you grow formal org structure and promote informal org structure required to foster agility. The tools are designed to support incremental evolution to a new normal when working at scale as change agents won't often have luxury of transforming an entire organization in one go. Hopefully this will make it easier to secure the permission they need to try experimenting with different ways to get organized.

Some will say a lot of good material already on the topic already exists, so why am I taking a crack at it?

Well, a good deal of content on the subject comes from a wide variety of sources. Having spent over a decade working on the subject, I have had to pull on a lot of different sources of content.

My team[1] and I have also integrated and extended a lot of this material into some thinking tools and facilitation exercises. This thought-ware has been refined from over a decade of experience across over several dozen agile change initiatives. I want to take all of the different ideas and experiences that my team has used to effect better organizational design, and get as much of these ideas as I can under one roof; presenting it in one consolidated package. I am hoping this will help others increase the agility in their organization by recasting their organization to one that will foster self organization and customer feedback.

Sources of Inspiration:

- David J Anderson's Kanban method[2], specifically using Kanban to perform long term planning with an eye to manage the dynamic formation of feature teams, and allocation of specialist pools
- the works of Donald Reinertsen[3], whose principles of flow shape many of my thoughts, but especially the idea that process and organizational structure can be defined based on reusable patterns, like the classic GOF Patterns do for programming.
- Jurgen Appelo's Management Books[4], especially his thinking on flatter organizations, wide job functions with narrow roles, and informal leadership
- The LeSS[5] concept of a traveller
- Beyond Budgeting[6] in general; it's principles, concepts and values

[1] https://agilebydesign.com/about_us/
[2] https://edu.leankanban.com/users/david-anderson
[3] http://reinertsenassociates.com/books/
[4] https://jurgenappelo.com/
[5] https://less.works/
[6] https://bbrt.org/

- the Oreilly books on Micro-Service Architecture[7] and Build[8] specifically where they intersect with concepts from Eric Evans Domain Driven Design[9], eg: Bounded Contexts, Domain Aggregates and Context Integration patterns to help teams align to solution architecture
- Recently, our team has been leveraging the excellent book Organizing for Complexity by Niels Pflaeging.[10] The book is an awesome read for a number of reasons. This text has helped me shift my conceptual model of how to describe various organizational structures.

The concepts from all these sources work amazingly well together, but it can be a tough slog for many to read all of this. It took a decade for us[11] to read all of it, try it, synthesize it, and learn from it.

Hopefully my work will be a more approachable, integrated path to getting a handle on all of this amazing content, all with real examples and lessons learned! I think these concepts can be learned and even mastered (if mastery in this domain can really exist) in a much shorter period of time, and I want this work to help.

[7] https://www.oreilly.com/library/view/microservice-architecture/9781491956328/
[8] http://shop.oreilly.com/product/0636920033158.do
[9] http://domainlanguage.com/
[10] https://www.nielspflaeging.com/books/
[11] https://agilebydesign.com/

From Industry to Uncertainty

Why do so many organizations seem so dysfunctional? A lot of it has to do with the fact that way we think how our organizations should be structured, and the way we think the people should be managed, is rooted in thinking that is over 100 years old.

The Industrial Era

Give me your cheap goods, your commoditized products yearning to be made available to the undifferentiated mass of consumers, and spacious, stagnant markets

Things looked very differently back in the turn of the last century.

I care, I really, really care…

Before the rise of the Industrial Organization, people played the primary role in creating goods for others. This was great in the sense that people could order to their unique needs, and service was familiar, even intimate. But production was expensive, very few goods could be produced this way. For certain, it meant that most goods were unavailable to a population that was for the most part both poor and un-educated

Commoditized and Cost Efficient

The challenge of this era long past was to get a limited number of goods to as much of the population as possible. And we needed to get those goods out for the lowest possible price.

Core goods that would significantly raise the standards of living for us all. Cars, radios, ovens, refrigeration, vacuum cleaners, you name it.

And we wanted to employ the population so they could afford to buy all of these items in the first place. Not to mention everyone gets a job.

The challenge of the industrial age was to generate wealth for the masses.

The Rise Of The Industrial Machine

Thanks in part to a paper published by Frederick W. Taylor, titled the *Principles of Scientific Management*, an Industrial Management Model gained wide prominence, and eventually became the model for the design and management of almost all of the organizations we know of, till this day.

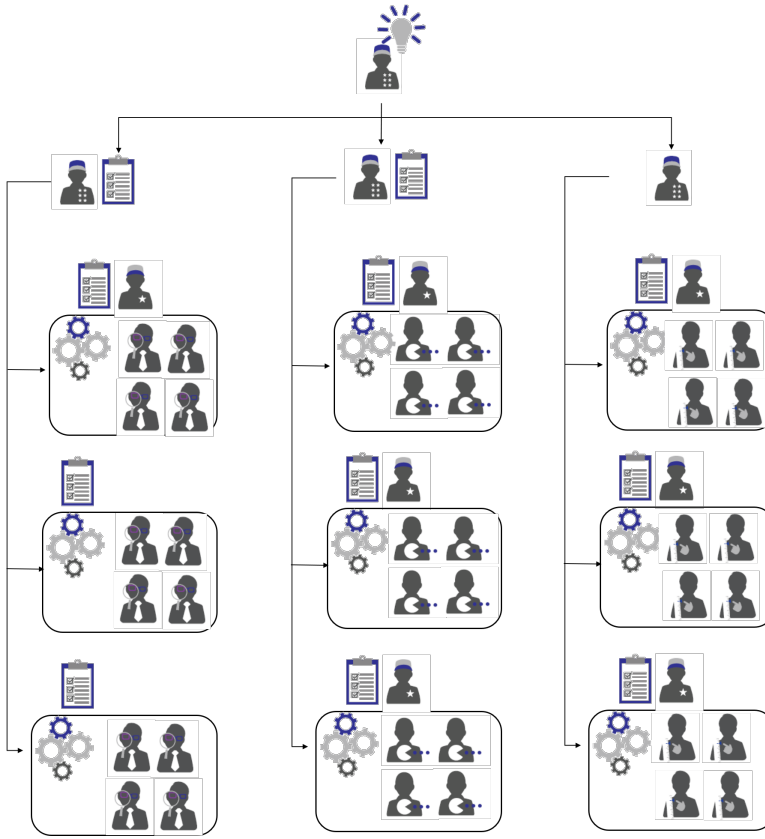
The industrial management model is geared towards attaining high efficiency. We want to lower costs and increase the markets access to our products.

I could do this all day…

This lower cost was achieved through standardization. Repetitive tasks were documented into detailed instructions that were carried out to the letter by workers. Managers were responsible defining these instructions, as ensuring there orderly execution. The educated few controlled the uneducated many.

In instructions we trust

Organizations scaled by **dividing** workers according to highly specialized tasks. People with separate skillsets were placed in separate parts of the organization.



The principles of standardization, division, and control were all fit for purpose for organizations that wanted to create wealth on a wide spread scale. More wealth was generated in a 100 years than the previous 1000.

And By wealth we mean wide spread wealth, we aren't talking about just a few fat cats here, (Yes they did well, fantastically well too) we are talking about the larger population.

Judged by these outcomes the industrial era organization was a fantastic success

But who really believes we are still in the Industrial Age?

It would not be a stretch to say that industrial era management and the industrial era organization has turned toxic in our current age era.

In many ways industrial era thinking is eroding our people's ability to effectively deliver value. Organizations exist to help people collectively create more value than they could by themselves. Yet the industrial principles of division and control and standardization are having the opposite effect. How many of us feel that our organizations make us collectively dumber, slower, and even eviler. How many of us bemoan the loss of humanity from working in such organizations?

Ch-Changes, Turn and Face The Strange

Times They Are A Changing, They Are Are a Rapidly A Changing, They Are a Accelerating a Changing, They Are a Exponentially A Changing, They Are...

Look color! it must be a new era...

Ok, so I need to pick my musical metaphors, and I am likely aging myself a bit, but the I hope the point is clear. The world of today looks very different from the world at the turn of the last century.

It must be quite a surprise to many of you. I joke, I joke.

I am sure most of you are keenly aware how different the era of today looks from the era that birthed the Industrial Organization.

Change is name of the day. Constant change, rapid change, exponential change, changing change. The only constant seems to be that the rate of change keeps accelerating.

Is it me or did this market get crowded all of a sudden?

It turns out that the industrial organization is a victim of it's own success. When our customers become wealthier, they also become much more discerning about what they want to buy. It's a good thing that wealth also brings a lot more competitors, all who are vying for the customers' dollar, and can provide all sorts of different choices to our customers.

that race just became an exceedingly fast race…

When customers have more to spend, and more suppliers are competing for that spending then product lifecycle get shorter, and shorter. Let's not forget the role that technology is also playing in disrupting familiar business models at an ever faster rate. Obsolescence can come in months, weeks, and even days.

Using my voice when I want to talk, is like so six months ago…

Oh and your customers now want their stuff personalized to their needs, wants, and tastes. So that design-it -once-and-repeat-for-everyone idea is now pretty much out the window. We are long past the point where markets are vast with spacious amounts of room to easily grow simply by doing more of the same.

Instead markets are highly segmented. Goods and services are personalized to variety of user personas. We are in the age of mass customization.

Finally the world itself seems to be rising up against us. Environmental issues, disasters, pandemics only add to our sense of a volatile world.

All of this adds up to an era typified by volatility, by uncertainty, by complexity, and ambiguity, (VUCA)

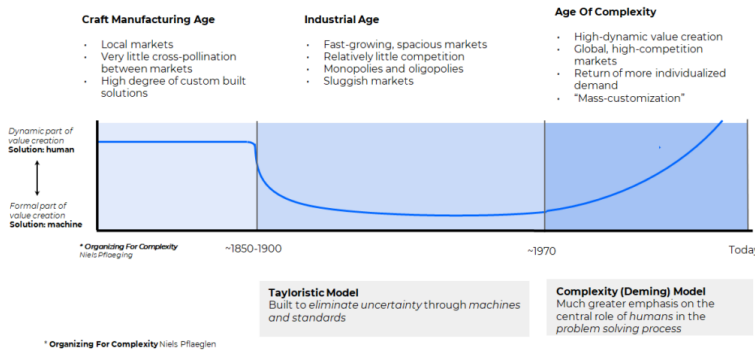
The challenge of today's era (this far) is to use market learning provide differentiated solutions that create customer delight and loyalty to an exact and exacting audience

From People to Machine, back To People Again

We just be cogs looking for a machine y'all!

As the industrial organization gained prominence, the machine become the engine that delivered value. Humans, were merely cogs in this machine. Automatons, parts that could be replaced with cheaper humans, or automated away with technology (the real machines).

While this was great for that initial burst of wealth, it wasn't great for our humanity. Thinking about people as machines is also a lousy way to take advantage of peoples innate ability to solve novel problems. Problem solving that we increasingly need for organizations to deliver customer value.



Humans are in charge again (for now)

And this is nothing new. Since at least the 70s people have been rising back to ascendancy. Organizations increasingly rely on knowledge workers to navigate between shifting market demand and constantly evolving technology.

Even old school manufacturing style assembly line work is requiring expertise in operating advanced robots, and programming these complex machines so that they perform in a way that is fit for purpose. Thanks to software, the vast majority of work is now knowledge work.

So we have gone full circle. From human, to machine, back to human again.

Time For A Radically Different Approach

Doing it this way always worked before, right?

If you need to move a lot of people across the ocean, use a cruise liner. It's efficient and cost effective.

But don't use a cruise liner to navigate inland, especially into uncharted waters.

You will hit an obstacle.

You will get stuck.

If you are lucky, you may be able to change course, eventually.

But you may capsize, instead.

Either way you will harm the boat, and more importantly, cause harm to the people in the boat.

Why am I so confused...

Likewise trying to navigate an environment of complexity and uncertainty with an industrial organization is a recipe for failure.

You will respond to slowly.

You will lose market share.

If you are lucky, you may navigate through the chaos, with huge effort.

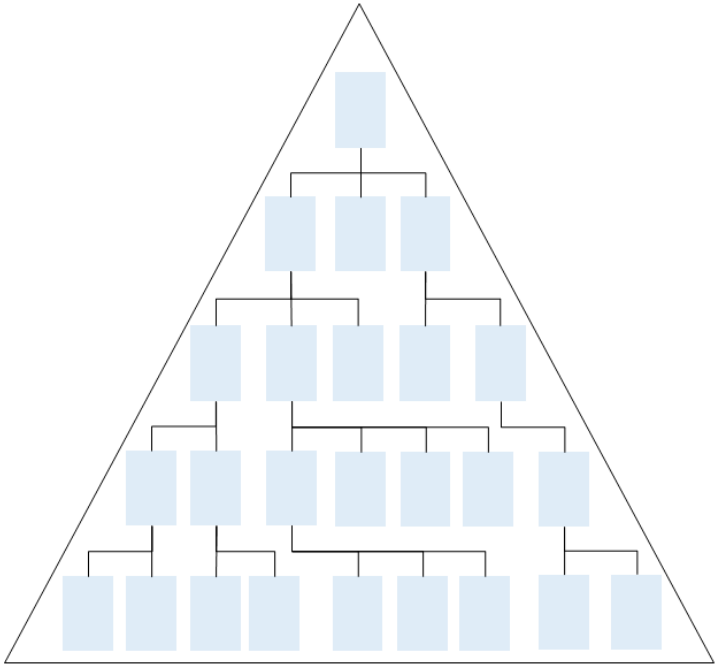
But you may go out of business instead.

Again, in either case, you will cause unnecessary strife to your workers, your leaders, and your customers in the attempt.

The Modern Organization

Our Modern Organization Does Not Remotely Resemble An Industrial Organization

The Industrial era organization, when looked at from a distance looks like a pyramid, with direction flowing from the top and information flowing from the bottom. When referring to this model, the emphasis is on command and control. Directives come from the top of the organization and are executed by lower levels in the hierarchy. Information resulting from activity within the lower levels of the organization and fed back to the top levels of the organization, the top of the organization then issues its next set of directives.



we are of the triangle

There are some fundamental issues with the industrial era organization in the face of market uncertainty. As uncertainty increases…

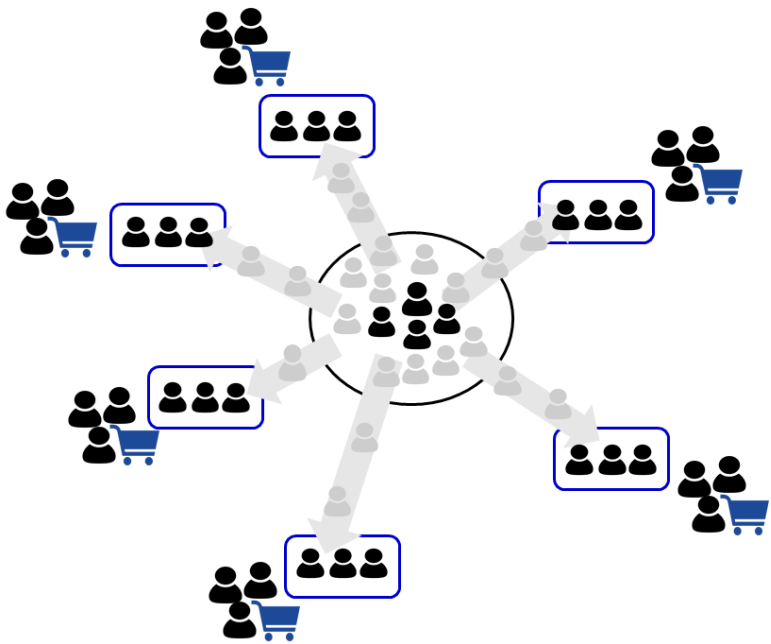
1. The top of the hierarchy doesn't have the context required to make the right decisions. The **wrong decisions** get made.
2. It takes to long for the top of the hierarchy to process information from lower levels of the organization in order to make decisions in a timely fashion. It takes **to long to react** to market change.
3. Accountability is taken away from the lower levels of the organization. The organization is **robbed of a wealth of intelligence and creativity**.
4. Silos form under distinct leaders at the top of the hierarchy. Internal, **departmental objectives take more focus** than mar-

ket outcomes do.

Given our current age does not resemble the Industrial age, you would think it should be obvious that a drastically different kind of organization is needed.

You would think. But most organizations are still run like industrial era machines. The majority of organizational leaders and their teams are operating in a very uncomfortable world where they are constantly fighting their management systems and org structures to deliver value. The mindset, skills, and the will to move to a new way of organizing seems to be in short supply.

But this is changing, we now have examples of real organizations operating under a new type of organization. With the people once again in ascendancy in the value creation process.



Organizing for the modern world is based on the idea that we decentralize authority into teams responsible doing the work. We group these people into cross-functional teams that have all the skills and permission they need to operate with autonomy required to own market outcomes. Instead of relying on one size fits all standards, teams continuously iterate and improved based on frequent market feedback. The operating metaphor is no longer the cooperate hierarchy, rather we think of people and teams as being part of a dynamic value network that flexes based on changes in the market.

For this to work we need a strong belief that most people want to work, and under the right conditions they enjoy it. We also need a strong belief that when teams with diverse skills are exposed to market feedback, that they will make better decisions than their bosses, who posses limited context in the face of growing complexity

Moving towards a more people centric organization is not done lightly.

It requires work. Work from your positional leaders, and work from every person across your entire organization. New practices and processes can illuminate parts of the path, but this is really starts with a shift in mindset, and that takes time. The good news is we have plenty of examples out there to draw inspiration from. Your path will be unique to you, but that doesn't mean you can't learn from others.

The other good news is that this new way of thinking and working already exists in your organization, if it didn't you wouldn't already be providing value in a complex world.

Digital technology background with social networking and interac - Multapplied Networks

The problem is that your value network is often underground, or add odds with the official org structure and management system. You value network is working in-spite of your management structure, not because of it. It is running against the "official" part of your organization.

Just like many pundits believe that communism in Russia lasted so long because of a vibrant black market that kept goods flowing across market participants. In the same vein, your organization is functioning because you already have a value network of self organizing knowledge workers, you just need to uncover it, make it official, and then remove the structural obstacles.

In my book I'll share ways to design, share, and test our modern value network based organization.

In Summary

- **Industrial Organizations** grew in response to the need to generate wealth for a largely **poor**, and **uneducated population** in **markets** that were **spacious**, **sluggish**, had **little**

competition, and had **huge potential for growth**.

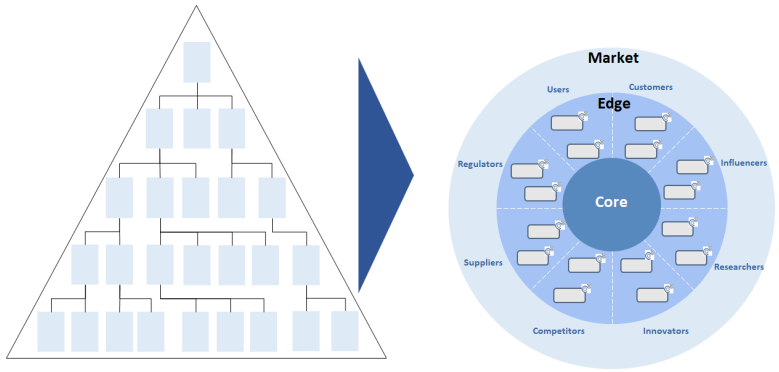
- The strategy of industrial management was **cost containment**, through principles of **division**, **standardization**, and **control** to achieve these outcomes
- A victim of it's unprecedented success, **today's organizations** are still largely **based on industrial era thinking**, thinking that is **turning toxic** in our current era of **accelerated change**
- **discerning** customers, **segmented** markets, **crowded** competition, **shrinking** product lifecycles, **rapid** technological innovation mean we once again **need to rely on people** and their **problem solving capabilities** more than we need to rely machines and their ability to do mindless, repetitive work
- The modern organization is structured around **decentralizing** decision making to teams that have the **autonomy** to own outcomes through **frequent market feedback**
- Moving towards a more people centric organization requires a **new mindset and practices** that allow us to **promote trust**, give up **control**, grant **ownership**, and overcome our **fear of failure**.
- Your value network of people **already exists** where you work, you just need to make if official, and make it core your **organizational operating system**

The Team is The Core Building Block

A Mental Model For Your Organization

Before we go further, let's pause and take a second to imagine a new mental model for organizations, one where we step away from picturing our organization as a triangle. as a hierarchy of power and control. While this model will not completely go away any time soon, we can decrease it's significant by picturing a model based on flow of value.

Niels Pflaeging in his excellent book *Organizing For Complexity*, provides one such mental model, one that we will refer to and build upon throughout this book. Using this model we go from imagining the organization as a triangle / hierarchy, where direction flows from the top to bottom, to a circular value network where feedback flows from the outside in to the center. Functional departments are de-emphasized, cross functional teams gain prominence. Teams act as cells in a value network that self form, organize, and connect to deliver value.



from *top to bottom* towards *outside in*

We will lean heavily on Niels' idea that *market pull* is what connects our teams across an organization. The activity of stakeholders externa to the organization initiates market pull. One or more market facing teams, placed at the "edge" of the organization respond to this pull, and as a result may request assistance from one or more support teams placed in a "core" zone, creating market pull inside of the organization.

It is through this new model of an organization that we can better respond to the needs of rapid feedback, co-creation, and meeting unique customer needs.

The Power Of Decentralization

We want to grow structures to make room for autonomy. The autonomy required for smart, capable and passionate people to self organize around value creation.

And that is the key to scaling out modern world. Stated simply, command and control doesn't scale in the face of complexity. Yes, it is more human, more moral to decentralize and provide people the space they need to make decisions, but it is also more practical.

Granting autonomy means we give up the illusion of centralized control. We decentralize decision making to people closest to the real work. We dismantle mind numbing bureaucracy and flatten the command and control hierarchy. Autonomy requires bravery, sacrifice, and a belief that people are fundamentally good. Without a higher calling, this belief can be hard to come by.

A lot of readers will be skeptical that this works in practice, and rightly so. "So, we just give up power and everything will work out? Really? Are you kidding me?"

Not quite.

the chaos self-(dis)organization

The Goal of Self-Organization

The goal of self organization is effective organization.

Giving up power does not mean we give all of the power to people to make any decision they want devoid of principle, or process, or guidance. We expect our staff, and our positional leaders to make decisions according to structures that we put in place. Structures

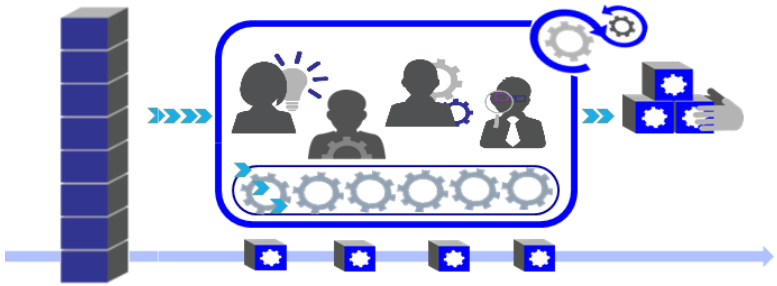
that are designed to maximize feedback, foster collaboration, and minimize bureaucracy. Structures that paradoxically constrain decisions making in a way that create an environment of increasing self - organization.

The agile team comes nicely equipped with several such *enabling constriants.*

Remember, the goal here isn't chaos, or anarchy. The goal of self-organization is *organization.* We choose to use self organization because it allows us *to be better organized* for the creation of value. Never let someone defend a lack of sensible structure with the excuse of self - organization

Why Agile Teams Work

The *why* and *how* behind agile teams is a well tread subject to many, but let's do a quick recap and touch on some of the key concepts.



we all live in a cross functional team, a...

Cross Functional

Agile teams by their very definition, are expected to have *cross functional capability.* Agile teams are formed around people in possession of, or can very quickly get access to, everyone and

everything they need to achieve on the outcomes they are trying to achieve.

Agile teams contain diverse skill sets

Independence to other teams is critical if we want teams to be able to make decisions while minimizing the impacts to other teams. Independent teams can increase their cross-functionality to own the planning, delivery, and operational activities required to truly realize their outcomes.

Market Outcomes and Feedback

Agile teams are given the *ownership* and *autonomy* to achieve specific market outcomes. Some teams are given this market direction, and some teams create their own, regardless, ownership of market outcomes is perhaps the *most essential attribute* of an agile team.

At a minimum, agile teams connect to the market by include people who represents the customers of the work. In tech, this means someone from "the business" is part of the team.

Agile teams have a strong connection to their markets

Better (much better) is when teams have *frequent interactions with real users*! We want a lot from our active customer(s). Prioritization and validation of value, yes. But also co-creation and problem /

solution exploration.

Agile team are guided by *market feedback*. Market feedback is achieved by delivering *smaller increments of value* to the market, and only work on *small number of those increments* of value at a time. Delivering a small number of small things means we dramatically increase the frequency we deliver value. We promote fast feedback and learning, we make problems obvious, and exposes opportunities for the team to improve.

Social Boundary

In the traditional working world, team membership is often a part time commitment. People are part of many teams. The team is never really together at one time, and the team, well never actually does any *teaming*. In contrast an Agile team is made up of *full time members*.

Agile team members have a strong social bond

This act of gelling, takes time. People need to learn how to trust each other and work together. So we want membership to be *stable*. If the roster is constantly changing, it becomes really hard to establish norms, and understand how to work with each other.

A stable roster of full time team members help set the team up as an *effective social boundary*, where both *formal value creation*, and *informal influence can take* place.

Flexibility

Team act as an effective container where team members are free to take up any role they want, as long as it serves the goal of the team. The team is expected to navigate the balance between what people want to do and what they need to do. This is easier than it sounds when we apply the concept of to how team members grow their skills over time, from highly specialized to *T- shaped people*.

We want people to be *flexible and be able to play many roles* on the team.

Agile teams can shift to accommodate change

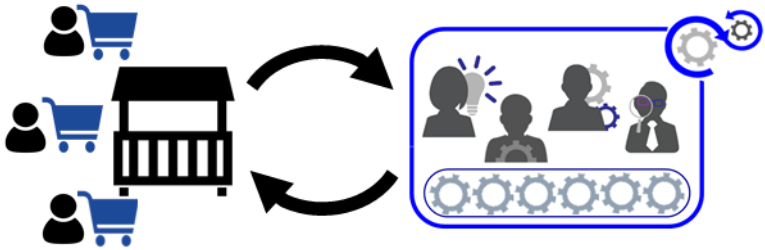
Flexibility of role, of process, of market direction is facilitated by *radical transparency* on the team. Agile teams achieve this radical transparency though a number of means such as visual backlogs, visual workflow management, and frequent team sessions held weekly and daily. This radical transparency allows the team to be both flexible and aligned to it's outcomes.

Agile teams also use these methods to inform a constant and virtuous cycle of *continuous improvement.* The team uses small experiments to adapt to constant change.

Agile Teams Deliver Value To The Market

The first constraint we can place on the people in our organization is that the majority of our people spend the majority of their time working in an agile team. Agile teams have direct and intimate contact with the outside. They are in constant connection with

the people outside of our organization that are impacted by the organization's purpose.

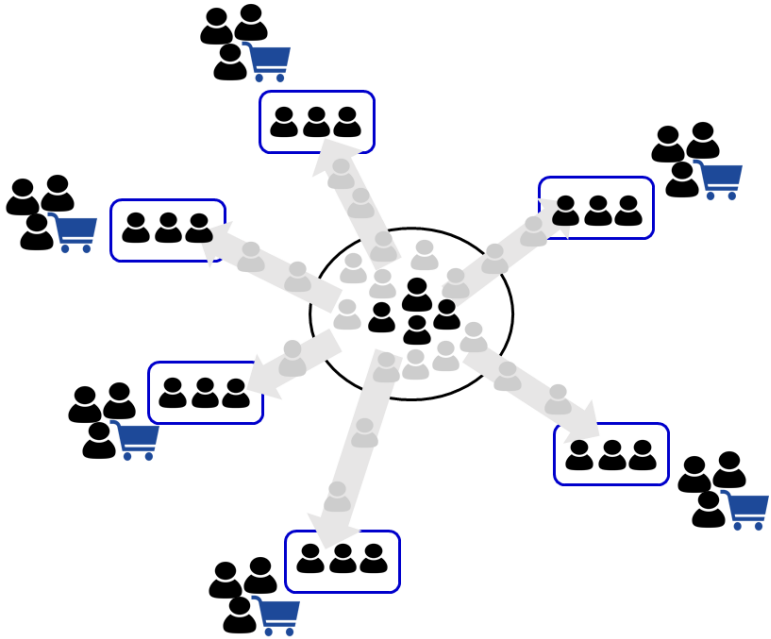


teams and markets is the way

The other core attributes of Agile teams that are key to enabling effective self organization, include:

- Ownership Of Outcomes
- Cross Functional Capability
- Committed Membership
- Stable Roster
- Customer Participation & Market Feedback
- Focusing on Small Number of Small Thing
- Independence from other Teams
- Radical Transparency through Visual Management and Feedback Loops
- Continuous Improvement / Change

This orientation has a profound impact on our organizational structure. Purpose driven organizations place an emphasis on creating structure around teams that can achieve market outcomes that fulfill an organization's purpose. Teams are connected directly to the market they serve, and are empowered to interact with the market actors who are necessary to achieving the organization's purpose.



get out of head office and get into teams

The Benefits of Agile Teams Don't Require Any Single Agile Methodology

It is important to note that these attributes can be applied to teams agnostic of any one agile framework, method, or methodology. There are many cases of organizations whose teams exhibit most or all of these attributes without regard to a single official agile practice, or even really being aware of agile at all.

That being said there is a wealth of agile material out there that provides some very compelling approaches, teams could do a lot worse than reading up and experimenting with them!

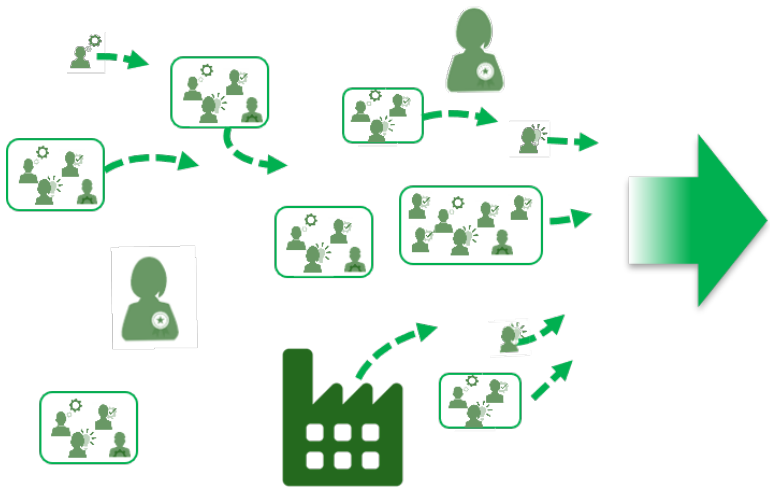
Again the point of agile teams is to provide an organizing construct that helps us *organize*. Agile teams do not mean anarchy, or a

free wheeling laissez-fair approach to value creation. Agile teams allow people to better organize themselves into cohesive groups that require minimal management intervention.

It Is The Combination Of Attributes That Enable Agile Teams To Be Effective

The more attributes you adopt, the better, it is the *combination* of radical work transparency, constant customer feedback, continual experimentation, and diversity of opinion that create a complimentary set of corrective mechanisms that keep individual teams iterating towards achieving the most meaningful outcome they can, in the best possible way, while minimizing the need for command and control.

This lack of command and control further fosters a sense of ownership and social density, which in turn decreases the need for any traditional management, a virtuous cycle.



autonomy and alignment is the goal

Unfortunately this idea depiction of agile teams seems to only rarely come to pass for organizations who adopt agile concepts. In .our next chapter we will start discussing the limitations, as well as how we can complement agile teams so they truly work well in the context of larger organizations,

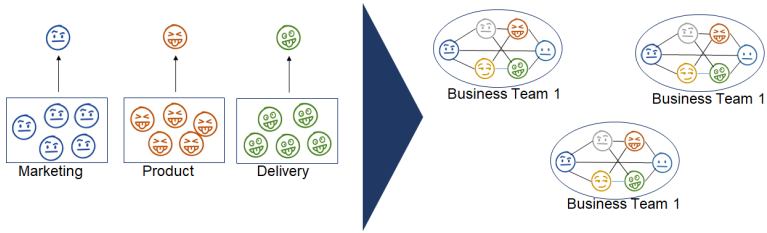
So stated fully;

Organizing Constraint 1: Deliver market value through stable, cross - functional, self- organizing teams that take advantage of radical transparency, full time membership, focus, and feedback. In other words Agile teams.

Functional Departments Grow Capability

When we use functional departments to deliver value we end up with single contributors who coordinate their work with other single contributors from other functional departments. In effect this means that the collaboration required to deliver value happens when people collaborate across their departments, The organizing structure works against you. Again, value creation happens outside of a functional hierarchy, and often it happens in spite of the functional hierarchy.

The functional department approach results in the need for a lot of coordination, this makes us reliant on bureaucracy and management to get things done.



from authority to capability

A much better model is to rely on functional departments and functional management to define and grow functional capabilities, to hire people that can possess a functional expertise, to provide training and learning, and to grow their careers to excel in a particular set of functional capabilities.

Delivery of value on the other hand is taken away from functional leads, and becomes the responsibility of cross-functional teams.

These teams can have leaders, but these leaders should not be confused with functional management, it is the leadership responsible for empowering and enabling teams to deliver value. This change in mindset from authority to mastery is not an easy one for many in positional leadership, but it is necessary if we want to organize for agility.

So stated fully;

Organizing Constraint 2: Use Functional Departments to Grow Capability In a Functional Skillset, avoid managing value creation through functional managers

In Summary

- The new mental model for the modern organization is a **value network where feedback flows inward** from the market, inwards to the teams in the organization.

- Decentralization and autonomy is not about anarchy, it is about creating a **better system of organization** in the face of complexity, **known as self-organization**
- Agile teams use **diversity of both skills and perspective** to increase their collective ability to self-organize.
- Agile teams try to be as **independent as they can** be from other teams, so that they can self organize without the need for manager coordination
- Agile teams are focus on **strong market outcomes**, and get as close as they can to their markets to achieve effective **market feedback**,often **co-creating value** with customers
- Market feedback is increased through **frequent delivery of smaller increments** of value
- An Agile team is made up of a **stable roster of full timers**, so that the team has a chance to develop the **social bond** required to be effective
- **Flexibility of role**, **workflow**, **value to deliver**, etc is paramount in an agile team. This flexibility is achieved through practices that enable **radical transparency** and **continuous improvement**.
- **Teams deliver value**, while **functional departments grow expertise**. Leading value creation and leading the development of expertise are two very different leadership activities and the two should not be confused with each other.

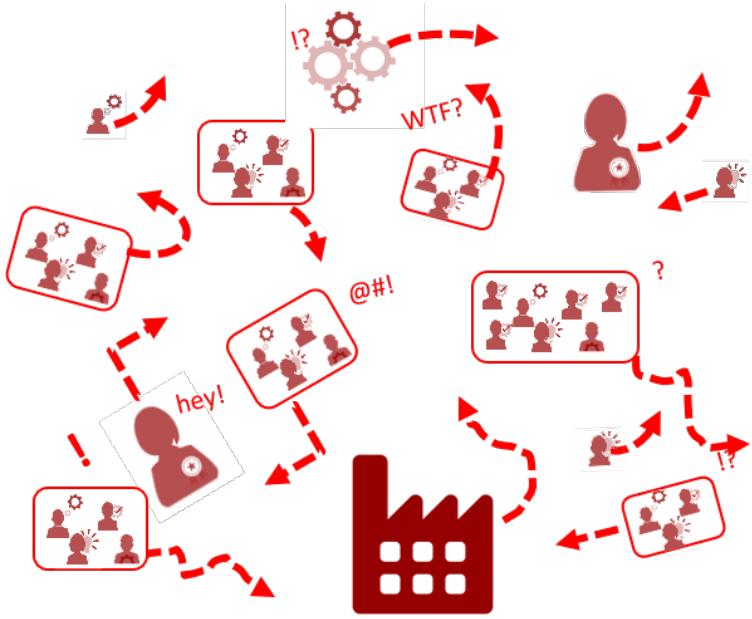
Other Building Blocks To Scale Agile Teams

The Need To Build Upon Agile Teams

There be some dragons in them hills of using agile teams. I know some of you will be skeptical that placing people in agile teams will result in an autonomous, self - organizing institution at enterprise scale.

And rightly so.

The reality is that the the number of organizations who are using agile that are operating according to my previous highly virtuous description of agile teams are vastly outnumbered by the number of organizations that are not. Organizations moving to agile teams often encounter some very real, almost intractable problems.



agile team chaos, yes it is a thing...

There are many reasons. Leadership mindset, organizational culture, industrial era values are huge contributors. But we can also look at the tangible ways we are organized.

There is a lack of understanding of how to extend the concept of agile teams to grow organizing structure at scale, and this can causes some huge and largely avoidable pitfalls....

- I put everyone I needed onto a team and the team is huge! 30 people! 80 people! 200 people!
- The work requires really specialized individuals, and its never the same ones! Someone is always idle on my team, and its never the same person at any given time!
- My software architecture is so fragmented that every team structure I can think of results in a spaghetti of dependencies! No way I can use teams to deliver value!

- I have too many customer, and they are all competing with each other! How I can I pick a dedicated customer to act as my customer representative for my team!
- My teams are all going in a different direction on something that would benefit from some consistency! I can't get this consistency if every team is self-organizing on every aspect of what they do!
- My teams is not allowed to deliver unless we get approval with legal/compliance/architecture/agile coaches/quality management/release management/customer experience/security/privacy/audit/finance/data/support/operations!

As a result a lot of people are disillusioned with the idea of agile teams. That a team model can solve the challenge of increased complexity. What follows next is retrenchment, disillusionment, and some really bad hacks that bring back old ways of working and thinking.

There is more to Organizing Around Agility Than Simply Placing All Of Your People into Agile Teams

But remember when I said Agile Teams are awesome? They are! But to make them awesome at scale we need to spend some time to understand some of the reasons why agile teams work. (Hint: it has nothing to do with frameworks or methodologies)

We need to understand what aspects of teams allow you scale them, and what parts of our definition of agile teams actually get in the way.

A common theme in this book, will be taking a concept that is familiar to most agile practitioners, and exploring why it works. We want to keep an eye on extending the underlying principle so that we can apply it at larger scale.

In order to do this, I am going to continue to challenge some sacred cows that are part of what I call the established agile dogma. Yes the agile-verse is now big enough that large tracts of it are more dogmatic than thoughtful, more conservative than leading edge. Such is progress I guess.

There Are No Independent Teams

Our first sacred cow, the idea of autonomous team. OK, let me be fair, let me state *there is no such thing as (completely) independent teams.*

Take a second to think about the value your team is trying to create. Perhaps we are talking a software application delivery team. The team has accountability for everything from prioritization, through delivery, all the way to roll-out and operations. Sounds like a truly independent team right?

Now think about all the things that team needs to deliver on any one increment of value. Is there another team in our organization that does market or competitive research? Do we rely on this research to make decisions? Should we? Is there a common business model the team is leveraging? Do other teams have input into it? Do we have input into it? Some other questions we could ask:

- Is there an overarching strategy being used to guide our work? Do we help build it or validate / invalidate it?
- Are there common cultural artifacts that we share?
- Do teams share a common physical work-space?
- Do multiple teams want to or need to share a common software platform, even if that sharing is in a completely voluntary or autonomous way?
- Do other teams help set that up?
- Does the team share any common data with other teams?

- Do teams share common customers?

Tired of these questions? Hopefully I have made my point. In an organization of any scale the answer to at least some of these answers will be yes. Even in organizations of high agility, independence, is a relative concept. It is safe to say we want teams to be as independent of each other as they can be.

I often think of independence as graduated concept, from contribute, to full delivery, to deliver and operate, to full ownership.

Even a team that fully owns market value creation will likely need help on what I'll call* indirect or enabling activities.* Think of a sports team. The team is responsible for winning, but a lot more activities go into winning than playing the game. Talent scouts, logistics, getting people into the sporting venue, career counseling, etc, but the team doesn't necessarily want to do all of these things. The team may want to focus on the game. Likewise, in a an organization of any scale, Teams may want to focus on direct value creation, and to receive support for help with everything else.

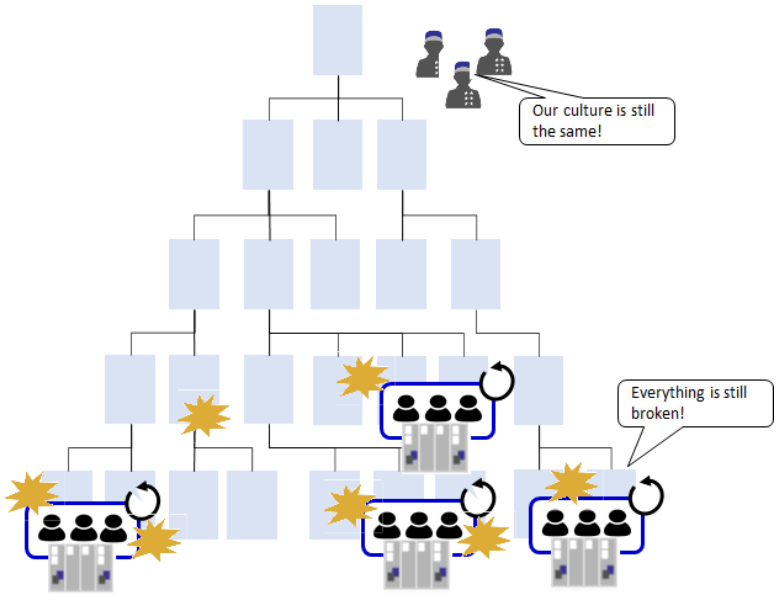
The trick to being successful with self-organizing, outward facing teams is to support them with a few other supporting structures that minimize our need for top down enforcement, hand offs, or approval gate. We want to set up supporting structures so that support is given to teams when they need it and how they need it.

But We Can't Scale Agile Teams With Traditional Org Structure

Many organizations fall into the trap of supporting agile teams using the same structure and mental model that has been in place for over a century. Hierarchy, control, coercion, and compliance. We continue to group people according to their specialized capability, and make them accountable to a functional manager. We

end up with agile team members being subservient to the multiple hierarchies in the traditional management structure.

Many agile transformations start on this wrong foot. We see a big emphasis on using agile to change the way people work, instead of a focus on making meaningful change to the way people are organized.



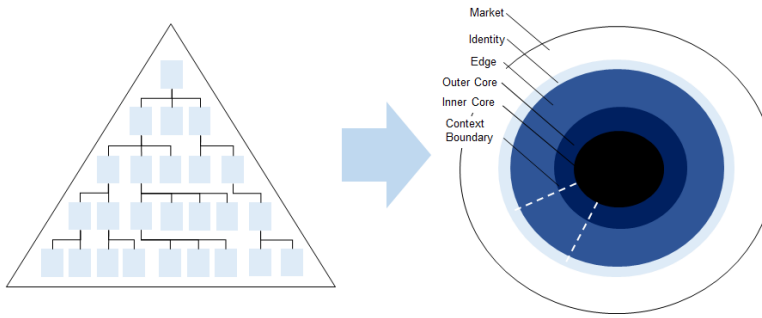
img

As I have stated previously Agile methods will offer only limited value to an organization that remains structured primarily around silos, that is inward facing, that are structured through command control.

Foundational Organizing Structures

When we think about organizing around purpose it is helpful to use a very different mental model. Instead of thinking about *layers of control and authority*, we think of defining our structure in terms of the *markets we serve and the outcomes we want to achieve*. Instead of using the hierarchical pyramid model of control, we will think of our organization as a set of concentric circles connected by *markets*.

Borrowed from Neils' book *Organizing for Complexity* we can represent an organization as people in outer layers that create demand that people in inner layers of the organization consume, creating a market oriented value network.



from hierarchy to value network

In this model people who are part of an outer zone which in turn are serviced by people that are part of inner zones. In effect each outer zone acts as a market to one or more inner zones. When using this new mental model it is useful to think in terms of the following organizational constructs.

The Teams everyone is in a team, each team may or may not look like a classically defined agile team; but the need for everyone to feel like they share an outcome with a smaller group of peers is paramount.

The Market are all the groups that exerts external pressure on your

organization.

The Identity establishes a common identity that directs self-organization towards value creation.

The Edge is made up of every team connected to the market and contains all roles that make decisions based on market interactions.

The Core is made up of teams that are intentionally deprived of market contact. These teams serve the Edge, providing services that the Edges chooses to use.

Context Boundaries: allow us to group teams in larger organizations based on a common context.

In larger organizations that number in the 1000s of employees, you may find it necessary to model a number of **Inner and Outer Cores**. An Inner Core serves the entire, or a larger portion of the enterprise. An Outer Core is closer the Edge, and directly serves a smaller number of associated Edge teams.

What's in a name

In Neils' book Organizing for Complexity, these zone have different names. Some alternative terms for each of these zones that I have used with clients include:

The Teams: Pods, Squads, Labs, Bands,

The Market: External Context, Environment, Situation

The Identity: The Sphere, Organizational Boundary, The Enterprise

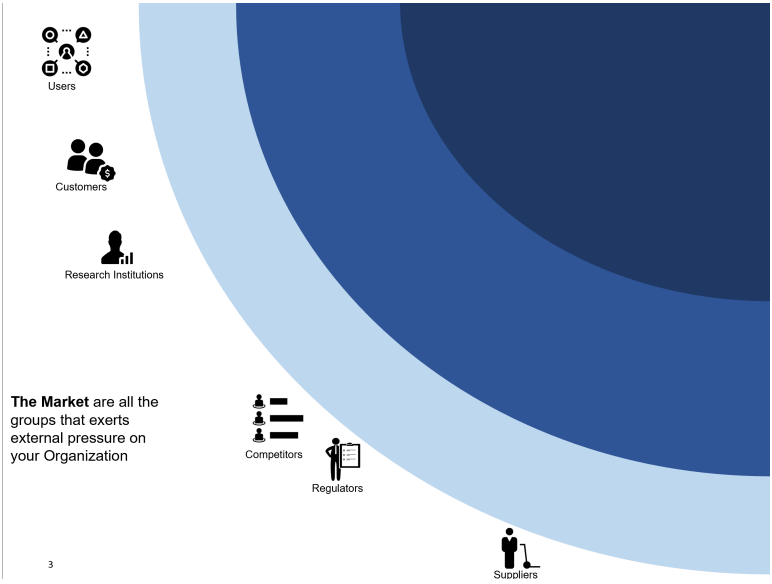
The Edge: Periphery, Market Facing Teams, Customer Teams, Value Centers, Feature Factory, Digital Factory,Garage

The Core: The Center, Support, Shared Services, The Glue, Infrastructure

Context Boundaries: Tribes, Guilds, Missions, Programs, Portfolios, Business Units, Subsidiaries, Divisions, Studios, Garages

Ok, lets go through each piece in a bit more details, and provide a few examples in an attempt to make it a little more real. Let's start from the outside and work are way inward.

The Market



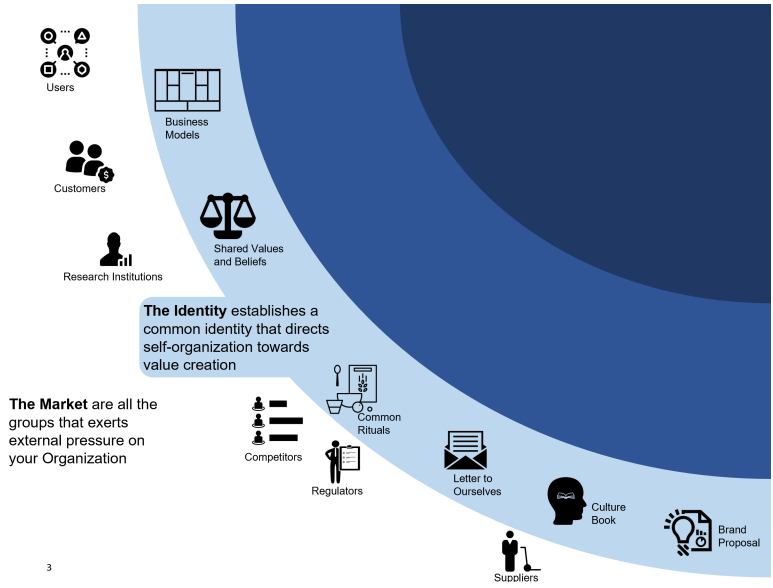
define your external actors and how they engage with your organization

The market are all the groups that exerts external pressure on your Organization. This list is not limited to customers and users, but **all stakeholders** that influence the way your organization provides services to the market. This list includes:

- **Users** of the products and services that you provide

- **Customers** who pay for those goods and services, either directly through some form of monetary compensation, or more indirectly by providing some other form of contribution such as their time, knowledge, or presence
- **Supporting Organizations** that help you engage with, or even better understand the Market you are in, such as Research Institutions, Educational Institutions, Industry Guilds, Technical Standards Committees, etc
- **Regulators** who place constraints and rules of conduct and validate compliance for organizations working in a specific market
- **Competitors** others outside the organization that serve the same or similar market

When starting to think about re-defining our organizational structure, it is very important that we start by identifying all of the players that our organization must interact with in order to deliver value to market. What roles do each of these market participants play? What is our primary means of engaging with them? there opportunities to exercise tighter feedback loops and better co-creation?



The identity guides self organization

The Identity establishes a common identity that directs self-organization towards value creation. **Unlike other "zones" in our org mental model, the Identity does not define where people are placed or who they engage with. In contrast, the Identity contains normative artifacts that describe who we are, and more importantly, *who we want to be,* as an organization.

I am not aware of any organization that has made a serious move to agility without paying attention to organizational identity.

Identity done poorly gives us propaganda. We get meaningless platitudes posted on walls, we get principles that leaders don't understand and don't follow. Identity done well give us some artifacts that inform the right culture. Artifacts that represent how leaders are behaving, or at least how they are striving to behave.

There is a strong emphasis on articulating how the organization can foster a culture where people are empowered to make decisions for themselves. Good identity artifacts give us something to point to when we fall back onto familiar old ways of thinking and behaving.

There are some great examples of artifacts we can use to help establish our organizational identity, almost all of these artifacts are informal, and focused on telling a story, and not on creating a false prescription of artificial precision.

Good identifying element include:

- **Business Models** formed using the practice of Business Model Generation[12]
- Taking the time to establish a common set of **Shared Values and Beliefs**[13]
- Full participation of **Common Rituals**, such as shared morning updates, standups, or periodic open spaces
- Publishing **A Letter To Ourselves**,
- Developing and sharing a **Culture Handbook**
- Developing and sharing an **Organizational Brand Proposal**
- A shared **Backlog of Customer Problems**

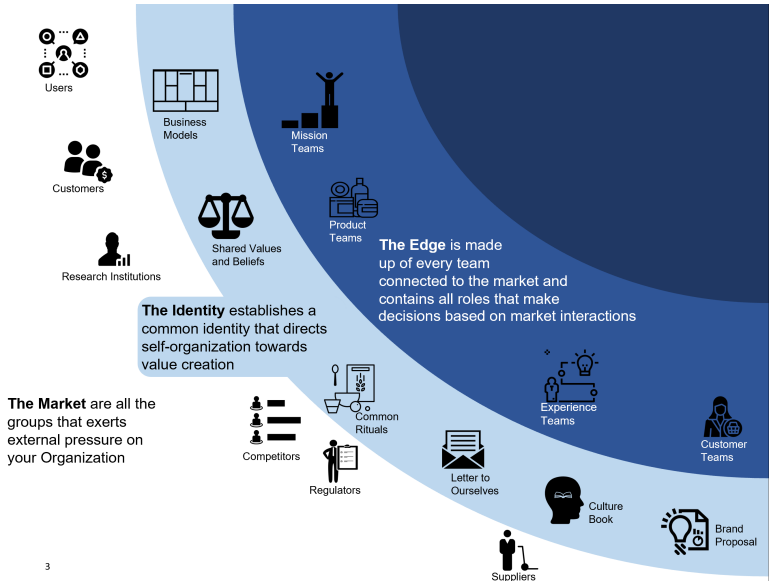
Poor Identifying elements include:

- Detailed Strategies / Blue prints or Road-maps
- Anything over a couple of pages
- Principles that are not immediately recognizable in the everyday actions of positional leaders
- Anything that cannot be challenged, updated for context, or revised based on learning

*Important: Identifying elements foster alignment and **inspire culture**, they don't lay out a recipe for people to follow.*

[12] https://www.strategyzer.com/books/business-model-generation

[13] https://www.notion.so/agilebydesign/Defining-Agile-through-Values-Beliefs-and-Behavior-26123c2f57fc47eba4940513baee00fe



the edge contains market facing teams

The Edge is made up of every team connected to the market. The teams in the Edge possess the skills and have the authority to perform all of the roles required to make decisions based on market interactions. This means *every* market interaction! If a senior executive is responsible for maintaining a relationship with a strategic partner, he does no do so from the center. If necessary he facilitates allocating the ownership of that partnership to an edge team. He may also join that edge team in order to perform his responsibilities as part of that team, even if he only joining that team on a part time basis. This is important! If we want our new organizational model to thrive in the face of constant change, we must move all market oriented activity to the Edge!

It is important to note that each Market Participant should connect to at least one Edge team. Mapping edge teams to market

participants is one of the first acts we should perform as part of organizational design.

There are many responsibilities that can be performed by Edge teams . Some of these include:

- Building new Product Features, deploying them to the market and validating them
- Customer On boarding, Maintenance and Support
- Research, Strategy, and Business Model Generation
- Marketing and Creative
- Vendor and Partner Management
- Engagement with Regulatory Bodies
- Participation in professional/standards/technical groups
- Organizational Investments
- Actuary / Credit / Other Financial Models that impact product and services
- Operational Activities that have a direct impact on market participants

This is a long list! It's important to think holistically about how we want to organize our people into teams within the Edge. It is extremely important to try to get this part of our organization right, and to continually refine how Edge teams are structured based on the latest feedback from the market.

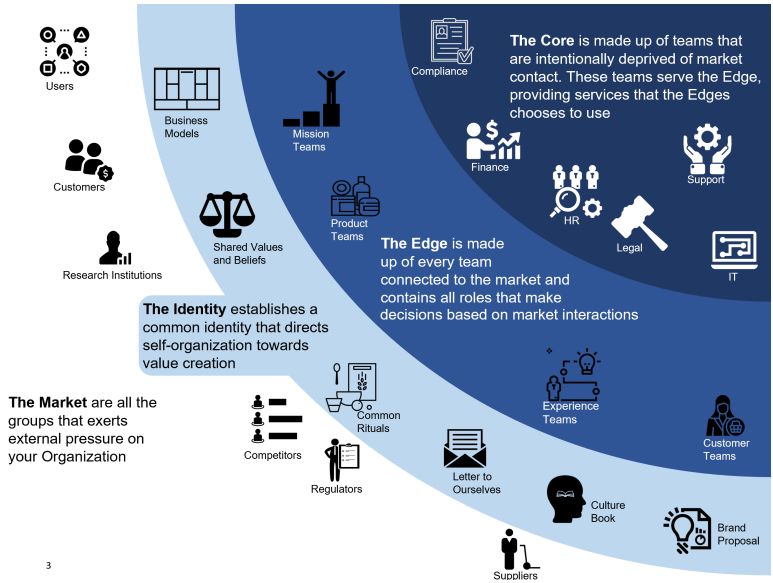
We also want to design our organizationso that team own as much as they can. This includes delivering new features, operations of daily services, marketing, customer engagement and support. Acting like micro-organizations in their own right.

Some approaches to organizing Edge teams include organizing teams based on:

- Organizational Missions or **Outcomes**

- **User Life-cycle** (e.g.: User Awareness, User Activation and Revenue, Chargeable Activity, Billing and Receipt, Support and Customer Care, etc)
- Specific **Customer / User Segments**
- **Products** offered to customers
- Common **Platforms / Systems**

There are pros and cons to organizing your teams on any of these attributes. It's fair to state that I have ordered this list by an agility rank. in other words when you organize by outcome you will find it easier for people to collaborate, self organize, and get market feedback then if you organize by system or platform. But, again context matters. There are many short term, and even strategic reasons to organize by customer or product or even platform instead.



the core serves the edge

The Core is made up of teams that are intentionally deprived of market contact. These teams serve the Edge, providing services that the Edges chooses to use. While we want to maximize each team's ability to self-organize around an outcomes, and to do it with the minimal amount of dependency possible, there will still be value in taking advantage of economies of scale for some forms of work.

Most people would agree that it would be wasteful for every team to define their own hiring process from scratch, or for each team to have their own means to pay employees. Technology teams may benefit from leveraging common technical platforms like deployment pipelines, or security and testing automation systems. It would be equally hard to argue that each team can independently define and maintain the overall organizational identity without some dedicated help to pull it together. Common capabilities like

User Experience, or Engineering, also can benefit from dedicated focus that spans multiple teams.

Virtually everything that you want to have some kind of oversight, consistency, economies of scale, or strategic perspective can go into the core. Some examples include:

- Regulation and Compliance
- Finance, Budgeting
- People Operations (Formerly known as HR)
- Legal
- Security and Privacy
- Information Technology
- Coaches
- Leadership and Management
- Any capability (Marketing, Engineering, UX, Product Ownership, etc)

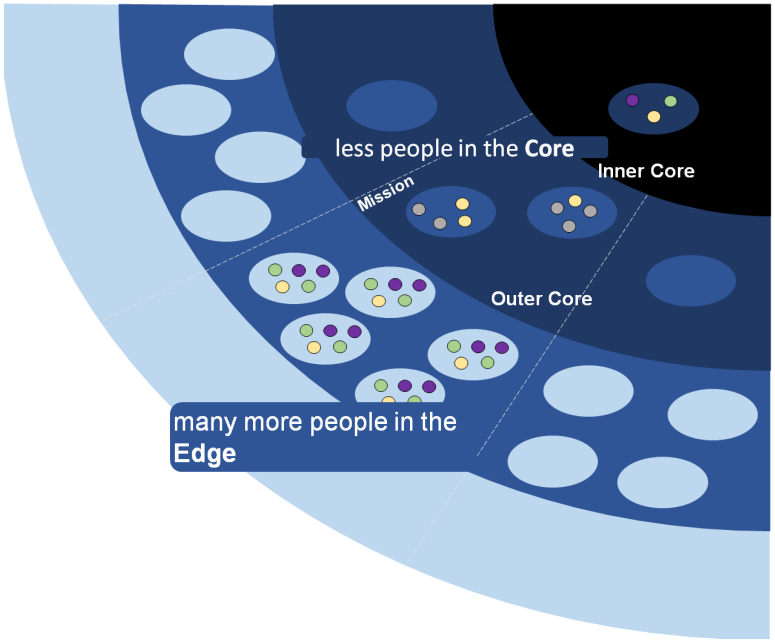
The Core is different from the Hierarchy

Most people would recognize these concepts as the traditional departments and organizational units you would see in a classic hierarchical organization. So what makes the Core different? The key design principles that makes the Core so powerful as an organizing element is that the product and services provided by the Core are strictly voluntary! The Core has to treat the Edge as it's market! This means if an Edge team can find a better service outside of our organization then it is free to do so. *Edge teams are not compelled to use anything from the Core.*

The Majority Works In The Edge

In general, as we move to a more agile way of working expect people working more people to be focused on market value. We

want to transition from people working primarily in a functional department or team devoid of market contact. Inorganizations with healthy autonomy and decentralization the amount of centralized support required goes down, way down. The amount of teams that work with the market goes up. The end result is more market value gets created in your organization.



the edge is where it's at

So stated fully;

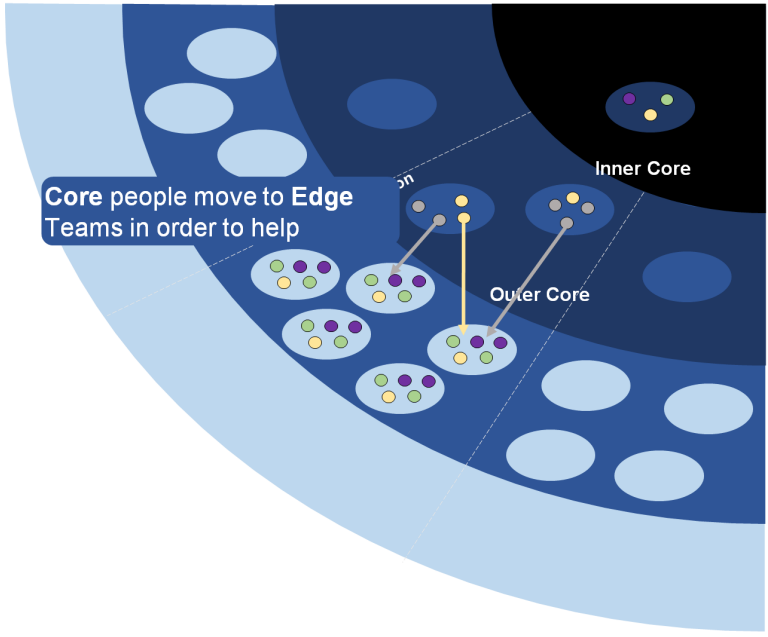
Organizing Constraint 3: The majority of people in the organization are in edge teams working on achieving market outcomes

The Core Travel to The Edge

Another way the core differs itself the traditional organization services model is how Edge teams engage with the Core teams. In the traditional enterprise model, the market facing people often send work to support people. The support people do their thing, and send the result back. IN the place of compliance and governance, market people submit their work for review and present it to support areas, who give their blessing to proceed.

As uncertainty goes up, this increasingly becomes a recipe for failure. Work returned by support teams often do not fit the needs of the market teams. Reviews are subverted to meet internal expectations of the support teams' processes and standards, not the actual risks to the market. Support people have a hard time getting the right market context they need to effectively support edge teams.

In our new model, we flip the direction of how people and work moves. Work goes to an edge team and stays there. When support is requested from a core team, someone from the core team physically (now virtually, thanks to covid) picks up and goes to the market facing team to provide the service. They becomes a part of that team for the duration of the request, we call this person a *travelling*. Alternatively support teams can provide platforms or knowledge to allow an market facing team to perform the service themselves, we call this *enablement*. Core team members may do both.



go edge young man

In only rare cases does allowing work to travel from an edge team to a core team and back represent a good choice to make. It may make sense for services offered by the support team that are highly commoditization, repeatable, and require very little collaboration between teams.

So stated fully;

Organizing Constraint 3: Edge teams avoid handing work to support teams in the core, people in core teams travel to edge teams to collaborate on the work.

Agility Does Not Come from Agile Teams

So lets take a breather, and tackle another agile cow.

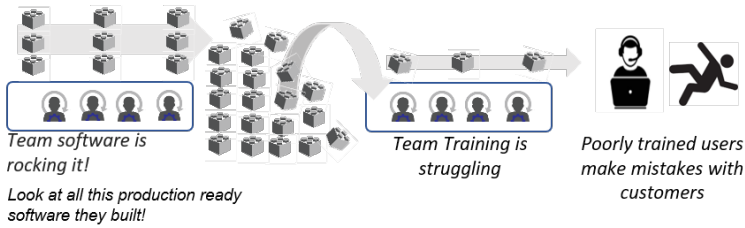
I am sure some of folks from the agile camp are going to come out with the pitch forks.

Ok, how can we say agility does not come from Agile Teams? Didn't I say previously that Agile Teams are a core building block for our new organization?

Hear me out. A lot of Agile thought-ware puts focus on the team. How is the team velocity? What are the team's practices? Can the team meet the teams' sprint commitments? How did the team retrospective go? Is the team blocked? What are team level impediments?

It's not that there is anything wrong with a strong team focus. We *want* a strong team focus. But a team view can be a myopic view, especially if you are working in a larger organization. We can optimize on one single team's health, but have no effect on actual value creation. In some cases team level optimization can even cause harm to organizational outcomes.

Let's take an example, Team A is a software delivery team that works with Team B who delivers end user training to customers. Team A is rocking it! They deliver and they deliver quickly! It's production ready every sprint! Team B is not keeping up. They can't get users trained fast enough to keep pace with all the changes to software Team A is making. Worse, Team B starts to compromise quality to keep up. This leads to poorly trained users. These users make mistakes using the software, which causes further problems with customers. As a result customers stop using Team A's software, market share is lost and profitability goes down. None of this is apparent from team practices, team metrics, or a single team focus.



agile teams can decrease agility

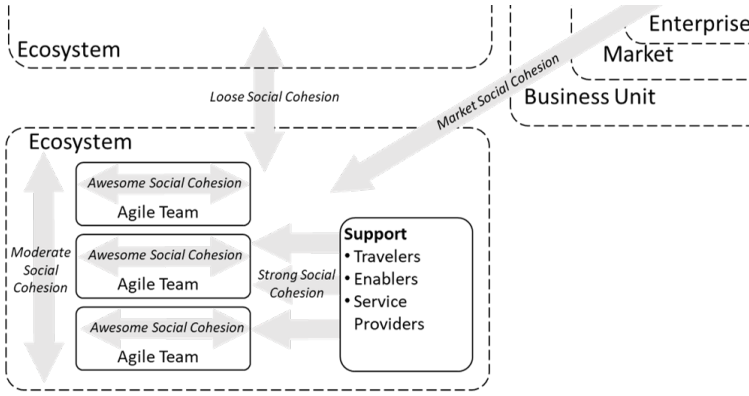
A lot of agile "scaling" frameworks attempt to address these and other scaling issue by adding a layer of processes to the mix. Some of the ideas coming from these frameworks are pretty decent, and some are frankly quite ridiculous. What the "more process is required to scale camp" does not always get is that Agile teams work because they create an environment of high *social cohesion.*

At scale we need social cohesion to be present in more than just within an Agile Team

Fundamentally, agile teams are awesome because they are a social construct that results in a high degree of social cohesion. Niels Pflaeging calls it *social density,* and discusses the dynamics of social density in his book Organizing For Complexity[14].

If we want to scale agility beyond a single team, we need to look for ways to scale social cohesion across the team and the organization. We can use formal and informal structure for this very purpose.

[14] https://www.nielspflaeging.com/books/



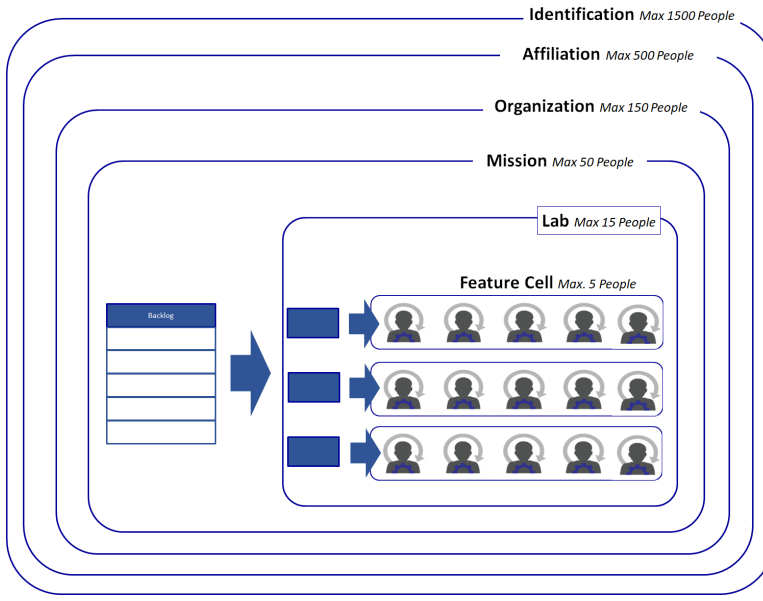
dude, it's about more than the team

So, let me rephrase my original statement.

Agility doesn't come from (just) agile teams, we need other social constructs as well.

Consider Using Dunbar's Number to Group People to Align to Social Cohesion (5, 15,35, 150)

Dunbar's number, or more correctly *Dunbar's numbers* suggests that there is a cognitive limit to the number of people one can maintain a stable social relationships with. It suggest that close nit groups should be not much bigger than 5 people. A person's active number, ie one where they maintains a fair amount of social contact is about 15. A larger network, one where people can still interact with at a reasonable frequency caps at about 50, and the total size of a network that can share a common mission or active identity should not go much past 150 people. When you get past these numbers, it's unrealistic to expect any form of meaningful collaboration. Shared acquaintance, and a common higher level identity is the best you can expect.



agility comes in all sizes

While it's unnecessary to pay strict attention to these exact numbers, they do provide a good guideline to set teams, as well as other context boundaries such as organizing missions/ portfolios for a set of highly connected teams, teams, as well as also larger organizational boundaries.

In my experience teams of 9 are good and past 15 they are unwieldy. It often makes sense to dynamically form individuals in a team into smaller groups of 3 to 5 people to process a small piece of work. With a team maxing out at 3 or 4 of these feature cells. Likewise a common mission with shared outcomes across teams becomes unwieldy past 50 people. I have personally seen larger business units have significant scaling and communicating problems when they get far past 150. Organizing support services so that a particular support team focus on a specific group of 50 to 150 people also has worked well in my experience. Again context matters, culture makes a huge difference, so use these numbers as

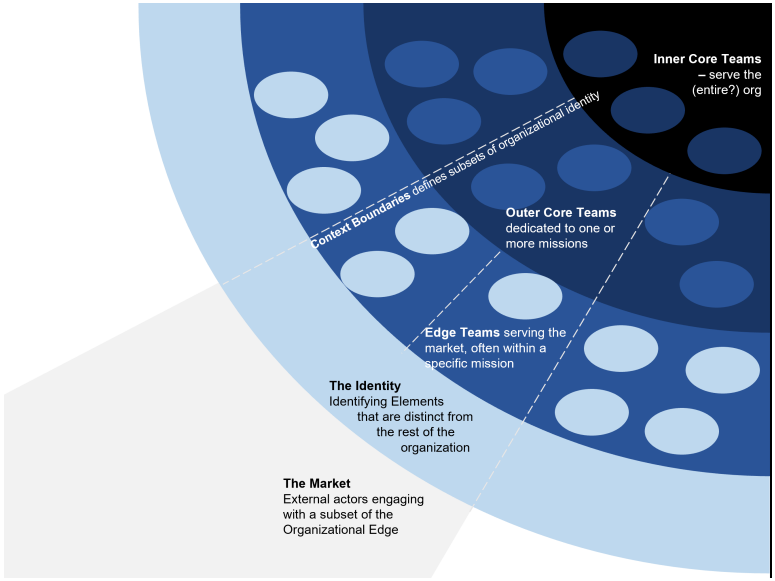
a general guideline only.

Organize Around Social Boundaries

In larger organizations I have found it helpful to define distinct subsets of the organization into **Context Boundaries.** We can use **Context Boundaries ** to group teams into units that share a subset of organizational identity, outcomes, and capability.

The concepts of *context boundaries* are based on my experience working with very large organizations. The idea was originally inspired by Eric Evan's concept of bounded contexts map, taken from his amazing book *Domain Driven Design*[15] . Bounded contexts are used to show explicit boundaries between teams and their code, boundaries that help establish communication pathways and decoupling for both the code and the people that write the code. This idea can be borrowed to help us demarcate the boundaries of a subset of a much larger organization, one that does not experience change at the same rate as the rest of the organization.

[15]https://dddcommunity.org/book/evans_2003/



An outer core is a set of core teams that are focused on supporting a subset of edge teams, all in the same context boundary. An Inner Core services edge teams from multiple context boundaries, perhaps the entire organization.

Edge teams along with their supporting, outer core teams are are often grouped together into a single context boundaries to help organizational designers work within the following constraints

1. **Lack Of Permission**: I have yet to be in a situation where I can facilitate change that spans an entire enterprise in one go. I typically help organizations that number tens of thousands of FTEs, specifically a subset of that organization, numbering from 100s of FTEs to perhaps just over a 1000. I need a way to compartmentalize the group that is moving forward, typically as a set of edge teams and outer core teams, from the parts that are not.
2. **Working with the As Is**: Really large organizations will be at different stages of their change, I need a way to represent

these more legacy parts of the organization, typically as inner core zones. I need to be able to move forward in an incremental way and represent incremental models.

3. **The Scale is just to big**: Even if I had the permission to change the entire organization, the scale is to big. Really, really big organizations are often a federation of smaller ones. It's often more feasible to work with the smaller pieces. Compartmentalizing thing into smaller bits often make the job easier.

As an example let's take a typical federated enterprise, each business unit has retained a good deal of autonomy to interact with their market. In this situation there are also some opportunities to leverage economies of scale and shared services.

1. Collections of edge teams, each serving their own market would be organized into their own context boundary, one context boundary per business unit. These edge would teams perform marketing, engineering, and operations required to fulfill their market outcomes.
2. We also have collections of Outer Core teams, each dedicated solely to edge teams in their business unit. These outer core teams would also be housed in the same context boundary as the Edge teams they are supporting. Examples include some localized DevOps capability, architecture, and Hiring / Recruiting
3. The company may also offer additional, complementary or even competing services on a global level, to both edge and outer core teams teams across the organization. These service would come from teams placed in the inner most core of the organization. Examples include a global People Operations group, Corporate Security, and Legal teams.

Every time we use an outer core, it is because we want to introduce some form of shared capability to only a subset of the edge teams

in organization. In effect we are partitioning the organization into smaller groups, establishing a smaller identity within the organization that spans multiple edge and core teams.

Inner and Outer Cores do not imply hand offs. We are not chaining market actors and service providers. A market request would *__NOT__* be serviced by a set of the following hand offs; Edge→ most Outer Core → Inner Core → most Inner Core. Rather teams placed in any Outer zones may be directly serviced by any team in an inner zone. So an Edge team could take advantage of an outer core *DevOps* team, as well as an inner core *People Operations* Team. The *DevOps* team could likewise take advantage of the *People Operations* team.

Context Boundaries can be used to group multiple Edge and Core teams according to:

- Missions or Outcomes
- Platforms or Systems
- Business Units
- Business Portfolios
- Common Business Models
- Some other common *Ecosystem* of participating teams

So stated fully;

Organizing Constraint 4: Demarcate organizing structures into discrete contexts based on social boundaries

Facilitating an Organization Mapping Exercise

<u>To Do: Integrate with chapter on impact mapping and add example</u>

Organization Mapping is a practice that allows you to map of market demand, required capabilities, and supporting organizing structure. It's a practice that becomes easier and easier over time as more and more if your organization starts working using backlogs and visual management. The first few times it can seem hard, really hard. You are trying to look at your organization from a lot of different perspectives to come up with a structure for your teams. It's easy to go to deep into analysis paralysis. It's also easy to miss an important detail, or to not give it the right amount of attention.

Before we get started

Before facilitating an Organizational mapping workshop it's important to lay down a few ground rules.

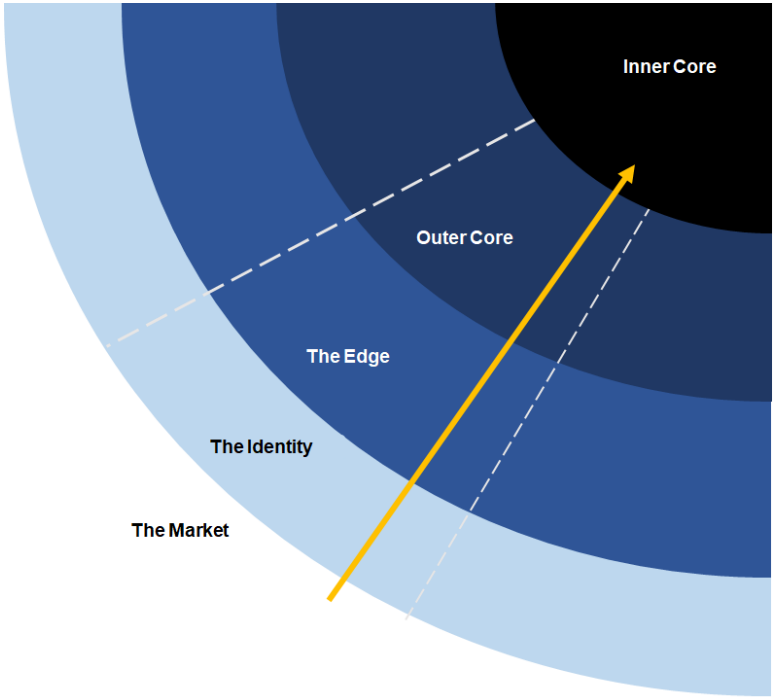
Invite Everyone

First try to involve as many people who are impacted by the change as you can. Emphasis on *invite,* the last thing you want is a room full of non participants. So make sure attendance is voluntary, make it clear that if someone does show up that they will be expected to

participate. Don't worry if you involve too many people, for this type of exercise it is easy for people to self organize into groups that are focused on the parts of the organization that they are involved with (or want to be involved with).

Start From The Outside and Move Inward

When facilitating this type of exercise it's generally better to start from the outside and move inward. If market players are unclear start by defining them. Then move inward by understanding what identifying elements are in place for the organization. Then spend some time to establish some context boundaries including missions and outcomes for teams. It's better to start defining teams at the Edge, then defining Core team from the context of how to serve Edge teams. It is usually not a good idea to start from the core and move outward to edge teams.



FacilitatinganOrganizationMappingExercise/Untitled.png

Use Sticky Notes and Whiteboards to Facilitate the Conversation

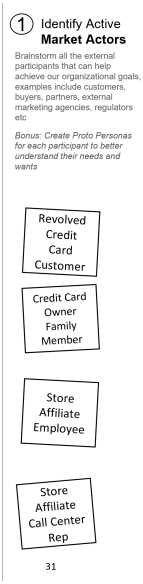
The agilists in the room will say “no kidding”. For the rest of us this is better if stated explicitly. We want to maximize the use of informal tools and we don’t want the design formally specified in a static document. It’ s important to stress that the org design you come up with is not set in stone. The one thing we can expect when working in uncertain times is that we can expect will change. So your org design need to change to accommodate this. The good thing is one people get used to it, refining your org and team structure starts to become relatively painless, but don’t let the need

for formal documentation get in the way.

We want to facilitate our workshop in a way that allows all participants to collaborate, and not have one person do the writing. So facilitator, give everyone sticky notes and sharpies, explicitly ask everyone present everyone to physically get involved, get them in front of a whiteboard or 3M style poster boards and get started!

1 - Identify Market Actors

It's often good to get started by brainstorming all the external participants that can help achieve our organizational goals, examples include customers, buyers, partners, external marketing agencies, regulators etc. At a minimum you want to just post every market participant you can think of, and start clustering them based on their needs and wants.



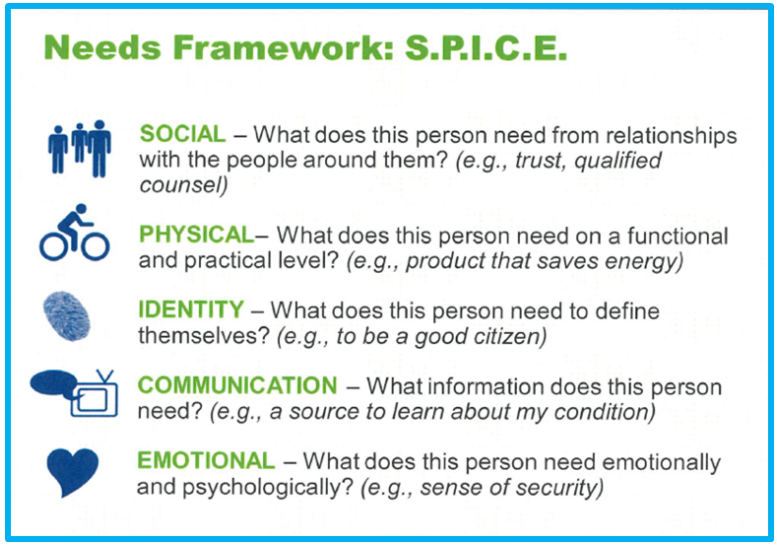
FacilitatinganOrganizationMappingExercise/Untitled1.png

Personas

You can also explicitly map needs and wants to each market participant. I have seen teams take a second and third pass on clarifying market participants, this time creating *personas* for each market participant. Personas are valuable when identifying our market participants because they help us align on:

- customer characteristics and expectations
- Customer needs and wants
- Expected customer behavior

The Spice framework is a nice, concise way to go about defining personas

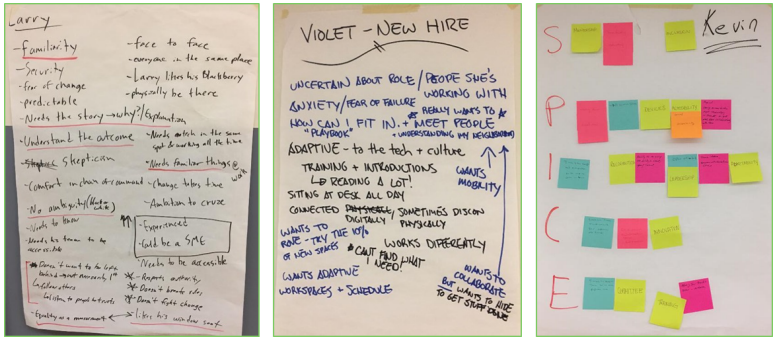


FacilitatinganOrganizationMappingExercise/Untitled2.png

In short, personas help us make better informed decisions that will inform our organizational design.

Proto Personas

A *proto-persona* is a good tool to quickly articulate a persona, think of it as an informal, high level description that is easy to create and easy to update. Use poster paper, whiteboards, and sticky notes to get proto personas out quickly.

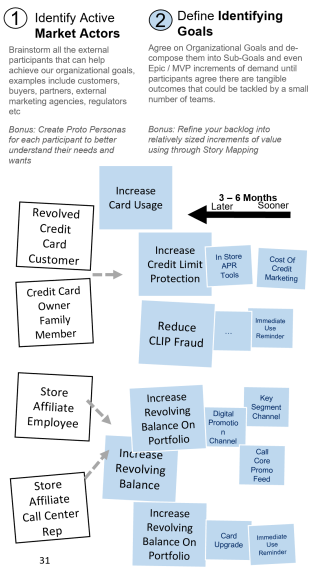


FacilitatinganOrganizationMappingExercise/Untitled3.png

2 - Define Identifying Goals

As you identify Market Actors, you will want to start clustering them according to Organizational Goals that align with each of these Actors. Start mapping Goals to Market Actors and decompose these Goals into finer grained Sub-Goals until you have a set of tangible outcomes that you feel could be tackled by a Mission and / or a small number of teams.

As part of this exercise it is often helpful to refine your Goals into the beginnings of an agile style backlog. In effect you start identifying some potential Epics, Features, Minimal Viable Products, etc that Edge teams and/or Missions could deliver to your Market Actors, and then you start doing some initial analysis to get a sense of their delivery priority.



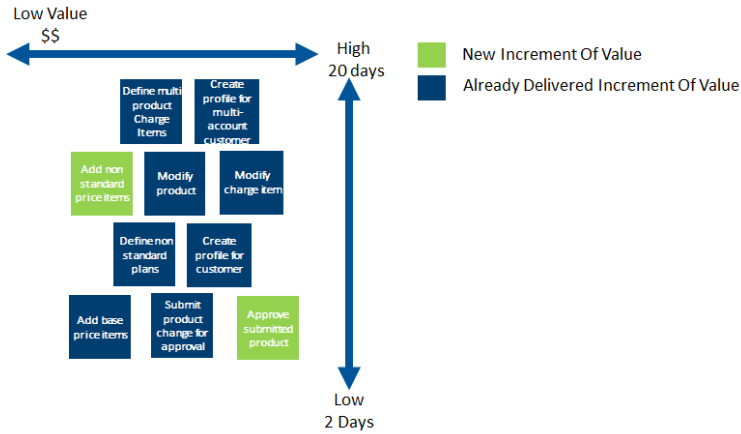
FacilitatinganOrganizationMappingExercise/Untitled4.png

Relative Backlog Sizing and Agile Long Term Planning

Depending on how well your participants knows your organizational goals and organizational demand, it may be beneficial to go a level deeper and further refine your backlog in order to facilitate a better understanding of the backlog for various Missions and their Teams. This allows you to make sure you base your organizational design on the value your Organization is planning to provide to market participants now, and on the value that is next in the queue. If you are facilitating an organizational design session and it appears that organizational priorities are unclear or conflicting you will likely need to step back and start building a backlog that contains larger increments of value you can start allocating to various Missions and teams.

Relative Estimating Done Quick

The first step to creating a backlog, one that provides a better understanding of value and impact, is to stack rank backlog items against each other, or better yet against other units of value that your organization has previously delivered. This allows you to quickly estimate an increment of value relative to something else. It then becomes easy to assign a estimate to backlog items based on where they are in the stack.



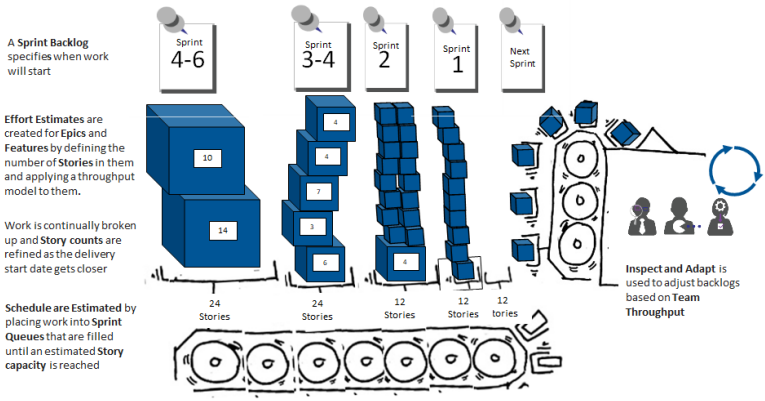
FacilitatinganOrganizationMappingExercise/Untitled5.png

We can use this relative stack ranking approach to size both the size/effort and the value / impact to the market we think it will have. This makes it easier to start prioritizing our demand, putting urgency on work that has higher relative value and lower relative complexity. Note this is not the end of estimating, but a quick way to do just enough discovery to define some reasonable mission and team structure

Using Story Throughput to Get An Understanding Of What is Up Next

As teams become more accustomed to delivering value using Agile methods, they often further break work into smaller increments known as Stories, and they start to track how many Stories they can deliver in a small period of time, typically a couple of weeks or a month in duratio. Many Agilists love to call this period of time a *Sprint.*

Once teams understand their story throughput (also known as sprint velocity) it becomes much easier to make informed decisions about when the next increment of value will make it's way to our existing missions and teams, giving us a signal that it might be time to revisit our Organizational Design.



FacilitatinganOrganizationMappingExercise/Untitled6.png

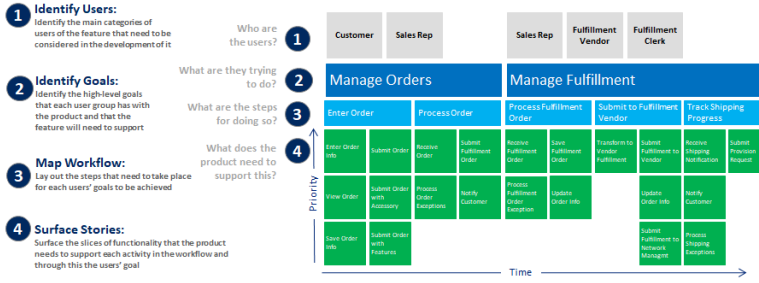
Getting Better Backlog Clarity Through Story Mapping

If Goals , Sub-Goals, Epics, Features, MVPs etc are still not clear enough for you to guide the Organizational Design dialogue con-

sider refining one or more items on your backlog through a Story Mapping session.

Story Maps allow you to map the flow of your solution in a hierarchical way that starts at Personas and their top level User Goals (known as Epics in agile speak), then moving to lower level User Actions and Steps, all the way down to micro units of value that can be delivered quickly by teams, ie Stories.

Story maps also allow you define independently releasable thin slices of value, or MVPs. This helps you prioritize small units of value that you can release to your customer. If you are churning on trying to understand the what, the why or the how of an increment of value, I highly recommend you try out story mapping.



FacilitatinganOrganizationMappingExercise/Untitled7.png

Establishing the Tactical Aspects of your Organizational Identity

If you don't feel participants In your workshop have a good handle on their Organizational Identity. I recommend taking a step back and spending some time on articulating Organizational Identity. There are many tools that can be used to help with this exercise. The goal of using these tools is to:

- Understand who our customers are, and what problems we are trying to solve for them

- Agree on the solution we can provide to customers to solve their problems
- Identify the key skills and tasks required to deliver that solution to our customers
- Articulate any key assumptions that should be validated

Some good tools are your disposal include:

- Business Model Canvas (https://en.wikipedia.org/wiki/Business_-Model_Canvas[16])
- Lean Startup Canvas
- Journey Mapping
- Impact Mapping
- Service Blueprint
- Cost Of Delay Estimation

Articulating Your Identifying Culture

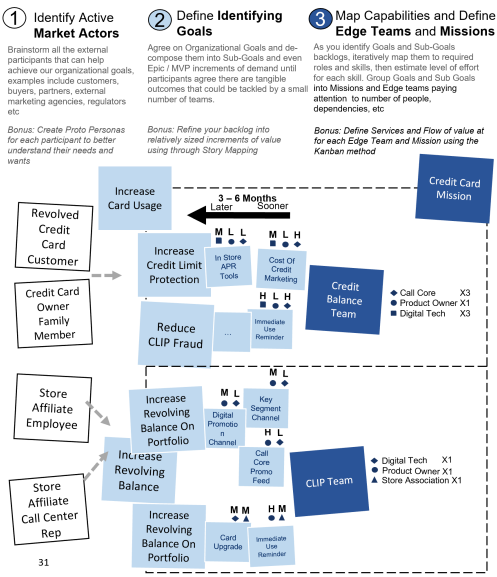
The tools mentioned above are focused on defining the tangible aspects of an Organization's Identity, this is important, but it is equally important to align on culture as well. Taking the time to at describe an aspirational culture, ie the Organization we want to be can seem to some to be superfluous, but can really set the stage as the Organization continues to grow. Some methods to describe an Organization's culture include:

- A Letter To Ourselves
- Values and Beliefs
- Mission Statements
- Culture Document

[16] https://en.wikipedia.org/wiki/Business_Model_Canvas

3 - Map Edge Capabilities and Teams

As you identify an ordered backlog that can guide your organizational design, take the time to list capabilities, skills, and/or roles required to deliver on this backlog. Spend some time to do a high level estimate for each capability you identify. Estimates can be very high level using T-Shirt sizing or some other relative ranking system. As the capabilities become clearer you will be able to cluster similar increments of value together to form missions and teams within those missions. This clustering could happen at the Goal, Sub-Goal, Epics, Feature or whatever level of granularity allows you to have enough critical mass to put teams and missions together.



FacilitatinganOrganizationMappingExercise/Untitled8.png

As you are doing this you can start approximating the number of people you require for a particular capability in specific Edge Team or Mission. At this point it is important for me to stress that this

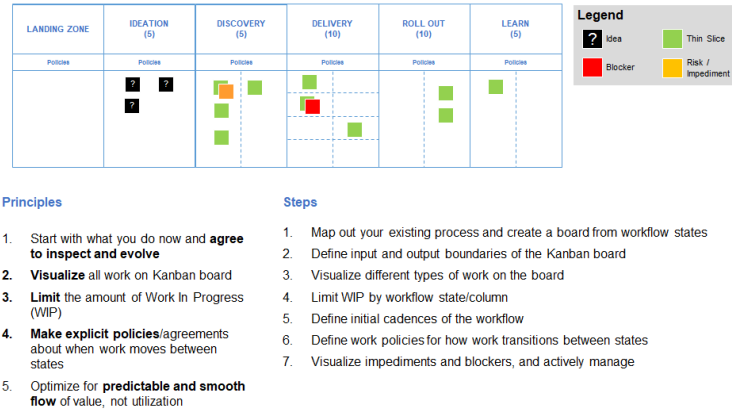
workshop can't be facilitated in a linear way, you will need to revisit how you are thinking about mapping participants to your backlog and supporting missions as you go.

Take care not to create structure focused primarily on systems, capabilities, or roles. Create teams and mission level structure around Market oriented Goals, Epics, or MVPs if you can.

Refining your Edge Team Structure Using Kanban

As We are designing Edge teams, we may want to get a better understanding of the work that those edge teams will do. This will allow us to make more informed decisions when trying to create context boundaries around teams and higher level missions.

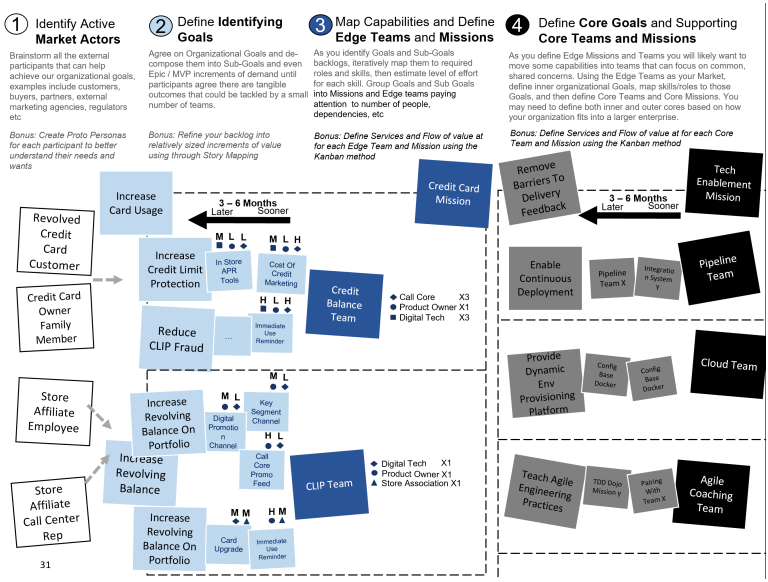
Kanban is an excellent method that not only enables teams to operational their value creation activities with agility, its a great mechanism to further refine your Mission and Team level structure. With Kanban, our effort to Organize for agility can be informed by cataloging market demand to Edge Teams into customer oriented services. Each of these services can them be mapped to one or more discrete work types (capabilities) which is then defined by an explicit and visual flow of work. Kanban is then used to visualize this flow of work in such a way as to spot bottlenecks and other impediments. This provides a rich set of observations that empower teams to continuously experiment on changes that further refine how we are organizing around value. I'll go deeper into how Kanban informs how we get organized in further chapters.



FacilitatinganOrganizationMappingExercise/Untitled9.png

4 - Define Core Capabilities, and Teams

As you define Edge Missions and Teams you will likely want to move some capabilities into teams that can focus on common, shared concerns. These are what Core Teams, and Core missions are for. When designing Core organizing structures, you use the exact same process that you would follow when designing Organizing structures for the Edge. In this cased you use Edge Teams and Missions as the Market Participants for your Core Teams and Missions. Use the the needs and wants of Edge teams to define Goals for your Core. You can then map skills/roles to your Core Goals, and further define and refine Core Teams and Core Missions. Of course defining your Edge and Core will be an iterative process, you dont need to finish figuring out your Edge before getting started on putting up some Core items as well.



FacilitatinganOrganizationMappingExercise/Untitled10.png

Inner and Outer Cores

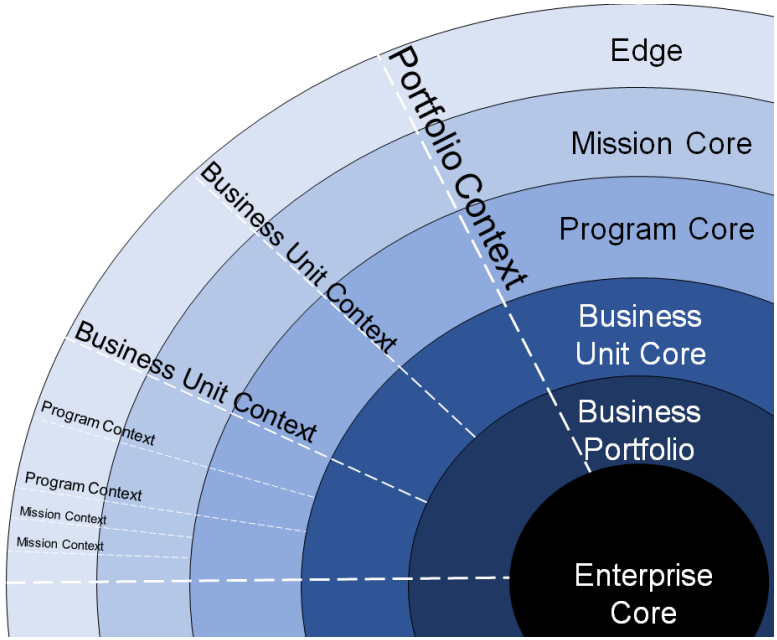
Likely you are working in a larger enterprise that contains capabilities and services that you are mandated / highly encouraged to use, they just come with the package of working in your enterprise. In these cases it is helpful to model these services as coming from on or more Inner cores. Examples of inner cores include Enterprise level services and managed Business Unit level capabilities. Neither are likely to be far along the Agile journey, but you may be surprised.

When working in this context all services that you are able to facilitate change with can be placed in your organization's Outer Cores. Services that are provided to a small number of teams, perhaps in a single mission can be though of as belonging to the outer most core. Services that are provided to a larger number of missions, a whole program, a portfolio of work, or a whole business unit can

be thought as belonging to Cores that are closer and closer to the center, ie they are progressive more inner cores.

A Special note On Context Boundaries

As you go through the process of defining teams and missions at the Edge and various inner and outer cores, use context boundaries to define larger structures. Some of these structures will be mandated as being part of your organizations enterprise, some of these will be structures that are defined as you facilitate organizing for agility.



FacilitatinganOrganizationMappingExercise/Untitled11.png

Context Boundaries follow the same narrow to broader categorization that cores do, in fact all wider Context Boundaries larger than a

team will often have at least one core team, the team responsible for managing that Context Boundary. Typically there will also be other core teams allocated to the Core zone defined by a wider context boundary, or else there is little justification in the existence of that Context Boundary. Again context boundaries can fall into familiar categories from smaller to larger:

- Mission
- Program
- Business Unit
- Business Portfolio
- Enterprise

Refining Your Organization Through Collaboration Patterns

RefiningYourOrganizingStructureThroughTeamCo/Untitled.png

When looking at ways to organize for agility, we often turn to the classic agile team as the "one structure to rule them all". Yet in my experience this is only one of several organizing structures that that help increase collaboration. This article provides a description of other ways people can organize to increase collaboration, documented as a set of Collaboration Patterns.

Why are Patterns Important

An intro to patterns and pattern languages

Design Patterns are something I fell in love with almost two decades ago when I was cutting my teeth as an application developer. In the face of a world that teemed with with complexity, discovering patterns was a welcome alternative to standard playbooks, processes, and answers presented as simple recipes that we could just copy and paste.

As we have already discussed, the one size fits all approach to managing our work breaks down in a world of constant market

uncertainty. What the other person uses to make himself successful can't be used to make yourself successful, at least not without substantial modification.

Software Developers have known this for a long time. The algorithm, data structure, or object interaction that works for one use case can't be blindly applied to the next use case. But common reoccurring problems could be solved by solutions that had common attributes. These solutions could be presented in such a way as to discuss some common implementation options. Trade offs and relationships to other solutions could also be shared.

The object oriented community is a place where Pattern Languages first became wide spread, even if it's not where they were invented. A *Pattern* is a solution that addresses a known family of problems. Presented with trade offs and relationships to other patterns. A *Pattern Language* is a map of these patterns and their problems that illustrate how these patterns connect to each other. Patterns allow us to get reuse because we don't make the fallacy of thinking that a pattern is a fixed solution. Patterns almost never get implemented without some form of modification. Patterns are also meant to be small, you could never implement an entire software solution using only a couple of patterns. You would likely use many of them, and you would use them to solve discrete aspects of your solution.

Patterns for organizing structures that improve agility

The idea that patterns can also be used to help get better organized came to me when I was reading various texts by Don Reinertsen[17]. Two of his books Managing the Design Factory[18], and Product De-

[17] http://reinertsenassociates.com/

[18] https://www.amazon.com/gp/product/0684839911/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=0684839911&linkCode=as2&tag=reinertassoci-20&linkId=PX2E35Y5Y2GXK4HU

velopment Flow[19] were instrumental in changing the way I thought about agility and agility at scale. One of the key things Don talks about are process patterns. Traditional Process design often suffers from designers trying to get alignment on top and medium level processes across an organization, one that spans many different contexts. In complex systems this approach doesn't work. A much sounder approach is to agree on common process patterns, small reusable solutions that can be adapted to be fit for a particular problem space. These process patterns can be connected together as required to define larger organizing solutions. Pattern maps are used to help navigate how to connect these patterns together.

Just like children can use a finite set of LEGO building blocks to build a much wider, almost infinite variety of larger creations. Don's writing on patterns caused me to think about the common approaches shared across official agile methodologies. and start start seeing these common patterns of collaboration.

While I was reading Don's work, I was also getting immersed in David J Anderson's[20] Kanban Method[21]. People often mistake Kanban for just being a visual management system that allows you orchestrate and improve the flow of your work. It is that, but Kanban is also a lot more. Once you reach a certain level of understanding, you will find that Kanban enables people to re-team as required to collaborate. Not only in stable, cross functional teams, the most obvious pattern we see in the agile world, but lots of other ways of forming together to collaborate as well.

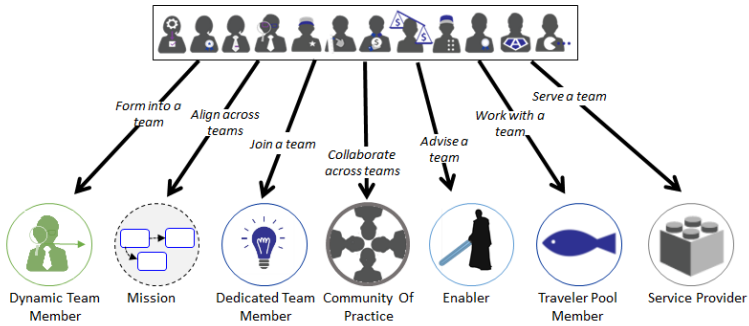
[19] https://www.amazon.com/gp/product/1935401009/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1935401009&linkCode=as2&tag=reinertassoci-20&linkId=U56AHKXXQVN4VZFY

[20] https://djaa.com/

[21] https://www.kanban.university/

Collaboration patterns

Collaboration Patterns discuss different ways people can can engage with each other to create value. Common attributes like the duration of the collaboration, the nature of the collaboration, and the outcomes are presented.



RefiningYourOrganizingStructureThroughTeamCo/Untitled1.png

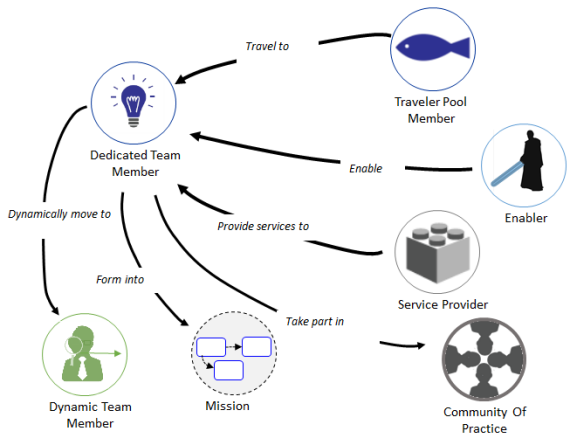
Collaboration patterns are helpful when you are refining your organizational design and want to layout how different skills and the people that possess them can engage with each other across your organization. These patterns also used to describe how members of Edge teams can collaborate with each other. They are especially helpful for people who provide support functions from the Core and want to understand their place in an increasingly agile world.

These patterns are:

- **Dedicated Team Member** - you are a dedicated member of a classic agile team
- **Enabler** - you are responsible for enabling multiple teams to use an organizational cross cutting capability, by providing guidance, a platform, relationships, and/or knowledge
- **Community Of Practice** - you are part of a group of people that share passion and knowledge and learning with each other about a certain capability

- **Traveler Pool Member** - you are part of a pool of people that travel to work closely with teams who have an ad-hoc need for your highly specialized and often difficult to learn capabilities
- **Service Provider** - you service the need of other people in the organization by fulfilling an order request, largely in isolation from the person making the request.
- **Dynamic Team Member** - You form into a team temporarily, to achieve a short term goal
- **Ecosystem** - you align a group of teams, pools and Enablers to a common overarching outcome, sharing a common backlog of work, and/or solution platform.

These patterns, and the different options that these patterns can be used form a *Collaboration Pattern Language* presented below.

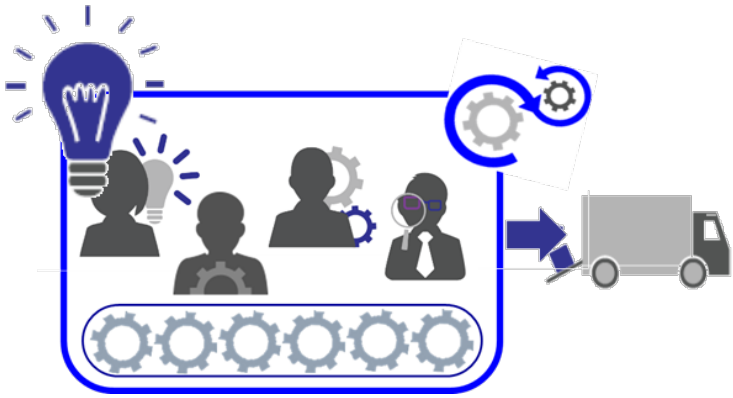


patterns to collaborate with agile teams

Dedicated Team Member

Summary

You are a full time member of a cross-functional, stable, market facing team (ie Classic Agile Team)



dedication, dedication, leads to value creation!

Problem

You are trying to create value in an environment of uncertainty, complexity, and constant change. A mix of skills and capabilities is required to successfully deliver that value. The need for your particular set of skills is significant and it is also stable over time. A high degree of collaboration is required between you and the rest of the team. The work is complex, emergent, and requires continuous learning

Solution

Collaborate as a *Dedicated Team Member* of a classic agile team. You and your team members posses enough cross functional capability to deliver on a market outcome.

When you choose to be a Dedicated Team Member, you are committing to the fact that you will be spending the majority of your working day with that team in order to achieve the goals of that team.

Discussion

Where possible we want most of our Edge teams to be comprised of people who are choosing to be Dedicated Team Members. We also want most of our people in the organization to be part of Edge teams. So, most of our people end up being dedicated Team Members of Edge Teams. This won't always be possible depending on context, but it is a goal to strive for.

In general, we also want Dedicated Team members to be part of a team that takes responsibility for end-to-end delivery of value, but this is not always possible. In some cases, we may have teams that focus on a particular phase of delivery such as discovery, delivery or operations.

Options

While the typical 6 - 8 team size is a generally recommended. I have seen teams smaller and a good size larger. While agile purists may balk at teams that get much bigger than 8, the reality is that members of larger social groupings can form into smaller groups to work on discrete items in their backlog.

Feature Cells

Feature Cells are a grouping of 2-3 team members who form to work one on discrete increment of value ie a feature. Team members can choose to form into feature cells as a part of the teams planning cadence, and then choose to be part of a different feature cell during the team's next planning cadence.

Dynamic Feature Teams

Some organizations choose to forego stable teams for the most part. Instead members of a larger Mission / Tribe / Portfolio form into teams based on looking at a larger, Epic / Goal level backlog, typically during a mission level planning cadence. In this model members of the team not only self- organize once they are in a team, they also self-select the team the form into,.

Dynamic feature teams may have some members of the team who

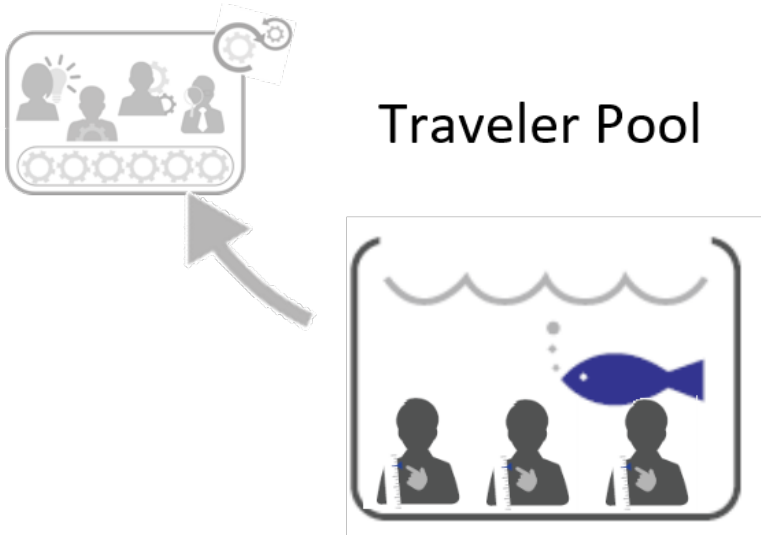
are considered stable,ie fixed members, while the remaining membership changes dynamically.

In the case of both Feature Cells and Dynamic Feature Teams, visual management systems ie Kanban can be used to help inform people on how to dynamically form.

Traveler Pool Member

Summary

You are part of a pool of people that travel to where the work is to collaborate closely with members of other teams.



Problem

Market facing team, have a need for your capability. But that need is ad-hoc in nature. Your skills are likely highly specialized and hard to learn. Your skillset can take a long time to learn and required extended training, intimate knowledge, and extensive experience

to be effective. There is a limited amount of people that have your skills set, and it is challenging to increase that number.

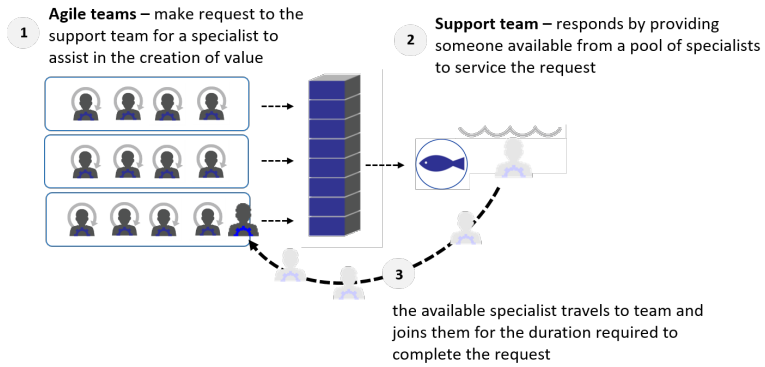
Demand from any one team for your skills tends to be infrequent, and/or inconsistent, so it is hard to justify your permanent presence on any one team. The nature of your work does tend to be complex, so when you are working with a team to create value, a high degree of collaboration between the you and other team members is required.

Solution

Organize yourself into a pool of *Travellers., apool is made up of all the people that exhibit similar capability. This pool services demand across multiple agile teams. Allow other teams to draw from you and your peers in the pool on a just-in-time basis. Use backologs, throughput planning, etc to prioritize the work, just like any other agile team would. When servicing a request from another team you travel to that team and work in a highly collaboration fashion, in effect becoming a member of that team for a short period of time. At the end of the request you go back into the pool.

Discussion

Traveler Pools allow you to minimize hand offs in the face of demand that does not lend itself to stable teams. One of the biggest mistakes people make when organizing teams is to structure teams based on capabilities, and then allow work to be distributed across multiple teams, relying on cross team coordination and hand offs between teams. We know hand offs reduce a team's ability to self-organize and respond to feedback. Hand offs reduce agility. The alternative option is to use pools of travellers instead, that way we *move people to the work*, rather than moving work to people.



Imagine we have a web team that delivers features on a web site that touch across a wide range of customer experiences and business products. The website needs to interface with a number of different back end system depending on the touch point and/or the product being exposed. When delivering digital channel website changes to the market, the digital team often need to ensure that a supporting changes is made to one or more of these supporting back end systems. But no two web site changes hit the same back end system across any particular calendar quarter. Most of these back end systems could be considered legacy systems, or at the very least highly proprietary. Making changes to each of these systems is done by a separate group of people, all who who possess very distinct and specialized knowledge from each other as well as your garden day variety object oriented developer.

Now let's scale the problem out to a more realistic enterprise scenario. Imagine we also have a similar mobile channels team, an IVR team, and a call center support channel team. These teams are also responsible for building systems that touch on the same range of products and customer experiences that the web team does, and need to likewise make similar types of changes across the various back end systems that the web team is.

In this case, organizing people who can make changes to these back end systems into separate **Traveller** pools, one for each system, is a

good way to go. Ensuring a feature works end end across multiple systems is a complex endeavor. One that is best performed without handoffs across teams. Organizing people into pools ensures that the ad-hoc nature of the work does not cause collaboration to suffer.

Other good examples where I have seen travelling work well is dedicated UX professionals, customer research and analytics, and larger / program level UAT expertise.

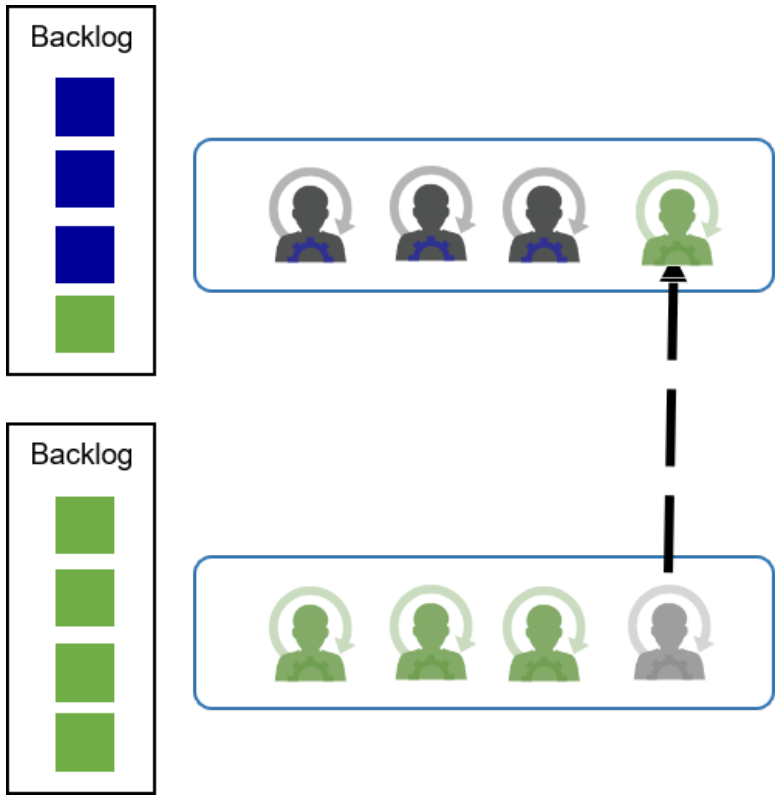
Being a member of a Traveler Pool is not always easy and can cause stress for the member of the pool. Maturity is required for a person to be able to switch from team to team and be able to work in an open and collaborative manner.

Traveler Pools can use visible backlogs and other agile cadences and practices to prioritize and manage their flow of work. This includes holding occasional standups, retrospectives, working on technical debt, or other internal improvements.

Options

Dedicated Team Members who also Travel

Dedicated Members of Edge Teams may also act as Travelers to other Edge Teams. This happens when it is not possible for two or more edge teams to completely isolate their work from each other. In the case where teams have dependencies on each other, it makes sense for a dedicated team member to go into traveler mode and work with the other team, rather than expect people across teams to work on a shared piece of work in isolation from each other.



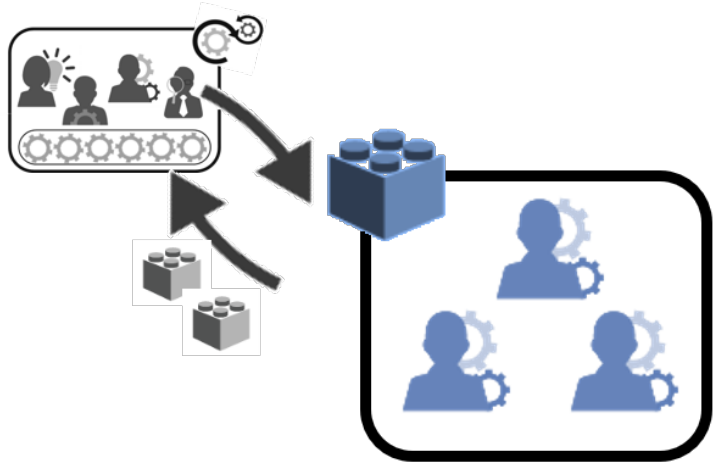
Traveling To Enable

Travelers can also ask members of dedicated team members to pair with them to start picking up their skills. As capabilities sharing starts to happen, the Traveler is able to shift his approach, taking the stance of an *Enabler*, moving away from doing the work, to guiding someone else on how to do the work.

Service Provider

Summary

You service the need of other people in your organization by fulfilling an order request, largely in isolation from the person making the request. The service provider works closely with other service providers who have the similar capabilities, and does not work closely with the team requesting their services.



Problem

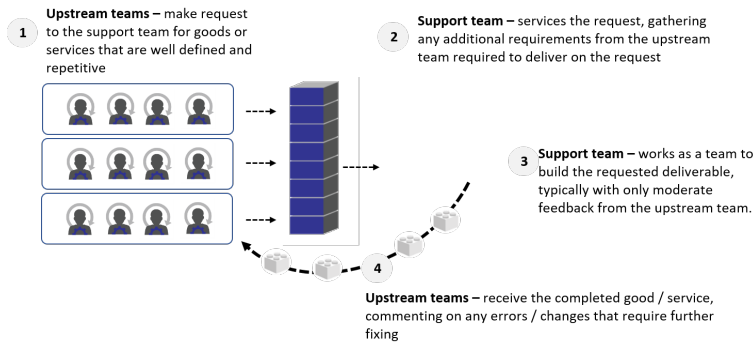
You need to fulfill requests from many different teams. Individual work requests tend to be smaller in duration / effort. Reducing the cost of the services you provide is paramount, in other words you are looked at as a cost center. Your work is considered to be highly repeatable and does not vary greatly from request to request. Economies of scale can be applied to share demand, reduce cost, and increase efficiency.

Solution

Perform your work as a *Service Provider*, as a part of a *Service Center*. A Service Center is a group of other similarly skilled service providers formed to service requests for a similar set of services. As a Service Provider you handle requests for your work using a

service request and respond, order ticketing mechanism. A work ticket is placed on a queue by a requesting team, a service provider working in the service center picks the ticket up off the queue, performs the work required, and then moves on to the next ticket.

Work can be thought of as travelling to the team in this model. You may provide periodic consults to your clients, but the vast amount of your work is completed independently of them. The majority of your time is spent in the service center, you deliver the output once the work is complete.



Discussion

Of all the collaboration patterns presented so far, this one will lead *to the least amount of organizational agility!* Consider other team collaboration patterns if you want to demonstrably improve organizational agility.

Service providers reduce agility because every time you are relying on them you are creating a hand off! Hand -offs erode self-organization. They cause lead times to go up, and they require coordination, typically management coordination across the teams.

Most service providers will not have your context. They won't understand why they are doing something for you, and will not understand why it is important to your customer. This lack of context causes poor results that often miss the intention of the

person who needs the work completed. The result is re-work and reduced efficiency, the very reason we want use a Service Provider in the first place.

Most back office functions are organized by placing Service Providers into service centers. They should not be. The majority of functions provided by HR, Legal, Compliance, security, IT, etc would be an order of magnitude more effective if they collaborated as Enablers (see below). or at a minimum Travelrers.

You will find some situations where the Service Provider is the best way to go. Activities that require a lot of manual labor and manual processing may need the economies of scale offered by this model. As software becomes more pervasive, and we continue to automate our work-flow, the need for such Service Providers should continue to diminish.

For the most part, the Service Provider is really an anti pattern. One of the biggest values the Service Provider pattern provides is we can describe the opposite of agile. Anyone responsible for defining a more agile organizing structure needs to describe the option we come to when we are not able to negotiate a more agile solution. We can represent these parts of the organization as they are now, as opposed to representing some idealized future.

Service Providers can start to increase their agility by adopting practices such as backlogs and visual managements system to introduce transparency and achieve better customer engagement, with an eye toward considering different collaboration patterns.

Options

Service Provider From the View of the Outside

A team may seem like a service provider from the perspective of the customer of the team, it provides a common service using a request response approach. That team could also appear to be comprised of *Dedicated Team Members* within a classic agile team when looked at from the perspective of the team. Unfortunately,

most agile teams in IT departments fall into the category. Few on the team have direct market contact, work comes to them from various organizational stakeholder instead. While the team may exhibit great internal agility, they are not situated to exhibit great market agility.

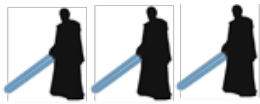
Intent Owner Traveler

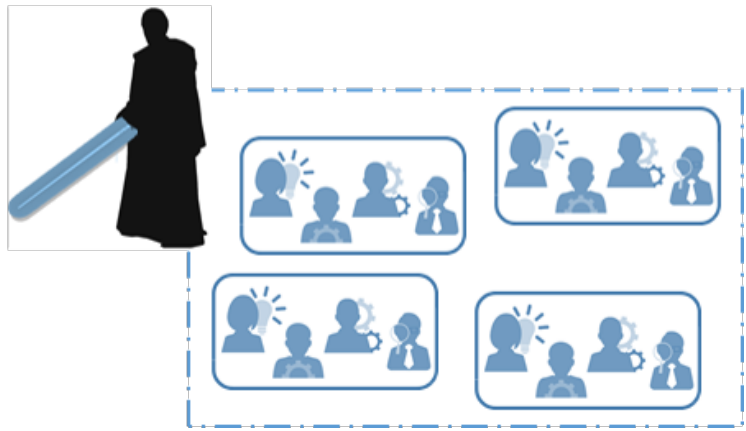
One way agile teams mitigate the impact of being engaged like a service provider is to try and get stakeholder / sponsors / users to come and spend time with the team and bring market connectivity with them. This is a reverse kind of travelling, the person that needs skills / capabilities in order to deliver value travels to the team. This *Intent Owner Traveler* collaborates closely with the Service Provider Team, in effect they operate much closer to spirit of an Edge team. As more an more intent owners spend more time with the team, the team starts to look more an more like a true edge team.

Dependent Edge Teams

When two teams agree to work on a complex piece of work that is shared across teams, and neither decides to pick up and move to work with the other team, than both team are acting like Service Providers to each other. They are now *Dependent Edge Teams*. Sometimes this can't be helped, but in most cases we are dealing with social inertia and a mindset that once someone is in a team, than they should stay in that teams. Remember, hand-offs kill agility, don't be that person who refuse the travelling option, and creates a hand-off.

Enablers





facilitating order in the galaxy?

Summary

You are responsible for enabling multiple teams to use an organizational cross cutting capability, by providing guidance, a platform, relationships, and/or knowledge.

Problem

You have a small number of people that possess a core set of capabilities and you want these capabilities to serve as foundational building blocks for as many teams as you possibly can.

You want to provide these capabilities in ways that give your organization a strategic advantage in the market, perhaps increasing

safety, security, or automation.

You want to provide these capabilities in way that allows teams to spend more time delivering market value, and less time reinventing "infrastructural" concerns.

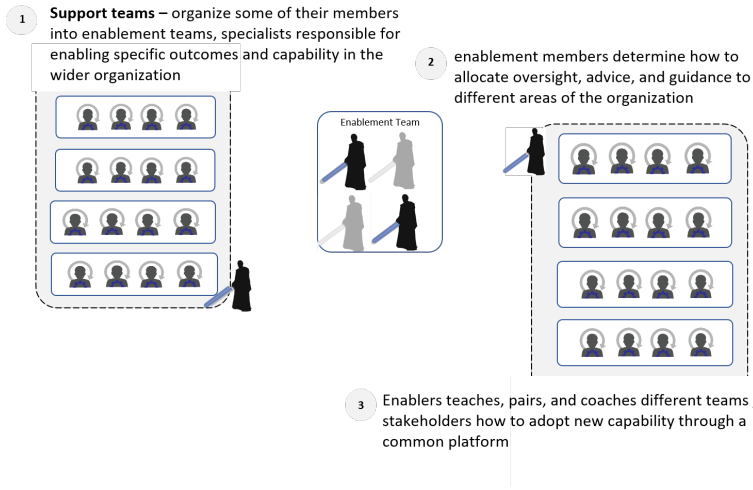
Solution

The small group that possess these capabilities collaborate with other teams as *Enablers*.

As an Enabler you are responsible for bringing your knowledge to team members. You may also provide a platform so that teams can self serve. You are not responsible for doing the work of teams, you are responsible for helping team members be more effective.

Discussion

Enablers often are members of a capability specific *Enablement Teams* that sits in the Core. Often members of these Enablement teams are each focused one a subset of the organization, like the Jedi Council or the Green Lantern Core (yes I am geek, quelle surprise), an Enabler can be assigned to watch over a closely related set of teams such as all teams on a program or mission, helping to bring order to their sector of the galaxy.



Enablers help teams be effective by empower them to operate in a self-service mode. Technology platforms, bodies of knowledge, teaching aides, etc are all tools of trade for enablers.

HR, Legal, Security, DevOps, Data/Reporting and Agile Coaching teams are all good examples of enablers, positional leadership, when acting as servant leader can also be thought of as Enablers. One of the biggest challenges with Enablement is that it is confused with authority. In traditional organizations many of the functions listed above maintain coercive power over Edge teams. When using an Enablement pattern, the services of an Enabler are *voluntary*. Teams are not mandated to use the services of an Enabler. We cannot effectively enable others through command and control as it will never lead to a durable change in behavior.

Example: HR recruiting Function

An example would be the recruitment and hiring function in HR. In a traditional org many teams would be mandated to go through an HR rep to perform the initial screening, setting pay bands, writing job descriptions, and engaging with talent firms. IN contrast, when

acting as an enabler, HR would provide a platform to make this job easier for teams. Outside of some minimal rules to ensure compliance with laws and avoidance of reputational risk, HR would not mandate its involvement in hiring, but rather it would offer help including:

- advice on how to effectively navigate the hiring process
- technology that streamlined the logistics of interviewing, assessing, and finally hiring a candidate
- sharing hiring assets (ie job postings) that other teams had created
- coaching on practices related to hiring candidates

The Enabler focus puts the the team being enabled in the position of being a customer of HR, as opposed to a subservient position. This ensures that HR services stay healthy, and suited to the needs of team!

Pattern Options

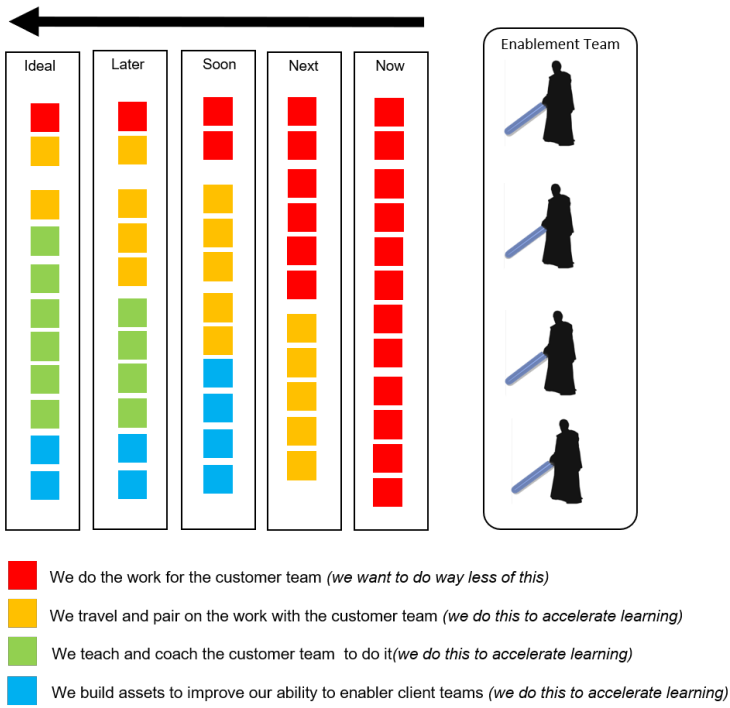
Enabler who Travels and Provides Services

It can be a struggle for people providing services to other teams to move into Enabler mode, at least at first. Until the organization reaches a certain level of maturity, teams will likely ask or even demand services from Enablers in ways that do not fit the Enabler pattern.

Sometimes this means working closely with teams and being hands on, either for a short or longer period of time.

As an example, a team of Agile Coaches should operate as the archetype of an Enabler. Yet coaches may need to temporarily play the role of Product Owner or Scrum Master if there's a desperate need. A Data enabler may be asked to help the business by creating reports rather than teaching them how to use the reporting platform. None of these activities lead to enablement of those teams, but sometimes what is practical trump what is pure.

On a positive note, teaching others is often accomplished through a do it, teach it, pair on it, and then coach it model. So doing the work can be thought of as a tip of the enablement spear. The *Enabler who Travels and Provides Services* provides a path towards this evolution. Enablement Teams can start reserving capacity to build up platforms / knowledge to enable self serve. They can then travel to customer teams to first do and show, then pair, then advise, and finally step back.



Getting agile teams comfortable with using Devops is an excellent example of how an infrastructure team can graduate from a Service Provider team, to a pool of Travelers, and then to a more purer form of Enablement. They can thoughtfully increasing the level of self service that teams can perform through a combination of platform,

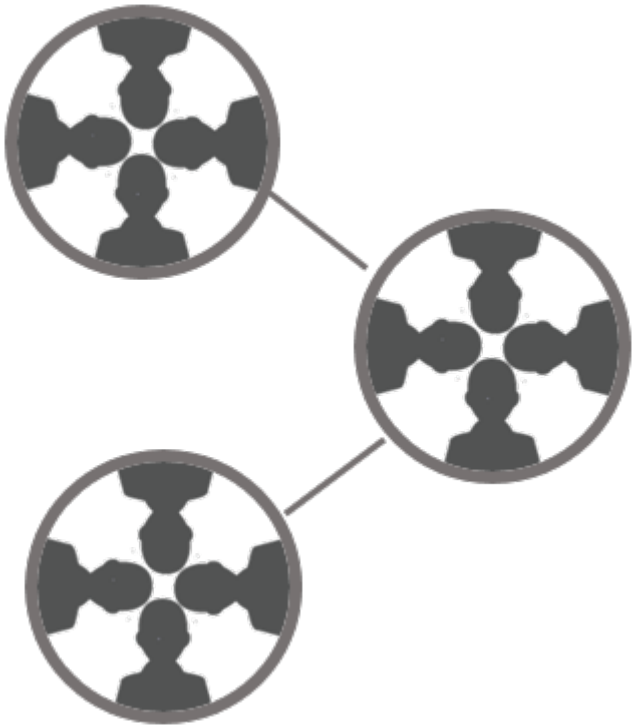
permission, and coaching.

The Devops team can use a highly visible agile style backlog to shape their demand to match the enablement model they want to achieve, including groundwork such as getting permission to infra, aligning with vendor / providers, marketing the offering, rolling out a platform, and building a body of knowledge.

Communities of Practice

Summary

You are part of a group of people that share passion, knowledge, and learning with each other about a certain capability



Problem

You want to to encourage grassroots level learning and shared understanding on a particular topic or capability. You may posses a functional capability that does not really have an organizational home in this new world. You want an opportunity to volunteer your time to collaborate with others to excel in a common craft.

Solution

You participate in a Community Of Practice. When you participate in a CoP, you collaborate to share knowledge amongst your peers. You spend some of your time creating a body of knowledge through consensus, and through sharing knowledge from practitioner experience in the field.

Discussion

Communities of practice are responsible for stewarding specific capabilities that are important to the overall enterprise. Unlike Enablers, Communities of Practice are often virtual; membership comes from interested contributors on other teams. CoPs may have dedicated funding and/or management. The difference between a community and a more traditional Center of Excellence is that communities do not hire, fire or manage people's careers. They are not resource centers. A communities mission is to foster a sense of community amongst like minded people, to help people move their craft forward. They hold learning events, tours, outings, etc and activity is driven largely by volunteers.

Options

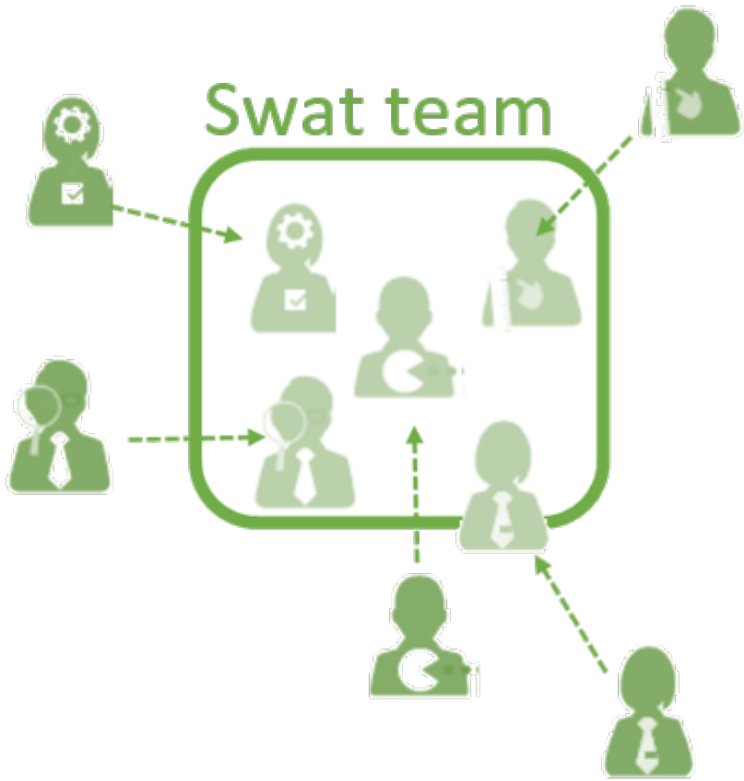
Enablement Vs Communities

It may be tempting to organize many capabilities as Enablement centers, for instance an Engineering Enablement team and a Product Owner Enablement team, or a UX Enablement team. As we try to increase agility, we want to also allow volunteerism to steer many of these efforts. Consider putting effort behind Communities Of Practice instead. Enablement teams can and should spend time fostering communities, a good sign of progress is when an Enablement team is able to retire, the capability is being fostered effectively through a community instead.

Dynamic Team Member

Summary

You form into a team just-in-time with other members drawn from a collection of other teams and departments.



Problem

You need to collaborate with members from a number of different teams at the same time in order to accomplish a complex task. You do not want to change your existing team structure to collaborate on this task. Perhaps the duration of the request is to short to justify such a change, perhaps their is a lack of will across teams and their managers to move people to different teams for an extended duration.

Solution

Form a *Dynamic Swat Team* with all of the people from the other teams you need to collaborate with. Often you and other members

of the team will agree to participating in one or more agile style sessions, and meet at a steady cadence. Shared discovery, shared stand-ups, and reviews are all examples of events a dynamic swat team could hold to improve collaboration. Dynamic Swat Teams will often agree to adopt some form of visual management to increase transparency.

Discussion

Use the Swat Team pattern is helpful when you don't want to restructure your teams. Examples include, fixing a severity one defect that spans multiple systems, converging on a fix dated, last minute opportunity, or supporting cut over during a merger and acquisition. Swat Teams are used to transcend official structure.

Options

The Project Team

Organizations that are beginning to consider agile as an approach will often form teams based on projects, under the purview of a project manager. Once the project is done team members will return to whence they came. This approach that is criticized for a number of reasons. Teams that are constantly reformed and disbanded never get a chance to become an effective team, self-organization never takes effect. Learning as a team never happens

This can be a reasonable first step. Experience gained can be used to inform a more ambitious adoption, one based on teams. It can be relatively straight forward to go from a project team to a longer lived team. But if agility is to improve then project teams are only a first, temporary step towards something more durable

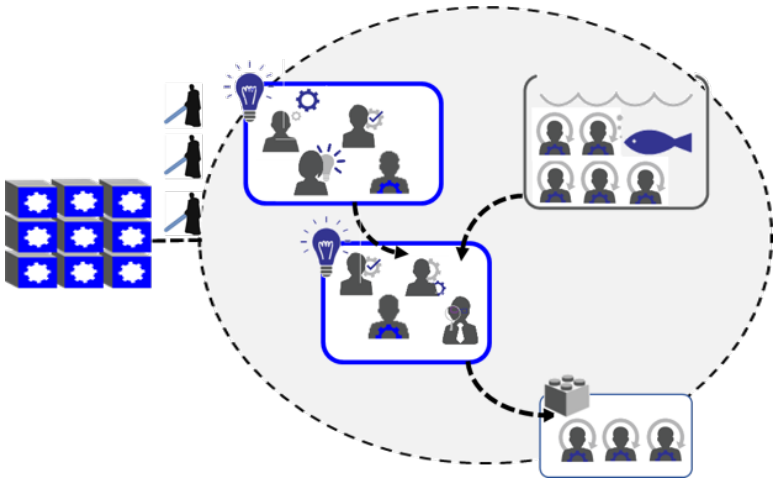
The Self Forming Team

A self forming team, such as a dynamic feature team could be thought as a form of swat team, albeit with more systematic attributes. Organizations will often use a visual management system, typically a Kanban system at the organizational, program or portfolio level to guide the process of self-selection. Members of different

teams can use a higher level Kanban to look for opportunities to swarm around the work as necessary to maximize flow of value.

Agile Ecosystem

You are part of a tightly connected ecosystem of teams focused on a common cause.



Problem

You are trying to deliver value to the market however there is more work to be done than could be accomplished by one single team. As you take on extra work required to scale, you spin up extra teams. Invariably teams run into challenges as teams start duplicating each others work, mis-aligning on outcomes, and in some cases causing conflict with each other.

Solution

Group all Edge Teams, Traveler Pools, and/or Enablers that focused on a common overarching outcome into a single context boundary, named for the outcome trying to be achieved.

Discussion

All members of a common Agile Ecosystem share a distinct subset of the market of the larger organization. Teams will typically share a common, higher level backlog of demand made up of coarser increments of demand (eg: Epics / Features / MVPs/MBIs etc) and will collaborate to deliver on that backlog.

Thanks to it's size, the Agile Ecosystem can be given larger responsibility than a single team often can. They can truly own an outcome, including operational activities, customer interactions, marketing, and support. We want to do everything we can to minimize dependencies across ecosystem. For this reason we want Agile Ecosystem to have their own core, with their own leadership, and dedicated enablers.

This will not always be feasible, but it is a design principle worth keeping front and center. One way to do this is to ensure that all dependencies between an Ecosystem and the rest of the organization are exposed through platforms, one that enables self service.

We need to be careful to not let Agile Ecosystems become to large, or it will be challenging for people to collaborative in any meaningful way. While there are no hard and fast rules, my experience is falls somewhere between 12 and 50 people. As we increase the size , we run the risk of fragmentation and churn, consider breaking up the Ecosystem into multiple, smaller ones.

Options

The Micro- Enterprise

Micro-enterprises can be thought of as miniature organizations within the larger enterprise. They are the idea of independent teams taken to their fullest extent.

I can't remember the first time I heard of the Term Micro enterprise, but I am very fond of the overview and description provided by Joost Minnaar and Pim de Morree in their excellent book Corporate Rebels[22]

"Micro - Enterprise are expected to engage their users throughout the entire business process: from R&D to product design to production and delivery. Users can offer feedback at any time in the process. In this way, Haier wants to turn the one-time customer into a lifetime user."

According to the authors of Corporate Rebels, one of the keys to the success of Micro-Enterprises is setting them up as Open Platforms. In this way, teams can connect both employees and users directly to each other via the Internet and digital tools.

The Micro - Enterprise is the pinnacle of business agility. Successful Micro - Enterprises grow within the overall enterprise, potentially spawning new Micro - Enterprises, unsuccessful ones are allowed to die.

Outcome Team

Another north-star in terms of agility is the idea that a Outcome team is no bigger than a single team. Each team has an independent outcome, and is effectively operating as an entire ecosystem, completely independent from other teams.

Sadly, I have rarely encountered this in my experience of traditional enterprises making the agile journey. Departmental protectionism,

[22] https://www.corporaterebelsbook.com/

dysfunctional back offices, arcane technology, shambling architecture, and a maze of bureaucracy and dependencies all acts as barriers. But this an ideal worth striving for, and there are numerous examples of organizations who have teams that have achieved it, why can't you?

The Program / Portfolio

For organizations who are following a more conservative path to agility, we can take the more traditional concepts of a large scale program, or a portfolio of initiatives and supplement them with an agile mindset. Many enterprise programs/portfolios have more control over how they deliver. They are staffed by a diverse mix from across the organization, outside of the traditional organizational structure. They have a common funding model that is based on achieving larger outcomes than individual projects. There is a healthy backlog of work, and a real challenge in coordinating it.

For these reasons, a program or portfolio can serve as a prime candidate for organizations starting to look for places to stand up a cohesive ecosystem of agile teams, enablers, and travelers.

In Summary

- Collaboration patterns are **reusable solutions** you can use to **define collaborative organizing structure** in a way that is **fit for purpose** while still relying on **experience and good practice** demonstrated in the field. The Patterns are:"
- **Dedicated Team Member** - you are a dedicated member of a classic agile team.
- **Enabler** - you are responsible for enabling multiple teams to use an organizational cross cutting capability, by providing guidance, a platform, relationships, and/or knowledge
- **Community Of Practice** - you are part of a group of people that share passion and knowledge and learning with each other about a certain capability

- **Traveler Pool Member** - you are part of a pool of people that travel to work closely with teams who have an ad-hoc need for your highly specialized and often difficult to learn capabilities
- **Service Provider** - you service the need of other people in the organization by fulfilling an order request, largely in isolation from the person making the request.
- **Dynamic Team Member** - You form into a team temporarily, to achieve a short term goal
- **Ecosystem** - you align a group of teams, pools and Enablers to a common overarching outcome, sharing a common backlog of work, and/or solution platform.

Defining Jobs and Roles in an Agile Organization

Traditional jobs and roles suck

Let's face it, most conversations we in have in the workplace about, job positions, roles, accountability, etc tend to be pretty depressing.

We get adversarial style discussions of accountability and expectation. We get told what we can and cannot do. We get hard boundaries.

Clear Job descriptions, distinct roles, positional expectations are all artifacts that are supposed to increase alignment within an organization. Everyone gets a clear and specific instructions, and everyone knows what to expect of everyone else. Sound great doesn't it?

Not to most people I talk to. Often when I ask people of what they think about their job, role and position in their organization, I hear a common refrain. Rigidity, constraint, blame, lack of empowerment. Why is this?

You are your Job and nothing but your job

In most organization you will get into a job. A job that has a very specific definition. The job definition is used to describe :

- the role you play (typically one role)
- the accountability you have

- the (very specific) boundaries of responsibility you can play in
- how you progress to the next level of your career
- the level and number of people that report into you

- your functional capability and what things you are allowed to do

This documentation will be largely static, and changed infrequently

Let's take an examples:

A business analyst is responsible for working with both business and technology to make sure requirements are clear, accurate, and well understood. The analyst will facilitate various requirements collection works shops, and is accountable for eliciting business needs for all stakeholders. The analyst documents these requirements and reviews them with Developers and Testers. The analysts reports into the senior analyst, and the analyst

The rise of the organizational undead

This type of job description put's the focus on individual effort rather than than the interactions across individuals. It starts and ends with an individual's functional duties. In the modern age, we want people who can play more than one role, playing to score rather than just playing to a position. When we frame people job's in an industrial matter, we put roadblocks in their way that interfere with them stepping outside of their job function. People have a reduced opportunity to provide value, and a reduced opportunity to learn how to add value.

Using highly specialized individuals implies lots of hand offs across members in a team. Work tends to be performed in a highly serialized manner, with individuals working in mini silos. These hand

offs create waste, and impede learning. We face constant bottlenecks.

Job definitions like these also seem to be perpetually out of date. You are guaranteed your job title is never in synch with you are doing now, or what you should be doing now. People are always asking for permission to get something done, permission they often don't get, after awhile people stop caring and stop asking.

Now some will say bad job and role definitions are no big deal, serious people just ignore that stuff right? In many cases yes, but wouldn't a different approach to jobs and roles that removes a lot of unnecessary nonsense?

The Sports Team: A Metaphor for Thinking About Jobs

wheee! I'm creating value! (eventually)

6 year old soccer

Who out there has watched their kids play a team sport at a young age. It is certainly a sight to behold, and it is a lot of fun. Some other truths..

- No one knows how/when to participate
- No one is taking accountability outside of coaches and referees
- Everyone is interfering with each other
- It's well suited to small groups of adults
- Teams do score! (eventually)

How many of us have felt like we were working in an organizational *six year old soccer game?* Some evidence you are in a 6 year old soccer game include:

- everyone shows up to every meeting
- everyone is consulted on every decision
- people come and go at random
- lots of overlapping and conflicting effort
- no ownership of outcomes

Now this is what I call expert value creation...

Table Top

Who out there worked for an organization where management got serious about putting some rigor back into the organization? Likely you ended up with…

- Individuals ending up with very specific job functions accountable to a very specific task

- A much tighter org hierarchy, with well though out spans of control at every level
- Lots of documented processes, that no one really understands,or follows
- No one is taking ownership of the end goal, outside of the most senior leaders
- jobs and roles are hard coded, no one changes their role
- The puck is constantly getting stuck in the corner, metaphorically got between the space of peoples job description, so overburdened leaders have to pick it up

How many of us have felt like we were working in an organization where we felt like a table top piece? Some evidence you are in a tabletop game include:

- you are told not do do something because it is someone else's job
- no one takes initiative to improve things, because it is the responsibility of some other department
- you need approvals from higher up the hierarchy to get anything done
- you get things done by flying under the radar, and begging for forgiveness rather than asking for permission
- lots of departmental protectionism, blamestorming,

You mean we can succeed and have fun?

True Team

Watching a professional team, or even a team of committed amateurs is an entirely different experience.

- Team members have positions, but position changed based on the play
- The team outcomes trumps any concept of role or position
- Interactions between team members trump individual accountability

Most of us actually have solid experience working in this type of organization, even if it's an organization that is outside the workspace. Some evidence you are on a True Team include:

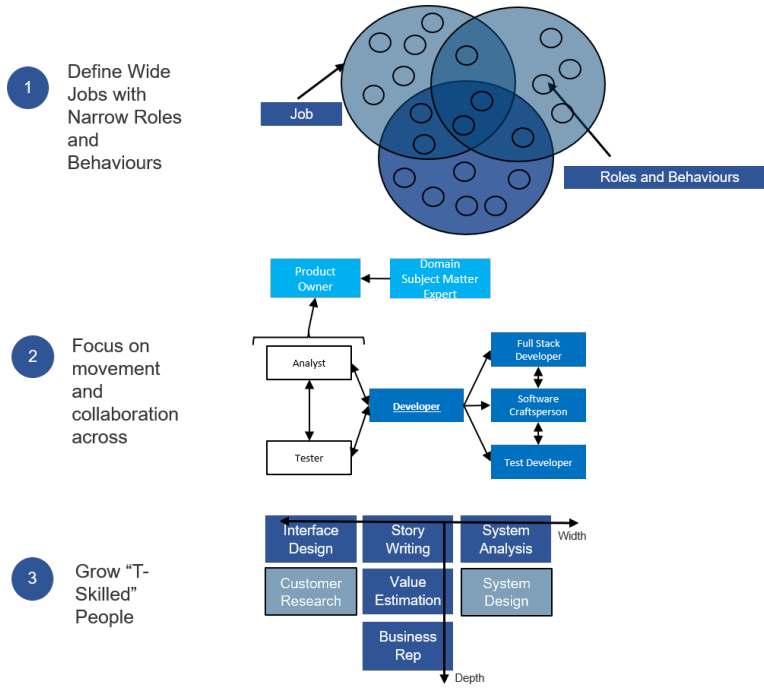
- you don't check if something is in your job description before helping

- you proactively seek advice and help from people more knowledgeable than you before trying something new
- you focus on getting to the outcome, not creating deliverables
- Specialists and specialization exists, but specialists train and mentor others on the team and level them up
- Leaders step in to help when required and get out of the way the rest of the time

People Play Many Different Roles

In our new model we will look at jobs and roles differently, using

- Small number of wide jobs definitions, made up of lots of narrow roles
- Roles describe though team oriented behaviors and movement across roles
- competency model based on T-skilled (Helicopter-skilled!) people



a job model for a dynamic world

In the new organization we don't just want cross-functional teams, ie a team of diverse individuals. We want cross functional *individuals* inside of our cross functional teams. The trick is that as people become more senior we want them to gain deep knowledge in one or two disciplines while at the same time expanding their expertise at a reasonable level across a number of other knowledge areas.

We want domain experts to get cursory knowledge outside their core domain of expertise.

We want delivery professionals such as a tester or an analysts to gain experience across the entire life cycle.

We want developers to be able to contribute code across the full stack.

We want product people to work in operations, and operational folks to learn how to define product features.

In other words, we want to encourage the development of *T-Shaped People*. Your knowledge literally expands deep and wide, like a T. Perhaps a better description is *helicopter shaped people*. We want people to go wide in a number of directions. Your depth of expertise still matters, but it's real value comes from educating others on how to help apply it. Mastery is really about pairing, mentoring, and teaching others to apply your domain.

Less Job Definitions

Let's start thinking about helicopter shaped people by having less job definitions than we ordinarily see in an organization, a lot less job definitions. And let's make those *jobs expansive, with lots of options* to different kinds of work. Taken this principle to it's extreme and we have organizations that only have one Job definition, *employee*. Valve , a gaming company, has no official job or roles defined. You are hired because they trust you to add value based on your skills and experience, and they trust you to figure out how to work with others to do so.

A more gentle approach may be to have a few job titles, perhaps 3,4, or 5 max, Each job would have a very wide scope of responsibility, within an over arching functional category.

More Roles that are Optional and Fluid

Jobs can be refined by the potential roles a person could fill as part of that job. These role often end up being fairly fined grained. An example job could be Business Expert. Roles in the Business Expert Job could include Experience Designer, Business Analysts, Functional Tester, Product Owner, Research and Analytics, Design Thinker.

We can add clarity to these roles by fleshing them out using *personas,* an idea taken from User Experience Design. We can focus our personas on team dynamics and interactions with others. Personas will be used to describe behaviors, behaviors that overlap with other roles. Finally we will empower teams to define and refine the roles and personas required as needed to meet the context of the team. Dynamism will trump consistency.

So let's take a look at our Analyst example again and see if we can describe the job a little bit differently. First of all we will change the job title to something a lot more broad, lets call the job *Business Specialist.*

A business specialist brings ownership, structured thinking, facilitation, communication, and a focus on business objectives to the team. The business specialist collaborates with business stakeholders to validate that the work of the team is accomplishing business objectives. The business specialist will work closely with other team members to clarify and validate work meets business needs. The business specialist will often engage closely with real users to gather insight and feedback.

The business specialist can play the role of an analyst (documenting required functionality) , tester (validating functionality), domain subject matter expert (providing expertise on functionality) or product owner (deciding on required functionality), as befits their experience and knowledge. Depending on their aptitude and interest a specialist may start to focus on user experience or facilitating customer insight as a design thinker. The entire team is expected to collaborate with the specialist to collectively determine the roles specialist can take on that will best serves the team's goal's.

Again there are a few concrete things that make this description more useful to an Agile Organization

- Jobs are containers for many fined grained roles that a person has the option to play

- Jobs and role descriptions provide ideas for ways people can collaborate and interact with each other
- Focused on team outcomes
- Are dynamic, and expected to change based on the make up and need of the team

Stated fully,

Organizing Constraint 5: Design Job specification so that people can play many different roles

Example - A Typical Tech Product Organization

Let's illustrate this concept by providing an example of what a job / role family could look like for a technology based product company, in the hopes that it will inspire you to look at ways with augmenting this in your organization. Take a large organization who that provides software enabled solutions to their end users, this could be a bank, an online retailer, or an organization that ships software to end users.

Three jobs

Let's limit ourselves to three jobs.

- **Engineering** people that perform the creation and operations of our technology solutions.
- **Product** people focused on achieving market success with our technology solutions, including supporting customers
- **Delivery** people that assist and facilitate the delivery and operations of our technology software.

That's it. Every one fits into those three jobs. Every possible permutation of what people can do to create value is slotted into one these jobs. We empower people to flesh out the details in terms of level, behaviours, roles, etc.. But official job titles are limited to these three.

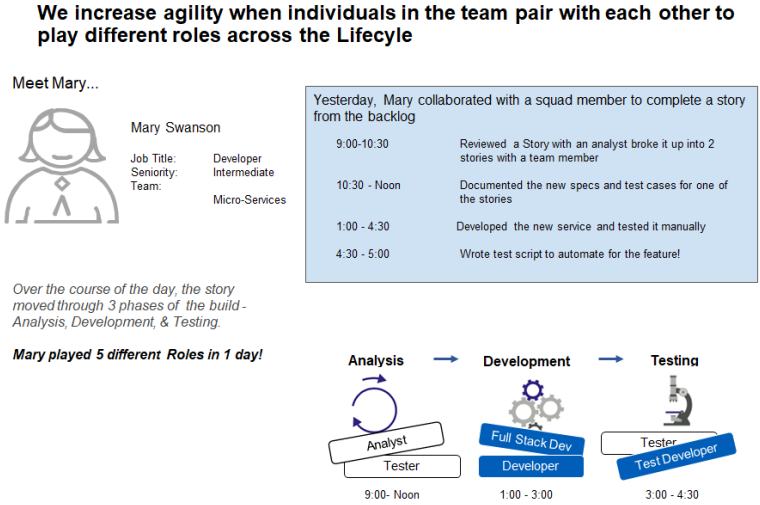
Remember this is an illustrative example only, and not some normative solution that is assumed to "correct" for your technology business. Many product focused organizations would be more comfortable with only two jobs, product and engineering, with delivery responsibilities being allocated across these two jobs. I have also seen another variant that pulls out Operations into its own distinct pillar. Depending on the domain, you may have domain expertise that is so specialized that you create a separate job for it, ie credit risk or actuary.

Many Behaviors and Roles

All of the different things people can do are captured as *behaviors* attached to *roles*. We can capture role behavior if we like into a set of *role personas*.

It's important to note that roles must be articulated in a way that show cases how people collaborate as part of a team, roles are not used to *mandate static boundaries* of responsibility.

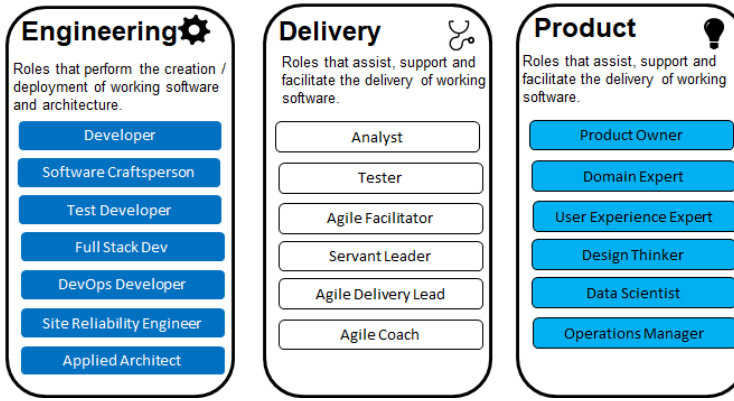
Roles and their behaviors for a team are best identified by the people who are part of that team, as required behavior can and will change based on the emerging needs of that team. There also can be value in sharing a common job/role model across an organization, it is critical that individual teams be able to tweak, and even up-end the the model as necessary to come up with the set of jobs, roles, and behaviors that fit their context.



I can do it, I really can...

We also want teams to self-organize to determine what team member plays which role. None of this is a one and done discussion, but something that is often revisited as part of team level planning and review sessions.

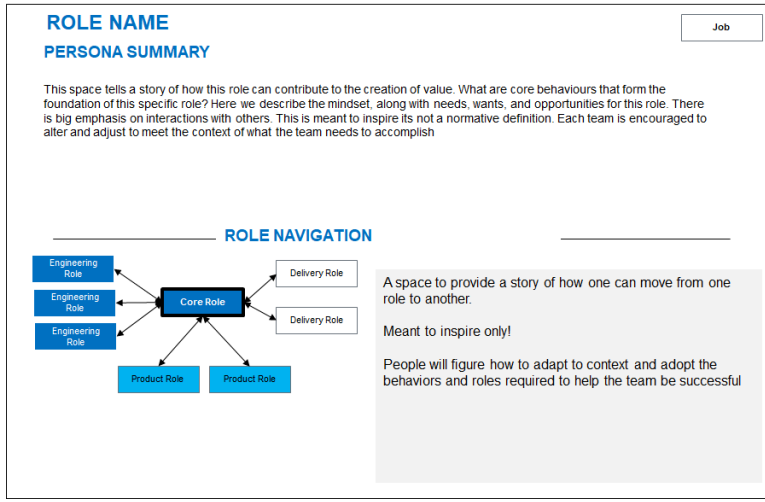
Here are a set of candidate roles that *could* serve as a reasonable starting point for the jobs in our hypothetical example.



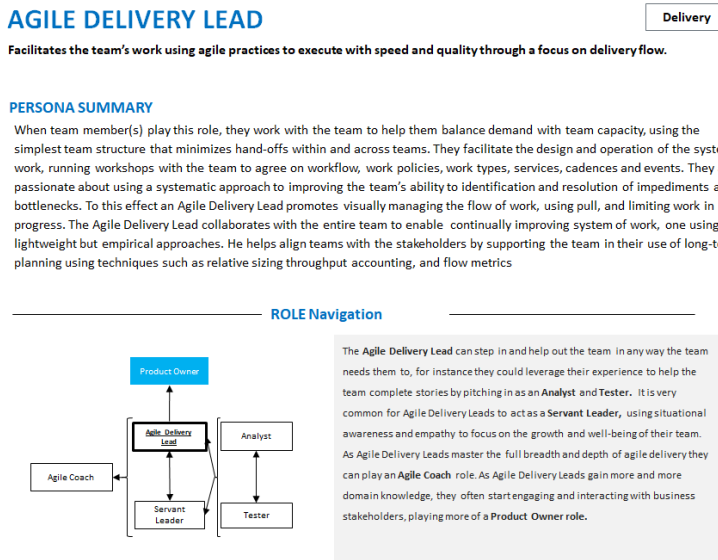
Role Personas

Borrowing a page from the User Experience world, one approach we can borrow is the use of personas, specifically *Role Personas*. We will describe personas through *stories*. Stories of how this role collaborates with others. Stories of how one could decide to take on this role. Stories of how to move to other roles.

These personas can serve as a good dialogue tool. Here is a template of a role persona, if you find the idea helpful, consider working with your organization / team to come up with your own template. Don't to get too lost in documenting all kinds of elaborate role personas. Treat this as an approach to help teams *across an organization to gain clarity* on the activity needed for them to be successful. If you or they have a different approach please use it!



Below is an example role persona I worked on with a client organization. It contains some pretty specific behaviors grounded in Kanban and lean thinking. This made sense for their context and where the organization wanted to go, again your needs will be different.



AGILE DELIVERY LEAD

Delivery

Facilitates the team's work using agile practices to execute with speed and quality through a focus on delivery flow.

PERSONA SUMMARY

When team member(s) play this role, they work with the team to help them balance demand with team capacity, using the simplest team structure that minimizes hand-offs within and across teams. They facilitate the design and operation of the system of work, running workshops with the team to agree on workflow, work policies, work types, services, cadences and events. They are passionate about using a systematic approach to improving the team's ability to identification and resolution of impediments and bottlenecks. To this effect an Agile Delivery Lead promotes visually managing the flow of work, using pull, and limiting work in progress. The Agile Delivery Lead collaborates with the entire team to enable continually improving system of work, one using lightweight but empirical approaches. He helps align teams with the stakeholders by supporting the team in their use of long-term planning using techniques such as relative sizing throughput accounting, and flow metrics

ROLE Navigation

The **Agile Delivery Lead** can step in and help out the team in any way the team needs them to, for instance they could leverage their experience to help the team complete stories by pitching in as an **Analyst** and **Tester.** It is very common for Agile Delivery Leads to act as a **Servant Leader,** using situational awareness and empathy to focus on the growth and well-being of their team. As Agile Delivery Leads master the full breadth and depth of agile delivery they can play an **Agile Coach** role. As Agile Delivery Leads gain more and more domain knowledge, they often start engaging and interacting with business stakeholders, playing more of a **Product Owner role.**

A couple of important points. This Role Persona does not describe what the role does from the perspective of an independent individual. Almost all behavior is described in terms of *collaborations with others*. This is important. If we can't describe jobs and roles in terms of our interactions, than we should not write them in the first place.

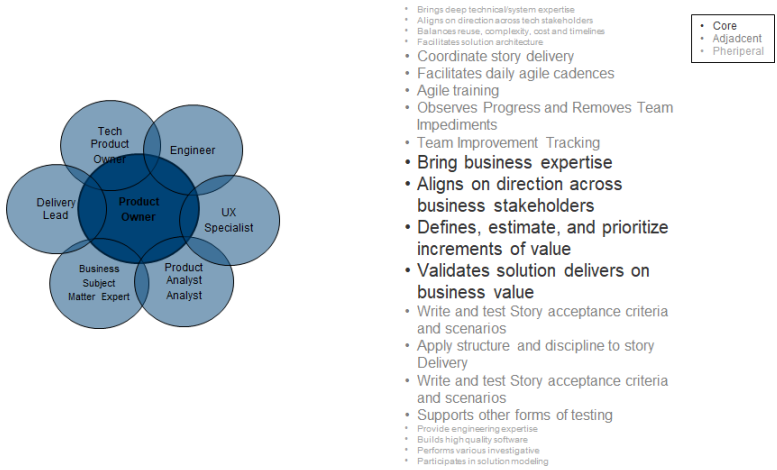
Also take note of the *role navigation* section. We want many people to play many roles! Taking the time to describe a role does not mean we want a person to play only one role, or that we want a role to be only played by one person.

Exploring behavioral adjacency across roles

One reason I have used these persona cards in the past, is to help organizations gradually step away from their addiction to defined roles and jobs with strict boundaries. This seems to be typical, even in organizations that are attempting to adopt agile. Agile methods like scrum and safe don't help, they get pretty prescriptive on the idea of the sanctity of certain roles like product owners and scrum

masters. Agility comes from role fluidity, sometimes we need to approach this fluidity in an incremental way.

Another technique to try if you are in an organization that is still stuck in a fixed job with hard boundaries is to encourage team members to refine their job and role description by listing what behaviors are exclusive to their job, what behaviors they can share with others, and what behaviors they can do in a pinch if needed. Try to move as many behaviors as you can into the adjacent bucket. Keep revisiting until the team becomes more comfortable working with a model where accountability is a team decision that is made based on the latest needs of the team.



DefiningJobsandRolesinanAgileOrganization/Untitled11.png

Above all any approach we use to clarifying roles and jobs needs to be done to encourage people to play more roles not less! It is ok and in fact recommended to treat this type of change incrementally. We don't need to go from the traditional rigid job descriptions to no job description all in one go.

In Summary

- Jobs in the traditional organization are *over specialized, specified*, and are a *poor fit for the collaboration and dynamism* required to succeed in an uncertain world
- Think of jobs like a *professional sport team,* we need both *expertise* and *pinch hitting* in order to succeed!
- Design Jobs so that we can encourage people to *play many different roles,* defined around behaviors tha*t focus on collaborating* with other roles to achieve valuable outcomes
- Moving to a more dynamic job model can be done *incrementally*

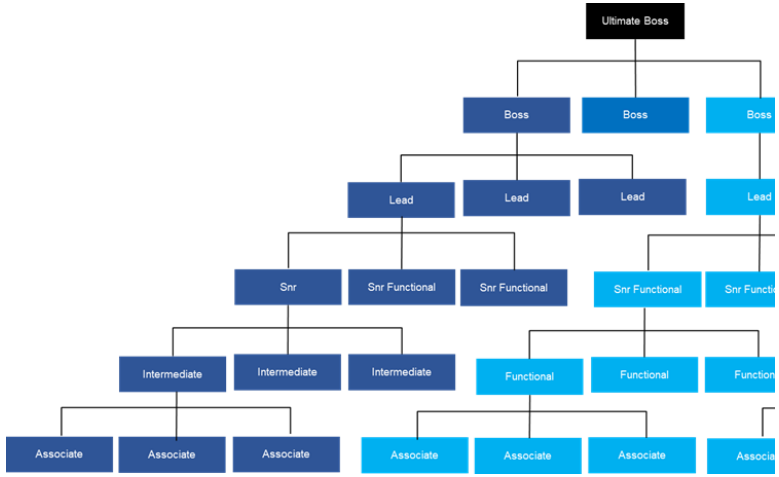
Reimagining The Hierarchy

The Industrial Hierarchy Is Broken

As I have already expressed in my book, I think reporting structure is also pretty broken.

Performance, recognition of expertise and seniority in the organization are all expressed through the number of people allocated to senior person. This leads to internal protectionism, in fighting, an inability to move people to where the value is, siloing, and people not ready for management being promoted into the role.

It also caused a lot of delay in decision making, as orders went down the hierarchy, and information cascaded back up. Decisions also get distorted and information degraded as travels up and down the hierarchy.



deep into the hierarchy we descend...

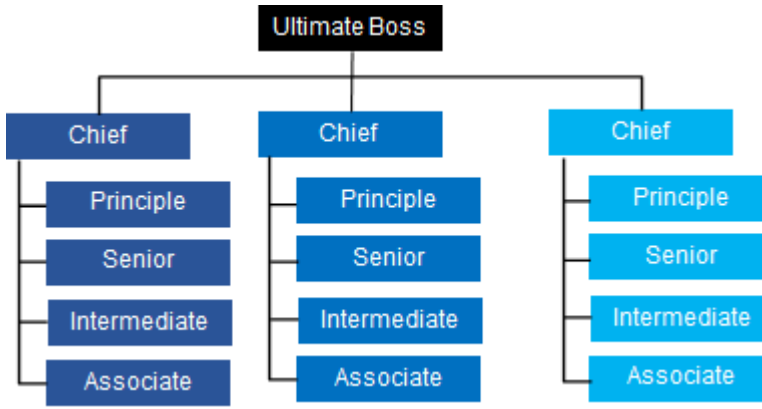
The concept of seniority does not go away in Agile Organizations. I know a number of people in the agile community will disagree with me on this. They will say that there is no concept of seniority in an agile team. That the team simply self organizes to come to an agreement. I have not seen this work without *some* official recognition of seniority. At least I have seen a need for official seniority in large organizations that are have started there agile transitions within the last decade. Maybe we can get rid of all forms of seniority at some point in the future, I'll keep an open mind.

Seniority Equals Mastery Not Authority

I think a pragmatic and practical approach for organizations starting a structural shift to agile is to decouple seniority from reporting structure. Rather, think of seniority as a *ladder of expertise* and competence*, often known as mastery. The top of the ladder is responsible for capability, community and competence within an

area of expertise. Everyone else is placed on the ladder according to experience, expertise and contribution. But everyone in the ladder reports to the top of the ladder, there is no other reporting structure!

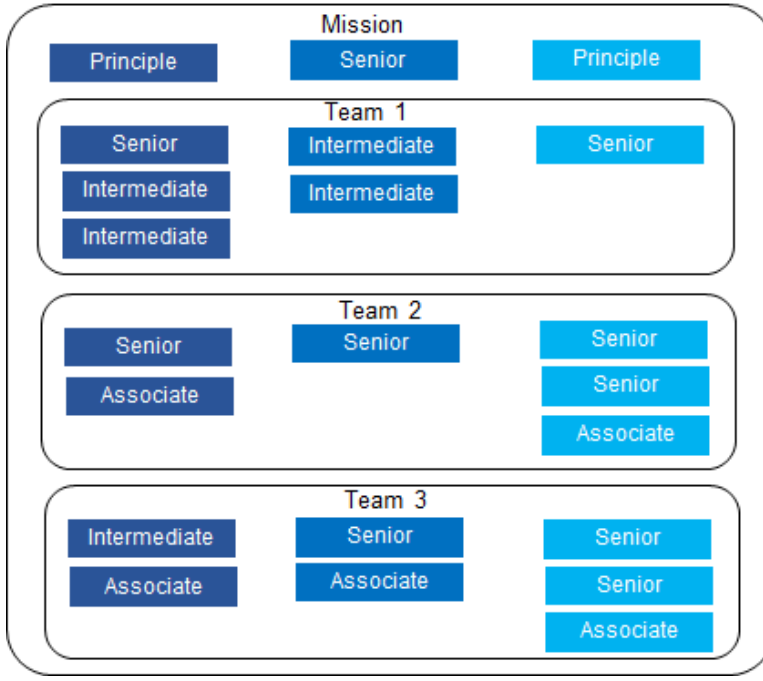
Taking our hypothetical example from last chapter we have the following ladders, with a shallow, limited hierarchy at the top.



shallow hierarchy and ladders of mastery

It is important to emphasize that the ladder is not used dole out tasks or assign work! Work allocation is managed by placing people into teams and possibly larger context boundaries, such as a portfolio, and asking those teams to self organize.

People with seniority may be placed in positions where they support more than one team, ie a mission or portfolio, but this is not strictly tied to a level in the organizational hierarchy. In this model a senior may shift from supporting a larger, but more stable portfolio of work, to directly contributing to a team level outcome that is more uncertain and complex.



delivery structure is decoupled from functional position

Note that self organization does not imply anarchy or even democracy! Within a team we expect more junior people to seek the advice of more senior people. Senior people will sometimes have to make a call where agreement can't be reached. Of course if senior people are constantly overriding more junior people than we know we have a deeper problem to resolve. It is the responsibility of senior people to both model and coach self-organization of team! This is not an easy thing to do, but it is critical if we are to avoid the problems associated with both implicit and explicit power structures.

Title can still be used to reflect seniority

I think a lot people confuse a lack of reporting structure with an absence of positional seniority. People fear that organizational

anarchy will result, with juniors not listening to seniors. My experience is you can remove most/all reporting structure and create a culture of advice / respect where more junior people can and will seek the advice of more senior people. My experience at Deloitte, a large and established consultancy was that the ladder approach can be very effective. Title was still be used to reflect seniority, but senior people did not have anyone officially reporting to them with the exception of partners that ran entire service lines. Reporting structure wasn't needed, there was a culture in place instead. Were there issues? Yes, and paradoxically not enough of them. The issue we struggled with was senior people's opinion still carried to much weight, and senior people had a tendency to roll over the opinion of people newer to the firm with less experience. Powers structures still formed in the absence of an official hierarchy, they were just a little more implicit. We will talk about this a little more later.

So, stated fully:

Organizing Constraint 6: Use seniority of position to reflect mastery of applying skill set to value creation, do not use seniority of position to denote authority over other people

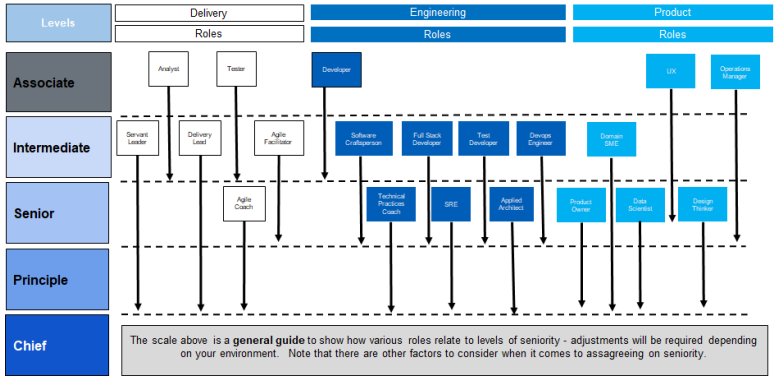
Decouple Seniority From Role and Job

If your organization uses a ladder, avoid having a different ladder for every role / job in your organization. If you can get away with it, use a single ladder description. Use generic descriptions of attributes like scope, expertise, and numbers of hats a person can play, and use this to provide a common understanding of what it means to advance the ladder. Above all keep it simple and concise. Helping an individual achieve positional seniority is helped through effective dialogue between people, not by a comprehensive capability framework! Here is an example from another client I worked with to describe their levels in their organizational ladder.

	Role Breadth	Domain Breadth	Level of Mastery	Scope of Responsibility
Associate	Demonstrates many behaviours that are "associate level" role (eg: Analyst / Developer)	Knowledgeable in one area of a system / application (ie module)	Working knowledge of core area of expertise Able to accomplish simple tasks independently and complex task with supervision Learns new skills / capabilities as required to complete tasks	Responsible for individual work items and contributing to squad outcomes
Intermediate	Demonstrates most of the behaviours across multiple associate roles and actively demonstrates some behaviours associated with 1-2 "intermediate roles"	Advance knowledge in multiple system /application areas of increasing complexity Working knowledge of an entire application / system	Deep knowledge and experience in core area of expertise Can accomplish more complex goals independently and able to work with increased uncertainty	Accountable for squad outcomes that require collaboration of numerous squad members
Senior	Demonstrates behaviors across 2-3 "intermediate level" roles and at least one 1 "senior level" role	Has an in depth knowledge of the systems/ applications in which the Squad(s) operates, including adjacent systems/ applications, and including business logic/context	Knowledge expert in core area of expertise. Significant experience in adjacent areas of expertise Actively improves the skills of others through training / coaching/ mentoring	Accountable for the outcomes and continued performance and improvement of 1 - 2 squads
Principle	Demonstrates the majority of behaviors across 2-3 "senior level" roles	Has an in depth knowledge of multiple systems/ applications that span the enterprise	Thought leader in a wide area of expertise with deep experience in adjacent areas Responsible for employees learning and growth for a knowledge area (eg Devops)	Accountable for outcomes across a portfolio / tribe (ie many squads)
Chief	Demonstrates the majority of all behaviors across 2-3 "senior level" roles and many of the behaviors of at least 1 role at the "chief" level	Unsurpassed knowledge in many systems/application/domain areas	Principle expert in multiple but related areas of expertise Oversees the growth and development across multiple knowledge areas.	Accountable for the output, growth, and capability of an entire department / Guild

Keep seniority simple

While some roles can be perceived as being more senior than others, avoid pinning role to exact level. Consider providing only a loose association between the two concepts if you feel you need this specify. Again an example from a client where team members wanted to map roles to seniority within their organization.



decouple role from position as much as you can

The Advice Process

When a model like this works, we end up replacing one monolithic and formal hierarchy with many small informal ones. People are expected to reach out to their team members, to people with expertise, and to people who are impacted, before making a decision. This known as the advice process.

Coined by Frederic Laloux[23], the advice process asks allows you to make any decision you want, provided you seek out advice from the right people first. A positive aspect of the advice process is that authority gets minimized, instead we have influence. Influence that comes from mastery. Influence that is loosely coupled to position. The advice process is a good way to counteract both implicit and explicit power structures, everyone is expected to participate.

Many agile artifacts work well when the people using them realize that they are excellent examples of the advice process at work. Story Maps, Kanbans, Executable Specs, Business Model Canvases are valuable not when you build them, but when you use them to get feedback from many stakeholders, sponsors and peers. They work even better when you get unexpected feedback from a person who is walking by. An early act of leadership is to demonstrate and encourage the use of the advice process to ensure that self-organization does not equal no organization.

Organizing Constraint 7: Implement decision making through the advice process,

In Summary

- The industrial hierarchy *conflates level with authority* to the detriment of value creation in the modern world

[23] https://www.linkedin.com/in/ACoAAAAGHpgBq9Ei3EvrUbRrLB09viMnDinmaJE

- *Flatten hierarchies* to the maximum extent possible, and *use positional laddering* instead
- Use *seniority to denote* mastery not authority
- Leverage the *advice process* to encourage good decision making *without imposing command and control* onto your workforce

Scaling Agile Behavior For Continuous Organization

Truly Stable Teams Erode Agility

Time to revisit our agile sacred cows, this one will for sure get some people's backs up.

Stable teams erode agility.

There, I said it.

This is a strong statement. And I admit I be trolling a bit to get your attention, while I have it I want to ask you to think about what Agile is trying to address.

Uncertainty.

Complexity.

Change.

Agile provides us a means to deal gracefully with a world that is complex and constantly changing, and it does so through a mindset and methods based on using feedback to respond effectively to this constant change. Agile teams constantly shift in terms of what they are delivering and how they are delivering it.

I have seen a lot of organizations do a reasonable job of figuring out there agile team structure yet only succeed in creating a mess of dependencies. Hypothetically, lets say you have analyzed the demand for your organization, mapped it to capability, formed an ideal set of teams around those capabilities, and staffed those teams

with people that possessed those capabilities. You can now sit back and let those teams deliver, right? You're done right?

TruthsMythsandLiesaboutTheAgileTeam/11.jpg

Wrong

Now something changes, perhaps the market has evolved, priority has shifted, our understanding of the work has grown, or the competitive landscape has been altered. We uncovered the need for a new capability. We can no longer feed an outcome or an initiative to a single team within our structure.

We have two options here.

TruthsMythsandLiesaboutTheAgileTeam/12.jpg

The first option is to keep our structure stable.

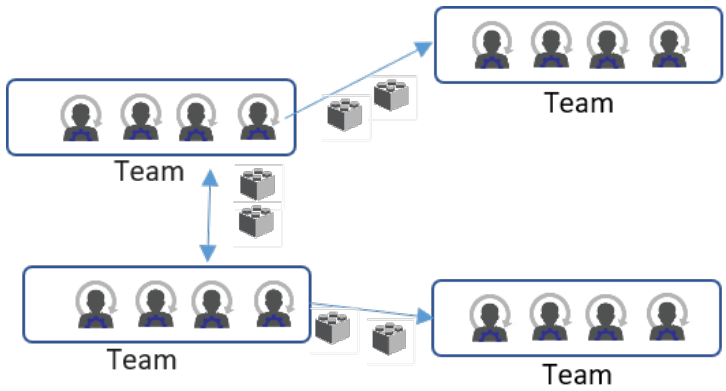
Perhaps some other team has those missing skills we need to deliver value? *Couldn't we send that portion of the work over to the other team?* We could, but now two team need to coordinate to deliver value, which *interferes with each teams ability to independently self organize.*

Maybe a piece of work involves more effort than we thought. *Cant we keep the team the same size and take longer to deliver?* We could, but we may *delay feedback and learning*, and depending on how important that feedback is to the team, we could deliver the wrong thing to the market.

Perhaps we have discovered a new opportunity, unlike anything we have done before as an organization, delivering it would require

collaboration that spans across everyone of our teams. *Couldn't we just coordinate the work across the teams?* We could, but I think you get the idea.

Coordination requires classical, old school management. One that is now informed by agile backlogs, stories, visual flow, etc, but it is still old school management. The more teams become dependent on each other, the more they they will require dedicated people to manage this dependency, and the more they will lose their ability to self organize.



All these dependencies look nice in a diagram, the reality sucks

The reality is that *stability and independence are concepts that contradict each other*. You need stability, absolutely. Without stability people in teams do not have a chance to well, team.

But stable teams are not static team.

So let me rephrase my original sentence

Static Teams erode Agility.

Static teams don't flex when demand changes. Static teams mean you have hand offs across your teams as the cracks in your original

team layout start to show. Static teams erode the independence of teams, and we need teams to have independence if we want self organization inside of teams.

The answer is an an evolutionary approach to designing team structure, conducted frequently

The Need For Continuous Evolution

Because we are working in a world where change is constantly accelerating, our team structure, not matter how well designed, is always going to be wrong.

Sometimes it will be a little bit wrong.

Sometimes it will be a lot wrong.

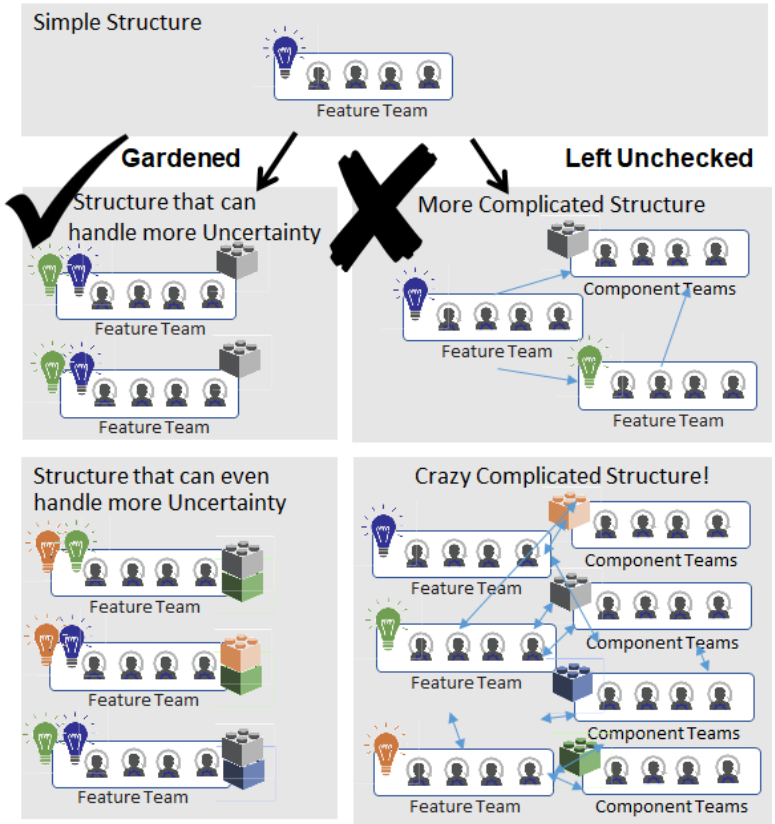
It will never be perfect.

Your team structure will always be catching up to changes in demand feeding it.

Eventually, your teams end up working on the wrong thing, or work will need multiple teams to work on it, or teams can't work on something as they lack the skills.

I have seen a nicely designed ecosystem of agile teams, left unchecked, devolve over time into a mess of dependencies in a matter of months. No team could deliver any item of value on their own, no team could own any particular outcome.

The answer is to continually *garden* team structure with an eye to adapting the simplest structure to meet the needs of the environment.

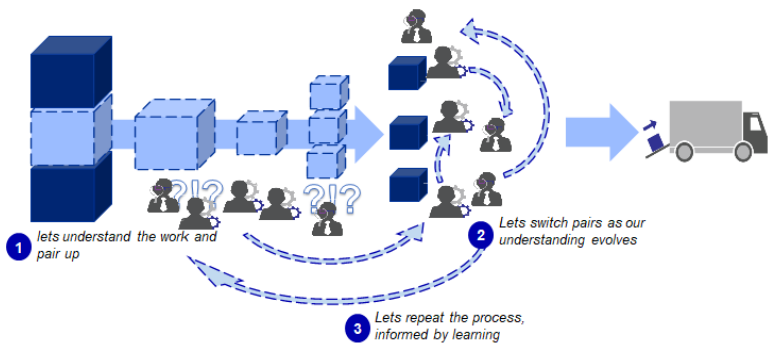


The good news is that people with agile experience already have a lot of experience in gardening the way they are organized, they are just more familiar with gardening at a smaller scale than what we are talking about now.

Agile Enables Teams To Organize

Continuous organization is a key part of Agile. Lets look at a day/week in the life of a stereotypical agile team.

- A subset of the team collaborates with stakeholders to shape the work so that it is ready for the team to work on, defining acceptance criteria, fleshing out details, decomposing it. etc.
- Weekly or even daily, the team collaborates to prioritize the next set of work to accomplish, creating a backlog for the team. The team will often divide into smaller groups, to "pair off" against individual items in the backlog.
- The team then meets frequently, perhaps daily, perhaps more often, to agree on how to do the work, and who does the work. Pairs are often switched as needed based on availability, capability, knowledge sharing, or pinch-hitting to help each other out.
- At the end of the cycle the team reviews what was accomplished with each other and stakeholders, they will take time to reflect on how they could have planned and delivered differently in an attempt to improve the way they work.

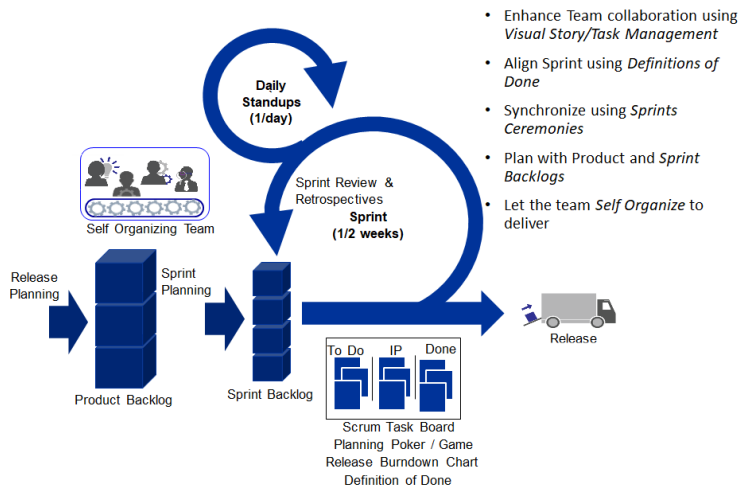


agile is a constant act of organization

Agile Behaviors of a Self-organizing Team

Within a team we see members re-configuring dynamically into twos and threes and fours as necessary to accomplish the team's overall goals. The following agile artifacts and practices help team members continually reorganize to meet the demands of the day:

- **Team Planning and Sprint Backlogs** allow a team and stakeholders to get organized around the highest priorities
- **Team Stand-ups and Reviews** provide a means for team members to reorganize around the work using the latest learning
- **Visual Work Management and Definitions of Done** make it clear how the team wants to work and how they are actually working, this makes it easier to understand when team members need help from each other



Armed with these, or similar practices, team members are continuously organizing, and re-organizing how they work together. Team members *self-organize* in the most effective way that can accomplish the goals of the team.

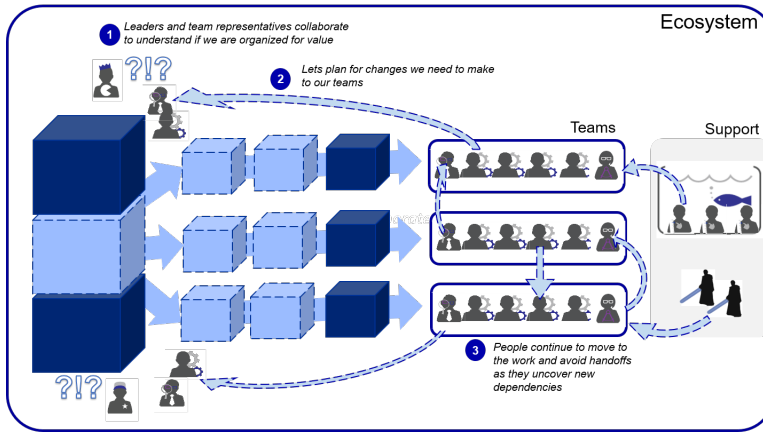
Scaling Agile Enable Larger Groups To Re-Organize

The concept of organizing and re-organizing around value is something that scales beyond a single team. Personally I have seen these organizing behaviors scale up to almost 200 people. That being said, it is better with a smaller group, say 30 - 50 people. This is the number of people that would fit into a medium sized *Agile Ecosystem,* enough people to deliver value for a product/ product family, customer/market outcomes, or organizational objective. Assuming that the teams within this ecosystem have spent enough time working with the agile concepts mentioned above, than we are ready to scale that behavior out to the overall ecosystem.

A day in the life for an *ecosystem of teams* could look like this

- ecosystem leadership and team representatives collaborate with each other and stakeholders to prioritize and shape the next set of strategic outcomes to be delivered to the market
- This group meets monthly, or more frequently to conduct cursory analysis on these high priority items, enough to understand what team(s) could deliver on it, as well as an order of magnitude effort to deliver
- Items are placed onto a cross team backlog, based on priority and team delivery capacity to deliver, team structure, including number of teams, team composition, and team size are evaluated and adjusted based on the context of this new demand

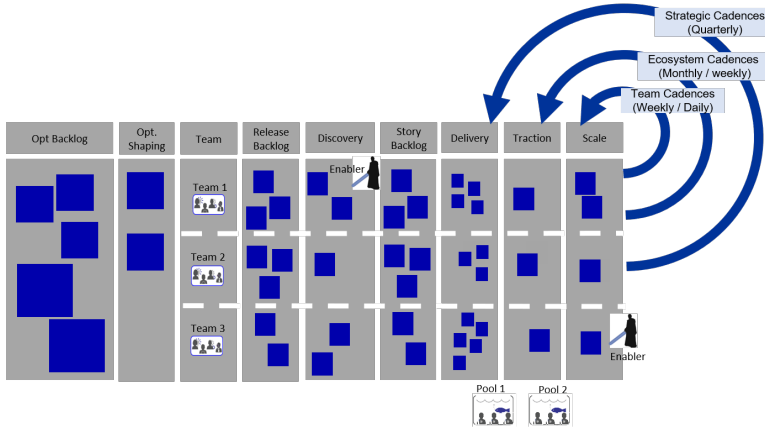
- leadership and other team representatives meet frequently, perhaps once or twice a week. They discuss how the current structure is working, they handle cross team dependencies, they discuss any changes in priority, and review other issues that could impact multiple teams. At any time team members may elect to travel to other teams or otherwise adjust team structure based on the latest learning
- At the end of the month or quarter both leaders and representatives from team reviews what outcomes were delivered to the market, and reflect on how on the ecosystem of teams could have planned and delivered differently



Agile Behavior For Re-Teaming

Within an ecosystem, we see teams dynamically re-configuring as necessary to accomplish the ecosystem's over arching highest priority outcomes. By revising some the team level agile artifacts and practices mentioned above, ecosystem members can continually *self - form* into teams to achieve outcomes.

- **Long Term Planning and Graduated Backlogs** allow multiple teams and organizational stakeholders to get organized around a set of overarching outcomes
- **Cadences aimed at Multiple Levels of Feedback** provide a means for members of multiple teams and support structure to reorganize around outcomes using the latest learning
- **Visual Flow Management of End to End Value Creation and Knowledge Worker Agreements (Policies)** by making it clear how work flows across the entire system of work, both within and across team and illustrating impediments and bottlenecks in a way that people to swarm around the work transcending team structure



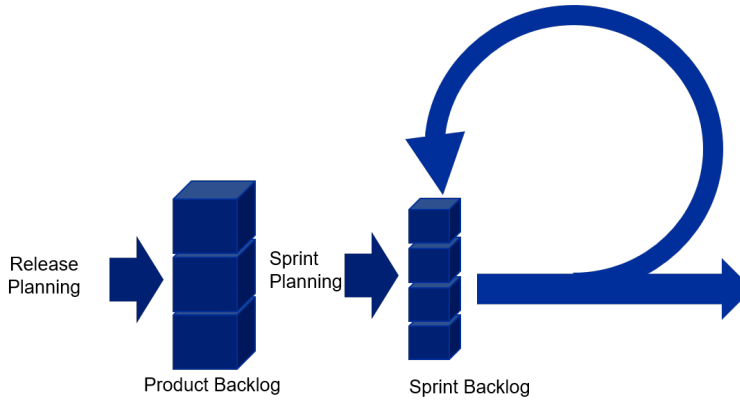
Facilitating Continuous Organization With Agile Practices

Forecasting Change With Agile Long Term Planning

Let's start by scaling *Team Sprint Planning* to *Agile Long term Planning*. Agile Long Term Planning is the act of applying team level sprint / team level planning concepts to a larger scale.

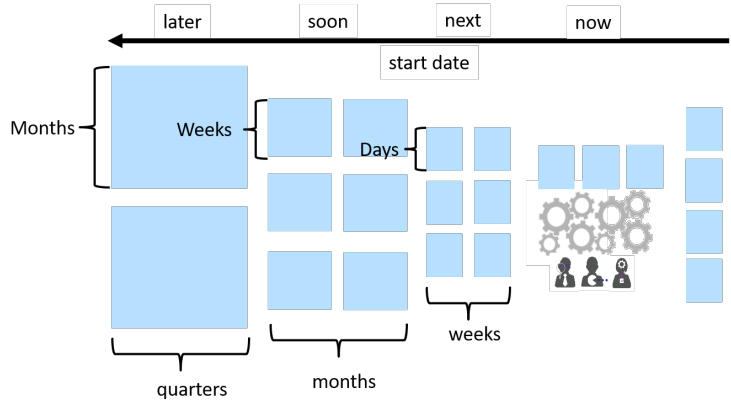
When we think of team level planning we often think of teams that:

- conducting release planning where Epics are placed into a product backlog
- conducting sprint planning where Epics are decomposed into stories and placed onto a sprint backlog based on team velocity



The Graduated Backlog

Applying planning to a larger scale includes expanding our concept of a backlog to what I call a G*raduated Backlog*. A Graduated Backlog encompasses both a larger *Time Horizon*, from days and weeks to months, and perhaps quarters, as well as a larger scale of *Value Increments,* from individual stories to larger increments; think Epics (Sagas?), Features, Business Value Increments, Minimum Viable Products, etc.

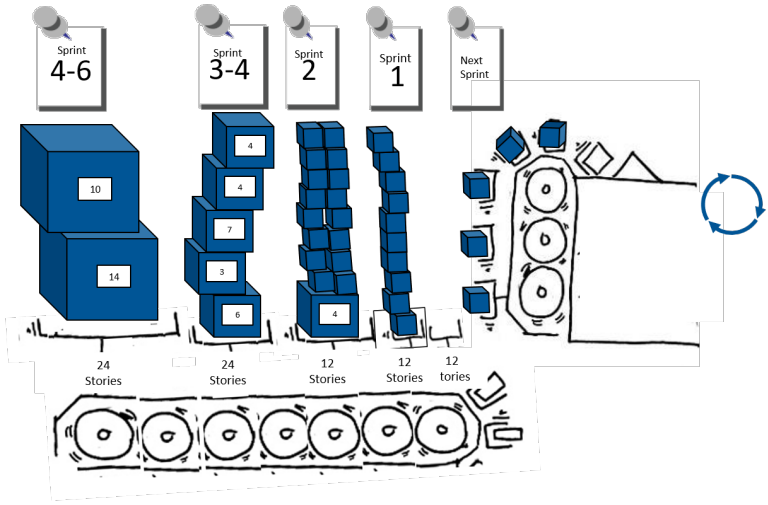


It is important to note that a graduated back log has an order of magnitudes less precision the farther away the work is from being *started*. When we don't expect to start work for a long period of time, we expect it to be poorly understood. As the team completes higher priority items, this work will "move left" as the start date of the work becomes closer to today's date. The team is expected to decompose the work the closer we get to starting it, iteratively breaking Strategies into Outcomes, Outcomes into Business Increments, and Business Increments into Thin Slices, and Thin Slices into Tests. (some teams break work down into tasks). A more agile friendly taxonomy would be Sagas⟶Epics ⟶ Features⟶Stories⟶Tasks, Let teams use what ever words they want to describe how to break things up, getting some alignment on this wording across teams within a single Ecosystem level is helpful but not mandatory.

Use Throughput To determine Where to Place Demand On the Graduated Backlog

But how can we anticipate when the team will be able to start a particular piece of work? We can do this by estimating the teams *throughput*. Agile teams deliver value by breaking larger request into smaller increments often known as *Stories*. The more teams

deliver, the better they can anticipate their likely story throughput over time, for example average stories per month, or average stories per week. This number can be used to place increments of value in a Graduated Backlog, and position according to likely starting week, month, or quarter (remember the farther out we go from today, the less precisely we want to define our start period).



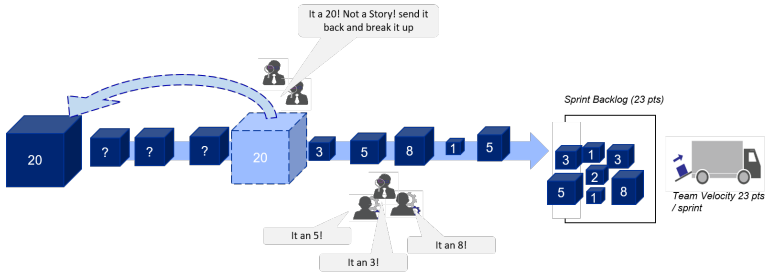
Not all teams have to agree on using stories to capture throughput. As long as the team is able to decompose work into relatively small increments that can be tested / verified for correctness, it doesn't matter what teams call the unit of work used to capture throughout. The key is that the work item can be tested. We want to track items that have gone through some form of quality check, tasks don't count.

What About Velocity

Many of you will ask why not use team *velocity,* a commonly accepted metric for communicating how much agile teams deliver? The short answer is that teams that are passionate about using ve-

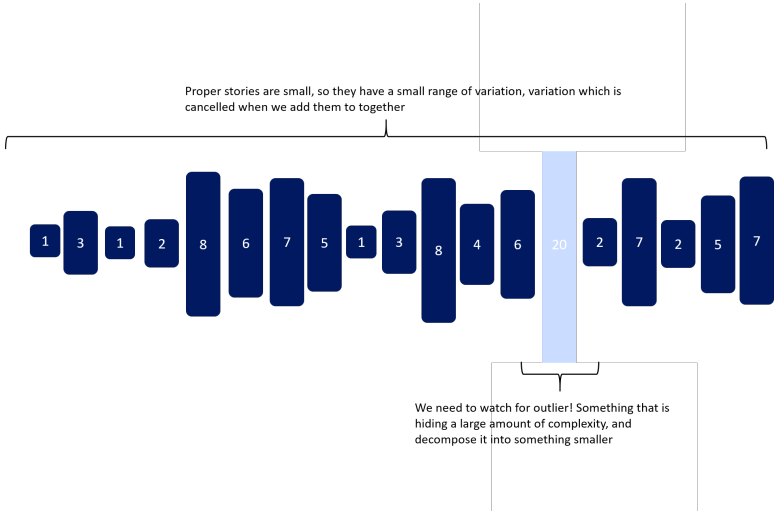
locity should be allowed to do so, but where we can use throughput we should. Why?

Velocity is based on the idea that teams take individual work items, often call stories, and perform relative estimation on them, expressed as a count of *story points*. These story points have no unit of effort in and of themselves. Rather, they are just a means to relatively rank each story in terms of complexity and effort. The team then estimates their typical velocity as the number of story points delivered end to end within a short time period, often a couple of weeks, frequently called a sprint.



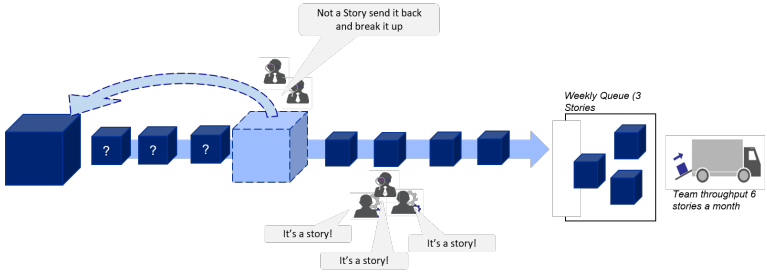
ScalingAgileForContinuousOrganizationalChange/velocity.png

So what is wrong with estimating points and tracking velocity? Well for starters, if we accept that stories are already small, and that the team will deliver quite a few of these in a short period of time, then points serve as a needless extra layer of abstraction. We can make the *exact same observations by simply counting the number of stories* that flow through our system of work.



ScalingAgileForContinuousOrganizationalChange/Story_variation.png

The biggest value of estimating stories in points is that the exercise provides an opportunity for teams ensure that they only work on *stories that are small.* Once a team has accomplished this, the difference between the complexity of one story vs another will be averaged out over time. Since stories are small, *they have a small amount of inherent variation.* What we really want teams to do is *watch out for items in the backlog that are not small.*



ScalingAgileForContinuousOrganizationalChange/throughput1.png

In other words we need to be *careful that our stories are actually stories*, and not masking a much larger piece of work, when this

happens we need to break it up into stories before we start working on it!

Some call this approach *#noestimates*. I disagree a little, I like to call this approach estimating where variation requires you to do so. An emphasis on throughput can save a lot of time and effort from the team, they no longer have to debate small differences in complexity between each story, differences that do not matter if we look at the medium or long term.

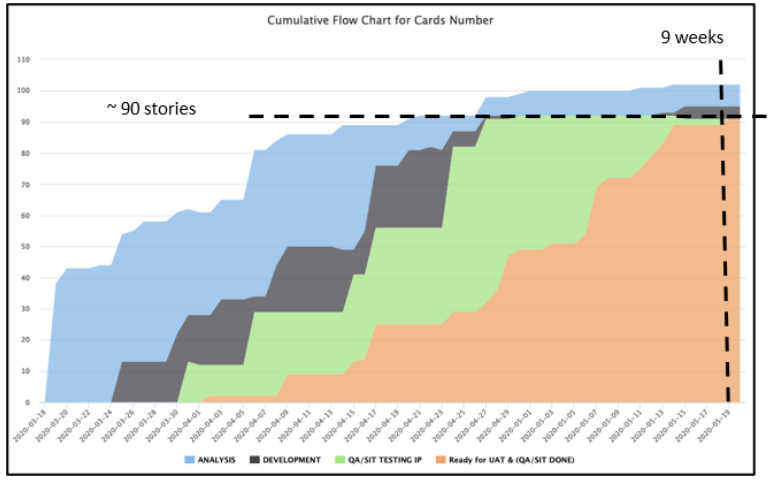
Throughput is also a much more accessible number to people outside the team, most stake holders can be shown what a story tends to mean for a particular team, we can provide them an example of a larger story and a smaller story. It then becomes very easy to express plans and progresses in terms of throughput of stories. How many stories do we think are in this feature? How many stories do we think we can do over time? How many stories have we tested so far?

Throughput allows us to Track End to End Value Creation

Throughput also allows us to look at the progress of value creation across the entire process, from idea to market feedback if we want to. Velocity is limited to tracking progress for the activities inside a very narrow time horizon, While many agile purists claim work can travel the entire value stream i one sprint, this is rarely the case for many organizations. Typically a sprint is just enough time for engineering activities to take place.

Throughput lets us track the progress of items that can take longer than a single sprint to deliver.

Let's take an examples, using flow metrics we have determined the capacity of a team, in this case they have delivered 90 stories over the last 9 weeks. This gives us a team throughput of approximately 10 stories /week or 40 stories /month.



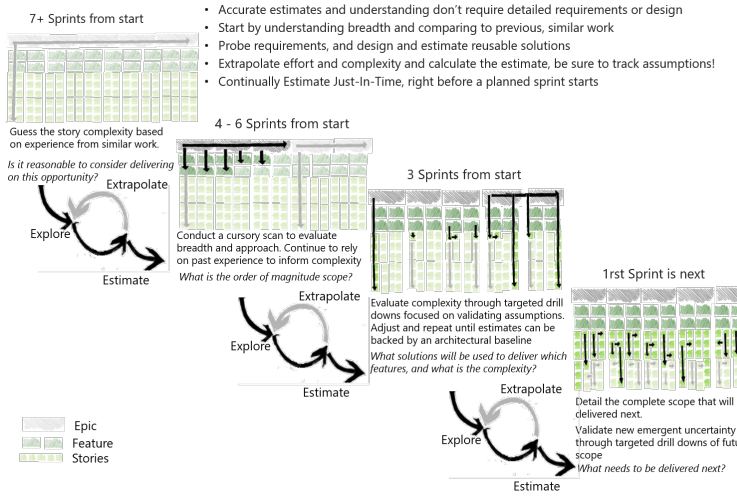
Fill the Graduated Backlog Through Relative Sizing

When a team is good at decomposing their work into small, incremental, testable units, then we can use the team's story throughput as a measure of a teams ability to deliver, and then estimate items in the backlog in terms of *number of stories,* in it. This is where relatively sizing can really shine! We can use it to quickly size new work by asking *how big is this piece of demand compared to something that was previously delivered?* Is it bigger or smaller? By a lot or a little?

I like to relative size visually, by placing the work on vertical axis and encouraging the team to place previous work initiatives based on a complexity scale. Estimating is simply an act of placing new items in the right spot according to relative complexity.

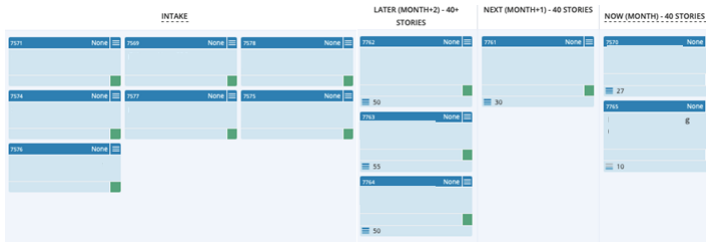
Go Deeper where you Need to, But No Deeper

Initially this is just a swag! We want experience to inform the conversation! Where there are to many unknowns the team will need to spend more time exploring the work using techniques like impact mapping or story mapping. It's important however to not facilitate these exercises with an eye towards doing a complete analysis. The team needs to take the vantage point of an explorer, you want to go wide for the most part, and go deep in order to uncover assumptions that are getting in the way of doing a reasonable, low precision sizing effort.



This relative sizing can be done with successive waves of refinement based on how soon the delivery start date is. Starting at the initiative level, then doing deeper dives to get a better understanding of the smaller increments of value, such as features, thin slices and eventually stories.

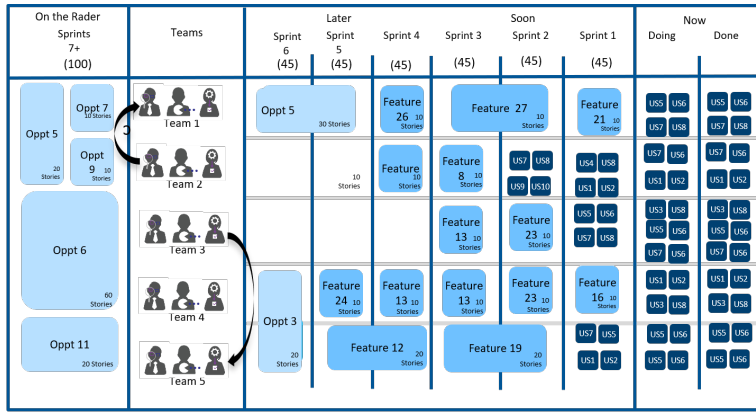
Again, taking our example team, if the team has been demonstrating a throughput of 40 stories a month, then the team could relatively size future work in terms of stories and get an at least reasonable understanding of when they could start a particular piece of work, assuming reasonable prioritization.



Again the point here is not a highly precise model. We are using reasonable approximation *to make imperfect information visible.* We

are providing a space where multiple teams and there stakeholder can collaborate and align on what the future *could* hold, with an eye towards *revising as we uncover new information.*

With a little bit of visualization, we can connect our backlog to multiple teams, say all the teams in an Agile Ecosystem. This allows us to see who is working on what outcome. We can clearly see if one team is overloaded, or if there is less demand for another team. In this way Long Term Planning allows team members to understand what outcomes will be worked on, and allow them to *move between teams based on that understanding.*



This type of view can be facilitated using physical card walls or electronic tools. What is important is that all members of the Ecosystem spend some time looking at it holistically and asking the questions

- are we organized for value?
- should I move to another team?
- should our team be working on a different outcome?
- how can I provide better support to another team?

There are an infinite variation of ways we can get people in an ecosystem to look at this visual of the demand coming in to the

ecosystem, typically we can extend the concept of team cadences to to a larger organizational unit.

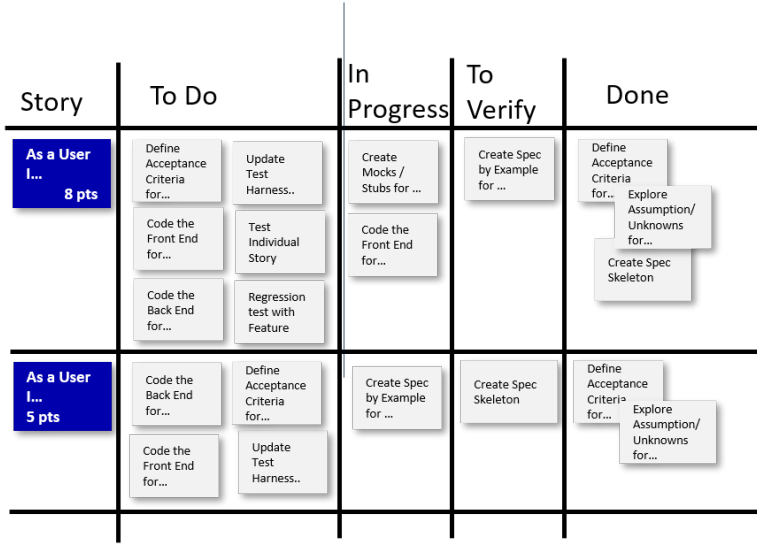
Organizing Around The Work With Kanban

Using Long Term Planning with a Graduated, Visible Backlog is just the beginning in terms of how we can use Agile to help an Ecosystem Of Teams organize around the value, as the complexity of our ecosystem increases, we can look to using Visual Flow Management with Kanban to enrich our team members understanding of the work, it's value, where things are getting stuck, and where to help out.

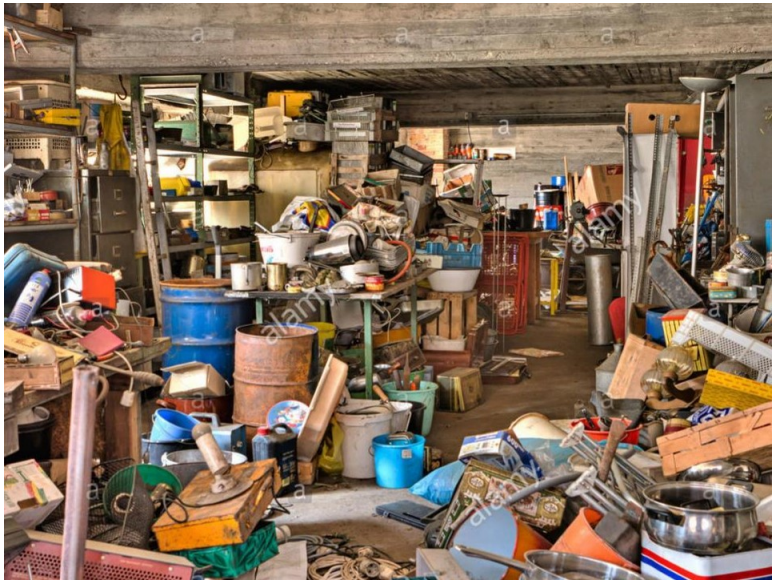
A bit on why Kanban is a nice way to scale out classic agile style visualization

There are many descriptions of Kanban out there that do a much better job of what Kanban is and why it can be so valuable to knowledge workers who need to collaborate in the face of constant change. In stead of repeating much of what has been said elsewhere, I'll discuss how Kanban can help teams take their agility to the next level, one based on end to end value and interacting with other people both within and outside of their teams.

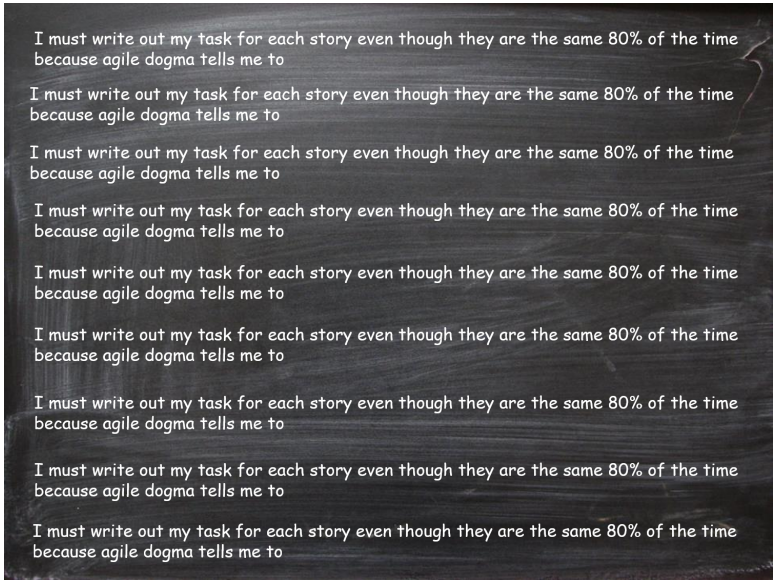
Let's start by describing your stereo-typical agile card wall. Here we can see stories that are broken up into tasks, with tasks going through a fairly simple flow. Once all the tasks are done we can mark the parent Story as done as well.



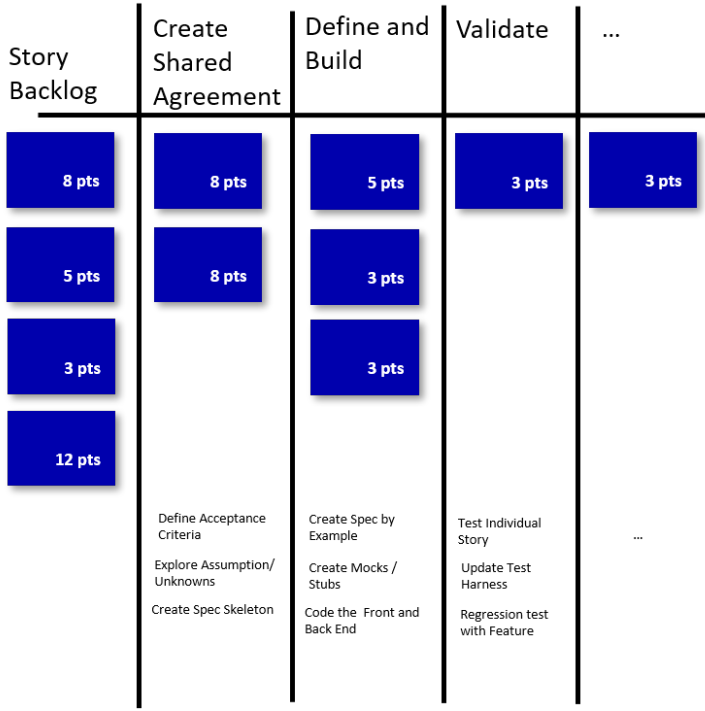
The problem with this form of visualization is that it can get very cluttered, very quickly. If were to look at an Ecosystems worth of work, we would have a swamp of information, and our visualization would not be giving us the information we needed to help people organize outside a very small scale.



What is interesting to note is that most of the information used in this type of visualization is noise, ie the tasks. Notice how the vast majority of tasks on the board are the same, I have witnessed this to be true for the vast majority of teams (we will talk about exception later). At some point writing these tasks out start to make you feel like a child being punished for talking in class.



Let's look at the Kanban approach to visualizing this work. Tasks are ordered according to the team's agreement of a *reasonable* workflow. The team collaborates to write out a set of knowledge worker agreements (Policies in Kanban speak) to create a common understand of the work to be typically done, and even the people that may take the lead in that state. Note that it is expected that this flow and the accompanying work agreements are expected to change, and to change constantly! The idea is not to set rules in stone, rather it is to give the team a lightweight way to continually update how they would like to flow their work.



It will be arguable to some about which form of visualization is better at the team level., at the end of the day teams should choose the style they go with. Where a Kanban style of visualization starts to really shine is when you start thinking about scaling your visualization to encompass a larger portion of value creation activities.

Scaling Horizontally

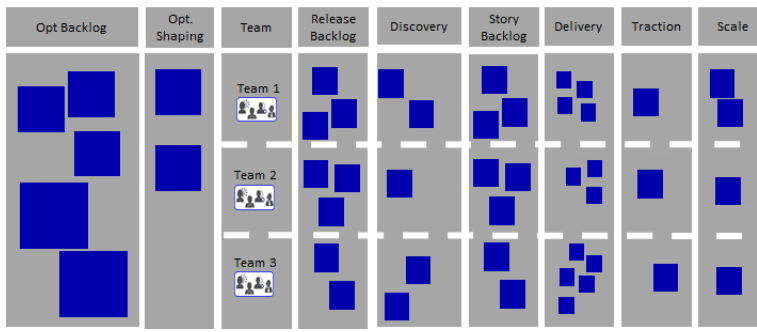
Kanban can be used to *scale horizontally*, connecting teams to other knowledge workers who may typically work upstream and downstream of the team as part of flow of value creation.

Kanban scales across the entire flow of value

By encouraging multiple teams, stakeholders, and other participants to collaborate in front of this style of visualization, they can start engaging with each other as if they are part of a common ecosystem, regardless of official organizational structure. We can encourage more frequent participation toward common outcomes and we can better align and engage across a wider horizon of activities.

Scaling Horizontally

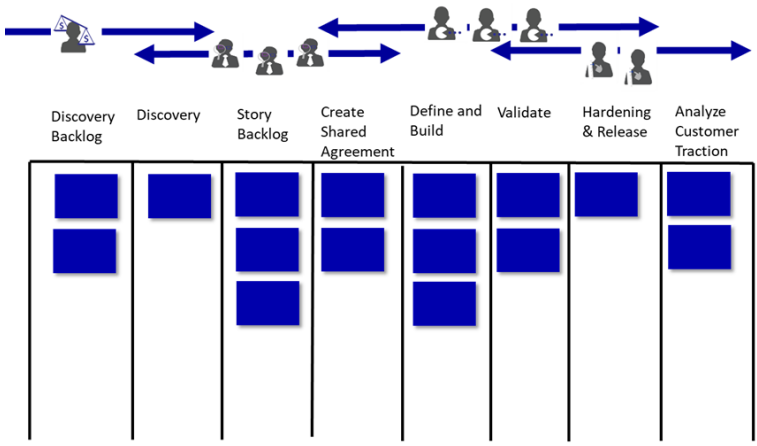
Kanban also allows us to scale our visual management vertically, to encompass multiple teams, an entire Agile Ecosystem, or even a larger organizational unit. Often with this style of visualization we abstract the work each team is doing to a higher level than individual stories. Individual tickets may represent outcomes, or thin slices. Even though this level of abstraction is less accurate, it is often enough detail to facilitate understanding of where our organizing structures are failing us, and where failures in collaboration are impeding value creation.



Kanban scales across multiple teams

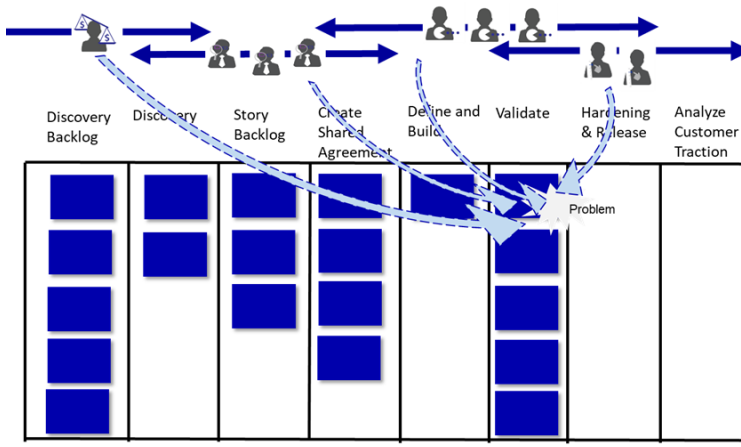
Using Kanban To Continuously Organize Around Value

Using Kanban in this way makes it much easier for people to understand the end to end flow of work, and allows a form of grass roots self-governance to take place. We get a common understanding of people's starting position within the flow of the work, and more importantly, we can also get a sense of how people move into new positions to collaborate with others. (Remember the Hockey Maturity Model metaphor?) We can start thinking about value networks not as hand offs across teams, but as concentric circles, with people overlapping to ensure work flows smoothly.



collaborating not handing things off

Kanban is working especially well when people are able to transcend any official team structure or hierarchy to swarm around a problem that is impacting the flow of value. I have seen this work not only in a team, but also across teams. and even across ecosystems, but not as often.

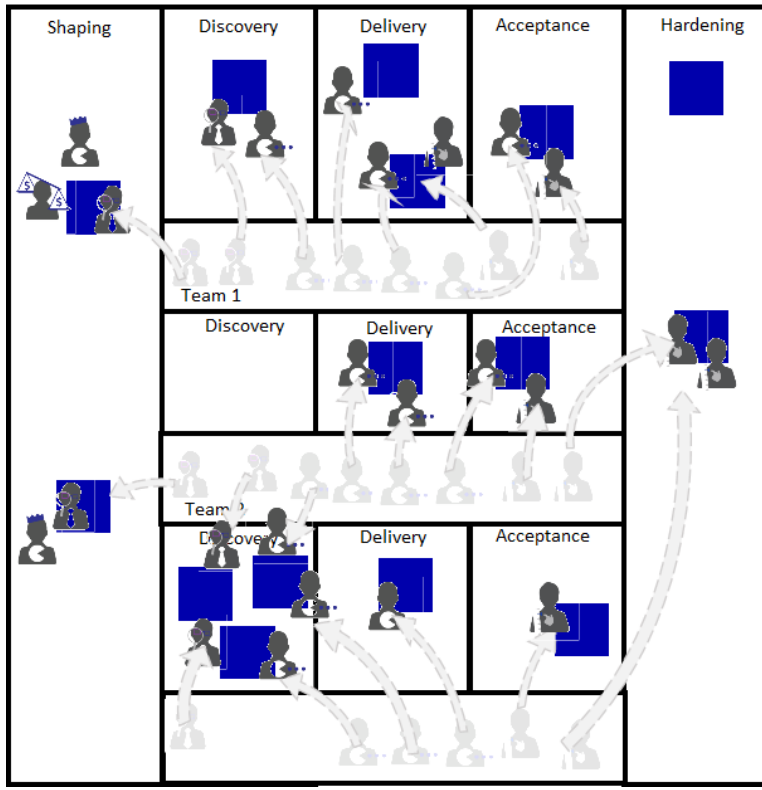


ScalingAgileForContinuousOrganizationalChange/Untitled4.png

I have had especially good experiences helping others use Kanban at the team of teams level, what I am again, referring to as an Agile Ecosystem[24]. Using Kanban at the ecosystem (others call it a program, or a portfolio, or a tribe) level allows people to organize not only across the flow of value creation, but also across departments, across teams, or over other "official org structure".

With the right attention to an ecosystem level Kanban system, people can start to form muscle memory in terms of how to collaborate in different configurations. People start to transition their thinking from static structure (team or otherwise) to a set of organic interactions that promote flow.

[24]https://agilebydesign.com/from-self-organizing-teams-to-self-forming-ecosystems/



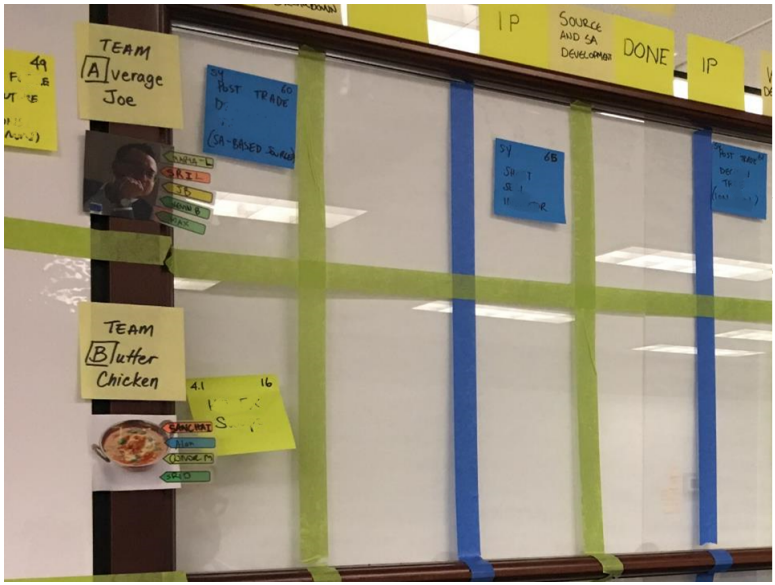
swarming around the value

Use Visualization to Illustrate Capabilities and Expertise in more Detail

Simple flags and other annotations can be used to articulate what capabilities we need to complete a piece of work, and what capabilities are in possession of a team within an ecosystem. This snapshot was taken from a Kanban System used to facilitate a large scale regulatory program that consisted of more than 11 teams. At the beginning of the program, a high level story map was used to understand what product and system experts were needed from

across more than seven different organizational departments. Team were quickly assembled by color coding each system or product expertise required for the program, matching people to the colors, and then assigning that color to Epics in the program level backlog, . Here we can see the people belonging to two different teams represented on the Kanban, right next to the Epic that team is working on.

This planning exercise took little more than a couple of hours. While it is true that we spent weeks ironing out the kinks, it was an incredibly fast process!

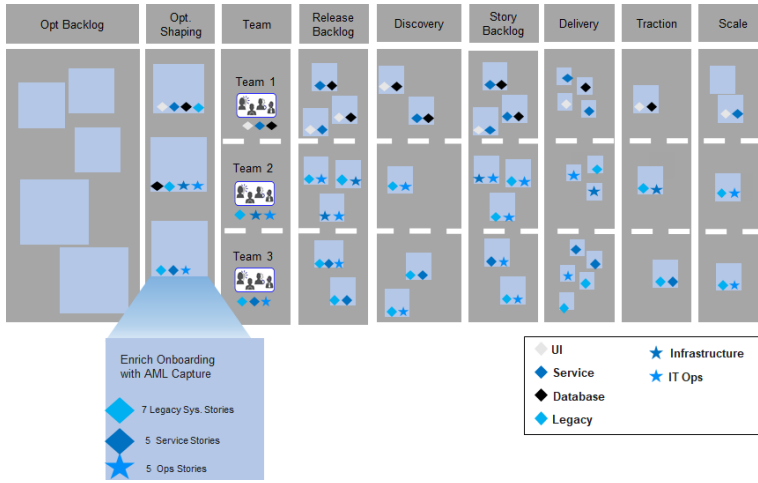


how did this suddenly become simple...

During this program, team composition and number of teams flexed based on the work flowing through the system. Some team members were more or less fixed to a team, while others acted more as travelers, and would shift across teams frequently, in some cases daily. The process of people moving across teams started as a manager led process. But over time it became a responsibility of

everyone working on the program. Planning, stand up, and review cadences were used to create a common awareness of how to organize around the value.

There all kinds of creative ways people can choose to illustrate the capabilities required to complete a piece of work, as well as the capabilities present in the current structure of teams. I have seen people use clever annotations, and team / individual "hockey cards" with simple icons to indicate various skill sets.



I can actually see what's going on…

This kind of visualization of capabilities on your Kanban system, can be taken a touch to far, especially if by the off chance geeks are involved in designing your Kanban system, :)

So take care not to let this get out of hand. Remember the point of visualizing capability is to make it easier for people self -form into the right structure that will allow for the creation of value with as few hand offs as possible!

we may have gotten carried away here...

There are limitless ways people can self-form around value, but I have seen some patterns as well. I will go over these in an upcoming chapter on Dynamic - Teaming patterns.

There are also different ways we can encourage people to attend to this kind of visualization at the Ecosystem level. I will go running various cadences at the ecosystem level as part of my next post.

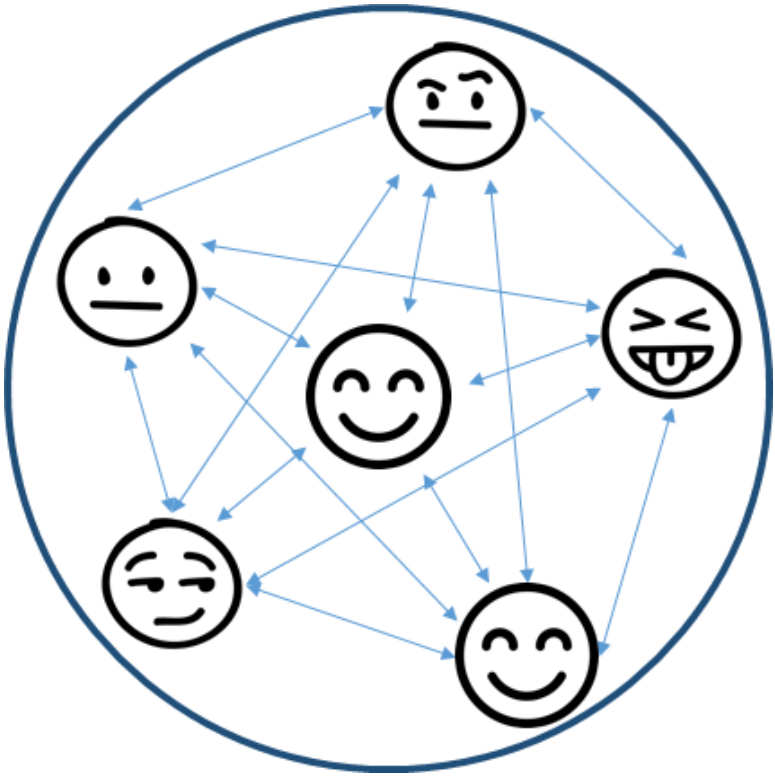
Balance Stability With Eliminating Hand Offs

Try To Keep Teams Stable

As I have stated previously, when people switch teams to often, they do not get the chance to gel, and do not get the chance to achieve high performance. Rework and distraction from part timer syndrome tend to be the norm without dedication of team members. Learnings that the team gained and any adoption of new culture and values also tends to ,evaporate once a team is disbanded.

So, it is very challenging to make teams work unless they exhibit some form of stability. Yes, for teams to work they need a common goal, and overarching mission, some constraints in terms of committing to transparency, feedback, and focus. They need space and

time so they can achieve social cohesion. Without some form of team stability we don't get this time or space.



we all live in a team with social density...

Strive To Eliminating Cross Team Hand Offs

We also need to be mindful that teams cannot effectively self-organize if they don't own the full scope of the work required to accomplish a mission. If a team has to hand of work to other teams in order to be successful, you have reduced their ability to self organize. They now need to organize with another team. Coordination is expensive, teams will slow down, and it will be

harder to react to feedback.

When a new piece of work comes into our organization, one that requires work from people across multiple teams, we are faced with a dilemma, do we keep the team structure as is, creating a need for a hand off across teams, or do we change our team structure, in turn introducing instability to the impacted teams?

Continuously and Deliberately Evolve Structure

The answer is to mindfully evolve your organizational structure as market demand changes over time. Thoughtful design of teams may not eliminate cross team dependencies, but it can minimize them.

Refinement of team design can be held at a cadence, done as part of the initial exploration of new opportunities. When we refine our make up of teams we need take care to minimize the impact that refinement has on team stability .

Better stated, we want minimize the changes we make to our team structure to those changes that allow us to minimize hand offs across those teams.

This is a balancing act, and not always an easy one, or even a possible one. Sometimes we end up sacrificing team independence for team stability, or vice versa.

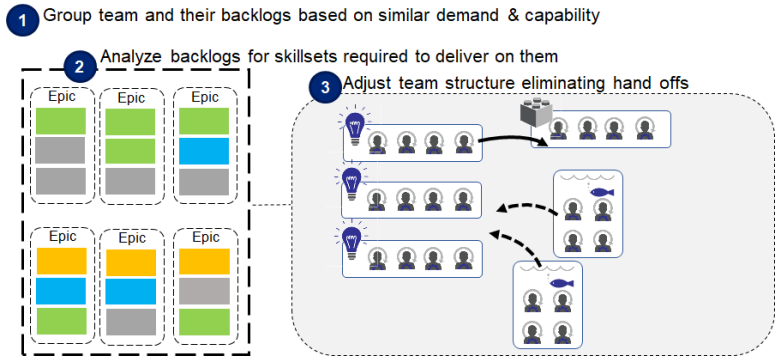
It is a counter intuitive truth that team independence trumps team stability.

So stated fully:

Organizing Constraint 8: Balance team stability while deliberately evolving team structure to eliminate hand offs between teams

Be Agile About Your Agile Teams

Visual backlogs, visual flow and other agile practices make it easier for people to self- organize around the work. This includes getting accommodated with a new team structure. But it takes courage, and it takes practice.



Refining team structures can be accomplished as part of an Agile style event, held at cadence, and dedicated to coordinating this type of planning. We can use flow visualization systems at the Initiative / MVP level to inform this cadence. We can facilitate informal, high level Epic /Feature analysis to inform our team structure. We can spot bottlenecks to inform where hand offs are acceptable and where they indicate a need for a structure with less hand offs. We will go through each of these techniques in the upcoming chapters in this book.

When demand fluctuates it tends to do so in a way that is recognizable to people that take the time to correctly observe it. Patterns start to form over time, patterns that not only leaders can see, but team members can recognize. Aided by a bit of Kanban style visual management, leaders and team members can observe changes in demand and participate in the deliberate and continuous evolution of team structure and how people are placed into teams.

Movement of people and change in team structure will start to

become repetitive. This repetition reduces the social strain caused when people move across teams. Team members know when to expect people to come and go. People will start to form a collective understanding of when they need to move to a different team. In effect stable teams evolves to become stable social fabric. A fabric that gracefully flexes to take advantage of new opportunities.

Using Kanban to Enable Continuous Evolution of Organizing Structures

- Kanban and The Promise of Better Service Delivery
- Dedicated Team Member
  - Feature Cells
  - Dynamic Feature Teams
- Traveller Pools
  - Stable Team Fronted Pool
  - Part Time Expert
- Intent Owner Traveler
- Service Provider
  - Service Oriented Value Network
  - Dependent Teams
  - Rotating Stand-up Participation
- Swat Team
  - Shared Discovery and Integration
- Enabler
  - Common Core Team Work Types - Servicing, Travelling, Enabling, Governing, and Platform Building
  - Travelling To Enable
  - Cross Cutting Product Owner
- Balancing Demand - The Path to true Enablement