BEST AGILE ARTICLES OF 2017

Co-Editors: Michael de la Maza, CEC • Cherie Silas, CEC

Best Agile Articles of 2017

Editors:
Michael de la Maza, CEC & **Cherie Silas,** CEC

BEST AGILE ARTICLES OF 2017

Edited By: **Michael de la Maza,** CEC & **Cherie Silas,** CEC

Cover design by Christopher Kyle Wilson
The text of this book is set in Warnock Pro & Myriad Pro
Book layout by THDesign, Inc.

First Edition: November 2018

Table of Contents

Table of Contents *continued*

Table of Contents *continued*

Table of Contents *continued*

***

Foreword

We are delighted to bring you this volume of the best agile articles of 2017. Our goal in publishing this book was to cull through the thousands of articles that are published every year to bring you a curated set of high quality articles that capture the latest knowledge and experience of the agile community in one compact volume.

Our purpose is twofold. First, we understand that it can be hard to figure out where to go when looking for ideas and answers. There are thousands of blogs, videos, books and other resources available at the click of a mouse. But that can be a lot to sort through. So, we thought we could be of some assistance. Second, we wanted to bring some visibility to many people who are doing good work in this field and are providing helpful resources. Our hope is that this publication will help them connect to you, the ones they are writing for.

Our intention is that this publication is to be by the agile community as a service to the agile community and for the agile community. What that in mind, we pulled together a great group of volunteers to help get this work into your hands.

The articles in this volume were selected by:

- A Nominating Committee of eight people with expertise in many areas including Kanban, Scrum, and professional coaching.
- The agile community. A call for nominations went out in mid-2017 and several dozen articles were nominated by the community.

The articles themselves cover a wide variety of topics including organizational structure, culture, and agile leadership. There is something for almost everyone here.

This is the first of what we expect to be an annual publication and we are thankful for the great participation by the agile community at large. If you would like to participate in delivering this publication in future years, contact us at the email addresses below.

Your co-editors,

Michael de la Maza, CEC
michael.delamaza@hearthealthyhuman.com
San Francisco, CA USA

Cherie Silas, CEC
cheriesilas@tandemcoachingacademy.com
Dallas, TX USA

9/27/18

Lean Startup has Changed Nothing!

By Pete Behrens

[**Note:** grayed text indicates hypertext links in original article.]

"Lean start-up has changed nothing."

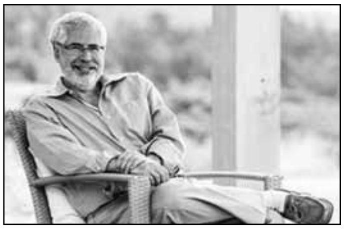
– Steve Blank

Don't take my word for it, take it from the one who founded the Lean Startup movement — Steve Blank. In a recent interview with Kurt Nickisch of the Harvard Business Idea Cast **podcast #588: When Startups Scrapped the Business Plan,** Blank discusses the growth and challenges in the Lean Startup Movement.

Who is Steve Blank?

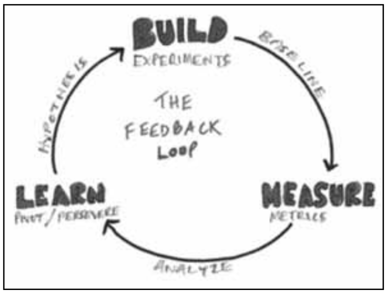
Blank, a serial-entrepreneur and adjunct professor of entrepreneurship at Stanford, is recognized for developing the **Customer Development** method that launched the Lean Startup movement, popularized by Eric Ries in his book **The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses**. After a decade of Lean Startup experiments in Silicon Valley, Blank authored a pivotal HBR Article in 2013 — **Why the Lean Start-Up Changes**

Everything, placing it on the many CEOs "To Do" list. Today however, after years of witnessing these innovative Silicon Valley startup mindset unable to penetrate the traditional corporate cultures, **Steve is changing his tune.**


© 2012 Eric Millette, All Rights Reserved
www.EricMillette.com 415-750-9999 USA

What is Lean Startup?

Lean Startup is discovery agility — as business encounters increased uncertainty in what product to build and how to build it, a lean startup approach seeks to learn more quickly and with less investment. Lean startup helps identify who your customers are, what problems they are struggling with, and guides learning through identifying and validating assumptions and pivoting (changing direction) through frequent feedback cycles.



"Lean Startup sounds a lot like Scrum!", you might say. You're right — I see Lean Startup as a more business-friendly and less codified version of Scrum. In many organizations I engage with, we incorporate Lean Startup language into a Scrum framework by adding assumptions and experiments into the backlog and using the Sprint framework to run experiments, learn and pivot. Don't take my word for it though — read the book.

Does Lean Startup Work?

GE, a company that I have been working with for almost a decade, co-developed and rolled out an entire corporate discovery model with Eric Ries entitled Fast Works — see the HBR Article: **How GE Applies Lean Startup Practices.** GE applies Lean Startup principles across all of their businesses from Healthcare to Appliances to Aviation.

Yes! Lean Startup is an incredibly effective framework, not only for those companies in an early startup phase, but also for established companies seeking to ingest more creativity, experimentation and innovation into their product development process. But again, don't take my word for it — Google case studies for yourself.

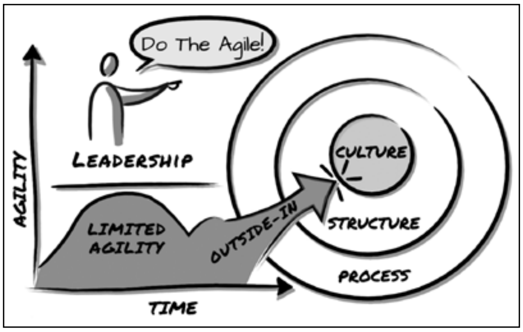
So Why has Lean Startup Changed Nothing?

First, let's look at the quote Steve Blank stated in the HBR interview…

> *"…lean start-up changed nothing, because I will contend that after three or four years of watching corporations try to adopt a lean methodology, that it hasn't affected the top or bottom line* ***for reasons that are a lot more cultural and organizational*** *than they are about whether you have an incubator, an accel-*

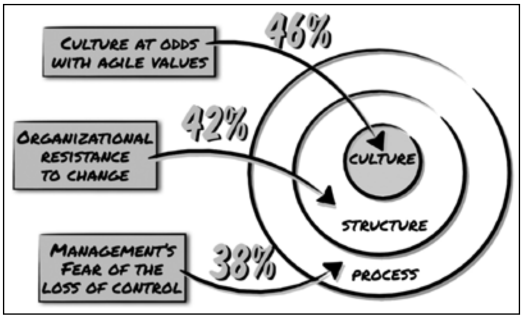
erator, or a chief innovation officer, and that was the surprise for me." – **Steve Blank** [emphasis added]

Blank modestly admits that Lean Startup has failed to make a significant impact to the bottom line of companies and governments across the globe. The reason? Corporate cultures and organizational structures treat Lean Startup as a virus to its way of living and attack it! Just like a bad organ transplant.

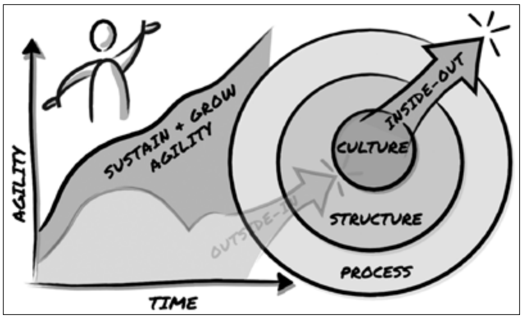


As I have been discussing this phenomenon for over 7 years now, Agile and all of its variant approaches like Scrum and Lean Startup are based on a shared set of agile values and principles (see **Agile Manifesto**). As companies venture into Agile territory, they will encounter these new *unusual* values and principles. Most companies, however, will only give a cursory glance at them — focusing primarily on following the practices.

Leaders often instruct their employees to "do" Agile (or Lean Startup). As these teams "do" Agile, they begin to encounter impediments, bumping into organizational structures and values that inhibit their ability to do Agile effectively. These are the cultural and organizational references which Blank is referring in his quote above.


CULTURE AT ODDS WITH AGILE VALUES
46%
ORGANIZATIONAL RESISTANCE TO CHANGE
42%
MANAGEMENT'S FEAR OF THE LOSS OF CONTROL
38%
CULTURE
STRUCTURE
PROCESS

Blank is highlighting what I have been seeing in the agile community for many years. Organizational and Cultural issues have been the top 3 impediments to deeper organizational agility for the past 10 years (see **VersionOne State of Agility Reports**)! A lack of cultural alignment, organizational resistance to change and a fear of a loss of control continue to limit agility. And when organizations approach agility from an outside-in perspective, focusing on "doing" practices of Scrum or Lean Startup — they fail to address these deeper organizational and cultural impacts.



However, when organizations approach Agility as a set of values to be integrated with and aligned to their own corporate values, and then explicitly define organizational structures to support and nurture their agility — Scrum and Lean Startup thrive.

This is the focus of the new Certified Agile Leadership (CAL) Program — helping leaders understand how to incorporate their own agile thinking and behaviors into their leadership practice to enable, support and grow more effective, adaptive, and empowering organizational cultures.

***

To read this article online with embedded hypertext links, go here:
https://trailridgeconsulting.com/lean-startup-changed-nothing/

About Pete Behrens

Pete guides senior executives to transform themselves and their companies to greater effectiveness and agility.

Pete is a Certified Agile Leadership (CAL) Educator, providing awareness (CAL 1) and practice (CAL 2) for improved leadership competency and value delivery. He developed the CAL Program for the Scrum Alliance in 2016.

Pete is a Certified Leadership Agility 360 Coach providing one-on-one 360 assessment, development and guidance for increasing awareness and agility of organizational leaders. He became a Changewise 360 Leadership Agility Coach in 2009.

Pete is a Certified Enterprise Coach (CEC) and a Certified Scrum Trainer (CST) engaging with organizations to improve team alignment and delivery. Pete developed the CEC Program for the Scrum Alliance in 2007 and became a CST in 2006.

Pete is the founder & Managing Partner of Trail Ridge Consulting, a co-creative coaching partnership building sustainable and healthy organizational agility. Their holistic framework-less agile scaling approach guides dozens of sustaining and healthy agile practices globally. He founded Trail Ridge Consulting in 2005.

Pete is on the Board of Directors for the Scrum Alliance, providing strategic consulting and guidance to transforming the world of work. He speaks at Agile Conferences, Scrum Gatherings and Agile Leadership Events, and Local Agile User Groups across the globe. His Board of Directors term is 2016-2018.

When Pete isn't visiting clients across the globe, he is probably biking or golfing near Boulder, Colorado.

Follow Pete on Twitter or LinkedIn, or Email Pete – pete@trailridgeconsulting.com.

If you want to innovate, don’t say so

By Sonja Blignaut

“Innovation is not so much about having ideas as it is about making connections”
— **Harold Jarche**

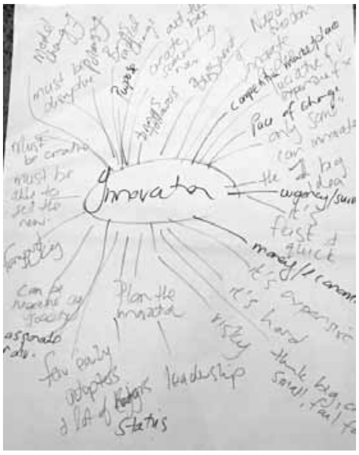
Last week I had the opportunity to facilitate a week-long “innovation sprint” for Agile 42, an international agile coaching company. After recovering from my time spent facilitating 39 other facilitators (on my own!) I finally had time to reflect on my experiences and learnings from the week. One of these is the importance of framing.

The framing of an event (i.e. what we call it) creates a container within which the process ro work unfolds. It therefore constrains the process in ways that are either enabling or disabling. Framing this event as an “innovation sprint” created expectations and dynamics in the group that could paradoxically end up stifling innovation instead of enabling it.

As this is a group of agile coaches, it made sense to use Agile terminology like “sprint”, however the term immediately evokes patterns from implementing Scrum (a specific Agile methodology) and tends to create expectations of a linear, iterative, time-boxed process accompanied by regular feedback and planning ceremonies. This brings the very real danger of premature convergence, and a focus on iterative problem solving. The pattern of iteration and regular feedback cadences also brings an expectation for regular check-ins to determine if “we are heading in the right direction”, which in this context is not a relevant question there is no “right direction”, especially in early stages of the process where diversity, enquiry and wide exploration are required.

The framing around “innovation” introduces different kinds of pressure and expectations into the process. Some of the participants voiced concerns about “not being particularly innovative” and many seemed to adopt a wait and see attitude about whether anything new will actually emerge. During our first ice-breaker exercises where we unpacked high and low dreams for the process, one of the low dreams for this event was that we would only end up with “tweaks and fiddles” as opposed to “real, out of this world innovation”. The pressure of having to come up with “that ONE big idea” often stifles, rather than enables innovation. To counter this, one of the first exercises I facilitated with the group was a narrative meaning making process around innovation. The aim was to at least externalise these meanings and their impact so that we could work with it in the group. We explored the following questions together as a large group:

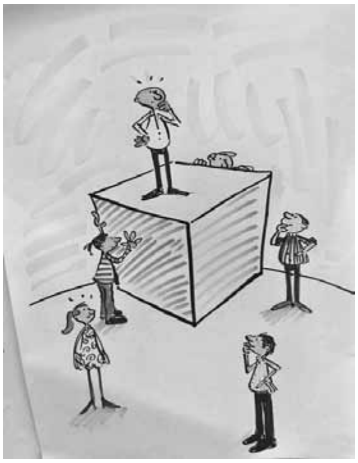
- What do "they" say about innovation? (They being the nebulous "other" that we often talk about, but never really know who "they" are)
- We then broaden it to look at the context in which these things are said
- Then we unpack the impact of these meanings on the group



The group highlighted two patterns that emerged here: one of pressure (innovate or die) and another of prejudice (only some people can innovate). Innovation was seen as "hard", about that "ONE big idea" that had to be lucrative. It is risky — expensive if you fail. Innovation is fast and quick; it's an imperative "innovate or die". On the other hand it's also a "buzz word". When asked to name this innovation narrative names like "we're all confused" emerged.

The group agreed that the impact of this on them as they're about to embark on this week-long process was a paradoxical feeling of pressure to come up with a brand new idea, while at the same time not believing it was actually possible. Simply externalising these meanings and acknowledging them created a much more open container for work the group needed to do in the workshop. Upon reflection later on in the week, many participants referred back to this process as a key break-through moment.

We also challenged the cliché of innovation requiring us to "thinking outside the box" … how can we think outside of a box when we often don't even know what that box is (and if it even exists).



We chose to reframe our process as "unfolding the box", this served to shift the perspective of the group to one of curiosity and enquiry, again taking the pressure off.

Agile 42 has a strategic partnership with Prof Dave Snowden, and we wanted to use this workshop to gain deeper practical insight in the Cynefin framework and other related methods. We therefore "unfolded the Agile42" box using multiple complexity based group activities such as …

- Future backwards to understand the current reality (today), future dreams (patterns in heaven) and fears (patterns in hell) of the group as well as gain insight into the shared corporate memory (patterns in the timeline). Here the group discovered that even though they are distributed across many different countries and continents, there were golden threads that tied them together.
- Cynefin contextualisation to surface sense-making patterns and biases.
- Archetype extraction – prior to the workshop, we collected stories about current agile practices (best and worst) using Sensemaker. We used these to extract emergent archetypes, visualised by a cartoonist in the workshop.
- We used appreciative interviews to surface and share success stories about client engagements as well as sales and marketing experiences. We then asked the ASHEN (Artifacts, Skills, Heuristics, Experience and Natural Talent) perspective questions and to surface key knowledge objects that exist in Agile42.
- We applied ABIDE (Attractor, Boundaries, Identity, Diversity & Environment) as perspective lenses to understand the market and "influencable" patterns.

All of these methods served to make patterns visible and keep the group from falling into old patterns of identifying problems and then trying to find solutions. One realisation that came from the group once the various perspectives were externalised was around innovation being much like a chef that combines existing ingredients into brand ties seemed disparate, each providing a different perspective or unfolding a different aspect of the box. There was no defined outcome, and no linear process where each activity built on another. Being complex methods, they were largely emergent and involved ambiguous instructions. Golden threads only became apparent on day 3, and some activities were "left hanging" and never really resolved. The majority of the people in the room were coaches, and as such are used to being the experts and the "ones who know". The inherent uncertainty and ambiguity that such a divergent process brings challenged this identity and brought greater empathy for how their clients feel when they experience similar processes.

When we did finally converge on day 4, the majority of the group felt that the discomfort was worth the end result. Many felt that the value was as much as the experience of the uncertainty of the process, and the associated learnings than the eventual outcome (a portfolio of experiments and ideas that will be actioned in the next few months).

One anecdote from a participant really struck me.

> *"Yesterday during a break, I went for a walk in the bush around the venue. I came across a deer in the path, and I immediately thought that I had to bring my son with me next time, as he'd love to see it. But then I realised that if I came back here expecting to see the deer again we'd probably both be disappointed"*

Such is the nature of innovation … most often it happens by accident. It's about serendipitous encounters and connections — we can create conditions where there is an increased likelihood for it, but in the end we can't mandate, manage or create innovation.

⁂⁂⁂

To read this article online with embedded hypertext links, go here:
http://www.morebeyond.co.za/if-you-want-to-innovate-dont-say-so/

About Sonja Blignaut

Sonja is a thinking partner to decision-makers and change agents. She specialises in applying methods rooted in complexity theory to enable change resilient, adaptive cultures and scaled agility.

Sonja has been working in the fields of narrative and complexity since 2002 and has consulted internationally with blue chip clients in a variety of industries. She is a sought-after speaker, trainer and facilitator.

Sonja is the exclusive South African partner for Prof Dave Snowden's company Cognitive Edge, offering training and consulting a variety of CynefinTM methods and Sensemaker.

www.morebeyond.co.za • twitter: @sonjabl • LinkedIn: http://linkedin.com/in/sonjablignaut

At the Intersection of Culture & Strategy

By Melissa Boggs

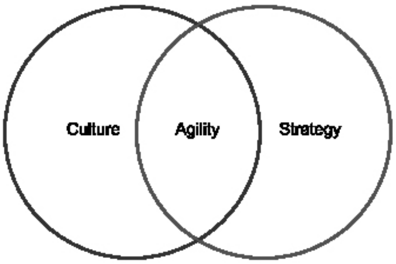
Culture: The traditions, habits, and behaviors of a group or organization.

Strategy: A plan of action or policy designed to achieve a major or overall aim.

How many times have you heard the words culture and strategy thrown around in the last few months? We throw these words around a lot in a corporate setting, without giving a lot of thought to their true meaning. In fact, you have probably heard Peter Drucker's quote, "Culture eats strategy for breakfast." in a presentation or on a website. I understand and agree with his general premise, that ignoring the existing traditions, habits and behaviors of a group can ultimately undermine or deter any strategy you develop. The successful execution of a strategy requires a supporting and aligned culture.

However, I'd also argue that culture needs strategy too. A group with shared values and ideals whose traditions, habits, and behaviors are healthy will need direction, and a plan. In fact, a healthy cultured group would likely demand it, in some form. The two are symbiotic, and a team or organization that has a healthy culture + a strong supported strategy can experience a deep level of alignment and productivity.

I am a huge fan of agile values and principles, and in my mind, they look a little like this:



This is why I love the line, "At the intersection of culture andstrategy". Being agilists really requires that we have a full understanding of both, and how agility is impacting (and impacted by) both. We are human beings, doing human work — but it IS work. We are in business to make money, make a difference, or a bit of both. Often, the word culture is perceived as “too fluffy”, and all about hugs and ping pong tables. I think in recent years though, organizational leadership has come to understand how crucial it is to understand, evaluate, and be intentional about what we do in that intersection.

That’s where agility comes in. When I read the Agile Manifesto, two things stand out: People and Business. The movement was a resistance to the notion that building software was akin to the manufacturing process: Wash. Rinse. Repeat.

People change. Business changes. Together, they need to be both flexible, and in sync. The manifesto reminds us that the two are inextricably linked. Individuals and Interactions (People). Working Software (Business). Customer Collaboration (Both!). Responding to Change. (Both!).

> *The requirements for our evolution have changed. Survival is no longer sufficient. Our evolution now requires us to develop spiritually — tobecome emotionally aware and make responsible choices. It requires us to align ourselves with the values of the soul - harmony, cooperation, sharing, and reverence for life.* – **Gary Zukav**

So how do we get there? When I work with organizations and teams, I like to start with their purpose. What is their mission? Do they have a defined set of core values? Is everyone aligned around those two things? Those are foundational to culture, strategy, AND ultimately to organizational agility. It’s not a silver bullet, but it gets us moving in the same direction. From there, we can develop a supporting strategy AND culture, and continually evolve both.

Agile values and principles (and the resulting movement they inspired) are focused on creating human-centric and human-respectful cultures. We aim to create cultures that bring out the best in us and our teams, and leverage those cultures to support dynamic and evolving strategies toward our goals.

***

To read this article online with embedded hypertext links, go here:
http://www.hummingbirdagility.com/blog/2017/11/28/at-the-intersection-of-culture-strategy

About Melissa Boggs

Melissa is a Culture, Agility, and Leadership Coach with ten years of investment in the agile movement. A Certified Enterprise Coach with an MBA, she blends education, experience, and enthusiasm to work closely with leaders on their most challenging business problems. Through training and coaching, Melissa encourages executive leadership to expose and understand the hidden strengths and weaknesses that exist within their culture, and how they can amplify what is best about the traditions, habits, and behaviors in their org. Most of all, she helps leaders and employees alike to introduce joy and inspiration into the workplace. You can find Melissa on Twitter as @HmngbirdAgility or at www.hummingbirdagility.com

Scrum Guide Sliders

By Zach Bonaker

[**Note:** grayed text indicates hypertext links in original article.]

It was Monday afternoon on her first day of work at a new company and Sarah was feeling anxious. Having studied Scrum over the last few months, Sarah had been hired to serve as a Scrum Master and she was eager to begin applying her new skills, however nothing she had been told today seemed congruent with her lessons. As Sarah listened to her new manager explain why the company's software required the organization to field a "UI team," "service team," and "back-end team," she felt especially troubled by the phrase she'd heard multiple times on this day: "No one does Scrum by the book."

"If no one actually follows the design of Scrum," she wondered to herself, "why did my Scrum trainer teach it to me?"

There's plenty more to this short anecdote; it's a brief introduction to a true story. And while Sarah eventually found a great deal of success, her experience with coercion to dysfunction in the name of "no one does Scrum by the book" is shared by far too many practitioners. I argue what passes for "Scrum," **more than twenty years since conception,** is so commonplace that **we've started accepting all of it (results, outcomes, experiences) as factual Scrum.**

While distressing, this reality is also quite understandable. **In a terrific blog post,** Ashok Singh said, "When people are not able to solve organizational problems, they come down to tweaking the framework to accommodate the dysfunctions." Meanwhile, we fear the dogmatic, prescriptive Scrum zealot, yet often find ourselves **trapped in a downward spiral in the name of pragmatism.** At the core of this pattern lies questions we might first ask ourselves: Before altering the Scrum framework for potential improvement, shouldn't we first achieve the maximum results from the framework itself? Without a baseline, how do we know we're tweaking the essence of Scrum for a competitive advantage?

Therefore, to help mitigate these risks for companies using Scrum—**"the Inevitable Avoidance Principle"** and false conclusions—I've been experimenting with (and enjoying wonderful success with) an exceptionally simple visual technique over the past year. I call it, "Scrum Guide Sliders."

The idea for "Scrum Guide Sliders" came from the mind of **Neil Killick** and his work in assisting organizations with agile-guided change. While mentoring Sarah and exploring ways to detangle the impact of "no one does Scrum by the book," **I serendipitously stumbled onto a Twitter dialogue between Neil and Bob Marshall.** As I considered how traditional "risk sliders" were used here to call attention to values, Neil's method for facilitating conversation and shared understanding of organizational mindset triggered an idea: What if we harnessed the same visual power for companies learning Scrum?

Therefore, the goal of the technique is threefold: reduce (or eliminate) false conclusions, create shared understanding of the behaviors needed for Scrum's success, and make visible what is often hard to see.

So, what exactly does "Scrum Guide Sliders" look like? Well, it might look like anything, as I hope you'll appreciate the simplicity and ease of customization… however, here's an introduction to the tool as I've been using it:

Download a spreadsheet version here — this is my personal version and it's free to download a copy, use, and enjoy! (Note: this version uses circle objects to create the "marker" for each slider. The circles don't work well with Google Sheets; try downloading a copy and using Excel for a better experience!)

Using Neil's dashboard as a model, 'Scrum Guide Sliders" extracts the contents of the **Scrum Guide** and organizes the essential components (and behaviors) into four sections:

- **The Development Team**
- **The Product Owner**
- **The Scrum Master**
- **The Sprint**

Within each segment, a series of "sliders" are housed that create a spectrum of behavior which might be described as:

"Scrum in Name Only" on the left <—> "Scrum by the Book" on the right

Given the context of my story about Sarah — and other stories you might have heard about "no one does Scrum by the book" — for the destinations on the right ("by the book"), I intentionally use verbiage directly lifted from the Guide wherever possible. For example, when describing The Development Team, the Guide states: **"They are self-organizing; No one (not even the Scrum Master) tells the Development Team how to turn the Product Backlog into increments of potentially releasable functionality."** I retain much of the phrasing here to reinforce the shared understanding of what Scrum is… and what it is not!

A few sliders I've included in my version are not explicitly stated in the Scrum Guide, however. For example, regarding The Scrum Master, one slider asks us to consider whether we have enough SMs to ensure the "process" occurs–versus–having enough

SMs to ensure teams get the coaching and mentoring they want. The Scrum Guide doesn't explicitly state how many teams a Scrum Master can work with; it simply points out that a Scrum Master is a required role. Therefore, I've added this slider based on my experiences and interpretation of the Guide, believing that this is a meaningful mindset change that helps us become successful with Scrum.

That last part is important: you're free to use my version of "Scrum Guide Sliders" as I've defined it, however this technique offers you an invitation to customize and add, change, or remove sliders. See something missing? Perhaps some of these sliders are unimportant to you? Experiment with sliders to fit your context, goals, and culture!

Use of this technique is exceptionally straightforward: In whatever way facilitates an appropriate conversation for the people you're working with, let people see the Scrum Guide Sliders visual and have dialogue over where a marker should be placed for each slider. In facilitating the discussion, you'll discover people begin to reveal their assumptions, share their feelings about the current working environment, and tell stories that hold a mirror to reality… and offer insight into what might improve if the slider moved closer to guide.

Lastly, I've found "Scrum Guide Sliders" have excellent synergy with a variety of methods to help decide on actions and experiments. In particular, three methods have proven to be very useful:

- **Force Field Analysis:** Setting an objective from the sliders (e.g., the "by the book" outcome of Sprint Review resulting in a revised or prioritized product backlog), use Force Field Analysis to uncover what's supporting and working against the objective.
- **Social Cause Mapping:** I learned this technique from **Dan Greening** and it should feel familiar to anyone using **Ishikawa diagrams** or **"five whys"**. With Social Cause Mapping, extract the "problem" or thing of interest from the sliders and, one person at a time, uncover possible causes that might be holding you back.
- **Perspective Mapping:** What might you discover about your organization when multiple groups (e.g., Managers, Scrum Masters, and Teams) place their own markers for each slider? Do the groups of people converge or diverge in their perception? What might improve if perceptions became more aligned?

It's important to reiterate the goal isn't to move every slider 100% to guide; the goal is to create shared understanding. I've found this technique enables us to easily have a conversation about "why?" When we say, "no one does Scrum by the book," this technique helps us see just how far away we are… and encourages us to ask what might be different if we were closer "to guide."

Geoff Watts summarizes this principle far more eloquently than I:

"…while measuring how strictly you are adhering to the rules and principles of Scrum is not the point, if you believe that an agile approach, such as Scrum, is a viable means of becoming successful then assessing yourself against the

> *application of that framework or process as a proxy metric of success might not be a bad idea."* — **Geoff Watts**, Scrum Mastery: From Good to Great Servant Leadership (71)

Good luck, have fun, and feel free to contact me (zbonaker@gmail.com) with comments, questions, or ideas for using "Scrum Guide Sliders!

⁂

To read this article online with embedded hypertext links, go here:
https://agileoutloud.wordpress.com/2017/06/26/scrum-guide-sliders/

About Zach Bonaker

Zach is an agile coach based in San Diego, California, USA and has more than ten years of experience assisting software organizations with improving working conditions and results. Acting as a "benevolent trouble-maker," Zach builds relationships to help transform people, systems, and structures towards safer and collaborative ways of delivering high quality software. Zach is an international conference speaker, frequent podcast guest, and contributor to the global agile community. When he isn't thinking about next-generation agile ideas, Zach can be found enjoying the sunny California weather and connecting with people all over the world.

Agile in Highly Regulated Environments

By Braz Brandt

[**Note:** grayed text indicates hypertext links in original article.]

"Individuals and interactions over processes and tools;

"Working software over comprehensive documentation;

"Customer collaboration over contract negotiation;

"Responding to change over following a plan."

For those of us in the Agile community, the **Agile Manifesto** is a wonderful expression of the True North of Agile software development – empowered teams, swarming to solve customer problems by collaborating closely with people who will actually use the things we're creating.

But for many, especially those who deliver software in highly regulated environments, the Agile Manifesto can seem downright hostile. When dealing with audit requirements and compliance, the thought of *Working software over comprehensive documentation* can result in Agile processes being dismissed out of hand.

With the pace of change happening in the world, that would not only be a shame, but organizations working in highly regulated environments would miss the opportunity to get ahead of competitors by leveraging Agile processes and principles.

Regulations – Prescriptive vs. Descriptive

When working in a highly regulated environment like healthcare, financial services, or dealing with reporting and regulatory audits for the US Federal government – I find it incredibly important to use a quick mental filter to understand the types of regulation my teams are working with. Broadly, I've found that regulations and their associated reporting requirements roughly fall into one of two types: **Descriptive** rules and **Prescriptive** rules.

Descriptive, Using Scrum

Descriptive rules seek to provide a definition of a system or process as it is so that it can increase the repeatability of that system or process. I've found the most frequent examples of these Descriptive rules used in internal auditing processes and also emerge in quality processes such as the **ISO 9001 Quality Management** standards.

A key factor in adopting Agile in regulated environments where Descriptive rules are in play is to make sure you work closely with whatever auditors you have, internal or external, who understand your documented processes. When I've introduced Agile processes into ISO 9001-compliant organizations, I quickly began close collaborative conversations with our auditors to make sure they understood the interactive and incremental processes we were introducing. Once we identified the gaps and differences in the documented process, I worked with our auditors to make sure our new processes were properly documented. *Descriptive rules* are made to be changed to meet the work, not to prescribe solutions! (SPOILER ALERT: We'll cover those in a second.)

Working with clients who primarily deal with these descriptive rules, I frequently look at the artifacts we can provide while using Scrum. The controls provided by a strict SDLC, especially around documentation, audit-ability, and traceability, can nearly always be met through well-written Acceptance Criteria and a light hierarchy of Epics to User Stories to Tasks. Further, most Agile software tools like JIRA, VersionOne, and Rally can provide for and automate the traceability documentation from Epic to production.

Prescriptive, Using Kanban

While *Descriptive* rules are designed to document a system as-is or as-it-should-be to provide a reference for a repeatable, quality system, *Prescriptive* rules are designed to create contracts and govern behavior. In organizations and teams ruled by Prescriptive rules, the sea-change in process and procedures introduced by Scrum can seem insurmountable.

As Agilists, it's important to remember that while Scrum may have emerged as the most popular of the Agile frameworks – to the point where most people mentally equate "Agile" and "Scrum" — it's far from the only Agile methodology or framework. When working with teams dealing with Prescriptive rules – such as those legally mandated by Federal agencies like the Food and Drug Administration or Securities and Exchange Commission — I nearly always fall back to Kanban.

Many of us conflate Kanban with the task board our Scrum teams use to make our daily work transparent. Kanban is a powerful Agile framework designed around documenting existing processes and applying rigorous focus toward maximizing flow. Using Kanban, we can find opportunities to incrementally improve our existing teams and processes.

By applying the **principles behind Kanban,** teams working under the constraints of Prescriptive rules can quickly adopt the principles of Agile.

- Start with what you do now;
- Agree to pursue incremental, evolutionary change;
- Respect the current process, roles, responsibilities, and titles

While this isn't an article about applying the Kanban principles and practices to your work, the practices themselves require embracing the Agile principles we know and love while also respecting the reality of the environment your teams exist within. Kanban's practice of *Visualizing the Workflow* aligns perfectly with Agile's embrace of transparency and openness; *Limiting Work-In-Progress (WIP)* closely aligns with the principles of simplicity and frequent delivery; *Improving Collaboratively* directly maps to the principle of regular reflection and improvement.

By using Kanban to map out your process, and then collaboratively look for opportunities to reduce bottlenecks and increase flow while also making and keeping process policies transparent and explicit, you can leverage Kanban to bring agility to your team and still meet Prescriptive rules and regulations.

Conclusion

When done well, Agile practices provide the means to create engaged and empowered teams who understand and embrace the need for regulations – both prescriptive and descriptive – and the means for those teams to own and optimize how their work is done while still meeting audit-ability, traceability, and regulatory requirements.

***

To read this article online with embedded hypertext links, go here:
https://agilevelocity.com/agile-transformation/agile-in-regulated-environments/?utm_campaign=Newsletters%20&utm_source=hs_email&utm_medium=email&utm_content=56619631&_hsenc=p2ANqtz-_koVY0Gq5q1e-HUNU4f6Vt6JyKEtPFVcXfVZgoZBTA-tKk2H3mWAlP4avFrDDz72cOb8HqT-NO3CBqNLYyJUNdDq5lA1Ow&_hsmi=56767593

About Braz Brandt

Braz is currently working as an Enterprise Agile Coach for Agile Velocity. He's been an Agile change leader and champion for nearly a decade, working in the non-profit, for-profit, government, and higher education sectors. During that time, he's been a ScrumMaster, Product Owner, Team Coach, and has most recently focused on management and leadership coaching — helping leaders, regardless of title, internalize their principle-based Agile mindset through coaching, training, and practices. He's currently enrolled in a professional coaching program with New Ventures West as he pursues certification with the International Coaching Federation as a Professional Coach.

When not coaching, Braz is a dad to six human and three canine children, and husband to his amazing wife.

What Kids Taught Me About Being Agile

By Maxime Castera

While agile initiatives now spread beyond IT within the corporate world, one may wonder why we tend to brand these changes as "transformations" or "transitions." After all, we are merely going back to a state of being which we have in fact experienced for years, not as adults but as children.

For several years, I ran my own small education business. I taught children from the ages of 6 to 10 about science and technology, specifically space and robotics, my favorite topics, with the help of LEGO® bricks, my favorite toy.

As I was busy with this endeavor alongside my daily job as a software development manager, I got plenty of opportunities to compare first-hand the difference between kids being agile and adults doing agile. This was a topic that surfaced in many agile coaches' narratives, but it was only then that I truly understood it.

Make a Plan, Then Change It

One of the most striking things I noticed during my students' activities is that when children are taking on a task, whether self-initiated or not, they always seem to have a plan. The plan is clear only to them and continually changes, but it eventually leads somewhere. Even if that means starting again, they reach the finish line quickly and iterate successfully.

It's no wonder the Marshmallow Challenge, your typical team building exercise in a corporate offsite event, is often said to be better accomplished by children than MBA grads.

Ask Why (And Truly Mean It)

When children ask "why?" they are truly curious and will likely dig further and further until there are no more answers to be had. They have neither a fear of asking nor a sense of shame, a quality which increases their learning and creative capabilities.

By contrast, adults tend to think twice before asking questions, and we feed ourselves on guesses instead. We tend to be paralyzed by our culture, our pride or fear of judgment from our peers. Either way, we miss opportunities to enrich ourselves, while children have already moved on to their next learning experience.

Flag It as It Is

Learning how to give feedback to each other can be a daunting task for agile team members, and we often associate criticism with its destructive meaning. In contrast, I would safely bet we have all experienced a very direct, sometimes crushing and often tactless (but always honest and "fairly" constructive) feedback from a child. We can all learn something from them.

Collaborate Rather Than Cooperate

I often tried to divide work and responsibilities among children in my workshops so that they could cooperate in building a larger LEGO® assembly. In most cases, my efforts initially failed, or at least felt like a failure. In fact, these little builders were often inclined to drop their own task to go and help a playmate realize his or hers.

By trying and iterating on the task together while sharing a goal, true collaboration was born and the children's learning experience was greatly enhanced. The opposite tends to be true in a professional environment, where a culture of individual goals and target-setting has traditionally reigned supreme.

While agile coaches can certainly help guide and scale your organization in the right direction, asking yourself what has happened with your inner child (minus the therapy…), or simply observing your own children, should help you open your mind to the changes to come.

***

To read this article online with embedded hypertext links, go here:
https://www.frontrowagile.com/blog/posts/115-what-kids-taught-me-about-being-agile

About Maxime Castera

Maxime currently owns the agile consulting firm Entervals (Entervals.com) based in Brussels, Belgium.

Entervals' purpose is to help organizations grow high-performance teams through trust, play and empowerment.

Maxime is also the founder of TeamPitfalls.com, a training method which exposes teams' dysfunctions using LEGO® robots-based workshops.

He is a certified PSM, PSPO, SPS at Scrum.org, a Scrum and agile development advocate and is passionate about new development approaches around lean product development and innovation.

He holds a Master Degree in Computer Engineering and Medical Imaging from the University of Poitiers in France.

Transcend the "Feature Factory" Mindset Using Modern Agile and OKR

By Felipe Castro and Alexandre Freire Kawakami

Charlie Chaplin's Modern Times

[**Note:** grayed text indicates hypertext links in original article.]

This article was first published at **Industrial Logic's blog**.

Agile adoption in most companies focuses on software delivery. Very few achieve business agility. When it comes to setting goals, the waterfall command-and-control mindset is still the norm: organizations use an annual, top-down process to create a set of static goals that is in direct conflict with being agile.

Waterfall goals and metrics turn teams into "feature factories" with no focus on delivering value. **As John Cutler describes,** many developers are "just sitting in the factory, cranking out features, and sending them down the line."

Marty Cagan highlights the huge missed opportunity of feature factories: "teams are just there to flesh out the details, code and test, with little understanding of the bigger context, and even less belief that these are in fact the right solutions." That is, the people closest to work have no influence on making decisions to help their customers or leverage existing solutions.

This failed version of Agile slows companies down and makes it harder for them to adapt to change while increasing risk and waste.

How can we even call them agile adoptions? Practitioners know that using Agile to deliver a waterfall plan has limited benefits: **70% of them report tension** between their teams and the rest of the organization, while **46% of agile adoption failures** are linked to company's culture and philosophy being at odds with agile values.

The alternative to transcend the "Feature Factory" mindset is to embrace Modern Agile's four principles. But how can we apply them in practice? How can we "do" Modern Agile?

There is one actionable tool for business agility that, if used correctly, will support the adoption of the four Modern Agile principles. This tool is **OKR (Objectives and Key Results),** the goal setting framework used by firms like Intel, Google, and Spotify.

The big difference from traditional planning methods? **OKRs** are set and evaluated frequently—typically quarterly. Furthermore, rather than being cascaded down the organization by the executives, OKR is bidirectional: teams create most of their OKRs in alignment with the company goals and then contract them with the managers in a bubble-up approach.

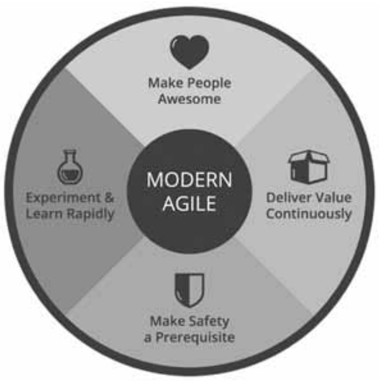
This approach provides a much more engaging environment for teams, who now feel responsible and accountable for the goals they help set, which they track on a fast weekly cycle.

Setting challenging goals is a fundamental tenet of OKR, which drives results and creativity. As **Amantha Imber reported,** research shows that if we put people in a role that challenges them, 67 per cent will demonstrate above-average creativity and innovation in their performance.

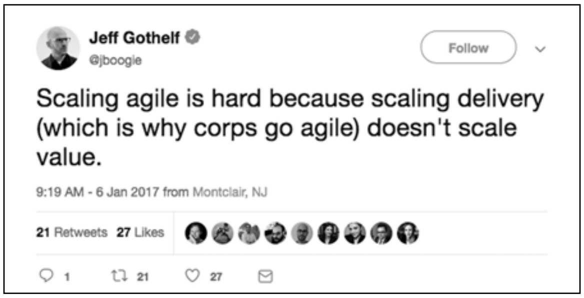
Dan Montgomery puts it well, "OKR is the day to day engine for organizational agility."

How can OKR support the four Modern Agile principles?

Deliver Value Continuously


Make People Awesome
Experiment & Learn Rapidly
MODERN AGILE
Deliver Value Continuously
Make Safety a Prerequisite

In Modern Agile we know that working software is not a measure of progress. While



Jeff Gothelf
@jboogie

Follow

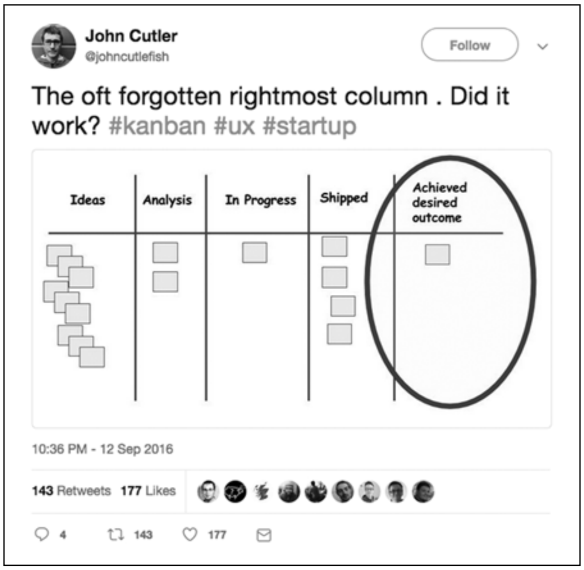
Scaling agile is hard because scaling delivery (which is why corps go agile) doesn't scale value.

9:19 AM - 6 Jan 2017 from Montclair, NJ

21 Retweets 27 Likes

1 21 27

an antiquated Agile mindset focuses on output-based metrics and concepts, such as the definition of done, acceptance criteria, burn-down charts, and velocity, Modern Agile knows that "done" only matters if it adds value.



John Cutler
@johncutlefish

Follow

The oft forgotten rightmost column . Did it work? #kanban #ux #startup

10:36 PM - 12 Sep 2016

143 Retweets 177 Likes

4 143 177

This old assumption that working software is a measure of progress rests on the belief that all software that works is valuable. Modern Agile teaches us to focus on continuously delivering real value to help make our customers awesome.

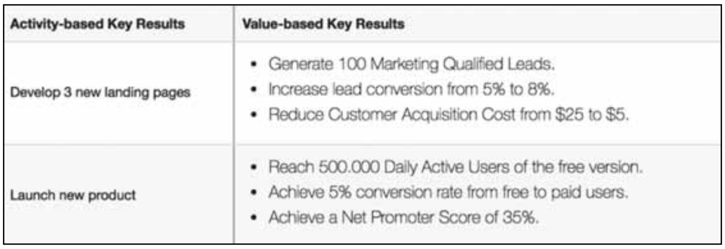
"The key to [defeating] waterfall is to realize that agilists value Outcomes over Features. The feature list is a valuable tool, but it's a means not an end. What really matters is the overall outcome, which I think of as value to the customers."

– **Martin Fowler**

Because it's just a framework, OKR can be used to measure outputs. The mere measurement of activities, however, is not a proper use of OKR and is incompatible with Modern Agile.

Practicing Modern Agile requires frequently setting and evaluating **Value-based OKRs,** which measure the delivery of value to the customer or the organization.

The two examples below clearly show the difference:



Activity-based Key Results	Value-based Key Results
Develop 3 new landing pages	• Generate 100 Marketing Qualified Leads. • Increase lead conversion from 5% to 8%. • Reduce Customer Acquisition Cost from $25 to $5.
Launch new product	• Reach 500.000 Daily Active Users of the free version. • Achieve 5% conversion rate from free to paid users. • Achieve a Net Promoter Score of 35%.

By adopting Value-based OKRs, teams can focus on delivering value. But how can they do that "continuously"?

Several years after the release of The Lean Startup, most organizations are still working for months without delivering anything to the end user. For them, continuous delivery is a distant dream. They are stuck with the old Agile delusion that showing software to stakeholders during a sprint review or demo is an adequate measure of progress.

The OKR quarterly cycle acts as the ultimate timebox to deliver value: every team has to deliver some value during the quarter. That way, teams move beyond acceptance criteria and the definition of done all the way into testing hypotheses and experimenting and learning rapidly.

Just as with any tool. OKR is not perfect and can be misused. We think that by using Modern Agile's four principles to guide your OKR practice, they can be a valuable and concrete starting point for your Modern Agile journey.

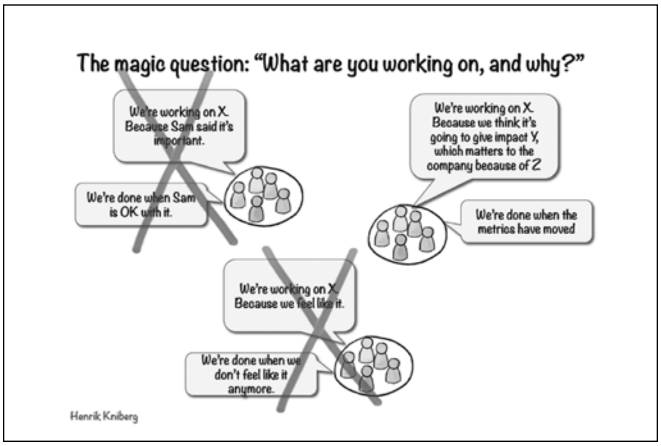
Make People Awesome

"If you're just using your engineers to code, you're only getting about half their value."
– Marty Cagan

The mindset that the team is incapable of deciding what to build is **toxic** and demotivating. The Modern Agile principle 'Make People Awesome' is grounded in providing people with opportunities to contribute their best ideas.

When your team has no voice regarding what to build and just empties their plate of backlog features one after another, they're not awesome.

To truly enable autonomous self-organizing teams, you need to give them the freedom to decide how to achieve the desired valuable outcomes. The role of the team has to change from: "delivering the features the stakeholders want" to "achieving the agreed Value-based OKRs."


The magic question: "What are you working on, and why?"
We're working on X. Because Sam said it's important.
We're done when Sam is OK with it.
We're working on X. Because we think it's going to give impact Y, which matters to the company because of Z
We're done when the metrics have moved
We're working on X. Because we feel like it.
We're done when we don't feel like it anymore.
Henrik Kniberg

Make Safety a Prerequisite

Following a fixed roadmap that lasts months or even years is a remnant waterfall behavior that still plagues several organizations that call themselves agile. They expose themselves to risk by having teams, in their most part only composed of developers, incrementally (and blindly) delivering (and to a staging environment, not production) a waterfall backlog, without any form of external validation.

"The only way it's all going to go according to plan is if you don't learn anything"
– **Kent Beck**

Furthermore, these plans are mostly devised by a single Product Owner or Manager, who doesn't even have access to production data that can help him/her understand the impacts of his/hers prioritization decisions. This sort of situation is demoralizing and unsafe. Mary Poppendieck, author of Leading Lean Software Development, wrote:

"Perhaps the biggest shortcoming of agile development practices is the way in which teams decide what to do. [...] for the longest time, answering these questions have not been considered the responsibility of the development team or the DevOps team."
– **Mary Poppendieck**

OKR Makes Safety a Prerequisite by ensuring that the teams collaborate on setting goals and deciding what to build (or experiment with) and adopt shorter feedback cycles, reducing risk and waste.

As **David J Bland wrote,** "[the process of] annual planning and budgeting collide with your efforts to adapt and change your roadmap as you learn in the market… Leaders are finally realizing that to make their organizations more agile, they'll need to start addressing some of these [fundamental] functions to achieve organizational agility."

Experiment & Learn Rapidly

It is only possible to Experiment & Learn Rapidly when we focus on outcomes and evidence rather than personal opinions. Outdated Agile is driven by the few stakeholders' definitions of what is valuable and accepted.

OKR replaces that subjectivity with measurable experiments that allow the team to learn and iterate. It enables teams to adopt practices such as Hypothesis-Driven Development, **as described by Barry O'Reilly:**

We believe <this capability>

Will result in <this outcome>

We will have confidence to proceed when <we see a measurable signal>

If our goal is a business outcome and we give the team the freedom to experiment towards that goal, small investments can lead to awesome results. In one such example, **a 20-minute feature tripled sales for 'Know Your Company',** while Eric Elliott delivered **"one Jira ticket that made his employer $1MM/Month".**

Conclusion

Just as with any other concrete planning framework, OKR is not perfect. Combining Modern Agile with the proper use of OKR can be a lightweight, joyful way for organizations to help their people achieve awesome results.

For more thoughts on Modern Agile and OKR, check-out **modernagile.org** and **felipecastro.com**

Thanks to Lael Gold, Joshua Kerievsky, Bill Wake and Tim Ottinger for early reviews.

⁂

To read this article online with embedded hypertext links, go here:
https://medium.com/the-alignment-shop/transcend-the-feature-factory-mindset-using-modern-agile-and-okr-c7a76f3c68b7

About Felipe Castro

Felipe is a business outcomes coach. He helps companies become more responsive, engaging, and aligned by adopting OKR, Silicon Valley's agile goal system. As an engineer who worked both as a product manager and as an HR consultant, Felipe bridges the gap between different functions and connects OKR with Agile, Lean, and product management to create outcome-driven teams. After training thousands of individuals, Felipe created the OKR Cycle, a straightforward approach to avoid OKR's most common pitfalls. He is the author of the upcoming book "Goals for Agile Organizations: Moving at Silicon Valley Speed with OKR." Felipe blogs at felipecastro.com.

About Alexandre Freire Kawakami

Alexandre is a programmer, coach, scientist, student and teacher who loves to push boundaries of agile software delivery. As Managing Director at Industrial Logic, he focuses on delivering quality software and services to solve our client's real world problems, and helps provide the tools needed to create a pathway to excellence for his team.

Five Lessons I'm Thankful I Learned in My Agile Career

By Mike Cohn

Since it's Thanksgiving week here in the United States, I took some time out of my schedule to reflect on some lessons I'm very thankful to have learned through my career. While these lessons are not unique to Scrum or even agile, each has been a big part of my success with agile.

For each lesson, I'll share what I learned and tell a brief story of how I learned it. In doing so, I'm hoping to help you avoid the mistakes I made before these lessons became second nature to me.

When There Are Two Ways to Do Something, Do It the Right Way

One of my first professional programming jobs was working in the litigation support division of one of the big consulting companies. Much of our work was driven by sudden demands from the opposition's attorneys. We developed software that helped our attorneys comply with those demands.

This often meant writing programs that would be used once, but that were needed within 24–48 hours and needed to be perfect. The boss who had hired me was a Unix shell scripting wiz and he'd managed much of the development with no concern for reuse. Every program was 100 percent new. He didn't even develop new programs by starting with old ones and extending or generalizing them.

This wasn't a bad decision necessarily. It was often the fastest way to comply with the attorney's needs. But over the longer term, I knew we could respond even more quickly if we started to assemble a library of common code. But would it be worth it? Did we even have the time to slow down 10 percent today to be 50 percent faster later?

Soon after joining the project, my boss was promoted and began working elsewhere on the same overall project. And I had a decision to make: continue as I had for the first two weeks on the project or focus on developing a reusable library.

I remember very clearly being in the office one Saturday with the two other programmers on the project, Sean and David. We discussed whether we should start building a reusable library or whether the urgency of deadlines was such that we had to stay focused on just getting things done. It turned out to be an easy decision. We agreed to start assembling the library every time we worked on something.

It was one of the best decisions I've ever been involved in making. The payback from that change came much faster than I ever expected because just a few months later we were faced with challenges we could not have met if we hadn't chosen reusability.

And so, the first thing I'm thankful for is that I learned the lesson:

When there are two ways to do something, do it the right way

The right way may seem more time-consuming or more difficult but in my experience, doing it the right way is always worth it.

Life Is Too Short to Work With People You Don't Like and Respect

One of the last jobs I had before Mountain Goat Software involved working in a highly dysfunctional culture. That culture was in place long before I got there, and, unfortunately, I didn't detect it during the interviews. I think perhaps I was so excited by the cool software I'd be involved in that I pushed the culture out of my mind and took the job.

It was horrible. I was one of five VPs reporting to our CEO. She decided one day that people needed to work more hours. Oh not because of some urgent deadline, just because. She assigned each of us VPs a different night of the week and we were expected to stay in the office until 7:00 P.M. so that employees would see us staying late. And that would motivate them to do so as well.

If you know me, you'll know I'm a complete workaholic because I love what I do. But I also have a mental screw loose. No matter how big the project is, I have a feeling that if we all stay late tonight — pull an all-nighter — we can finish the project by tomorrow. What? Linux rewritten? Let's work through the night and finish it! The rational part of me knows it can't happen, but it doesn't stop me sometimes from trying.

My point is, I work long hours. But I work them on my terms. I like finishing my day at a reasonable time--perhaps 5:00 P.M. and then exercising for a bit before having dinner with my family. And then I'll work more — often much more — later in the night.

But tell me I must stay until 7:00 P.M. and the anti-establishment part of me kicks in, and I want to rebel. Even though I was often in the office until 7:00, or later, the boss telling me I had to do it pissed me off. And I didn't like the message it was sending others. I would literally wait until the CEO left (normally no later than 5:15) and then I'd go tell anyone still in the office to go home.

The CEO wasn't the only dysfunctional person in that company. There were many. There was the architect who wiretapped the boardroom so he could listen in on meetings. There was the developer who claimed to be allergic to our building and went out on medical leave two days after starting. I could go on.

One day our CEO announced some new ludicrous policy — I don't even remember the specifics. I was in another VP's office when he read her email to me. We both looked at one another and came to the same conclusion:

Life is too short to work with people you don't like and respect.

We both decided we would never again work with people we didn't like and respect. It just isn't worth it. Within a month, we'd both left that company, and we haven't looked back. Today we each run successful agile coaching and training businesses. We've never been happier.

If you find yourself working with people you don't like and respect, work to get yourself out of that situation. You'll be much happier.

Removing Someone from the Team Never Hurts as Much as You Think It Will

Firing someone is never easy, even when it's for just cause. I had to fire one system engineer who was stealing hardware from our company and selling it online. The police had caught him fencing expensive hardware that he'd purchased for the company, was missing from our data center, and matched serial numbers from the vendors.

I had no doubt about his guilt. His trial was pending. Yet my boss was reluctant to support me in the decision that he needed to go. My boss's reasoning was, "Do you know what they do to pretty boys like him in prison?" Yes, he really said that to me and hesitated over doing the right thing because he'd seen too many movies. Still this guy had to go, yet it was hard to fire even someone like this. It's always hard to fire someone.

Sometimes firing someone is hard because the team has become highly dependent on that person as the only one who can do a certain job. And you're worried that if you fire that person, the team will be slowed dramatically.

My experience is that those fears are way overblown. In a couple of cases I had to fire someone who was the only person who knew how to do some vital thing or was the only person familiar with some very crucial code. But letting those individuals go and watching their teams quickly learn whatever needed to be learned taught me that.

Removing someone from the team never hurts as much as you think it will.

Teams are amazingly resilient. If there is something specific to be learned, they will dig in and learn it.

Also, by the time a manager realizes someone needs to be fired, gathers the energy to do it, and gets the human resources group on board with the idea, the team has thought the person should be fired for months. Teams typically realize these things much faster than managers.

Letting someone go should never be done without significant deliberation. But it's never as painful as you fear it might be.

The Smartest Person in the Room Is Not Smarter than the Whole Room

A lot of leaders work their way up their careers by being very good at what they do, having that be noticed by someone senior, and then getting promoted. And so many leaders and managers have for at least part of their careers been the smartest person in the room.

And when a decision needed to be made, they would state their opinion, defend it, win the argument over how to do things, and very often lead the team to the right decision. But no matter how smart the smartest person in the room is, it is highly unlikely that the smartest person is smarter than the collective wisdom of the entire team.

I learned this lesson early on in my career. I was a software team leader, which meant I had perhaps 3–5 more years of experience than the average person on my team. And, I probably was the smartest person the room when we were debating design decisions and such.

I remember one debate in which one of the more junior team members changed his opinion without much argument at all. I asked him why and he said that because I was the team lead, I must know best. And that scared me. Even when it might have been true in some cases, I never wanted my teammates backing off their positions and agreeing with mine just because of some job title I had.

Since then I've hated job titles. They are how we present ourselves to the outside world. Within a company, job titles should be meaningless.

I am positive that this junior developer did sometimes have better ideas than I did. And what a shame it would have been if he'd been reluctant to share because I had a fancier job title than he did.

This incident and other similar ones led me to learn that

The smartest person in the room is not smarter than the whole room.

No matter how smart the one person may be or how much experience that person might have, the collective wisdom of the team is greater. The best ideas and decisions will be born from the discussion rather than from the mind of just one person.

If You Don't Manage Expectations, Expect to Fail

I learned this next lesson from perhaps the least technically savvy boss I ever had. But he understood the importance of managing expectations.

We were building a large call center system that was to be used by nurses who worked in our company. The project ultimately was very successful and was instrumental in helping the company grow from 100 employees to over 1,600 employees within a couple of years.

But if I hadn't shifted my efforts a few months before the end of the project, that same project would have been viewed as a complete disaster.

The problem was that the nurse's expectations for what the software would do were through the roof. They had somehow started to believe that the software was going to do things that still aren't possible 20 years later. Some of the things they had just made up in their heads — and then shared among themselves — were amazing.

I was aware of this disconnect between expectations and reality. But I was too busy with the overall technical aspects of the project to worry about it. Until one day my boss gave me some advice that made me reconsider. He told me that to be successful, the project needed to do two things:

- provide the necessary functionality
- meet or exceed user expectations

He educated me that if we failed at either one, the project would be viewed as a failure. I knew right away that we could never meet or exceed the nurses' current, inflated expectations. Since I couldn't change our technical capabilities, I began working to change our users' minds.

I immediately shifted the focus of my time, spending about half of each week talking to the nurse users about what the system would and wouldn't do. I traveled to each of the company's four call centers around the United States every few weeks and presented the equivalent of a sprint review to nurses in each location.

I'd learned the lesson that

If you don't manage expectations, expect to fail.

If I had stayed focused purely on the technical delivery of that product, our users would have looked at it and said, "Is that all there is?" Their unrealistic (impossible!) expectations would have led them to be disappointed in what was actually a very good product — one which ultimately made billions of dollars for that company.

Giving Thanks for These Lessons

There are many more lessons I'm thankful to have learned. I often have to live through an experience a couple of times before the truth hits me. Each of the truths I've shared with you so far is something that I learned relatively early and that had a significant impact on my career.

But there's one last lesson that I need to share:

I couldn't do what I do if it weren't for you.

I wrote above that I love what I do, and that's why I'm a workaholic. Except most days it doesn't really feel like work to me. I couldn't do what I do if it weren't for the people who visit this blog or who subscribe to my weekly tips.

And for that I am very thankful to each of you.

What Lessons Are You Thankful to Have Learned?

What are you thankful to have learned in your career? Please share a lesson or story in the comments section below.

⁂

To read this article online with embedded hypertext links, go here:
https://www.mountaingoatsoftware.com/blog/five-lessons-im-thankful-i-learned-in-my-agile-career

About Mike Cohn

Mike is the author of three of best-selling books on agile (*User Stories Applied for Agile Software Development, Agile Estimating and Planning,* and *Succeeding with Agile*). He is also a Certified Scrum Trainer, keynote speaker, and an in-demand coach to companies throughout the United States and around the world.

As the founder of **Mountain Goat Software,** Mike specializes in helping companies adopt and improve their use of agile processes and techniques in order to build extremely high-performance development organizations. His weekly email **tips on succeeding with agile** are read by over 100,000 people.

With more than 20 years of experience, Mike has previously been a technology executive in companies of various sizes, from startup to Fortune 40. He is a co-founder of both the Scrum Alliance and the Agile Alliance, the home of the Agile Manifesto. He can be reached at mike@mountaingoatsoftware.com.

Change Artist Super Powers: Empathy

Esther Derby

Some people seem to think that empathy has no place at work…that work requires a hard-nose, logic, and checking your emotions at the door. But, in periods of change, emotions—which are always present, whether we choose to acknowledge them or not—surge to the surface. Ignoring the emotional impact of change doesn't make it go away. Rather, attempts to depress or devalue people's response to change may amplify emotions.

Empathy is the ability to recognize and vicariously identify someone else's experience and emotions. Empathy enables you to understand someone else's point of view, the challenges posed by the change, what they value, and what they stand to lose by changing.

Empathizing doesn't mean you have to feel the same thing, think the same way, make the other person feel better, or fix the situation so everyone is happy. Demonstrating empathy means you listen, acknowledge, and accept feelings and points of view as legitimate. Empathy is fundamentally about respect.

Three kinds of empathy play a part in change.

Emotional empathy, understanding another's emotions and feelings. This is what usually comes to mind first when people hear the term. Emotions are a normal part of change—from excitement, to grief, puzzlement, loss of confidence, and anger. Too often, people who "drive" change dismiss these responses and urge people "just get on with it."

Cognitive empathy means understanding someone else's patterns of thought and how he makes sense of his world and events. Understanding how others think about things may help you frame a new idea in a way that meshes with their views. That also helps you—you'll know more about the obstacles and issues you are likely to encounter.

Point-of-View empathy combines a bit of both of these, and it allows you to say genuinely, "I can see how it looks that way to you." Once you extend that courtesy to someone, he is more likely to want to see how the situation looks to you.

Empathy provides information that helps with change in at least two ways:

You can refine your ideas about the change based on local information, which people are more likely to share when you make an effort to listen and connect with them.

People are more likely to listen to you when they feel listened-to.

The more you listen, the more you learn about the needs and values of the people facing a change. And that is the key: People rarely change because someone has a bright new idea. They change to save something they value. But you won't learn that unless you empathize.

⁂

To read this article online with embedded hypertext links, go here:
https://www.estherderby.com/2017/04/change-artist-super-powers-empathy.html

About Esther Derby

Esther started her career as a programmer, and over the years has worn many hats, including business owner, internal consultant and manager. From all these perspectives, one thing became clear: our level of individual, team and company success was deeply impacted by our work environment and organizational dynamics. As a result, Esther has spent the last twenty-five years helping companies design their environment, culture, and human dynamics for optimum success.

You can reach Esther at esther@estherderby.com and follow her on twitter @estherderby

Agile Coaching: An Awful Truth

By Bob Galen

[**Note:** grayed text indicates hypertext links in original article.]

I attended an impromptu agile coaches gathering about a year or more ago. It was a "coaching the coaches" session and it was very valuable. But an aspect of it has stuck with me ever since. One that I've mulled over and over and would like to share.

There were a group of coaches in attendance from the same client engagement, a large, multi-billion-dollar organization that had been going Agile for a couple of years.

When they decided to go agile, one of the first things the client did was reach out to an agile coaching firm for help. On the surface, that sounds like a good thing to do. However, the firm was largely staff augmentation focused, so that was their background and comfort zone.

They reacted like they would for any similar engagement. They recruited 10 disparate agile coaches, minimally vetted their experience, and aggressively negotiated their rates. Then they negotiated a global agreement with the client and on-boarded the coaches.

There was no engagement strategy nor much consistency across the various coaching approaches. There was also no coaching team. Instead, there was simply a group of coaches thrown into a very lucrative situation. And as coaches are wont to do, they started coaching…

Rates

Let's take a diversion to approximate the cost of this endeavor. While I'm not privy to the exact rates, I know the ballpark. Each coach was probably signed up for ~$1,200 / day while the client charge rate was ~$2,500 / day.

The run-rate for each coach was ~$625,000 annually. For ~10 coaches, the firm was paying ~$6M per year. For a 2-year engagement, the total cost was approximately $12M - $15M, including coaching, certifications, and other training.

That's sort of money should inspire and create phenomenal results, right?

Teams

The client quickly ramped from zero Scrum teams to about 150 Scrum teams. So, the coaches played a significant part in quickly scaling up the organization's teams.

Their primary focus was downward to the teams. If you measured their success by how many teams were spun up and how quickly that was done, then they were quite successful.

Ultimate coaching costs per team were ~$100,000.

Back to the Coaches

But let's back to the clients' coaches in our meeting. To a person, they were sad.

It seemed while they were largely successful in getting teams on-board with agile, they realized it wasn't enough to transform the organization.

They learned (and many had known before they joined) that you can't transform an organization at a team-only level, that any solid transformation needed the full engagement and participation of management and leadership.

Haunted

Part of the sadness at the meeting was the coaches were approaching the end of their engagement. The client organization felt that their value proposition had declined and the initial goal of achieving agile had been accomplished.

But the coaches knew differently. While the teams had been assimilated, the organization's leadership style remained the same. And the overall pre-agile culture remained the same.

In other words, the agile teams were largely alone in their environment with no amount of leadership, management, or true cultural support. The coaches knew that the teams fledgling efforts would eventually revert to their previous approaches, that they would not stand the test of time.

Being professional coaches, they were quite sad about their efforts not resulting in sustainable change. They seemed to be wracked by questions like:

- Why wasn't there on overarching coaching strategy at the beginning?
- Why weren't we hired as and formed into a team for the engagement?
- Why wasn't there more of an on-site coaching leadership presence?
- Why didn't we challenge management and leadership more to engage and be a part of the transformation?
- Why didn't we intervene when the organization clearly misunderstood the nature of an agile transformation?
- Why did we continue to coach aggressively downward, when we knew that upward was the better direction?
- And most daunting, why did we continue to coach when we knew we weren't making an impact in the best interest of the client's goals? Why

didn't we leave instead of just cashing our checks and going through the motions?

And to be fair, it wasn't just the coaches who should have been asking these questions. Their firm should have been doing so as well. Especially since they were driving the overarching engagement strategy (or lack thereof) for this client's agile transformation engagement.

In the End, A Tremendous Waste

The reason I brought up the funding model, was to show the incredible investment the client made in this effort. But it all seemed for naught.

In the end:

- The coaches felt like they had failed their *Prime Directive,* to coach an organizational-wide agile transformation. And they did fail.
- The organization felt that they had done what was asked of them. They went agile. But from an impact perspective, they all knew that very little in the way of significant change (outcomes, performance, quality, culture) had changed. They had also failed.
- And they had spent $15M in the process, for essentially another failed initiative.

From my perspective, this is an example of an incredible waste of effort, time, and funding. And it could have all been avoided with a much different strategy and approach.

Now I've joined the mood of those coaches. This entire tale makes me SAD! And what's even SADDER is this is not a unique outcome. This happens incredibly often in agile transformations.

I've shared this tale so that you might avoid a similar outcome. Here are a few related posts that might be helpful to plot a different journey.

- **http://rgalen.com/agile-training-news/2014/6/9/agile-coaches-were-coaching-the-wrong-people**
- **http://rgalen.com/agile-training-news/2014/7/21/coaching-leadership**
- **http://rgalen.com/agile-training-news/2014/11/23/agile-coaches-trainers-have-you-walked-in-the-shoes-of-technical-management**

One where you, as an agile coach, take a much more balanced and effective approach in your organizational coaching. Where you establish a leadership partnership early-on that trusts and engages your coaching at all levels of the organization. Where you spend more time "coaching UP" than you do "coaching DOWN".

Or where and when this doesn't happen, you consider congruently moving onto greener coaching pastures.

Stay agile my friends!

⁂

To read this article online with embedded hypertext links, go here:
http://rgalen.com/agile-training-news/2017/2/7/agile-coaching-an-awful-truth

About Bob Galen

Bob is an Agile Methodologist, Practitioner & Coach based in Cary, NC. In this role, he helps guide companies and teams in their pragmatic adoption and organizational shift towards Scrum and other agile methodologies and practices. He is Director, Agile Practices at Zenergy Technologies, a leading agile transformation company. He is also President and Head Coach at RGCG.

Bob regularly speaks at international conferences and professional groups on topics related to software development, project management, software testing, and team leadership. He is a Certified Enterprise Coach (CEC), Certified Scrum Product Owner (CSPO), and an active member of the Agile & Scrum Alliances.

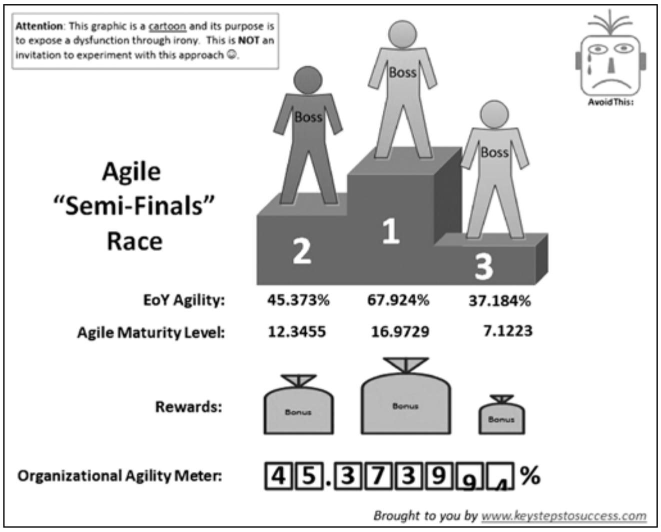
He's published three agile focused books: *The Three Pillars of Agile Quality and Testing* in 2015*, Scrum Product Ownership,* in 2009 — 2'nd Edition in 2013, and A*gile Reflections* in 2012. He's also a prolific writer & blogger (www.rgalen.com) and podcaster (at www.meta-cast.com).

Bob may be reached directly at bob@rgalen.com

Addressing Problems, Caused by AMMS

By Gene Gendel

[**Note:** grayed text indicates hypertext links in original article.]


Attention: This graphic is a cartoon and its purpose is to expose a dysfunction through irony. This is NOT an invitation to experiment with this approach ☺.
Avoid This:
Agile "Semi-Finals" Race
Boss
Boss
Boss
2
1
3
EoY Agility: 45.373% 67.924% 37.184%
Agile Maturity Level: 12.3455 16.9729 7.1223
Rewards:
Bonus
Bonus
Bonus
Organizational Agility Meter: 45.37394 %
Brought to you by www.keystepstosuccess.com

Nowadays, for too many organizations, **Agile Maturity Metrics (AMM)** have become a trusted way to measure improvements of agility at personal (individuals), team and organizational level.

However, it is not always apparent to everyone that AMMs are different from Agile Check-Lists (e.g., **classic example** of Scrum Check list by H. Kniberg) and this can often lead to problems and dysfunctions:

Check-Lists are just a set of attributes that are usually viewed on-par with one another; they are not bucketed into states of maturity (other logical grouping could be applied though)

On contrary, AMMs place attributes in buckets that represent different states of maturity, with one state, following another, sequentially.

With very rare exceptions (favorably designed organizational ecosystems), there are three potential challenges that companies face, when relying on bucketed AMMs:

1 – System Gaming: If achieving a higher degree of agile maturity is coupled with monetary incentives/perks or other political gains (for many companies that are driven by scorecards and metrics, this is the case), there is will be always attempts by individuals/teams to claim successes/achievements by 'playing the system', in pursuit of recognition and a prize.

Attention: This graphic is a cartoon and its purpose is to expose a dysfunction through irony. This is NOT an invitation to experiment the presented case ☺.
Avoid This:
GOAL:
45.3734999 % Agile
by EoY!!!!
(Пере)выполним годовой план
за три квартала !!!
Brought to you by www.keystepstosuccess.com

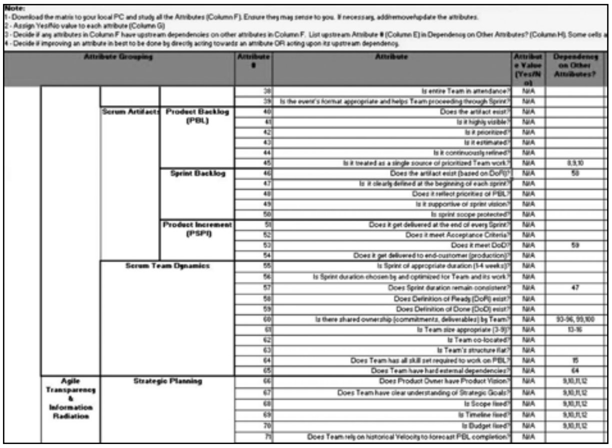
Note: Translation of the text in gray: *"(Пере)выполним годовой план за три квартала!!!"* = "Will meet/exceed the annual plan in three quarters!!!"

2 – Attribute-to-Maturity Level relationship is conditional, at most: Placing agile attributes in maturity buckets implies that attributes in higher-maturity buckets have more weight than attributes in lower-maturity buckets. However, this is not always a fair assumption: weight/importance that every organization/team places on any given attribute, while defining its own maturity, is **unique to that organization/team.** *For example, for one team, "...being fully co-located and cross-functional..."* could be much more important than *"...having Product Owner collocated with a team..."* For another team, it could be the other way around.

3 – Correlation between attributes is not linear, at system-level: Regardless of buckets they are placed in, many agile attributes are interrelated systemically and impact one another in ways that is not apparent, to a naked eye. For example, placing *"Scrum Master is effective in resolving impediments"* attribute in a maturity bucket that comes before the maturity bucket with *"...Organization provides strong support, recognition and career path to Scrum Master role..."* attribute, dismisses the real cause-and-effect relationship between these two variables, misleads and sets false expectations.

To avoid the issues described above, it would be more advisable to treat every identified agile attribute as a system variable, that is on-par with other system variables, while assuming that it has upstream and downstream relationship. In many situations, instead of spending a lot time and resources on trying to improve a downstream variable (e.g., trying to understand why it is so difficult to prioritize a backlog) it is more practical to fix an upstream variable that has much deeper systemic roots (e.g. finding an empowered and engaged product owner who has as the right to set priorities).

Below, is the list of agile attributes (a.k.a. system variables) that are logically grouped (check-list) but **are not** pre-assigned to levels of maturity (all flat). Some examples of suggested system-level correlation between different attributes are provided (cells are pre-populated).



Note:
1 - Download the matrix to your local PC and study all the Attributes (Column F). Ensure they may sense to you. If necessary, add/remove/update the attributes.
2 - Assign Yes/No value to each attribute (Column G)
3 - Decide if any attributes in Column F have upstream dependencies on other attributes in Column F. List upstream Attribute # (Column E) in Dependency on Other Attributes? (Column H). Some cells a
4 - Decide if improving an attribute in best to be done by directly acting towards an attribute OR acting upon its upstream dependency.

Attribute Grouping				Attribute #	Attribute	Attribute Value (Yes/No)	Dependency on Other Attributes?
				38	Is entire Team in attendance?	N/A	
				39	Is the event's format appropriate and helps Team proceeding through Sprint?	N/A	
		Scrum Artifacts	Product Backlog (PBL)	40	Does the artifact exist?	N/A	
				41	Is it highly visible?	N/A	
				42	Is it prioritized?	N/A	
				43	Is it estimated?	N/A	
				44	Is it continuously refined?	N/A	
				45	Is it treated as a single source of prioritized Team work?	N/A	8,9,10
			Sprint Backlog	46	Does the artifact exist (based on DoR)?	N/A	58
				47	Is it clearly defined at the beginning of each sprint?	N/A	
				48	Does it reflect priorities of PBL?	N/A	
				49	Is it supportive of sprint vision?	N/A	
				50	Is sprint scope protected?	N/A	
			Product Increment (PSPI)	51	Does it get delivered at the end of every Sprint?	N/A	
				52	Does it meet Acceptance Criteria?	N/A	
				53	Does it meet DoD?	N/A	59
				54	Does it get delivered to end-customer (production)?	N/A	
		Scrum Team Dynamics		55	Is Sprint of appropriate duration (1-4 weeks)?	N/A	
				56	Is Sprint duration chosen by and optimized for Team and its work?	N/A	
				57	Does Sprint duration remain consistent?	N/A	47
				58	Does Definition of Ready (DoR) exist?	N/A	
				59	Does Definition of Done (DoD) exist?	N/A	
				60	Is there shared ownership (commitments, deliverables) by Team?	N/A	93-96, 99,100
				61	Is Team size appropriate (3-9)?	N/A	13-16
				62	Is Team co-located?	N/A	
				63	Is Team's structure flat?	N/A	
				64	Does Team has all skill set required to work on PBL?	N/A	15
				65	Does Team have hard external dependencies?	N/A	64
	Agile Transparency & Information Radiation	Strategic Planning		66	Does Product Owner have Product Vision?	N/A	9,10,11,12
				67	Does Team have clear understanding of Strategic Goals?	N/A	9,10,11,12
				68	Is Scope fixed?	N/A	9,10,11,12
				69	Is Timeline fixed?	N/A	9,10,11,12
				70	Is Budget fixed?	N/A	9,10,11,12
				71	Does Team rely on historical Velocity to forecast PBL completion?	N/A	

*Please, go to this link (***http://www.keystepstosuccess.com/wp-content/uploads/2017/10/KSTS_AMM.xlsx**) to *download the matrix to your desktop, amend the list of attributes if you feel that your situation calls for modification, and then use "Dependency on Other Attributes?" column to better visualize system-level correlation between the attributes are of interest to you and other related attributes (some examples are provided).*

***

To read this article online with embedded hypertext links, go here:
http://www.keystepstosuccess.com/2017/10/addressing-problems-caused-by-amms/

You Get What you Ask For: Agile Coaches–"Centaurs"

By Gene Gendel

[**Note:** grayed text indicates hypertext links in original article.]

Why are there so many **troubled agile** "transformations"? We frequently hear the following answer: "because companies lack senior leadership support". True. And let's not trivialize this: without strong and genuine support by senior leadership (beyond slogans and "support in spirit"), without selecting a deep, systemic approach to problem resolution, companies can only expect localized, peripheral and, most likely, short-term improvements.

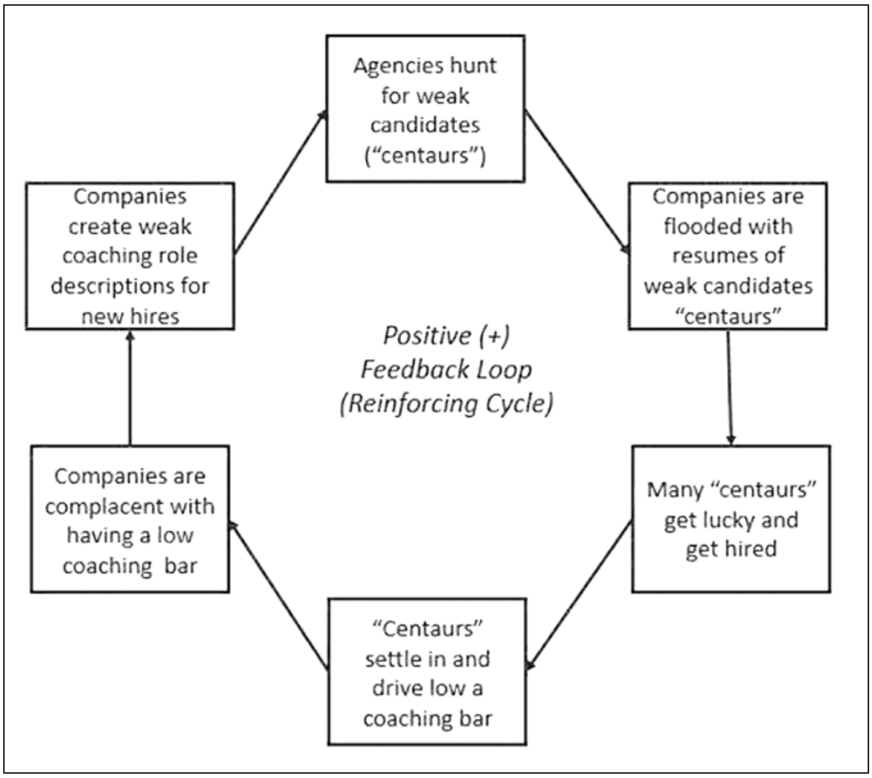
But is there anything/anyone else that can be conveniently held accountable for failed agile transformations?

How about ineffective agile training and coaching? *[Note: If you are interested in learning more about some of the most common challenges with agile training, please visit* **this page.** *This post is about coaching.]*

...There is a vicious cycle that hurts so many companies (can be also considered as a self-inflicted wound):

➞initially, companies set a low bar for coaches, based on poor understanding of a coaching role ➞low quality coaches (**quasi-coaches-"centaurs"**) are hired, most of whom are not even coaches, but rather people that have mastered agile jargon and know how to impress HR and uninformed hiring managers ➞weak coaches (most of whom have minds of conformists, not challengers) cannot effectively guide companies to fix systemic weaknesses and dysfunctions ➞teams and departments don't really improve; rather create a superficial appearance/illusion of progress (often, to impress senior management) ➞companies lose faith and stop seeing value in coaching ➞companies start trivializing a coaching role ➞ companies decide not to spend more money on high quality coaching ➞ cheaper, even less effective, coaches are hired (or internal, misplaced people are refurbished into coaches, overnight, as per Larman's Law # 4) ➞ initially, low-set coaching bar, is lowered even further...and so on....

Graphically, it looks something like this:


Agencies hunt for weak candidates ("centaurs")
Companies are flooded with resumes of weak candidates "centaurs"
Many "centaurs" get lucky and get hired
"Centaurs" settle in and drive low a coaching bar
Companies are complacent with having a low coaching bar
Companies create weak coaching role descriptions for new hires
Positive (+) Feedback Loop (Reinforcing Cycle)

As a result, what was initially meant as a strategic organization- improvement effort, now takes on a form of just another system-gaming change management fad that ultimately leads to a failure and responsibility/blame-shifting.

What are some of the reasons why the above happens? Here are some suggested reasons:

- Companies don't understand the **essence of agile coaching role:** it is viewed as another "turn-on switch" management function
- Leadership does not feel a **sense of urgency (p. 14)** to make changes and exempts itself from being coached: people are too busy and too senior to be coached; they find coaching trivial
- Certain organizational pockets are genuinely resistant to/feared of changes that can be brought about by real coaches (as per **Larman's Laws 1 – 3**)
- Market over-saturation with unskilled recruiters that hunt for low-quality coaches and contribute to the above cycle: this further lowers a company's chances to find a good coach
- This list can be extended….

Who is responsible for initiating this vicious cyclic dysfunction? Does it really matter if we identify guilty ones? Maybe it does, but only, as a lessons-learning exercise. What probably matters more is how to break out of this cycle. Where to start: discontinue low-quality supply (coaches) or raise a bar on demand (by companies)? Usually, demand drives supply and if so, ***for as long as companies remain complacent and reliant on outlived staffing/head-hunting approaches, cold-calling techniques, and ineffective HR-screening processes, performed by people that poorly understand the essence of an agile coaching profession, while trying to procure cheap "agile" resources or treat seasoned professional coaches, as "requisitions to be filled", a coaching bar will remain low, and companies will be getting EXACTLY what they have paid for: coaches-centaurs.***

Big Question

"What should companies be looking for when hiring a coach?"

An organization should be looking much father and beyond of what is typically presented in a resume or a public profile of a candidate: usually, a chronological list of an employment history or a long list of google-able terms & definitions, popular jargon or claims of experience in resolving deep, systemic organizational challenges with Jira configurations . Much more attention should be paid to the following important ***quantitative*** characteristics of a coach:

Coaching Focus: What is an approach and/or philosophy to coaching does a coach have? This will help a company understand an individual mindset of a coach.

Coaching Education AND Mentorship: What active journey through education, mentorship and collaborative learning in coaching and related activities over significant period has a coach taken?

Formal Coaching Education: What has contributed significantly to a person's coaching journey, including courses on topics of facilitation, leadership, consulting, coaching, process, and other related activities which have influenced a person's coaching practice? Such education may not have to be degree-related (training and/or certification from any recognized institution could be sufficient).

Coaching Mentorship & Collaboration: How a coach developed a skill/technique or received guidance to a coaching approach and mindset? Respect and recognition of mentors — matters here.

Informal Coaching Learning: What important topics outside of Agile/Scrum literature have impacted a person's coaching philosophy? This increases chances that a coach is well-rounded, beyond standardized book learning.

Agile Community Engagement & Leadership: Does a coach engage in agile user groups, gatherings, retreats, camps, conferences, as well as writing, publishing, reviewing, presenting, facilitating, training, mentoring, organizing, and leading agile events? An active participation and leadership in the agile community is a good demonstration that a coach has not developed herself within a unique organizational silo, by self-proclaiming and self-promoting, but rather has diverse and 'tested' industry experience.

Agile Community Collaborative Mentoring & Advisory: Does a coach mentor or advise other individuals (not for pay) on how to increase their competency or development? Is a relationship on-going, purposeful and bi-directionally educational?

Coaching Tools, Techniques and Frameworks: Does a coach develop awareness and understanding of tools, techniques and frameworks while engaging with organizations? Has she customized or developed anything that was client/engagement-specific?

In addition to quantitative characteristics, here are **qualitative** characteristics of a good coach:

Coaching Mindset

- How does a coach react when an outcome of coaching was different from what she had desired? In the past, how did a coach address this situation?
- How, based on clients' needs, a coaching mindset had to change? In the past, what compromises did a coach make? What was learned?
- What new techniques or skills did a coach learn, to meet a client's needs?

Coaching Competencies

- Assess – Discovery & Direction
- Balance – Coaching & Consulting
- Catalyze – Leadership & Organizations
- Facilitate – Focus & Alignment
- Educate – Awareness & Understanding

Coaching Specialties

- Lean / Kanban
- User Experience / Design
- Scaling Agile / Enterprise Agility
- Technical / Quality Practices
- Organizational Structures
- Lean Startup
- Product / Portfolio Management
- Organizational Culture
- Learning Organizations
- Non-Software Application
- Business Value / Agility
- Technical / Product Research
- Multi-Team Dynamics
- Organizational Leadership
- Organizational Change

*[**Note:** The above, is based on guidelines provided by Scrum Alliance application process for CTC and CEC.]*

While running some risk of sounding self-serving (very much NOT! the intent here): please, be mindful and responsible when you select guidance-level professionals in your agile journey.

⁂

To read this article online with embedded hypertext links, go here:
http://www.keystepstosuccess.com/2017/07/you-get-what-you-ask-for-agile-coaches-centaurs/

What Should Agile Leadership Care About?

By Gene Gendel

Agile frameworks (e.g., Scrum, Kanban, XP), individuals' roles & responsibilities, processes & tools, metrics & reporting, burn-up charts, estimation techniques, backlog prioritization, agile engineering practices, agile maturity models etc. — all of them are important attributes of a typical agile transformation. However, NONE of them are first-degree-of-importance system variables that are responsible for transformation success. Most of them, are good superficial lagging indicators of agility but they are all corollary (secondary and tertiary) to another much more important system variable.

What is the most important system variable that defines a company's agility? It is **Organizational Design** — the most deeply rooted element of organizational ecosystem that defines most of system dynamics.

When organizational leadership decides to take an organization through an agile transformation journey (it could take years, sometimes), it [leadership] needs to acknowledge that real, sustainable agile changes are only possible if deep, systemic organizational improvements are being made. For that, leadership needs to be prepared to provide to its organization much more than just *support in spirit,* accompanied organizational messages of encouragement and statements of vision. Leadership must be prepared to intimately engage with the rest of an organization, by doing a lot of real **"gemba"** (genchi genbutsu (現地現物)) and change/challenge things that for decades, and sometimes for centuries, have been treated as de-facto.

What does it really mean for leadership to engage at System Level? First, it is important to identify what a system is: what are a system's outer boundaries? For example, one of the most commonly seen mistakes that companies make when they decide on "scope of agile transformation" is limiting its efforts to a stand-alone organizational vertical, e.g. Technology – and just focusing there. Although this could bring a lot of local (to IT) success, it may also create unforeseen and undesirable friction between the part of an organization that has decided to change (IT) and the part of an organization that decided to remain 'as is' (e.g. Operations, Marketing). For example, if Scrum teams successfully adopt CI/CD, TDD or other effective engineering practices that enable them deliver PSPI at the end of every sprint, but business is not able to keep up with consumption of deliverables (too many approvals, sign offs, red tape) then the whole purpose of delivering early and often gets defeated. Then, instead of delivering to customers soon, in exchange for timely feedback, teams end up delivering in large batches and too far apart on a time scale.

A successful Agile Leader must treat an organization, that is expected to transform, as a **sushi roll.** Just like seaweed alone does not provide a full spectrum of flavors and does not represent a complete, healthy meal, one single department (e.g., IT) is not sufficient enough to participate in agile transformation efforts. Other organizational layers need to be included as well, when identifying a slice for agile transformation experiment. A slice does not have be too thick. In fact, if organizational slice is too thick, it might be too big to "swallow and digest". But still, even when sliced thinly, an organization must include enough layers, to be considered as a 'complete meal'.

Note: A great example of treating an organization as a sushi role, while making it more agile, is **Large Scale Scrum (LeSS)** adoption.

So, what are some key focus areas that every Agile Leader must keep in mind, while setting an organization on agile transformation course?

- Location strategies. Geographic locations.
- HR policies (e.g. career growth opportunities, compensation, promotions)
- Budgeting & Finance
- Intra-departmental internal boundaries and spheres of influence
- Organizational Leadership Style
- And some other areas that historically have been considered as …untouchable…

All the above listed areas are defined by Organizational Design and can be better understood through **self-assessment,** done by organizational leaders at all levels.

***

To read this article online with embedded hypertext links, go here:
http://www.keystepstosuccess.com/2017/09/what-should-agile-leadership-care-about/

“Who are the Judges?” Who Decides on Who is Gonna Coach?

By Gene Gendel

[**Note:** grayed text indicates hypertext links in original article.]

Lets kick off this post with the quote from another recent discussion that generated a number of strong comments from experienced professionals:

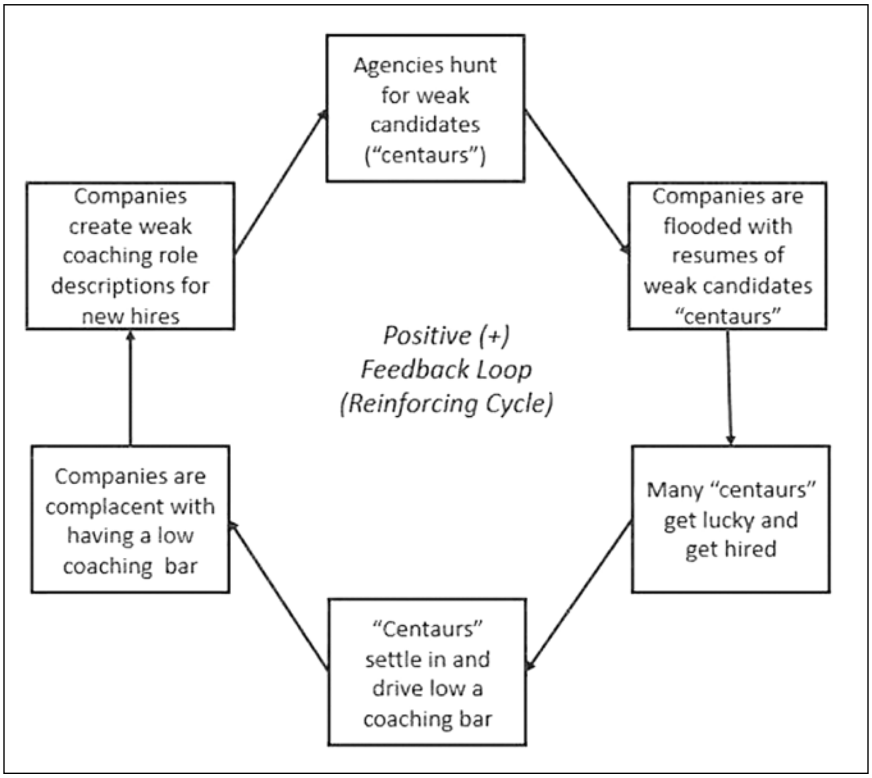
> *“…as long as companies remain complacent and reliant on outlived staffing/head-hunting approaches, cold-calling techniques, and ineffective HR-screening processes, performed by people that poorly understand the* ***essence of an agile coaching profession, while trying to procure cheap “agile” resources (using “preferred vendor lists”) or treat seasoned professional coaches, as “requisitions to be filled”, a coaching bar will remain low, and companies will be getting EXACTLY what they have paid for:*** **coaches-centaurs**“

To summarize, the purpose of the above referenced discussion was to increase awareness about implications of ineffective coaches and coaching that exists in abundance today. Here, lets look at some root causes why this problem exists.

Who Defines the role of Agile Coach?

For the most part, organizational understanding of a coaching role is weak. Definitions of a coaching role that flow around, suggest that companies are still confused about what coaches do. Definition of a coaching role is frequently lumped together with the role of a project manager, team lead, business analyst, Jira/Rally/Version-One administrator etc. While some of these other roles, could represent potentially relevant past experience for a coach, lumping all of them together in one all-inclusive role description, delimiting them by a commas or forward slashes, is ironic, to say the least. Many of these “ace pilot/submarine captain/NHL star” roles create a conflict of interest not just for people that step into them but for everyone else who gets affected by interaction. Very often, inaccurate definition of a coaching role leads to inappropriate behaviors by a coach, such as attempts to seek authority and organizational power, exhibition of command & control behavior, competition with people being coached for ownership of deliverables, monetary incentives and other perks.

Once a poorly-defined coaching role description hits the street, it enters a vicious cycle — reinforcing feedback loop. (described in detail **here**):


Agencies hunt for weak candidates ("centaurs")
Companies are flooded with resumes of weak candidates "centaurs"
Many "centaurs" get lucky and get hired
"Centaurs" settle in and drive low a coaching bar
Companies are complacent with having a low coaching bar
Companies create weak coaching role descriptions for new hires
Positive (+) Feedback Loop (Reinforcing Cycle)

*(**Note:** the above illustration excludes other system variables that may have effect on the variables and variables' relationships shown above).*

This vicious cycle usually leads to one inevitable result: over-time (usually months; sometimes a few years) companies realize that agile coaching did not bring about enough sustainable organizational improvements, as it was expected. This further leads to two outcomes, both of which dependent of senior leadership vision and goals:

- Companies seriously re-assess their own initial actions, acknowledge mistakes made, and then improve coaching standards and elevate the bar in favor of real, experienced coaches
- Companies, try to water down mistakes they have made, trivialize a coaching role for a lack of it's benefit and, and by doing so, further reinforce the loop above

Who Really Makes Decisions and Why?

Rarely, senior executives take an active role in a coaching hiring process; exceptions exist but they are rare (usually, exceptions are seen when things become very urgent – **page 14**). But even when they [executives] do engage in the process, it is usually more the act of a formality, to ensure that a hired person "fits the culture". Of course, and very ironically, one of the key expectations from an experienced coach should be to challenge an organizational structure (both, at enterprise and team level), and since culture is corollary to structure (**Larman's Law # 5**), the latter would change (would be challenged) as well. But this is not something that too many senior executives would like to hear.

For the most part, a hiring process is delegated to first- and sometimes second-line management, as well as internal agile champions that oversee and own agile transformations. While Larman's Law # 1, historically, has defined the attitude of middle management towards fundamental changes that challenge a status-quo, the recently added Law # 4 neatly describes "contribution" by some internal agile champions. And while exceptions do exist, trends and statistics speak louder.

Let's imagine the process by which an organization wanted to hire an agile coach (as employee or consultant — no difference):

In this process, the interviewers – are individuals described in **Larman's Law # 1 and # 4.** On the other hand, an interviewee, is a seasoned agile coach, with long enterprise- and team-level track record: she is a system thinker, dysfunctions challenger, a real organizational change agent.

Impact on a hiring process by Larman's Law # 1-type Interviewers

At an interview, a coach-candidate meets with first- and/or second-line managers that also expect that a coach will report into them, when she joins a company. During a discussion, interviewers hear from a coach certain things that coaches usually bring up, uninhibitedly:

- Simplified overall organizational structure, where developers receive requirements and communicate on progress, by interacting directly with end customers, not middle-men
- Flattened team structure, where developers self-organize and self-manage. Overall reduction of supervision and resource management, in favor of increased autonomy, mastery and purpose, by individuals that do work
- Harmful effects of **individual performance appraisals** and subjective monetary incentives, especially in environments, where team commitments and team deliveries are expected

Unsurprisingly, the biggest question that many interviewers walk out with, after interviewing such a candidate is: *"What will my role be like, if this coach is hired and brings about above mentioned organizational changes?"*

Impact on a hiring process by Larman's Law # 4-type Interviewers

Knowledge and experience of a coach-candidate supersedes that of internal agile champions and process owners. Some of the discussions a coach elicits, and answers provides, by far exceed expectations (not to be confused with a term used in a **performance appraisal process**) of her interviewers. Some suggestions and ideas shared by a candidate are a great food for thought for senior executives but not at a level, where Larman's Law # 4-type coaches are authorized to operate. Interviewers clearly see that a coach-candidate, if on-boarded, soon may become a more visible, influential contributor than the interviewers themselves. A coach may also bring about some organizational turbulence that will take out of comfort zone some individuals that are resistant to changes.

What are the odds that this experienced coach-candidate will be given a "pass"? What are the odds that she will be even given a chance to speak to senior executives involved in a hiring process, to attempt to influence them, to open their eyes, to offer a deeper system perspective on a situation, to make them think and talk about the forbidden?

Slim-to-none.

And this is one of the ways, in which organizations that are complacent about agile improvements, shoot themselves in a foot: they very effectively disqualify qualified agile coaches and by doing so, reinforce the feedback loop illustrated above.

***

To read this article online with embedded hypertext links, go here:
http://www.keystepstosuccess.com/2017/08/who-are-the-judges-who-decides-on-who-is-gonna-coach/

About Gene Gendel

Gene is an Agile Coach, Trainer and Organizational Design Agent. Gene is a proud member of the small community (about **94 people worldwide**) of **Scrum Alliance Certified Enterprise Coaches (CEC).** Today, he is the only CEC who resides in NY State. Gene's goal is to help organizations and individual teams with improving internal dynamics, organizational structure and overall efficiency. He strives to engage at all organizational levels: senior- and mid-level management, teams and individuals. In his work, Gene uses various methods, tools and techniques to strengthen learning of others and to ensure that teams and individuals gain autonomy after he "coaches himself out of the job". Throughout his long career, Gene has served small, mid-size and large companies, domestically and abroad.

Gene is a well-recognized member of global and local agile communities, where he influences people via open-space agile collaboration workshops, coaching retreats, group events and presentations.

Gene is a well-recognized blogger and publisher. He is the co-author of the book *Agile Coaching: Wisdom from Practitioners* (**free pdf**). His collection of personal essays ("The Green Book") can be also found **here.**

Gene strongly supports Scrum Alliance (SA) in its efforts of "transforming the world of work". He is an active member of SA working group of coaches and trainers that have been involved in improving SA certification/education programs, by aligning them with natural career paths of agile professionals: **Team Level Coaching Certifications (CTC)** (Gene is also one of co-creators of the program) and **Enterprise Level Coaching Certifications (CEC)**. Through these efforts, Gene tries helping highly qualified individuals to differentiate themselves from **lowered standards.**

An Agile Approach to Software Architecture

By Gene Gotimer

[**Note:** grayed text indicates hypertext links in original article.]

Summary

For an organization transitioning to agile development, creating software architecture isn't incompatible with your new processes. Consider the principles in the Agile Manifesto, involve team members who will be using the architecture in its development, and reflect and adapt often, and you will end up with an architecture that meets the needs of your team and your enterprise.

Software architecture is the design and specification of the rules by which software will be built and by which components of the system will behave and interact. It could be as high-level as "We will build out the solution using REST services" or as detailed as naming the particular services to be developed and what data we expect to pass in and out of each.

Architecture also includes establishing design considerations for the development team, such as "no returning null values" and "all code must be peer reviewed before release," as well as producing and maintaining a list of third-party libraries that are approved for use when building code.

When adopting agile, enterprises often ask, "When do we create our software architecture? How much of my architecture do I create up front?" My instinctive response is, "Approach architecture just as we approach everything else—by using agile."

When I look at the **Agile Manifesto,** several principles stand out that guide the development of an agile software architecture.

Designing Architecture Based on Agile Principles

Working software is the primary measure of progress.

Architecture specifications, design documents, approval processes, etc., may be important, but only when they bring us closer to our goal of working software. These artifacts are only a means to an end, not a goal themselves, so they must never be prioritized over delivering working code.

This principle is closely tied to the Agile Manifesto value of working software over comprehensive documentation. If we spend too much time documenting our architecture instead of building working code, we are moving away from agile principles.

The best architectures, requirements, and designs emerge from self-organizing teams.

"Ivory tower architectures" developed by architects that are not involved in the day-to-day development of the software may not fit the immediate needs of the software and the team. The team has the best vantage point to figure out what they actually need to build and how that aligns with architectural guidelines.

If you consider that the customer for the architecture is the development team, this principle and the related Agile Manifesto value of customer collaboration over contract negotiation suggests that a big, up-front architecture created by an architectural team will not be as effective as embedding an architect into the team working with the developers throughout the project. A creative collaboration involving someone who will actually be using the finished product will lead to an architecture that fits the team needs as well as the requirements of the enterprise. An architectural contract handed down from on high will often not.

Simplicity—the art of maximizing the amount of work not done—is essential.

Up-front architectural designs often try to accommodate anticipated future needs that may never actually be realized. That causes more work for a payoff that never arrives. Building and designing based on perceived future need is not agile and wastes effort. Try using team-based design, where your design is incrementally built along with the code, refactoring it as you go.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Architecture is there to guide the team to a solution that fits the needs of the software and the enterprise. If the team members aren't involved with the development of the architecture, they may not understand the objectives and values, so they are more likely to be prone to following the letter of the law without understanding the spirit. Give them the environment and support they need, and trust them to get the job done.

Continuous attention to technical excellence and good design enhances agility.

Agile doesn't mean "no design" or "no architecture." Best practices still apply and will help the team develop more effectively. A well thought-out architecture makes it easier to change directions as customer needs change or become better understood.

Too much architecture can make it more difficult to adapt. Agile architecture must strike the right balance for the team, the software, the environment, and the enterprise.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

What was expected in the early stages of a project almost never matches with reality as the project matures. This holds true for design and architecture just as it does for requirements. By designing iteratively and reflecting on what is providing value and what is hindering the team, the team can adjust the architecture as it is tested and proven through actual use, and they can reevaluate and adapt it as the project changes.

Fitting Architecture into the Agile Process

In order to be agile, we must develop the architecture in an iterative manner, beginning with enough definition for the team to start development. The architecture has to guide the team in a direction that will give them the best chance for success, without specifying so much that they don't have the flexibility to build the software in an agile manner.

There is a trade-off between allowing a team to choose what makes sense for their project while ensuring they don't pick technologies that are incompatible with the rest of the enterprise. This trade-off is making sure the team builds software that aligns to the overall enterprise architecture without forcing the team to use an architecture that won't be effective for their project.

Like any mature agile process, an agile approach to architecture relies on doing just enough definition up front to get started, gathering feedback as we go, adjusting as needed, and iterating frequently to keep architecture and design in sync with the emerging application.

Prior to the first sprint, a high-level enterprise or system architecture should be created (if it doesn't already exist) and discussed. As part of each sprint kickoff, team-based design is used to update the as-is design from the last sprint to account for feature additions and enhancements that will be made in the current sprint. As stories are tasked out for development and testing, the team will use this new "to be" design to help determine what will need to be implemented in order to satisfy the expected design.

Sprint review meetings should present how each story fits into the bigger picture of the whole application and the enterprise. Sprint retrospectives should include architecture and design in the discussions of what is working and what isn't.

In all facets, architecture should be treated like any other part of the agile process:

- By starting with experienced architects and industry best practices, we can integrate the project requirements with the enterprise requirements and standards to be confident in a good foundation
- By incorporating retrospection and review in the process, we can make minor adjustments to both the architecture and the software being developed to ensure that we meet the enterprise's needs
- By driving that feedback using quality tools, security tools, testing, and other objective metrics, we ensure that we aren't reacting to guesses and anecdotal evidence
- By reflecting and evaluating continuously, the course corrections remain minor and incremental and can adjust to changing architecture requirements throughout the life of the project and as needs evolve

- By collaborating with the development teams, we can be sure the architecture reflects real-world needs and isn't just an "ivory tower" architectural approach that makes sense on paper but fails in actual use.

For an organization transitioning to agile development, creating software architecture isn't incompatible with your new processes. Consider the principles in the Agile Manifesto, involve team members who will be using the architecture in its development, and reflect and adapt often, and you will end up with an architecture that meets the needs of your team and your enterprise.

⁂

To read this article online with embedded hypertext links, go here:
https://www.agileconnection.com/article/agile-approach-software-architecture

About Gene Gotimer

Gene is a senior architect at Coveros, Inc. (www.coveros.com), a software company that uses agile methods to accelerate the delivery of secure, reliable software. As a consultant, Gene works with his customers to build software better, faster, and more securely by introducing agile development and DevOps practices such as continuous integration, repeatable builds, unit testing, automated functional testing, analysis tools, security scanning, and automated deploys. He has successfully brought these techniques to commercial and government clients. Gene feels strongly that repeatability, quality, and security are all strongly intertwined; each of them is dependent on the other two, which just makes agile and DevOps that much more crucial to software development. Gene can be reached on Twitter @CoverosGene or email at gene.gotimer @coveros.com.

The User Story Needs A Remodel. Here's Why

By David Hawks

[**Note:** grayed text indicates hypertext links in original article.]

User Stories have become the standard way Agile teams capture requirements and were introduced almost 20 years ago as a part of XP (Extreme Programming). To put it in context, that's four presidents and 14 iPhone models later. A lot has changed and it's time we upgrade how we define and communicate work for teams.

Most teams are using user stories to document requirements and to align expectations between stakeholders and the delivery team. But building features with the goal of satisfying stakeholders is not good enough anymore. We should be focused on satisfying the customer.

The Biggest Lie In Product Development

The lie product departments and stakeholders tell themselves is that they know everything up front. That getting a bunch of subject matter experts in a room will allow them to make intelligent priority and scope decisions. The problem here is that most of these decisions are gut instincts that are not grounded in facts. They are "I think" or "I feel" decisions. Worse yet, these decisions aren't validated until they are implemented and deployed into production. This is a very long and expensive feedback loop.

Also, our current requirements processes assume we actually know the right features to build. Stats show that more than half of the features we build are rarely used. Think of the word "requirement." If something is a "requirement" it sounds like a fixed need. The statistics show us that most "requirements" are not used; therefore are not really necessary to the user. These "requirements" are really just guesses. If it's an educated guess, then it's a hypothesis. Maybe we should change from thinking about "requirements" to thinking about defining "hypotheses?"

The Relationship Between OKR's And Business Objectives

We still need to have some clear goal or target. Today our goals are defined in the form of scope or the largest "epic" and then broken down into smaller user stories. But again this assumes we know the answer. Here, I think we could borrow from the OKR (Objectives and Key Results) world. What if instead of defining the "epic", we defined the business "objective" we are trying to achieve?

Once we have a defined objective, we should engage our team to brainstorm ideas on the best way the object can be achieved and these ideas could be defined as a set of hypotheses. One way to prioritize would be for us to identify all of the "assumptions" we are making about our customers, product, market, solution, etc. We could examine the risk and impact of these assumptions and by looking at the riskiest assumptions, it might help us **prioritize** which hypothesis to prove first.

Agile Teams And Hypothesis Creation

We talk a lot in Agile about self-organizing teams, however, most of these teams are still focused mostly on how to best "deliver." I think these teams should not only be focused on delivering, but also engaged in understanding the problem domain and creating hypotheses around solutions. We don't want the product owner and stakeholder just throwing requirements over a wall for teams to implement.

We have promoted story decomposition techniques for years so that we can deliver iteratively. But too often we still wait until we have a full solution before we deploy to production and get real feedback from the market. How could we learn faster? What if we could run small "experiments" to prove our hypothesis or validate our assumptions without having to build the whole solution? The idea would be for the team to determine what is the smallest "experiment" that can be run to learn more about the problem or solution. Most importantly learning that can be validated by our customers, not stakeholders.

The emphasis of this approach would be for us to validate our ideas as fast as possible. Or is it really to invalidate our ideas and assumptions as fast as possible?

***

To read this article online with embedded hypertext links, go here:
https://agilevelocity.com/agile/user-story-needs-remodel-heres/

About David Hawks

David, Founder and CEO of Agile Velocity, is a Certified Enterprise Coach and Certified Scrum Trainer who is passionate about helping organizations achieve true agility beyond the basic implementation of Agile practices.

David's primary focus is to guide leaders through their Agile transformation by helping to create successful transformation strategies and effectively manage organizational change with a focus on achieving real business results.

Eco Leadership, A leadership approach for the ecosystems of tomorrow

By Chris Hoerée

Why is "Leadership" so important today?

In the last few years, many CEO's, entrepreneurs and coaches have been prototyping new ways of collaborating, new business models, self-managing teams, lean and agile principles, etc. The book of F. Laloux, *Reinventing Organizations,* O. Scharmer's Theory U and the many learning communities that emerged have been a catalyst in this movement of organizational innovation for myself and many others.

I was given the unique opportunity to help building an organization based on the principles Laloux described in his book. We implemented an inspiring higher purpose and strong company values, a shift towards an agile innovation culture, where people evolved based on their roles & talents, new co-creative practices integrated in the day-to-day operations. Our biggest challenge was the implementation of the principles of wholeness and self-management. Being able to be your whole self at work requires courage and a conscious authentic self that is not trapped into ego or fear for rejection; hence the need for a safe environment for people and personal development. Self-management asks from people to take individual responsibility and initiative while at the same time working together as a team. It requires high levels of awareness of group dynamics and skills in interpersonal communication and facilitation. These skills are not often part of the business training of professionals and managers. We experienced that for operating an organization based on Teal or similar principles, a new organizational, and as a consequence, a new leadership culture has to be established.

Reinventing Organizations means Reinventing Leadership.

I saw my observation confirmed in the 2016 Global Human Capital Trends report of Deloitte University Press among 7000 business and HR leaders in 130 countries: **"Leadership for a new kind of organization" was the top of mind concern of business & HR leaders globally in 2016.** As companies have to be more agile and customer-focused, they are shifting from traditional, functional structures to interconnected, flexible teams. A new organizational model is on the rise: a *"network of teams"* in which companies build and empower teams to work on specific business projects and challenges. These networks are aligned and coordinated with operations and information centers. In some ways, businesses are becoming more like movie production teams, with people coming together to tackle projects, then moving on to new assignments once the project is complete. This new structure has important implications for leadership development.

89 % of executives in the survey rated the need to strengthen, reengineer, and improve organizational leadership as an important priority. 56% of executives report their companies are not ready to meet leadership needs.

Although the acknowledgement of WHY Leadership is hyper relevant today, there are a myriad of answers on the questions WHAT leadership is.

What is Leadership? — The origin of the word

In order to touch on the essence of the meaning of a word, it is often useful to go back to its roots, its etymology. The root words of "leading" can be found in Indo-European and Germanic languages: the Indo-European root word "leit(h)" means to go forth, to cross a threshold, to leave, to die; the Old English "lædan" means march at the head of, go before as a guide, accompany and show the way, carry on, pass (one's life); the root word "liðan" means to travel, to go forth (from Proto-Germanic laidjan). Interestingly, there is a clear difference in the root of the words "lead" and "manage": manage is derived from the Latin root word "manus", what means "hand". There is a clear connection between managing and handling or controlling things. The word "lead" comes from several root words meaning "to go (forth)".

"Leading" is about crossing a threshold, going places never seen before, and then, going forth, guiding, showing the way.

"And as we let our own light shine, we unconsciously give other people permission to do the same." Nelson Mandela

The roots of the word leadership essentially talk about a "process": **a personal process (a journey towards consciousness) as well as a collective process (to show the way to others).** Travelling, crossing a threshold is the first step in the Hero's journey, according to Joseph Campbell. Facing the unknown with openness and trust, receiving what is emerging is part of the hero's quest. But there is also a social level: the hero returning to society, transformed, reborn, is sharing his/her knowledge, showing the way, guiding others to new places never seen before. While "managing" is about handling, command and control, "leading" is about a process of learning and transformation that starts with a personal journey of growth in consciousness.

What is Leadership?—Leadership models & their qualifiers

Leadership as such is open and can serve the good or the bad.

> *Leadership needs a qualifier to embed it in a time and ethical context. The qualifier shows the belief system from which you operate.*

“Trait Leadership” for example, is based on the assumption that (innate) personal characteristics are responsible for leader effectiveness independent of the situation. The underlying belief is that a universally agreed list of leadership qualities exists. “Situational Leadership” on the other side, believes that effective leadership is task-relevant; the most successful leaders are those who adapt their leadership style to the abilities of the individual or group in front of them. The assumption here is that leaders can change their behavior at will to meet different circumstances. It neglects the impact of unconscious beliefs, fears and habits, with other words the human psychology. “Transformational leadership” is a concept of leadership assuming that a leader works together with others to identify needed change and create an inspiring vision to guide and execute the change. A transformational leader enhances the motivation and performance of the people he/she is leading by connecting their sense of identity and self to a project and to the collective identity of the organization. Transformational leaders are role models and have the ability to inspire others based on their charisma and vision. This model implies a strong presence and personal leadership and strong interpersonal skills, but it does not necessarily imply shared or co-leadership. The “Primal Leadership” model of Goleman, author of “Emotional Intelligence”, is based on a lot of studies showing that emotional competencies like self-awareness, authenticity, empathy, service attitude, collaboration skills, have an enormous impact on the effectiveness of leadership.

It is not the purpose here to list all current leadership models (generative, integral, servant, etc.). Most models recognize the following elements as key to leadership: **strong inner leadership and presence, vision, engagement, emotional intelligence and action.** To me, this would apply to somebody like Wangari Maathai, the woman who started the Green Belt Movement and won the 2004 Nobel Peace Prize.

eCo Leadership — leadership for a sustainable positive future

I said before that the “qualifier” going with the word leadership shows the values and beliefs that are underlying the leadership model. For me, this time and ethical context in which the needed leadership is embedded, is the following:

The challenge of today, on the level of the individual, the organizations as well as socio-economic systems, is to co-create a sustainable positive future for all participants and the planet we are living on, using our individual and collective intelligence and creativity and combining technological innovation with universal wisdom.

This requires a multi-layered concept of leadership:

1. The basis, **on the individual level, is Presence and Leadership Consciousness:** the individual leader will have to go on his/her own quest, finding answers on the universal questions: who am I, where do I come from, where do I go to? I have called this the personal Why-How-What, the discovery of your personal purpose, identity and contribution to the world. This process is necessary if we want to move from an ego to an eco-awareness, from organizations that are like machines towards organizations that are like living systems. A personal quest results in renewed inner strength and embodied consciousness, a certain easiness, flow and vibrant presence. The most important in this process is to let go of fear and doubts, attachment to old patterns and distraction.

It takes a lifetime but start with the first step. The first step is this crossing the threshold, then, confidently going forth, acting with courage and responsibility.

2. On the relational and organization level, the leadership that is needed to lead ecosystems or networks of teams rather than traditional hierarchical companies, is **Co-leadership.**

Shared/distributed/rotating/collective leadership acknowledges peer influence and engages in consultation and coordinated action. Co-leadership is more a relational or group process than a position and assumes interpersonal influence, dialogue and mutuality. A Co-leadership team is operating based on roles rather than positions. It requires an **in-depth understanding of systemic patterns and dynamics** from the members. In a living system, everyone and everything has to have its place, and even though there is no hierarchy in the traditional pyramidal way, there is always an invisible order or archetypal structure within a system. This invisible order and its consequent dynamics can be healthy and natural or can be unhealthy and unproductive. They need to be brought to consciousness in the team in order to make necessary changes and transformations. The leadership team has to build its unique tribal order — are we a herd of horses, a wolf pack, a flock of geese? — and distribute roles and responsibilities on the more day-to-day pragmatic level. This type of work asks for trust, open dialogue and authenticity in a group. Co-leadership therefor requires strong **interpersonal skills** from members, specifically non-violent communication and conflict handling skills, and the ability to facilitate open conversation and co-creation.

3. On a bigger picture level, **in the social and economic field,** leadership will have to deal with increased complexity, "chaordic" principles and organic and disruptive change: simultaneous top-down, bottom-up, diagonal, and circular change processes. Partnering, co-creation and open innovation across organizational and/or sectorial boundaries will be desirable and necessary. **Leading eco-systems** will become the new norm. The required skills and knowledge of leadership teams working with living systems and transformation processes is a combination of new technology, fresh and fluid innovative thinking and a deep and intuitive understanding of natural and human processes.

Paradoxically enough all this complexity is in our nature as humans, it is in nature and life itself and readily accessible as a source of intelligence and wisdom. Our biggest challenge is to unlearn old habits and (re-)discover, learn and adopt new practices of communicating and collaborating.

Therefor, the qualifier for the leadership concept described here is **"eCo Leadership".** eCo refers to:

- Leadership Consciousness
- Co-leadership
- Leading eco-systems
- Nature and natural wisdom

Hence, I propose the following definition of eCo Leadership:

"eCo Leadership is the ability of a group of individuals by their presence, creativity and wisdom to inspire others to co-create the purpose of an organization (company, non-profit, community,...)."

For me, the 7 **ingredients of eCo Leadership** are the following (ingredients can be qualities, skills and practices that can be developed by individuals and teams and are characteristic of an eCo Leadership culture):

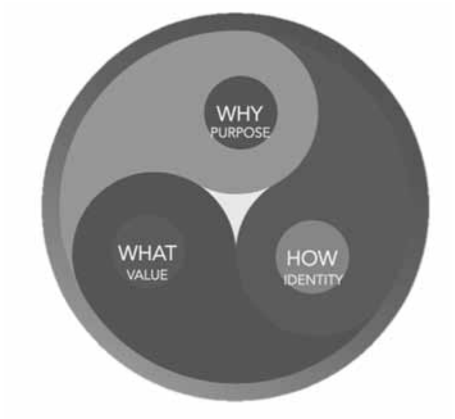
1. **Consciousness**
2. **Presence & Personal Leadership**
3. **Higher Purpose**
4. **Interpersonal Sensitivity**
5. **Curiosity, creativity**
6. **Courage & Action**
7. **Engaging others & creating community**

Now that I have established the qualifier and underlying belief system, definition and characteristics of eCo Leadership, I will give a framework for the "How" without going into details since every Leadership coach applying it, will have his/her own style and approach.

HOW to develop eCo Leadership? — a framework

The 7 leadership ingredients I mentioned before will have to be developed at three levels: personal, interpersonal, and on the level of organizations and eco-system.

I believe that Leadership development has to start from the inside out. A personal quest for attaining or refreshing **Leadership Consciousness** will be the first step. The model I developed as guidance and structure for this quest is the "Personal Why-How-What". Although this is a process of self-discovery, it is best experienced in both individual as well as interpersonal learning. This is why I favor a retreat format. Individual mentoring can be part of this and/or follow the retreat. The methods used in this phase can be: mindful meditation, nature walks and conversations, visualization, shamanic journeying, intuitive writing & drawing, circle talks, systemic constellations, deep listening & dialogue, action & reflection, prototyping…



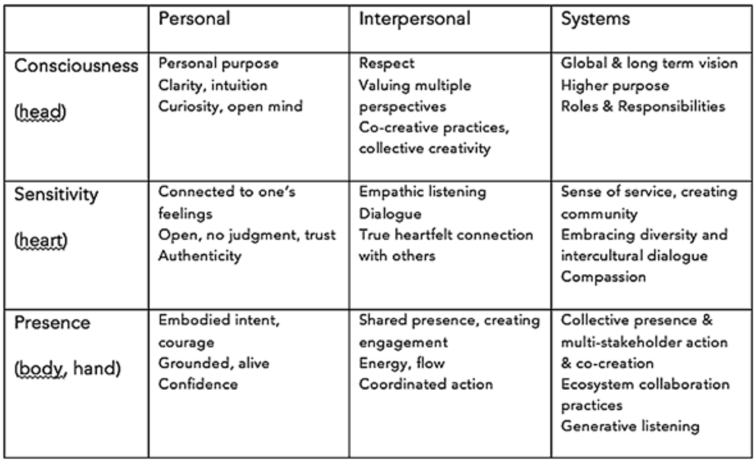
The result of the process is to get clarity on your personal purpose, on your heart's fire, on what drives you and gives your real meaning to your life and work.

You will rediscover your true identity, your talents, your essence, what gives you energy and joy. You will become more conscious about your unique way of being, perceiving and interacting, your patterns and style as a leader. You will explore your place in a group, your qualities and pitfalls in a co-leadership role. You will explore and crystallize your Work in the world: what you want to contribute, create, realize.

The next levels of continued learning are on the **interpersonal and organizational level,** this means, it is developing eCo Leadership skills and practices like participative methods, holding the space for others, facilitating deeper levels of listening and conversation, facilitating circle ways, etc. … This can be done in subsequent open retreats or in-company training with specific Leadership teams can.

Inspired on Integral leadership concepts and the ideas brought forward by O. Scharmer, I propose the following **framework for developing eCo Leadership qualities & practices.**

It takes into consideration the 3 layers of leadership: personal, interpersonal and systems (organizational + ecosystems) levels and 3 important facets of human psychology: "Consciousness" including intuition and mind, "Sensitivity" including emotional intelligence and compassion and "Presence" including intent and action.



	Personal	Interpersonal	Systems
Consciousness (head)	Personal purpose Clarity, intuition Curiosity, open mind	Respect Valuing multiple perspectives Co-creative practices, collective creativity	Global & long term vision Higher purpose Roles & Responsibilities
Sensitivity (heart)	Connected to one's feelings Open, no judgment, trust Authenticity	Empathic listening Dialogue True heartfelt connection with others	Sense of service, creating community Embracing diversity and intercultural dialogue Compassion
Presence (body, hand)	Embodied intent, courage Grounded, alive Confidence	Shared presence, creating engagement Energy, flow Coordinated action	Collective presence & multi-stakeholder action & co-creation Ecosystem collaboration practices Generative listening

Framework for developing eCo Leadership qualities & practices

In my work with Leadership teams, my experience has been that learning to recognize the level of listening and conversation is an eye-opener for people: the shift from downloading to empathic and generative listening, from debate to dialogue is a radical change in culture. Practicing new ways of listening and conversing changes the culture in a leadership team. I often experienced in my work with management teams, that a talking circle gave people energy although it would take a lot more time than the many rushed business meetings that leaders sit in all day. Sometimes, "slower" practices as circle ways are felt to be inappropriate for the day-to-day business, where an often artificial sense of speed rules. But once a leadership team is familiar with these new methods and recognizes the value of it, the spirit of these methods can be integrated in a creative way in day-to-day practices.

Circle talks are a very accessible way to start re-learning to listen to each other, to utilize silence and space in the conversation, to move from exhausting debates to authentic open dialogues.

The conscious implementation of new "practices" is an important part of changing the culture of an organization. Practices are new habits, habits are the cornerstones of a culture.

An obstacle for establishing a new culture of listening and conversation often is the physical environment: hi-tech meeting rooms where it is impossible to move the tables out and make the simplest and most natural of geometries: a circle.

Conclusion

In the last couple of years, Laloux's Reinventing Organizations, Theory U and the U-Labs of O. Scharmer, the Teal and Integral movements, Enlivening Edge community and our local Community of Practice (2BeLinked) were for me and many others, a catalyst and accelerator for change. Although all these ideas as such were not "new-new", the momentum was created to make them the "new normal". Suddenly a true higher purpose had its place in the business world. Coaches, entrepreneurs, CEO's, professionals from all sectors and backgrounds courageously started prototyping new ways of collaborating, operating with new business models, self-managing teams, circle ways…

My personal quest was an intensive integration of skills and experiences from very different fields of life. I could deploy intuitive practices that before were not heard of or talked about in the business world.

My personal purpose is to contribute to a sustainable positive future for the generations to come, for life and the planet by developing the leadership capacity and collective wisdom needed for that, by helping people to unfold their untapped abilities and creativity and supporting the co-creating of ecosystems in a new economic paradigm.

References

Gauthier A. (2011), *Emerging Concepts and Forms of Integral Leadership: Embodying a Radically New Development Paradigm,* Integral Leadership

Review Global Human Capital Trends (2016), *The new organization: Different by design,* Deloitte University Press

GLOBE multi-cultural Leadership research: R. House a.o.(2004), *Culture, Leadership and Organisations, The*

GLOBe study of 62 Societies, Sage Publications

Goleman D., Boyatzis R., Mc Kee A. (2002), *Primal Leadership: Learning to lead with Emotional Intelligence,* Harvard Business School Press

Hofstede G. (2010–2011), Cultures and Organizations, Software of the Mind, McGraw-Hill

Hurst A.(2014), *The Pupose Economy. How Your Desire for Impact, Personal Growth and Community Is Changing the World,* Elevate, USA

Laloux F. (2014), *Reinventing Organizations: A Guide to creating Organizations inspired by the Next Stage of Human Consciousness,* Nelson Parker

Porritt J. (2013), *The World We Made. Alex McKay's Story from 2050,* Phaidon

Scharmer O. (2009), *Theory U. Leading from the Future as It Emerges,* Berrett-Koehler

Sharmer O. (2008), *Uncovering The Blind Spot of Leadership,* Executive Forum Publishers, Inc. San Francisco

***

To read this article online with embedded hypertext links, go here:
https://medium.com/@chris_hoeree/eco-leadership-a-leadership-approach-for-the-ecosystems-of-tomorrow-bcd9c41a941

About Chris Hoerée

Chris is a Leadership & Organization Coach with 20 years of experience facilitating change processes and strategy development. She has a Master in Psychology & Post-Graduate studies in Business. She trained as a Systemic Coach, Deep Democracy practitioner, Scrum Master & Creative Facilitator. She worked for 10 years at the European Head Office of Toyota where she was in charge of Brand Strategy & Culture Change. Chris has had her own consulting & coaching company since 2001. During the last years, she specialized in helping leaders to "reinvent their organizations", to become Teal, agile, self-managing companies, driven by a strong purpose and culture. Chris has a talent to activate creativity and collective intelligence within teams and to help translate these in concrete action. Her own drivers are passion for people, a belief in the power of imagination & collaboration and a strong feeling of responsibility to contribute to a positive & sustainable future for our planet.

chrishoeree@earthways.eu

www.earthways.eu

https://www.linkedin.com/in/chrishoeree/

British Airways: A Brilliant Example of How Cost-Cutting Increases Costs

By Rowan Jackson

[**Note:** grayed text indicates hypertext links in original article.]

In the 1990s, Sir Colin — later Lord — Marshall, Chief Executive of British Airways was being interviewed by a journalist. The latter asked him, as leader of a famous brand, what he feared most. Sir Colin said something along these lines: "the pilots can be ill, the food can taste bad, the plane may be late and we lose the passengers' baggage. I know I can fix these things and I will. But the thing I fear most is our Information Systems going down. We are critically dependent on our IT people for delivering our customer experience and for our survival. Our IT is of strategic importance and I keep my Chief Information Officer really close to me. Our IT is so important we would never outsource it." Sir Colin was a deeply experienced leader who had invested very heavily in creating the unique British Airways' customer experience for the "world's favourite airline". He made mandatory attendance at a training program called Putting People First and attended in person at the end of every program to take employees' questions. All those in leadership positions went through its sister program Managing People First and he attended that one too. A charismatic forthright, rigorous and determined leader, he made British Airways a customer driven company.

Sadly, none of his successors have had the IQ or the EQ (nor the training he had as a Purser in P&O) to understand the world's favourite airline customer experience or to keep it going. Since the Marshall days the British Airways' customer experience has been progressively eroded by a succession of cost cutters. We have had Ayling, Eddington, Walsh and now Alex Cruz who has just had his beard singed with what is probably the most catastrophic meltdown of any information systems in modern times. As a public relations disaster, it is up there with United's Dr David Dao event.

Here is a vignette of what it meant to one high-margin customer:

The customer was flying from JFK to LHR in business class. Half an hour before the flight was due to leave, the information board said: "flight delayed", and then soon after "cancelled", without reason. The customer was told to go back through security to the BA desk, where BA's staff had no idea that the flight has been cancelled or why.

The customer was then told to collect his checked-in luggage from the returns area and "in the meantime the BA staff will look at other flight options". The bag was returned last despite being a business class ticket holder, at which point the customer had to join the back of the queue (fortunately the priority queue) to check in for the next flight. Once at the front of the queue the desk agent told the customer that their new flight "is departing now"; he needed to rush for it. The customer checked with the check-in desk that the luggage would also make the flight, and the desk confirmed this. The customer then rushed to the departure gate and just made it before the doors were shut.

After landing at LHR and waiting at the luggage belts it was apparent that his luggage had not make it onto the plane. On enquiring with the service staff about the location of the bag, there was no record of it. The customer filled out a lost bag form along with instructions to deliver the bag in the evening, as no one would be at home if the bag was returned in the daytime.

The following day the customer received a text message from BA stating that their luggage would be delivered to his home address between 1pm and 3pm, a time when he was at work and was unable to receive it. Delivery was rescheduled for the evening of the next day.

All-in-all a poor customer experience delivered by British Airways. Bad communication about the delay and subsequent cancellation of the flight. Followed by a poor procedure to rebook and recheck-in people and luggage. And then finally poor handling of the return of the luggage despite the customer being explicit about the time when they would be available to take delivery.

Ryanair immediately took the opportunity to make public the fact that none of their IT is outsourced and that backups exist in three different countries.

This sad story is all because of cost-cutting Cruz and his naive team who have now increased costs for British Airways. At time of writing costs are estimated to be £120 million. You may assume that this figure will double or even treble when the costs of customers switching to other airlines, for ever, is taken into consideration. Running a full-service airline using operational effectiveness techniques just does not work and never will.

I'm writing this on a British Airways plane, on the tarmac at Heathrow, away from the jetty, where we are waiting for take-off from Frankfurt. The plane has been delayed by two hours by thunderstorms in Germany. Not British Airways' fault. But the cabin crew are still insisting we have to pay for the Marks & Spencer's food, rather than giving us free drinks and food as Virgin did when I experienced the same problem several years ago. They don't get it, nor clearly do their managers. It's not the crew's fault but it is Cruz's. Their managers are not encouraged to empower them, a key component of a high performing customer experience.

I have been a member of British Airways Future Lab for over four years. This is the British Airways method for obtaining detailed feedback from customers. It is done via a special website and a series of questions we members answer every week. It appears most of the members are, like me, Gold Card holders. These are the 20% of customers who provide 80% of British Airways' revenue.

Over the last few months I have noticed a trend in the comments on Future Lab. Many are frustrated that British Airways does not seem to pay any attention to what we say and takes no action on our suggestions. This is despite that, from reading the comments, it is clear that members of Future Lab want British Airways to be successful and make the airline competitive, improving and to have an excellent customer experience.

And it is not just the customers that are unhappy. Currently about 58 cabin crew are resigning at Gatwick every month; this is not sustainable.

So here are a few messages for Messrs Cruz and Walsh.

1. Firstly, Mr Cruz, resign. It happened on your watch, do the honourable thing get out of the way and let somebody who really understands how to run a full-service airline take on the job
2. Mr Walsh: find Cruz's replacement from a decent airline; I recommend Cathay, Singapore or Emirates. If not, do what Apple did and go to a top-class hotel chain. They know how to create a branded customer experience.
3. Stop trying to compete with the low-cost airlines; as I have said many times on Future lab: THEY ARE NOT YOUR COMPETITION! Lufthansa, Cathay, Finnair, Qatar, Emirates, Singapore, United, Delta and Etihad are.
4. Switch all of the energy that you currently devote to cost-cutting into reduction of any errors, waste and rework that is destroying the BA branded customer experience. Singapore Airlines is the best example of doing this but, if you cannot attract anyone from them, go to Nissan in Sunderland. They know how to reduce failure demand; that is demand on the system caused by failure (often called rework or doing it right the second time when you got it wrong the first time). The average organisation has 35% re-work, and cutting that delights customers, employees and shareholders. It also increases loyalty and profits.
5. Take a rigorous and very detailed look at the way you select, train, lead and subsequently develop your people. Focus selection on the right attitudes required to deliver the customer experience which itself is about 70% related to people and 15% to do with the product. Put in place lean methods for continuous improvement of the customer experience. The new Chief Executive needs to follow Sir Colin's example and attend every training program to demonstrate how important they are to him.
6. Take a leaf out of Ryanair's book: bring all of your information systems in-house and have at least three back-up systems in different countries so that if one goes down you have a failsafe back-up. Don't outsource it to India or to Spain, it is far too important to do that.
7. Stop cutting the customer experience. Get rid of the Marks & Spencer's

food and paying for it, you are not the ghastly Ryanair and should be moving in the opposite direction to the lightweight O'Leary. Then really make a very big deal about free baggage, free food and drink, easy-to-get-to airports and genuine service offering as in "To Serve To Fly". Make your customer experience the centre of your differentiated service offering. Don't copy, differentiate!

8. Re-train all of your customer-facing staff and make the training mandatory and assessed. Get rid of some of the old deadwood (and goodness me there is plenty of it) and enhance the customer engagement with newly selected and properly trained staff. I've only ever handed out two Golden Tickets and one of them was to a ground crew person who sorted my lost baggage. Make properly paid people the focus and the gathering of customer feedback the two issues that you pay attention to at the highest level. Do not focus on satisfaction but on the performance against what your customers expect. Satisfaction is an idea beyond its sell-by date. **Oh and for goodness sake pay them properly!** If you would like to know more about how to do this then click here
9. Engage much more at CEO level with British Airways Future Lab. Invite us to come and talk to you, in-depth and often. Bring us into your closest decision-making, listen to what we say carefully, act on it and go on using us. We are willing to do it for you providing that you engage with us.
10. Don't outsource your call centres! They are critical to your customer experience and should be run by well-paid and well-motivated BA employees not by some people who do not understand your culture or how to delight your customers.

Do that and you will have, just possibly, a chance to recover from this disaster.

Fail to do so and Sir Colin will continue to spin in his grave

***

To read this article online with embedded hypertext links, go here:
https://promisingoutcomes.com/british-airways-a-brilliant-example-of-how-cost-cutting-increases-costs/

About Rowan Jackson

Rowan is a Strategy Development & Implementation specialist and Executive Team Facilitator. He has worked both as a consultant and as a change management practitioner in companies. Rowan has had two careers; the first, 20 years in the Royal Marines where, after serving in Commando units, he spent two years in The Royal Household. After this, Her Majesty The Queen invested him as Member of the Royal Victorian Order (MVO). He is the Chairman of Promising Outcomes Limited.

Escaping Method Prison

By Ivor Jacobsen & Roly Stimson

[**Note:** grayed text indicates hypertext links in original article.]

Key Takeaways

- An understanding of the concept method prison with its side effects gurus, method wars and the zig-zag path, and why it is "the most foolish thing in the world".
- Escaping method prisons by adopting the Essence standard and getting a common ground on top of which to deal with your methods.
- The team gets a) practices easier to teach/learn/change/compare, b) practices easy to apply giving guidance in daily work and stimulating teamwork, and c) a practice library, from which practices can be selected and composed to entire methods.
- The executives get a) the organization to move from essentially being a craft to essentially being an engineering discipline, b) a forever learning organization with a practice library that continuously is improved as teams learn more and more, and c) a tool to measure progress and health for existing projects independent on which method is being used.
- The industry gets industrial scale agile — from craft to engineering.

Background

The world has developed software for more than 50 years. Software has changed virtually every aspect of our lives so we cannot live without it. Thus, the software industry has been very successful. We could choose to be happy and continue doing what we are doing.

However, under the surface everything is not as beautiful: too many failed endeavors, quality in all areas is generally too low, costs are too high, speed is too low, etc. Obviously, we need to have better ways of working or, which is the same, we need better methods.

This is not a new observation. Over all these 50+ years we have been searching for a better method. In some ways our methods of developing software have dramatically changed over time, in other ways they have stayed much the same. As an industry we have followed a zig-zag path moving from paradigm to paradigm and from method to method, changing very much like the fashion industry inspires wardrobe changes. Every new method adoption is generally a very expensive, demoralizing affair. It is expensive because it means retraining the software developers, the teams and their leaders. In some cases existing software may even have to be rewritten in order to work more efficiently with new software. It is demoralizing because the more experienced developers feel they have to re-learn what they already know.

Companies, especially larger ones, realize that a great method provides a competitive advantage — even if it is not the only thing you need to have. They also realize that their method must be explained and explicit so that it can be applied consistently across the organization. And, they realize that one size doesn't fit for all they do – they need a multitude of methods.

Here a **method** provides guidance for all the things you need to do when developing software. These things are technical, such as work with requirements, work with code and to conduct testing, or people related, such as work setting up a well-collaborating team and an efficient project, as well as improving the capability of the people and collecting metrics. The interesting discovery we made in 2013 was that even if the number of methods in the world is huge it seemed that all these methods were just compositions of a much smaller collection of 'mini-methods', maybe a few hundred of such 'mini-methods' in total. These distinct 'mini-methods' are what people in general call **practices**.

In this paper the term method also stands for related terms such as process, methodology, method framework, even if these terms strictly speaking have a different meaning.

1. What is a Method Prison

Let's take a look at four of the most well-known methods (called method frameworks) for scaling agile: The Scaled Agile Framework (SAFe), Scaled Professional Scrum (SPS), Disciplined Agile Delivery (DAD) and Large Scale Scrum (LeSS).

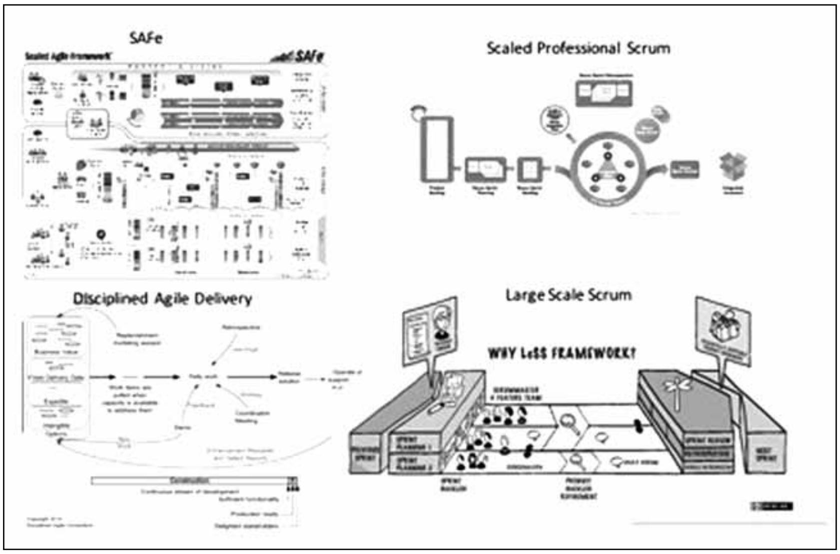


Figure 1 Big pictures of four well-known scaled agile methods

They are all popular and used by organizations around the world. They deliver value to their user organizations in both overlapping ways and in specific ways. Overlapping means that they include the same practices, specific means they have some special practices that makes the difference. If an organization applies one of these methods its users usually don't know anything about the other alternatives.

What are then the problems?

1. They are all monolithic – non-modular.

Most methods (not just the four ones discussed here) are monolithic meaning they are not designed in a modular way. This means that you can't easily exchange one module with another one and keep the other practices intact.

A method can be tacit — in the heads of people - or explicit — described at different levels of detail. A lot of software in the world is developed using tacit methods. Organizations using a tacit method are generally not aware of the problem with method prisons, even though they are often the most caught in the prison because they can't explain their method and they consequently can't easily change it.

Instead, what we want is a library of reusable modules, which is being updated as users learn more and more. Since every method is just a composition of practices, we want reusable practices. Teams and teams-of-teams should be able to easily agree on their own method by mixing and matching the practices they want to use from the library and compose them together.

2. They have their own individual presentation style.

Every method has its individual specific structure, and uses its own style and terminology to describe its selected practices. The owner of the method has decided about these important aspects for themselves without following any standard. As a result, its practices are incompatible with practices from other methods.

3. They have a lot in common — but it is hidden.

Moreover, though every method has some unique practices, it has a lot more in common with others. Every method "borrows" practices from other methods and "improves" them. So, what is common is hidden behind new terms and "new" features. We use quotation marks to indicate that it is not really exactly "borrowing" that happens, and it is not always "improving", but due to misunderstanding or reinterpretation of the original practice, it often becomes a perversion or confusion of the original. Likewise the "new" features are typically not completely new at all, but new name for an evolution or variation of a previously existing practices ("new bottles for old wine").

4. Every method is controlled by a warden — the guru

The guru has decided which practices should be combined into his or her method, and in some cases extended the method with practices "borrowed" and "improved" from other methods. The method reflects the particular perspectives, prejudices and experiences of its guru, and not to what we as a development community have collectively learned. Methods should reuse what the team or organization considers the best practices for their specific challenges and purposes, and not those selected by one single guru independent of these considerations.

5. Every method is branded and often trademarked and copyrighted.

Other gurus are now, if its users like practices from other methods, forced to "borrow" these practices and "improve" what could have been re-used. This way of working doesn't stimulate collaboration with other gurus, on the contrary. Given the investment in time and capital by the gurus of these other methods, they must defend their turf with feverish determination, resulting in method wars.

As a consequence, adopting a method — published or homegrown — means that you are stuck with a monolith, presented with its individual style, using many practices that are common but you don't know it, guarded by a guru who has branded his method making it difficult to reuse. Your method cannot easily reuse practices from a global practice library. Instead, you are in a method prison. You are stuck with how the guru of your method has decided things are done while working with his/her method. To be clear here, we are not suggesting that gurus consciously try to put you in a method prison; they just continue do what we as an industry have done since our origin, because we didn't know anything better.

Thus, once you have adopted a method, you are in a method prison controlled by the guru of that method. Ivar Jacobson, one of the the authors of this paper, was once one of the gurus governing the Unified Process prison. He realized that this was "the most foolish thing in the world" (of course the software world) and it was unworthy of any industry and in particular of such a huge industry as the software industry. Recently similar ideas have been expressed by others, (e.g., see [0]).

We as software professionals need to put a stop to this ridiculous development. We want people with creative practice ideas to collaborate and together provide libraries of reusable practices to the world. We want them to serve the whole industry and not be forced to create branded methods.

2. A History of Methods and Method Prison

Since we started to develop software and adopted published methods we have had method prisons. Moreover, method prisons have some side effects that we also need to eliminate, the three most negative ones are the reliance on gurus, the method war and the zig-zag path. Our history will focus on how methods have created method prisons and their side effects. We will do that from two perspectives: lifecycles and practices.

2.1 Gurus, Method Wars and Zig-Zag Paths

Why is the reliance on a guru bad?

1. We all understand that relying on a single method/guru is risky. Big companies cannot accept the risk that individuals outside their domain of control should play such a vital role in their way of satisfying their clients. No single method can possibly effectively contemplate the endless variables that arise from the variety of working environments, industries, individual companies and their employees.
2. You effectively ransom your organization's own future competitiveness and ability to adapt, survive and thrive. In the future the method guru

decides if and how the method prison is changed over time. And if you don't like it, or it doesn't match your strategic direction of travel and associated needs, there is nothing you can do, because you are stuck inside this method prison, unless you want to suffer the cost and pain of moving to yet another different method prison.

Many organizations don't realize they are in a method prison. It is easy to understand why not. They have not identified any problems because they haven't seen how it could be different than today. The problems are too abstract without a solution to them. Once upon the time users didn't know that software should be built using components, e.g. java beans. Similarly, they didn't know they needed use cases or user stories to capture requirements. And so on. However, once they got it, and started to use it, they saw the value. Similarly, once they see that they can have access to a global library of practices, which are continuously improved, and from which they can select their own method, they won't go back to what we have today.

It is easy to understand that branded methods put you in a method prison. However, the situation for in-house developed methods is not different, just not so visible. What about agile methods? Most agile methods are today light in description. However, they also suffer from the same problem of not supporting reuse, mixing and matching practices, building a practice library, etc. We also advocate very light descriptions focusing on the essentials, but with the ability to extend with details when desirable.

There is a *method* war going on out there. It started 50 years ago and it still goes on — jokingly we can call it the Fifty Year's War, which has been even longer than the Thirty Year's War in Europe early 1600 (which was also a "religious war", incidentally). There are no signs that this will stop by itself.

It is a war because, as the situation has been and still is today, it is very hard to compare methods. We have not had a common ground to work as shared reference. Methods use different terminology, terms that could be synonyms have been adorned by some small differences and these differences are overemphasized, and terms that are nearly homonyms, but not quite, make any comparison very hard to do. Gurus and their followers talk about their method in religious terms, with a lot of passion and zealotry, which makes reasoned comparison and evaluation much harder. Not standing on a standard platform makes it impossible to compare methods and have a rational discussion on methods.

Once upon a time we had a large number of different notations to describe elements in software engineering. Then we got the Unified Modeling Language (UML) standard in 1997 and all these different notations were replaced by one single standard — the notation war was over. Notations are only one aspect of methods, so we need a similar standard for all other aspects of methods, a standard that allow for all the diversity needed from methods.

The software industry has followed a *zig-zag path* from paradigm to paradigm and from method to method.

1. With every major paradigm shift, such as the shift from Structured Methods to Object Methods and from the latter to the Agile Methods, basically the industry throw out almost all they know about software development and started all over again, with new terminology with little relation to the old one. Old practices are dismissed as garbage, and new practices hyped. To make this transition from the old to the new is ex-

tremely costly to the software industry in the form of training, coaching and tooling.

2. With every major new technical trend, for instance service-oriented architecture, big data, cloud computing, internet of things, the method authors also 'reinvent the wheel'. They create new terminology and new practices even if they could have reused what was already in place. The costs are not as huge as in the previous point, since some of the changes are not fundamental across everything we do and thus the impact is limited to, for instance, cloud development, but there is still significant and foolish waste.

Within every such trend there are many competing methods. For instance, back early 1990 there were about 30 competing object-oriented methods. The issue is that all these methods suffer from the five problems resulting in method prisons. This is of course to the advantage of method authors whose method is selected, even if this was not their conscious intention.

We need to eliminate the need for a continued zig-zag path.

2.2 Lifecycles and Method Prisons

From the ad hoc approach used in the early years of computing, came the waterfall methods. There were hundreds of them published. Some of the most popular were Structured Analysis and Design Technique (SADT), Structured Analysis / Structured Design (SA/SD) and Information Engineering (IE). They had their greatness from 1960 to year 2000.

The waterfall methods were heavily influenced by the practices of construction project management — the mantra was "find ways to build software like civil engineers build bridges". They described a software development project as going through a number of phases such as requirements, design, implementation (coding), and verification (i.e. testing and bug-fixing).

Around the year 2000 they were more and more replaced by iterative methods originally introduced by Barry Boehm's Spiral Model of Software Development and Enhancement, and methods such as RUP and DSDM, and later simplified and further popularized by agile practices such as XP and Scrum. All the four methods introduced earlier, SAFe, SPS, DAD and LeSS, apply an iterative lifecycle.

Of course, all different methods were accompanied by method prisons, and we relied on gurus and perpetuated the method wars.

2.3 Practices and Method Prisons

Since the beginning of software development we have struggled with how to do the right things in our projects. Originally, we struggled with programming because writing code was what we obviously had to do. The other things we needed to do were ad hoc. We had no real guidelines for how to do requirements, testing, configuration management and many of these other important things.

We have had three major eras in software engineering (years are just approximate):

- 1960-1980: The Structured Methods Era,
- 1980-2000: The Object Methods Era, and
- 2000 – now: The Agile Methods Era,

resulting in the zig-zag path from era to era. We don't want any more eras and no zig-zag path in future.

The Structured Methods Era

In this era the most popular methods, such as (e.g., SADT, SA/DT, IE), all separated functional process logic from data design. They did this for what were good reasons at the time - because computers at that time were designed exactly like that — with separate program logic and data storage structures. They were used for all kinds of software development — including both "Data Processing" and "Real-Time" systems, following the common parlance of the time. The value of the function/data approach was of course that what was designed was close to the realization – to the machine – you wrote the program separate from the way you designed your data. The systems were hard to develop and even harder to change safely and that became the "Achilles heel" for this generation of methods.

The Object Methods Era

The next paradigm shift came in the early 1980s, inspired by a new programming metaphor — object-oriented programming, triggered by a new programming language Smalltalk. The key ideas behind Smalltalk were much older, being already supported by Simula in 1967. Around 1990, a complement to the idea of objects came to widespread acceptance. Components with well-defined interfaces, which could be connected to build systems, became a new widely accepted architectural style. Components are still the dominating metaphor behind most modern methods.

With objects and components a completely new family of methods evolved. The old methods and their practices were considered to be out of fashion and thrown out. What came in was in many cases similar practices with some significant differences but with new terminology, so it was almost impossible to track back to their ancestors. A new fashion was born. In the early 1990s about 30 different object-oriented methods were published. They had a lot in common but it was almost impossible to find the commonalities since each method author created his/her own terminology and iconography.

In the second half of 1990s the Object Management Group (OMG — see omg.org) felt that it was time to at least standardize on how to represent drawings about software — notations used to develop software. This led to a task force being created to drive the development of this new standard. The work resulted in the Unified Modeling Language (UML). This basically killed all other methods than the Unified Process (marketed under the name Rational Unified Process (RUP)); the Unified Process dominated the software development world around year 2000. Again a sad step, because many of the other methods had very interesting and valuable practices that could have been made available in addition to some of the Unified Process practices. However, the Unified Process became in fashion and everything else was considered out of fashion and more or less thrown out. Yes, this is how foolish we were.

The Agile Methods Era

The agile movement — often referred to just as "agile" — is now the most popular trend in software development and embraced by the whole world. The Agile movement changed the emphasis away from the technical practices, placing the team, the work and the people front and center.

As strange as it may sound, the methods employed in the previous eras did not pay much attention to the human factors. Everyone understood of course that software was developed by people, but very few books were written about how to get people motivated and empowered in developing great software. The most successful method books were quite silent on the topic. It was basically assumed that one way or the other this was the task of management. With agile many new people practices came into play, for instance self-organizing teams, pair programming, daily standups.

Given the impact agile has had on the empowerment of the programmers, it is easy to understand that agile has become very popular and the latest trend. Moreover, given the positive impact agile has had on our development of software there is no doubt it has deserved to become the latest trend. And, while some agile practices will be replaced by other, better, practices, agile as a philosophy and attitude is not a fad that will pass away. It will stay with us for the foreseeable future.

To summarize

Though the different eras have contributed knowledge and experience, and a lot of it is specific for each era, they all resulted in a continuation of the method war controlled by a few gurus.

3. What to do to Escape Method Prisons

It took us a while to understand what was wrong with how we have dealt with software development methods (see [1] and [2]). However, once we had seen the "most foolish thing in the world", it didn't require a genius to figure out that the key to put an end to it was to find a common language with a common terminology or in one word a common *ground,* which we can use when talking about and using practices and methods. Thus in 2009 the SEMAT community was founded with the mission to "re-found software engineering…[1] include a kernel of widely agreed elements that would be extensible for specific uses" [3].

We need to find a common ground

Most methods include (or imply) a lifecycle, technical practices and people practices. Thus there is something we have in common. However this is hidden and not easy to discover, because different gurus describe these things using different vocabulary and language. Thus the common ground we are searching for includes a vocabulary and a language. We called the vocabulary the *kernel* and the language the *kernel language.*

Common Ground = Kernel + Language = Essence

Starting with the kernel

Given that the kernel is intended to help describing methods and practices, it needs to contain "things" that are or should be perceived as always prevalent in any method. In essence, what are the things we always have, always do and always produce when developing software[2]? We, the team of SEMAT volunteers (about 20 people from around the world), working with the kernel, agreed that these things called the *universals* should be "applicable no matter the size or scale of the software under development, nor the size, scale or style of the team involved in the development". "In essence it provides a practice independent framework for thinking and reasoning about the practices we have and the practices we need. The goal of the kernel is to establish a shared understanding of what is at the heart of software development."

As an input to the work on finding the kernel in 2010, the three founders of SEMAT (Ivar Jacobson, Bertrand Meyer and Richard Soley) wrote a vision statement [4]. The three of us understood that finding the kernel needed to be guided by criteria and principles. We first agreed on some *criteria for inclusion* of elements in the kernel (see [4] for more complete description of the criteria).

Elements should be: *universal, significant, relevant, defined precisely, actionable, assessable and comprehensive. Relevant* was explained as "available for application by all software engineers, regardless of background, and methodological camp (if any)" and comprehensive as "applies to the collection of the kernel elements; together, they must capture the essence of software engineering, providing a map that supports the crucial practices, patterns and methods of software engineering teams".

We also identified the following general principles deemed as essential to finding a kernel (also in [4]): *Quality, simplicity, theory, realism and scalability, justification, falsifiability, forward-looking perspective, modularity and self-improvement. Theory* meant "the kernel shall rest on a solid, rigorous theoretical basis", realism and scalability "the kernel shall be applicable by practical projects, including large projects, and based where possible on proven techniques", *self-improvement* "the kernel shall be accompanied by mechanisms enabling its own evolution".

Moreover, the vision statement [4] also formulated what features the kernel should have: Practice independence, lifecycle independence, language independence, concise, scalable, extensible and formally specified. Scalable was explained as the kernel must support the very smallest of projects — one person developing one system for one customer — it must also support the largest of projects, in which there may be systems-of-systems, teams-of-teams and projects-of-projects. Extensible meant the kernel needs to possess the ability to add practices, details and coverage, and to add lifecycle management and to tailor the kernel itself to be domain-specific or to integrate the software development work into a larger endeavor.

With these criteria, principles and features the SEMAT team set out to find the kernel.

Followed by the language

To explain the universals in the kernel and also practices and methods we need a language. Using just English is not precise enough so we need to have a formal language with syntax and semantics.

The language must be designed for its principal users who are professional software developers participating in a software development endeavor. The language must also allow competent practitioners to create and improve practices without having to learn an advanced language.

The language should support four principal applications: Describing, simulating, applying and assessing. From [4]: "The concept of state is likely to play an important role in the kernel language, to represent work progress."

The same vision statement gave rather specific requirements on the language. For example "The language should be designed for the developer community (not just process engineers and academics)", which is an important requirement asking for a more intuitive and more engaging user experience in working with methods than what has been available today. Another example of a requirement is that the language must provide "validation mechanisms, so that it is possible to assess whether a project that claims to apply a given method element … actually does, and is not just paying lip service to it."

We need more than a kernel — we need practices and methods

The role of the kernel and the kernel language is to be used to describe practices and methods with a common ground. To get there, a useful common ground had to be applied in describing a large number of methods. We needed to agree on what a practice and a pattern is [4]. We said for example: "A practice is a separate concern of a method. Examples are … iterative development, component-based development", "every practice, unless explicitly defined as a continuous activity, has a clear beginning and an end" and "every practice brings defined value to its stakeholders".

With these principles, values and requirements in the baggage the SEMAT team had got a good idea of WHAT was needed to escape the method prison.

4. How to Escape the Method Prison

From idea to tangible result is a long way. We first had to get a common ground.

4.1 Essence — the common ground of software engineering

As a response to "the most foolish thing in the world", the work on an escape route from method prison started in 2006 in Ivar Jacobson International (IJI). In 2009 the SEMAT community was founded and in 2011 the work was transferred to OMG, which eventually gave rise to a standard common ground in software engineering called Essence [5]

Essence became an adopted standard in 2014. Thus Essence didn't come like a flash from "the brow of Zeus", but was carefully designed based on the vision statement [4].

We were also inspired by Michelangelo: "In every block of marble I see a statue as plain as though it stood before me, shaped and perfect in attitude and action. I have only to hew away the rough walls that imprison the lovely apparition to reveal it to the other eyes as mine see it." We felt that we from all this mass of methods had to find the essence so we paraphrased it:"We are liberating the essence from the burden of the whole."

And by Antoine de Saint-Exupéry:"You have achieved perfection not when there is nothing left to add, but when there is nothing left to take away." We took a very conservative approach in deciding what should be in the kernel and what should be outside the kernel. It is easier to add new elements to the kernel than to take them away.

An Essentialized practice/method is described using Essence and it focuses the description on what is essential. It doesn't mean changing the intent of the practice or the method. Essentialization provides significant value. We as a community can create libraries of practices coming from many different methods. Teams can mix and match practices from many methods to get a method they want. If you have an idea for a new practice, you can just focus on essentializing that practice and add it to a practice library for others to select; you don't need to "reinvent the wheel" to create your own method. This liberates that practice from monolithic methods, and it will open up the method prisons and let companies and teams get out to an open world.

4.2 Using Essence

Instead of giving the whole theory behind Essence, we will show its usage by presenting a practice described on top of Essence — using Essence as a platform to present the practice.

We have selected to describe User Story as an example of an Essence practice — calling it here User Story Essentials. Figure 2 below shows (not to be read in detail) the set of 14 cards that represent the headline essentials of the practice.

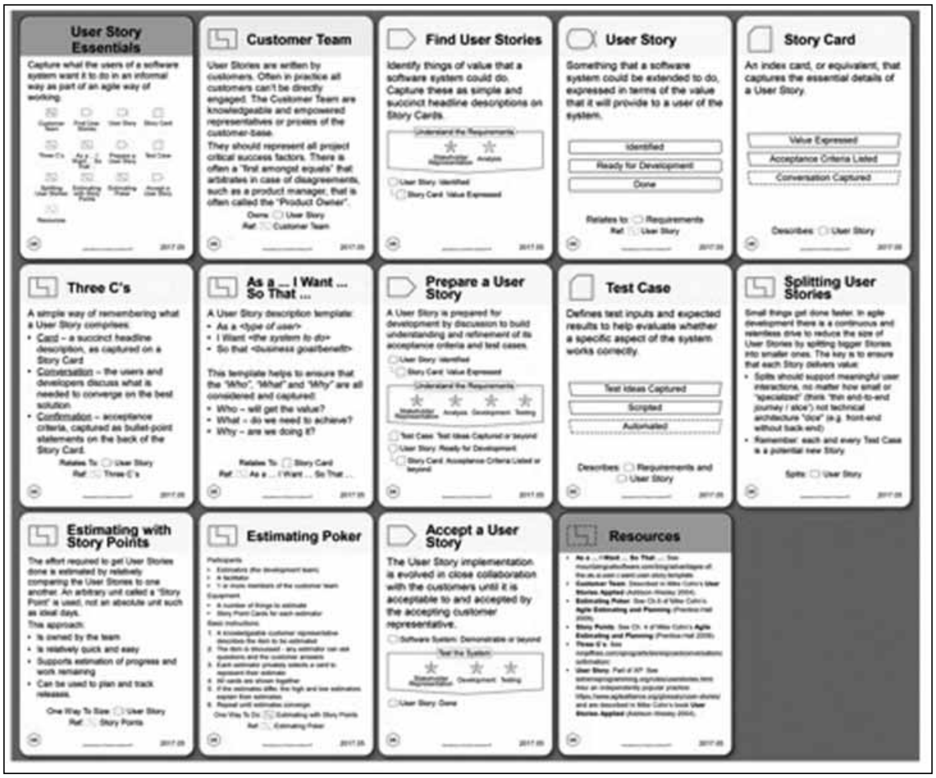
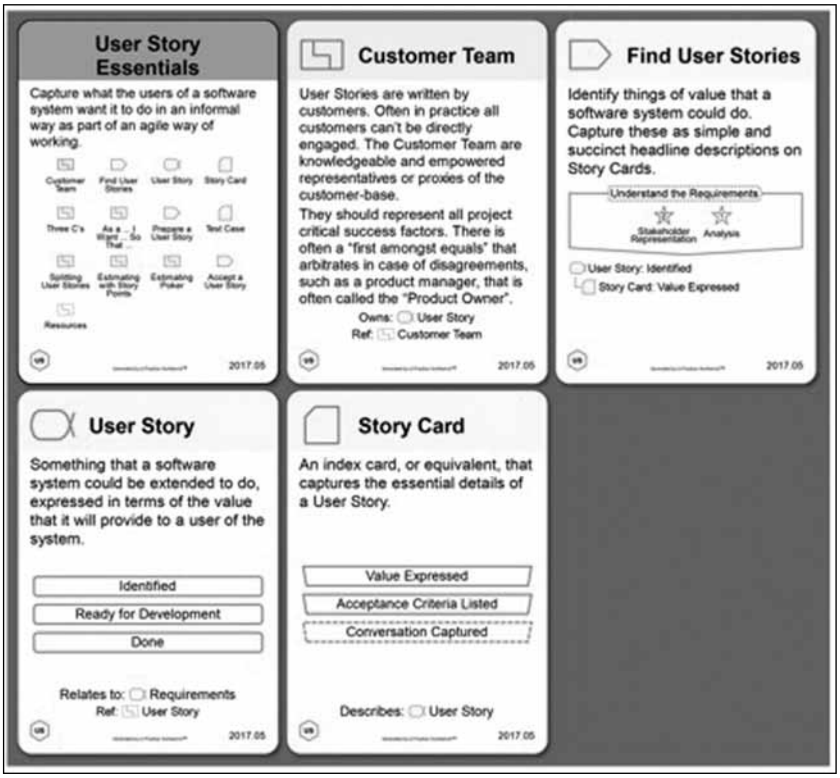


Figure 2: The User Story practice as an example of an Essentialized Practice

It is not our intention to here describe the entire practice but to give you a good understanding of what an essentialized practice look like.

Thus, we have selected a representative set of cards being briefly described next.


User Story Essentials

Capture what the users of a software system want it to do in an informal way as part of an agile way of working.

Customer Team
Find User Stories
User Story
Story Card
Three C's
As a ... I Want ... So That ...
Prepare a User Story
Test Case
Splitting User Stories
Estimating with Story Points
Estimating Poker
Accept a User Story
Resources
2017.05

Customer Team

User Stories are written by customers. Often in practice all customers can't be directly engaged. The Customer Team are knowledgeable and empowered representatives or proxies of the customer-base.

They should represent all project critical success factors. There is often a "first amongst equals" that arbitrates in case of disagreements, such as a product manager, that is often called the "Product Owner".

Owns: User Story
Ref: Customer Team

Find User Stories

Identify things of value that a software system could do. Capture these as simple and succinct headline descriptions on Story Cards.

User Story: Identified
Story Card: Value Expressed

User Story

Something that a software system could be extended to do, expressed in terms of the value that it will provide to a user of the system.

Relates to: Requirements
Ref: User Story

Story Card

An index card, or equivalent, that captures the essential details of a User Story.

Describes: User Story

Figure 3: A selection of five cards form the User Story Essentials practice

User Story Essentials (Overview Card) – gives an overview of the practice in terms of:

- A brief description that gives an insight into why (benefits) and when (applicability) we might use the practice
- A contents listing — showing named practice element icons for all the elements within the practice (each of which is described with its own card).

Note that the color coding gives an immediate visual indication as to the scope of application of the practice — in this case we see that the practice is:

- Mainly Yellow cards — the Essence color coding for the Solution area of concern — telling us that this practice is concerned with the software system we are building and/or its requirements.
- One Green card — the Essence color coding for the Customer area of concern – telling us that the practice also concerns itself with how we

interact with business / customer area concerns such as the Opportunity and the Stakeholders.

- Zero Blue cards — Essence has three areas of concerns, the third color coded in blue standing for the Endeavor area of concern. The User Story Essentials practice has no cards in this area.

Note also that in this case there is a strong separation of concerns between the Solution and Customer concerns that User Story Essentials addresses and the Endeavor space, which includes concerns such as the Team and how we manage the Work. The practical impact is that this practice can be used with any number of different management practices that mainly operate in the blue Endeavor space, such as a timeboxed, Scrum-style approach to work management or a continuous flow, Kanban-style approach.

Customer Team (Pattern Card) — patterns give supporting guidance relating to other elements and/or how these relate to each other, in terms of (in this case):

- Textual description — encapsulating the critical aspects of the guidance that the pattern provides.
- Named associations — showing which other element or elements the pattern relates to primarily — in this case the User Story element.
- A Reference Link — to a named Reference on the Resources card – which in turn provides one or more pointers to sources of more guidance or information. The Resources card is one of the 14 cards in Figure 2 describing the practice.

Essentialized practices can de described at different levels of detail. The cards in this practice don't attempt to provide all the information for example that a novice team would need to successfully apply the practice. If history has taught us anything it is:

- No amount of written process enables novices to succeed without expert support.
- The more words there are the less likely that any of them will be read.
- Instead of "borrowing and rewriting" other people's words when it comes to the more voluminous detailed supporting guidance, it is better to simply reference the original sources of this guidance.

Essentialized practices such as this one work on the principle that novice teams need support from expert coaches to be successful. The cards become a tool for expert coaches to use to help teams to adopt, adapt and assess their team practices, or for expert teams to use in the same way.

Finally note that, when presented electronically as browsable HTML images, the association and reference links can all be navigated electronically, as can other link elements on other cards.

Find User Stories (Activity Card) — gives guidance to a team on what they should actually do, in terms of (in this case):

- A description of the activity.
- An indication of the Competencies and Competency Levels that we need for the activity to be executed successfully. For instance the card requires Stakeholder Representative competency at level 2 and Analysis competency at level 1 (all of which is defined in the Essence kernel, and can be immediately drilled into from the electronic browsable HTML and cards)
- An indication of the space that the Activity operates in — i.e., what "kind of thing it helps us do" (the generic kernel "Activity Space" — in this case "Understand the Requirements")
- An indication of the purpose of the activity expressed as the end-state that it achieves — in this case a User Story is Identified and a physical Story Card produced that expresses the value associated with the User Story.

Note that activities are critical because without them nothing actually ever gets done — it is remarkable how many traditional methods inundate readers with posturing and theorizing, without actually giving them what they need, which is clear advice on what they should actually do!

User Story (Alpha) — a key thing that we work with, that we need to progress, and the progression of which is a key trackable status indicator for the project — you can think of Alphas as the things that you expect to see flowing across Kanban boards, described here in terms of:

- A brief description that makes clear what this thing is and what it is used for.
- A sequence of States that the item is progressed through — in this case from being Identified through being Ready for Development through to being Done. (Think of these as candidate columns on a Kanban Board — although teams may want to represent other interim states as well depending on their local working practices).
- The "parent" (kernel) Alpha that the multiple User Stories all relate to (the Requirements in this case).

Story Card (Work Product Card) — gives guidance on the real physical things that we should produce to make the essential information visible — in this case a key defining (though often forgotten) feature of the User Story approach is that we use something of very limited "real-estate" (an index card or electronic equivalent) as the mechanism for capturing the headline information about what we want to build into the Software System. The Work Product is defined here on the card in terms of:

- A brief description.
- The Levels of Detail that we progressively elaborate — in this case indicating that initially we ensure that we have captured and communicated the associated value, and that we also need to continue on at some stage to list the acceptance criteria — the dotted outline of the third level of

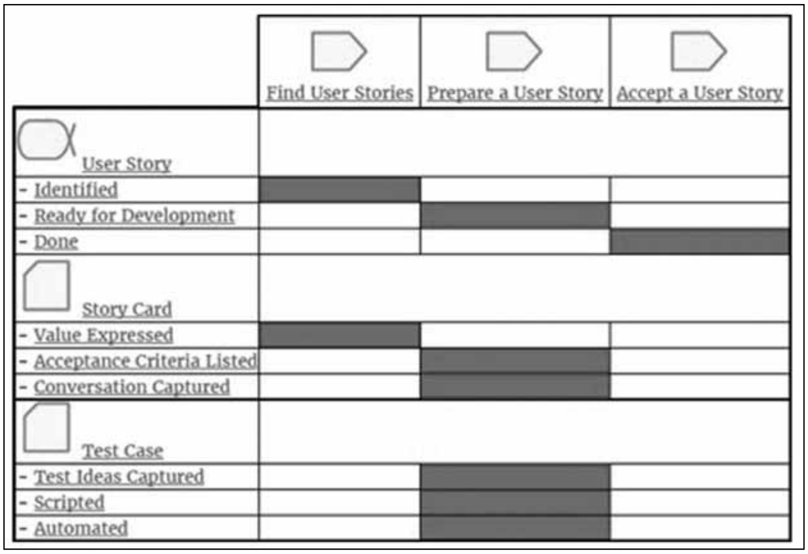
detail indicating that we may or may not capture associated conversations – for example in an electronic tool if we are a distributed team.

- The Alpha that the Work Product describes — a User Story in this case.

Putting it all together

We have now described a representative subset of the different types of card which are used in the User Story Essentials practice, so we will not describe the other cards because the story will rapidly become familiar and repetitious (which is part of the value of using a simple, standard language to express all our practice guidance).

Now we understand what all the cards mean, we also need to understand at a high level how the whole practice works. The cards themselves give us all the clues we need as to how the elements fit together to provide an end-to-end story – which activities progress and produce which elements, but it is also here useful to tell the joined-up story in terms of end-to-end flow through the different activities.



	Find User Stories	Prepare a User Story	Accept a User Story
User Story			
- Identified	▨		
- Ready for Development		▨	
- Done			▨
Story Card			
- Value Expressed	▨		
- Acceptance Criteria Listed		▨	
- Conversation Captured		▨	
Test Case			
- Test Ideas Captured		▨	
- Scripted		▨	
- Automated		▨	

Figure 4: State Progression Matrix showing end-to-end flow through the Activities

- First we need to Find User Stories. This brings one or more User Stories into existence in the initial Identified state, each documented by a Story Card with just enough information to ensure that the User Story has its Value Expressed.
- On a Story-by-Story basis, we will select a User Story that we wish to get done next, and use the Prepare a User Story activity to progress the User Story to be Ready for Development, which involves ensuring that we have the Acceptance Criteria Listed on the Story Card, and during which we may also get any supporting Conversation Captured. As part of this same activity we also fully elaborate the associated Test Cases.

- The final activity that this practice describes is how we work to Accept a User Story, the successful completion of which moves the User Story to the Done state.

Notice that this "chaining" of Activities primarily via the state of the things that they progress does not over-constrain the overall flow. It does not, for example, imply a single-pass, strictly sequential flow. We might, for example, iterate around the different activities for different User Stories in different ways. Exactly how may be further constrained as part of adopting other practices. For example, if we use the User Story practice in conjunction with Scrum, as is very common, we may agree the following general rules as a team:

- Do the Find User Stories before we start our First Sprint, but also allow this to happen on an ad hoc basis ongoing.
- Do the Prepare a User Story activity before the first Sprint and then during each Sprint for the User Stories for the next Sprint, in time for Sprint Planning.
- Aim to Accept a User Story as soon as it is done, to get all the User Stories selected for the Sprint Done before the end of Sprint Review.

Some of the key features and benefits of essentialized practices as illustrated by this one example are:

- The practice is tightly scoped — it tells us how to do one thing well, and does not constrain or limit any of our other choice when it comes to other practices we want to use in other spaces (Scrum, Kanban, ...).
- The practice is VERY concisely expressed — it's a little compressed in the above graphic, but when "life-size" the cards in the practice together represent roughly the equivalent of a side of A4.
- The practice is accessible and can be interacted with — the cards are used in all kinds of ways — including making an annotated team way of working instantly visible, self-assessing the adequacy of local practices and prioritizing improvement areas.
- The practice is expressed in a simple, standard way — now you un-

To summarize the general rules and principles illustrated here:

Essence distinguishes between elements of health and progress versus elements of documentation. The former is known as **alphas** while the latter is known as **work products**. Each alpha has a lifecycle moving from one alpha state to another. **Work products** are the tangible things that describe an alpha and give evidence to its alpha states; they are what practitioners produce when conducting software engineering activities, such as requirement specifications, design models, code, and so on. An **Activity** is required to achieve anything, including progressing Alphas and producing or updating a Work Product. Activity spaces organize activities. To conduct an activity requires specific **Competencies**. Patterns are solutions to typical problems. An example of a pattern is a role, which is a solution to the problem of outlining work responsibilities.

Essence in defining only the generic standard "common ground" defines no work products, activities or patterns, since these are all practice-dependent. It defines 7 alphas with its states, 15 activity spaces and 6 competencies, which are all practice-agnostic. Practices defined on top of Essence introduce new elements or subtypes of the standard kernel element types.

derstand these 4 cards from User Essentials, there are no barriers to understanding any other Essence practice from any other source — just because you like this User Story practice, you aren't now captive in its method prison — you are free to roam the open market to select any other practices from any other sources.

- The practice "plugs into" the Essence standard kernel, thus ensuring it interoperates in well-defined ways with any other essentialized practices.
- This same fact enables scope and coverage of any practice to be instantly assessed (our practice adds activities into the Essence kernel activity spaces "Understand the Requirements" and "Test the System", but adds nothing to the other 13 activity spaces outlined by the Essence kernel ("Implement the System", "Deploy the System", ...) — so if this is the only practice we adopt, it is clear that we have no agreed or defined way of doing these other things (which may or may not be a problem, but is a clearly visible fact ...).
- It contains all the essentials – you may or may not be doing many other things, but if you are not doing this set of things in this kind of way (or locally modified equivalent things, or possibly explicitly NOT doing one particular aspect for a clearly understood and well-articulated reason) then can you reasonably claim to be doing "User Stories" at all?

4.3 Reflection

In section 4.2 we presented the User Story practice essentialized without first presenting the Essence kernel and language. We presented the practice with "Essence in Stealth Mode", to coin an expression we have got from Paul McMahon. However, underneath the essentialized practice we rely heavily on Essence. In our example User Story is a sub-alpha related to the *Requirements* kernel alpha. The "Find User Stories" activity is allocated to the "Understand the Requirements" activity space and so is the activity "Prepare a User Story", while the "Accept a User Story" belongs to the activity space "Test the System".

We have attempted to show that practices are easily understood even without first giving a long and, to many people, boring introduction to Essence. This has been done in many other papers and books already, see [6] - [10]. Thus, here we will just mention some important things you may need to take away.

When the SEMAT volunteers designed Essence as a response to HOW to escape the method prison, particular attention was paid to the "simplicity clause" that "the kernel shall only include essential concepts", which the team interpreted as the guidelines for a method or practice should focus on the essentials.

- The experience is that developers rarely have the time or interest to read detailed methods or practices. Starting to learn the essentials gets teams ready to start working significantly earlier than if they first have to learn "all" there is to say about the subject.
- The essentials were defined as a rule of thumb being about 5% of what an expert knows about the subject.

- Some teams and organizations need more than the essentials, so different levels of detail must be made optional.

The SEMAT team also knew we had to come up with a new user experience to teach practices. The current way of doing it through books and web sites didn't help during actual work — books are dead descriptions and not active guides. We searched for a more engaging way of working and found inspiration in modern work on gamification. We used cards, as you have seen.

We also consistently applied the principle of 'Separation of Concerns' in many different contexts (for general discussion see http://en.wikipedia.org/wiki/Separation_of_concerns). Practices are separate concerns, which can be composed into methods through a merge operation, known in Essence as "composition". The kernel is also a separate, more abstract, concern, on top of which practices can be composed, also merged.

In summary, Essence enables us to escape from method prisons because it sets out a common description of what all methods have in common, and a standard language for talking about this common ground and about all our practices. This means we are free to select essentialized practices from any source we choose, including from our own organization as well as external sources, and free to mix-and-match them with practices from other sources, in order to get the best from all worlds, without being locked in to any of them.

5. Out of the Method Prison

Many companies are now in the process of essentializing their existing methods. For instance, in the words of Tata Consulting Services (TCS): "TCS has engaged with all of its core industry partners like SAP, Oracle, Microsoft and others and also the clients of TCS and is working with the core methodology teams of those companies to help foster the collaborative adoption of the Essence standard and turn this de-jure standard into a de facto standard."

These companies get reusable practices available in a practice library. Teams and organizations are able to mix and match practices from different methods and create their own ways of working. Today, we believe that there are around hundred practices described on top of Essence. Ivar Jacobson International has developed about 50 practices and made 25 of them available in a practice library at **https://practicelibrary.ivarjacobson.com.**

Those companies are getting out of their method prisons. They don't rely on gurus anymore. They won't follow a zig-zag path, but they expect a sustainable evolution. The method war is over for them. However, getting out of method prisons is not all they are expecting. They have much higher ambitions. They are on a path to industrial-scale agile — moving software development from primarily being a craft to primarily being an engineering discipline, but still being agile in both software development and in working with methods.

To be successful we still will rely on the craftsmanship of our empowered teams, but this will be underpinned with a shared base of codified engineering practices that can be reused in different permutations and combinations across different technical domains and project types. This will enable us to maintain high levels of craftsmanship consistently across all our projects, and to sustain this indefinitely through future challenges and changes.

We also need a supporting organization with a learning culture open to new ideas and comfortable with experimentation. Discussing this is out of scope for this paper, but we refer to papers already published (see [8]-[10]).

Essence is also making inroads in the academic world. Universities around the world are teaching Essence to a varying degree. Here a quote from Professor Pekka Abrahamsson, "At one of the largest technical universities in Scandinavia, Norwegian University of Science and Technology in Trondheim, in the Spring of 2017, we have successfully taught Essence in Software Engineering course to 460 students … Essence empowered students to gain control of their project, work methods and practices. We have finally moved beyond Scrum and Kanban … Data and results convinced me and thus my Software Engineering education in the future will be driven by Essence."

Maybe this move to Essence is "the smartest thing in the world" to these companies and universities.

References

1. Hastie S., Linders B., McIntosh S., Ferreira R.M., Smith C. **"Opinion: What 2017 Has in Store for Culture & Methods".**
2. Jacobson I., Meyer B. Methods need theory. Dr. Dobb's Journal, 2009.
3. Jacobson I, Spence I. 2009. Why we need a theory for software engineering. Dr. Dobb's Journal, 2009.
4. Jacobson I., Meyer B., Soley R. 2009. Call for Action: The Semat Initiative. Dr. Dobb's Journal, 2009.
5. Ivar Jacobson, Bertrand Meyer, Richard Soley. "Software Engineering Method and Theory — a Vision Statement", Feb 2010.
6. Object Management Group, **"Essence - Kernel And Language For Software Engineering Methods",** November 2014.
7. Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence and Svante Lidman, "The Essence of Software Engineering: The SEMAT Kernel," Communications of the ACM, Volume 55, Issue 12, December 2012.
8. Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence and Svante Lidman, "The Essence of Software Engineering: Applying the SEMAT Kernel", Addison-Wesley, 2013
9. Ivar Jacobson, Ian Spence and Pan-Wei Ng. "Agile and SEMAT: Perfect Partners", Communications of the ACM, Volume 11, Issue 9, Oct. 2013
10. Ivar Jacobson and Ed Seidewitz, "A New Software Engineering," Communications of the ACM, Volume 57, Issue 12, Pages 49-54. December 2014.

11. **Ivar Jacobson, Ian Spence, Ed Seidewitz. "Industrial-scale agile: from craft to engineering",** Communications of the ACM: Volume 59 Issue 12, December 2016.

***

To read this article online with embedded hypertext links, go here:
https://www.infoq.com/articles/escape-method-prson

About Ivor Jacobsen

Dr. Ivar Jacobson received his Ph.D. in computer science from KTH Royal Institute of Technology, was rewarded the Gustaf Dalén medal from Chalmers in 2003, and made an honorary doctor at San Martin de Porres University, Peru, in 2009.

Ivar has both an academic and an industrial career.

He has authored ten books, published more than a hundred papers and he is a frequent keynote speaker at conferences around the world.

Ivar is a father of components and component architecture, work that was adopted by Ericsson and resulted in the greatest commercial success story ever in the history of Sweden, and it still is. He is the father of use cases and Objectory, which, after the acquisition of Rational Software in 1995, resulted in the Rational Unified Process, a widely adopted method. He is also one of the three original developers of the Unified Modelling Language. But all this is history. Ivar founded his current company, Ivar Jacobson International, which since 2004 has been focused on using methods and tools in a smart, superlight and agile way. This work resulted in that Ivar became a founder and a leader of a worldwide network, SEMAT, which has the mission to revolutionize software development based on a kernel of software engineering. The kernel has been realized as a formal standard called Essence, which is the key idea described in this paper

About Roly Stimson

Roly is a Principal Consultant with Ivar Jacobson International (IJI) with more than 30 years' experience in applying software methods to complex development challenges. For the last 15 years Roly has been involved with iterative, incremental, lean and agile methods. He has contributed to the development of IJI's kernel-based EssUP practices, SEMAT's OMG Essence standard, and IJI's Essence-based Agile Essentials and Agile at Scale practices.

Becoming A Non-technical Scrum Master

By Jeremy Jerrell

[**Note:** grayed text indicates hypertext links in original article.]

If you work in the tech industry then you've no doubt noticed the recent explosion in **demand for qualified Scrum Masters.** But, what's more interesting is the surge in interest from individuals hoping to fill these positions who are coming from outside of the tech industry. In fact, in the last month alone I've received several questions from individuals without a technical background who are interested in becoming a Scrum Master. While the background of each individual is different, all of the emails end with a common question: "Can someone who doesn't have a technical background find success as a non-technical Scrum Master?". While opinions on this may certainly differ, I'm thrilled to say that in my experience the answer to this question has been a resounding "Yes!".

Learning The Skills To Be A Great Scrum Master

While there's no question that having a strong technical background can be advantageous to becoming a successful Scrum Master, it's in no way a requirement. In fact, some of the best Scrum Masters that I've ever worked with have come from completely non-technical backgrounds.

So, if a technical background isn't a requirement for becoming an effective Scrum Master, then what *is?*

The Scrum Master role is all about fostering collaboration across your team. And to do this effectively you need to be able to build connections with others and communicate with them in a way that makes sense to them. Therefore, great Scrum Masters need to be great communicators and understand how to adapt their communication style to a variety of situations and individual preferences.

Teams who are new to the Scrum framework are often skeptical of the number of ceremonies and artifacts that are required to adhere to the rules of the framework. It's your responsibility as a Scrum Master to communicate the value of each of these ceremonies and artifacts in a way that resonates with each member of your team so that everyone is on board with fully participating in the framework.

In addition, great Scrum Master have a knack for setting short-term goals with their teams and then visualizing the steps necessary to reach those goals. In this way, the Scrum Master helps their team understand the value of each of their sprint goals and helps them build a plan to reach those goals.

But What About Technical Skills?

Both of the skills mentioned above are non-technical in nature, meaning that anyone from any background could be capable of employing these skills successfully. But, does this mean that there are no technical skills required for becoming a great Scrum Master?

Not exactly.

Truly effective Scrum Masters also possess a deep understanding of how their organization delivers software. However, this is not the same thing as being a skilled software developer. Great Scrum Masters have a clear mental picture of all of the steps their organization takes to deliver an increment of product to market and how each of those steps fit together. This doesn't mean that the Scrum Master understands every line of code used to bring a product to life, or the specific nuances of each step of the product's deployment pipeline, but they do understand how each step of that process fits together and how changes to one step can affect other steps.

In addition, while they may have a deep understanding of *their* organization's software development process, they also understand that their organization's process is almost guaranteed to differ from another organization's process…and that this difference is ok.

The goal of understanding *their* process is to better position them to spot impediments that could be affecting their team, especially those impediments that their team may not even see themselves. And another goal is to help them to spot opportunities to improve and optimize that process so their team can deliver software more effectively.

So while some level of technical understanding is necessary, the good news is that this level of understanding can be learned on the job by anyone willing to invest the effort to do so.

Finding Success As A Non-Technical Scrum Master

But despite the skills above, a large part of your success as a non-technical Scrum Master will depend on how willing your team is to accept a Scrum Master from a non-technical background.

For some teams, this won't be an issue. They'll be happy to have the aide of a great Scrum Master and won't care about your level of technical chops…or lack thereof. For others, however, this may be more of an issue.

For some teams, a Scrum Master from a non-technical background may need to work a little harder to gain their team's trust. This can be particularly true for teams who are new to the Scrum framework and are already a bit skeptical of the Scrum Master's role to begin with. But don't despair, if you find yourself in this situation all hope is not lost.

When working with a team whose trust you may have to work harder than usual to earn, your first order of business should be to invest in building strong relationships with each individual on the team. This can pay huge dividends since people whom you have strong relationships with will be more likely to support and follow you as you begin to drive change in your role. Or, even if they don't always agree with you, they'll at least be less likely to publicly detract from you when they disagree.

But beyond this, you must also work to really learn the skills and responsibilities that are expected of a Scrum Master. And then, make a visible effort to put these same skills to work bettering the lives of your team.

As mentioned above, it's not unusual for teams are who are new to the Scrum framework to also be skeptical of the Scrum Master role in general. Often this skepticism is rooted in a general lack of understanding of purpose of the Scrum Master role as well as the value that this role can bring to their team.

But, by working to truly develop the skills of an effective Scrum Master you'll not only start to show your team how you can add value to their work but you'll also demonstrate that the Scrum Master role is a craft of it's own that requires a commitment to mastery comparable to their own roles and therefore worthy of their respect.

Beginning Your Journey To Becoming A Great Scrum Master

So, you're confident that you can become a truly effective Scrum Master even without a technical background but you don't know where to start? Luckily, getting started is easier than you think.

First, there are a wealth of books and **online courses** available to help you deepen your knowledge of your craft and to teach you the specific skills you'll need to be successful. Becoming an effective Scrum Master is a career-long pursuit and there's always more to learn, but luckily you'll never be at a loss for inspiration.

Second, finding an experienced Scrum Master who can serve as a mentor can be an incredibly effective way to accelerate your own growth as a Scrum Master. A great mentor can give guidance as to what materials or learning would be most appropriate for where you are in your journey, provide insight and advice to problems that you may be facing based on their own experience with similar problems in the past, or just act as a sounding board and listen encouragingly as you reason out the best approach for yourself. If you're looking for a mentor, a great place to start are the more experienced Scrum Masters in your own organization, Scrum Masters from outside organizations that you may encounter at local user groups or conferences, or even those Scrum Masters who can provide **coaching and mentoring** remotely via the internet.

And finally, jumping into your role with both feet is the most effective way to quickly find success as a Scrum Master. Truly effective Scrum Masters are great communicators and great facilitators, but above all, truly effective Scrum Masters are great problem solvers. This is because every situation you'll face will be different and therefore the problems you'll face with one team will differ from the problems you'll face with another team. Great Scrum Masters don't have all the answers, but they excel at putting their problem solving skills to work to find those answers. And there's no better way to do this, then to dive headfirst into your first team and start solving these problems for yourself.

***

To read this article online with embedded hypertext links, go here:
http://www.jeremyjarrell.com/becoming-non-technical-scrum-master/?utm_source=Great+Agile+Tips&utm_campaign=a40f3fc47f-AUGUST_2017_BLOG_POST&utm_medium=email&utm_term=0_3ddfeb9350-a40f3fc47f-329233165

About Jeremy Jerrell

Jeremy is an agile coach and author who helps teams get better at doing what they love. When not creating agile training courses for industry leading sites like Pluralsight.com, Jeremy mentors Scrum Masters and Product Owners to help them reach their full potential.

He is a highly rated speaker throughout the United States as well as a syndicated author whose articles and videos have appeared on sites such as InfoQ.com, StickyMinds.com, Pluralsight.com, FrontRowAgile.com, Simple-Talk.com, and ScrumAlliance.org.

Jeremy resides in Pittsburgh, Pennsylvania with his wife and three children and is an avid runner. He loves to discuss all topics related to agile methodologies and can be reached by Twitter at @jeremyjarrell or at his website, www.jeremyjarrell.com.

Implications of Enterprise Focus in Scrum

By Ron Jeffries

[**Note:** grayed text indicates hypertext links in original article.]

When Chet and I began thinking about our talk for the 2017 Scrum Gathering, we considered the quotations below, which seemed to us to be in conflict, at least in practice.

Agile is Mindset.

– Steve Denning (and earlier, Alan Shalloway)

I invented Extreme Programming to make the world safe for programmers.

– Kent Beck

As we talked more about what we wanted to say, we found some common ground. Let's dig in a bit:

Enterprise Focus

Steve Denning is active in bringing Agile ideas into the corporation, writing in Forbes and elsewhere. The three words above sum up his viewpoint quite well for just three words. If people up higher in the organization take on the "Agile mindset", they'll begin to build a company that gains the benefits that Agile can bring. It's hard to argue with that.

Agile is Mindset.

– Steve Denning

The world of "Agile" is strongly focused on the corporation these days. The magic words are "Scaling", "Enterprise Coaching", "Agile Leadership", and the magic methods include Enterprise Scrum, LeSS, and the particularly aptly named SAFe. The Project Management Institute is well under way on a second assault, er um, effort to bring Agile ideas to Project Management, in a joint effort with the Agile Alliance. The Scrum Alliance is generating new certificates as rapidly as their PDF formatters can generate.

Corporate "Agile" is everywhere. And that's not a bad thing, because the ideas can't thrive until the mindset has been spread widely enough within the enterprise.

Effective Agile Development

I invented Extreme Programming to make the world safe for programmers.

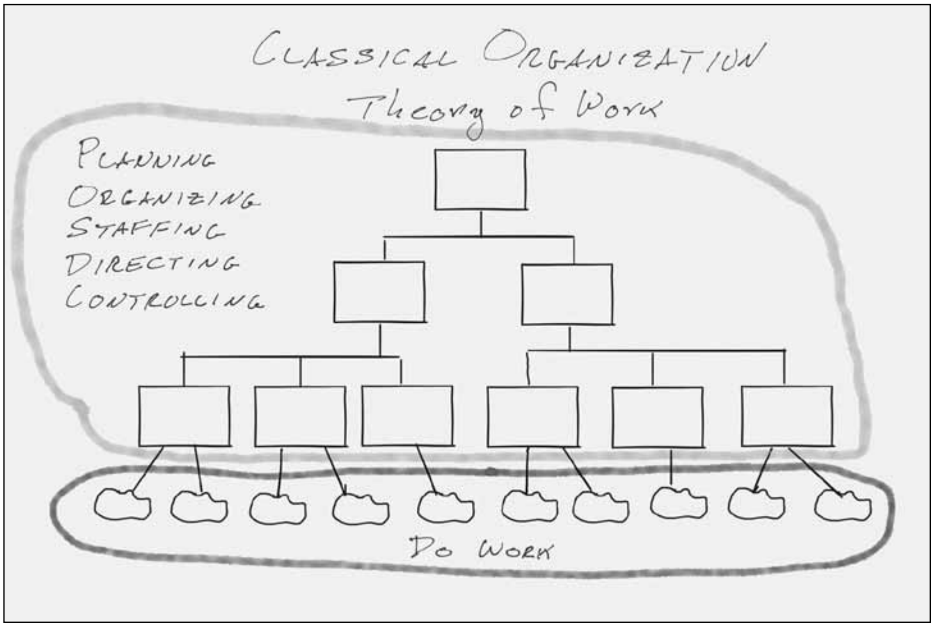
– Kent Beck

Agile Software Development done poorly makes the world unsafe for programmers, as discussed in the entire Dark Scrum series on this site. Perhaps more interesting to the enterprise: Agile Software Development done poorly drops the benefit of the ideas almost to nothing.

It's easy to see that we all have a common goal: do Agile Software Development well. Let's explore what that means to our organization, our investment, and our management.

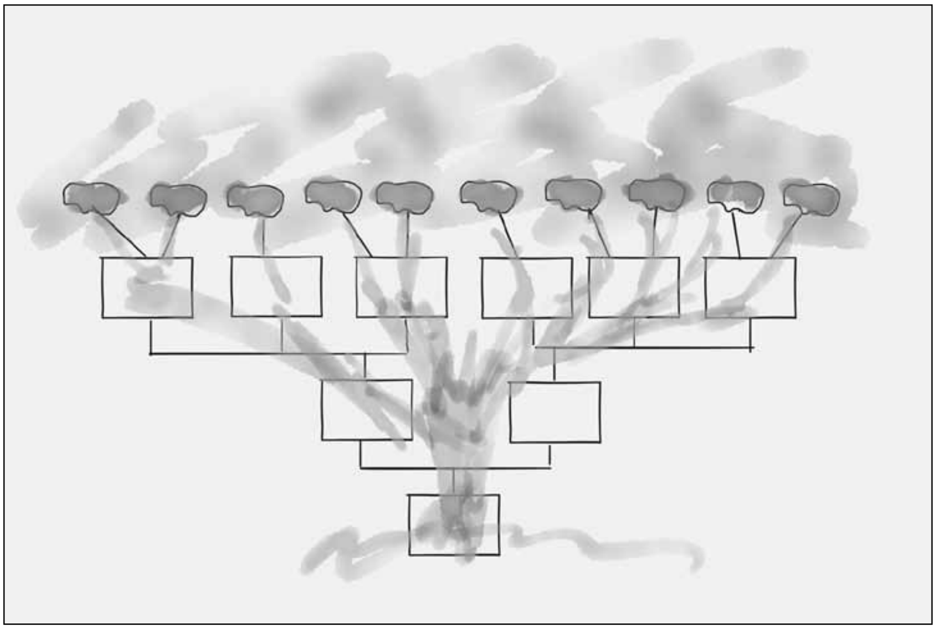
Classical Organization

One way of looking at this is that the conventional corporation is built like an upside down tree, with powerful managers at the top, pushing commands and control downward, until finally good stuff is squeezed out of the company at the bottom, where most of the actual work gets done. When a company has this mindset, Agile becomes an unprecedented opportunity to micromanage product development teams. **See "Time was ..."**.

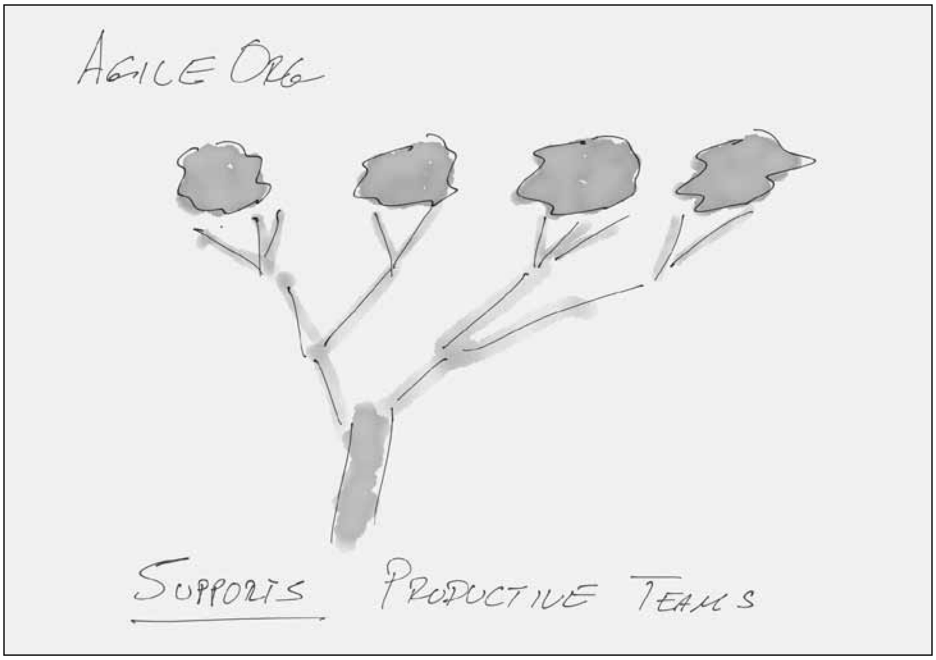


Agile Organization

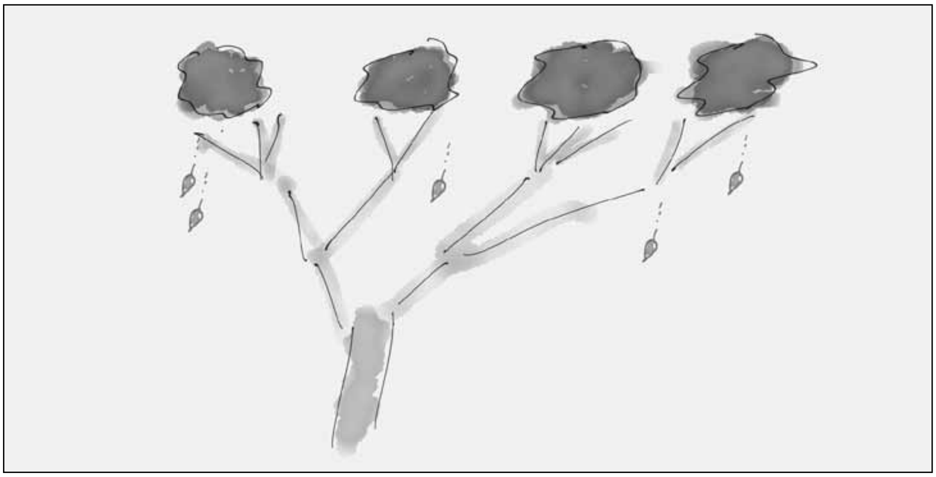
For Agile ideas to work well — frankly for any modern management approach to work well — the corporation should be thought of more like a real tree, with the trunk of the C-level and the branches of management all supporting the healthy green leaves where the work gets done.



When organizational support is properly in place, and when the development teams are executing Agile ideas well, the result is a healthy tree. (Let's pretend that the picture below looks like a healthy tree. :)



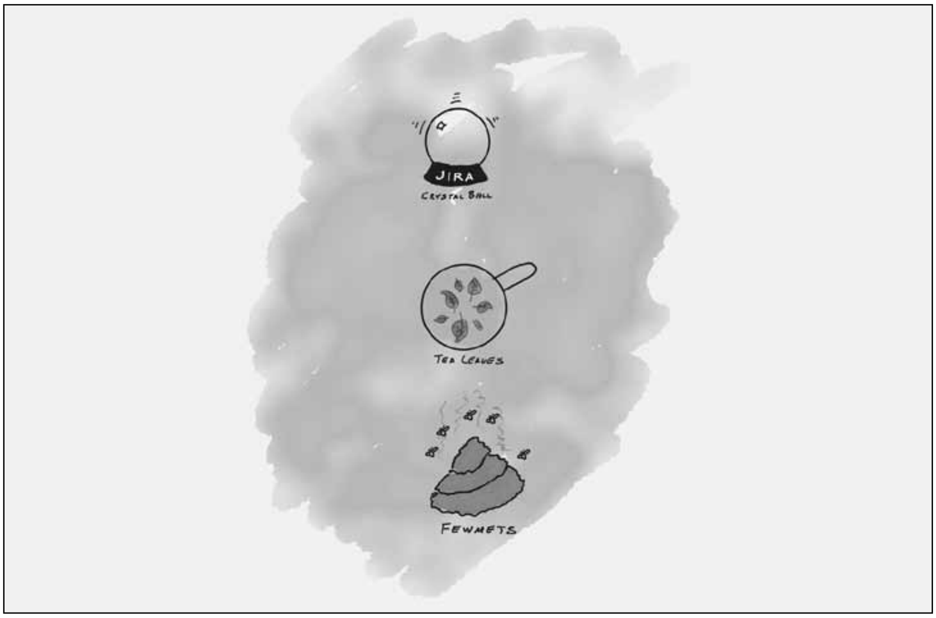
Unfortunately, when support is not suitable, or when development teams do not know how to build software in an Agile fashion, the leaves are not as productive. They turn brown and begin to fall away.



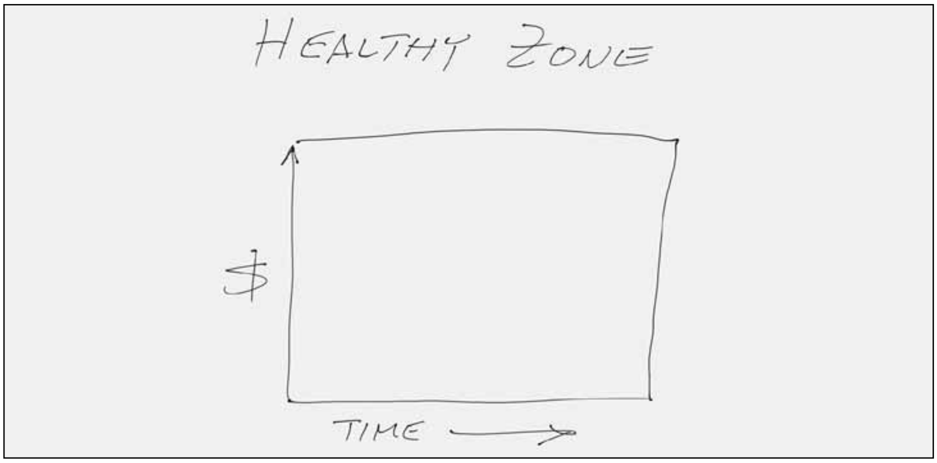
This leads, all too often, to what we call Dark Scrum, where well-intended stakeholders mistakenly oppress the teams, actually reducing effectiveness while trying to improve things. This is unpleasant for the team. Far more important to the organization, it inevitably results in slower progress and a weaker product.

Proxy Management (Fortune-Telling)

Classical managers are used to considering proxies rather than actual progress. They look at "earned value" or how many Jira items are done, or how estimates compare to actuals. While these indicators used to be the best we could get, in today's Agile terms they are little better than reading tea leaves or finding animals in the forest by examining their droppings.

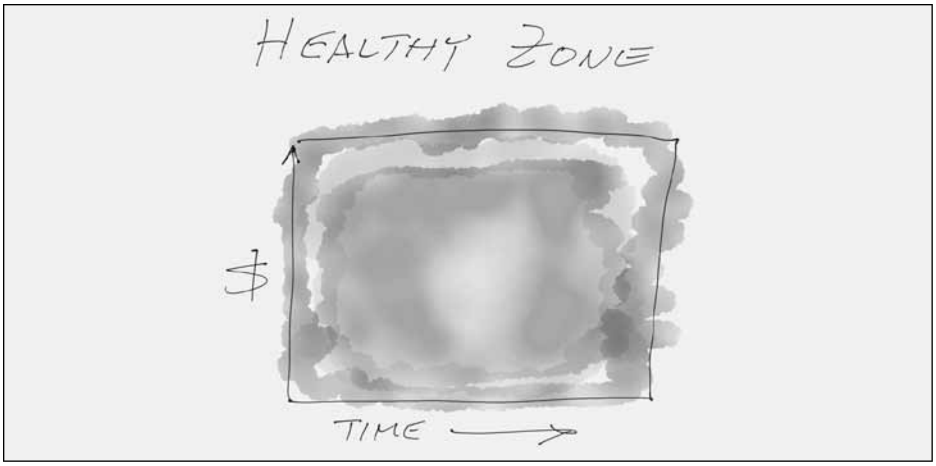


To find a better way, let's begin by considering the "Time-Money Box". Every manager — probably every employee — has some flexibity in how much they can cost the company, in terms of time and money, before they get in trouble. Inside the zone, they are healthy. Outside, they may be in trouble. Typically, the amount of time or money we can waste gets smaller as we get closer to the leaves of our tree. And that's exactly what we'll count on.

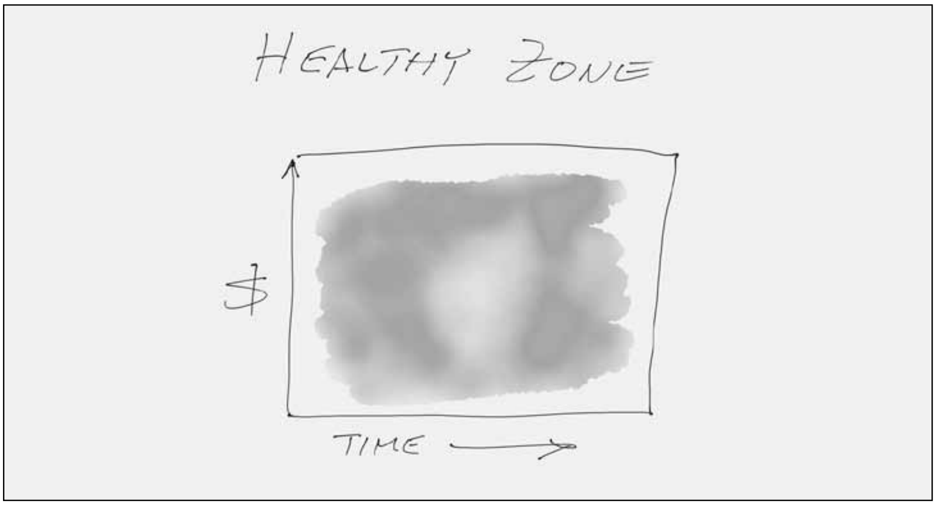


Invest with Time-Money Boxes

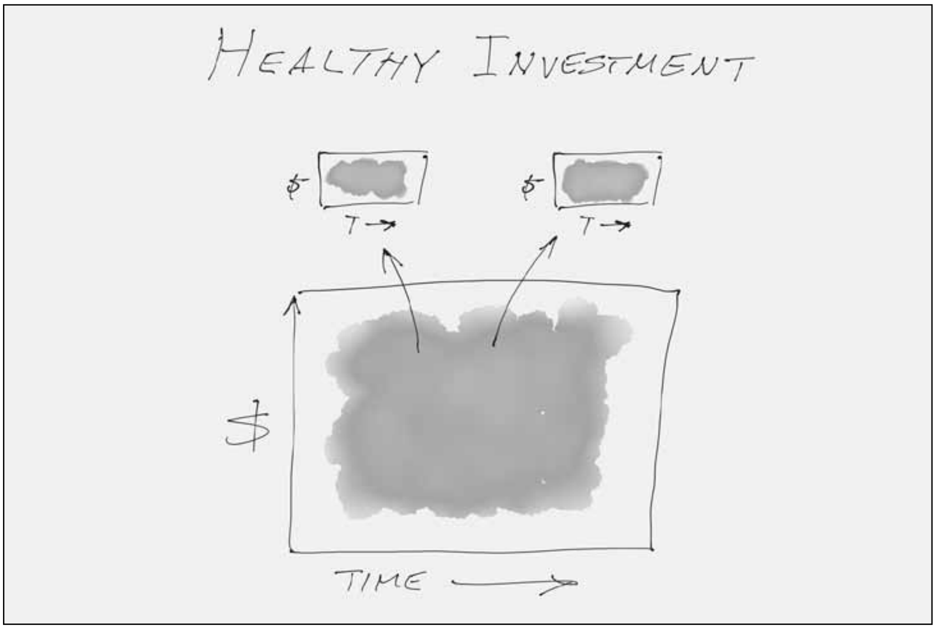
Generally, no manager is entirely sure how much flexibility they have if they get too close to running out of time or money. As they get closer to the edges, they become more nervous and begin inspecting and guiding their projects ever more carefully and actively. This is natural, and prudent, even though it's often not very effective.



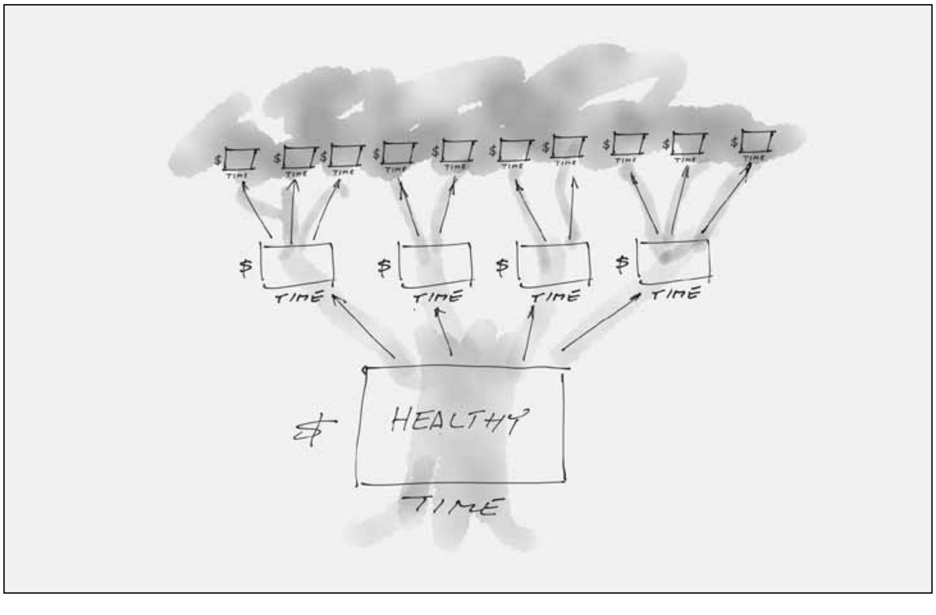
The manager does best to stay well inside the healthy zone. In terms of their projects, that means that they want to be sure everything is going well, In conventional management of conventional projects, you're back to the crystal ball. In the case of an Agile method, there's a much better way.



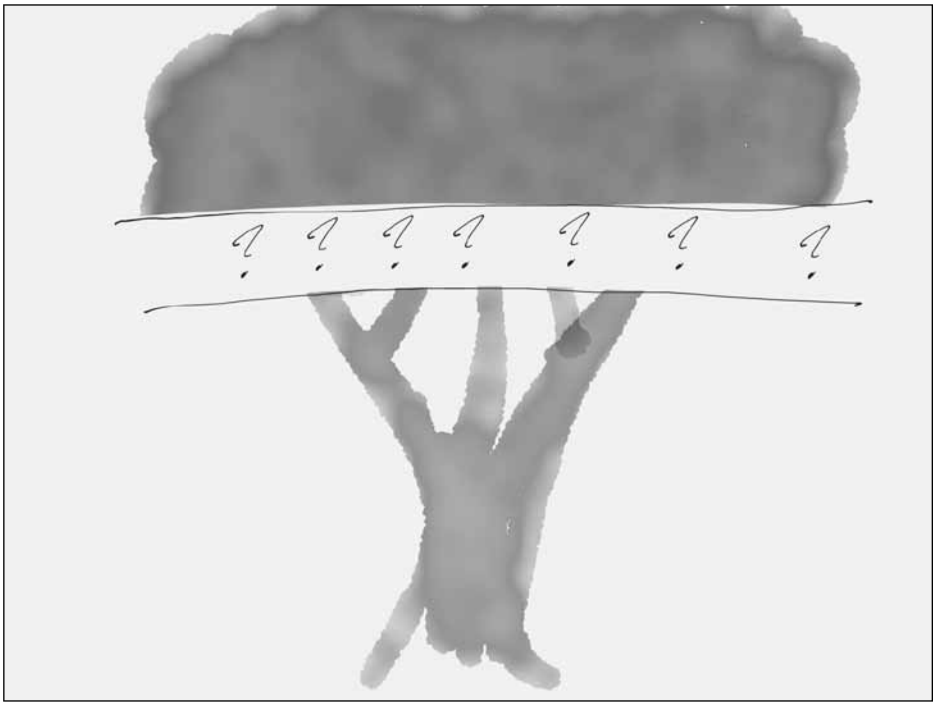
To be sure that they stay well inside their healthy zone, the manager provides a smaller Time-Money Box to those who report to them. By monitoring the effort at the end of each of these boxes, the manager has a clear understanding of how the effort is going and has flexibility to help it succeed.



We repeat this investment pattern all the way out to the leaves where product development is done.



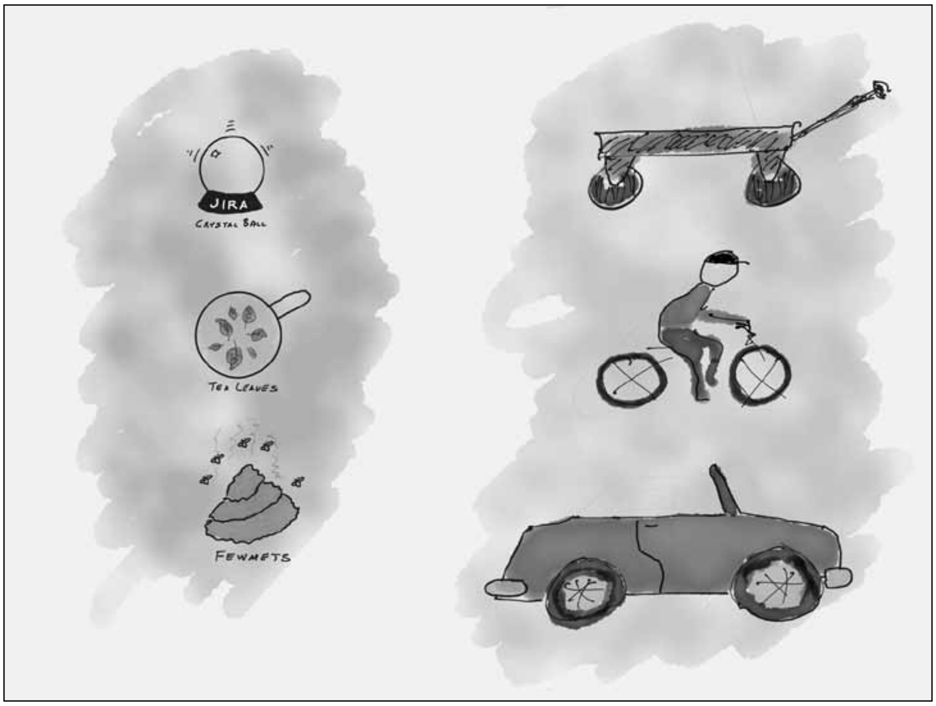
Then, however, we hit an important question. Here we are, out at the leaves, with our Agile product development going on. How can we find out better than with tea leaves and fewmets, how the product effort is progressing?



Manage by Increment

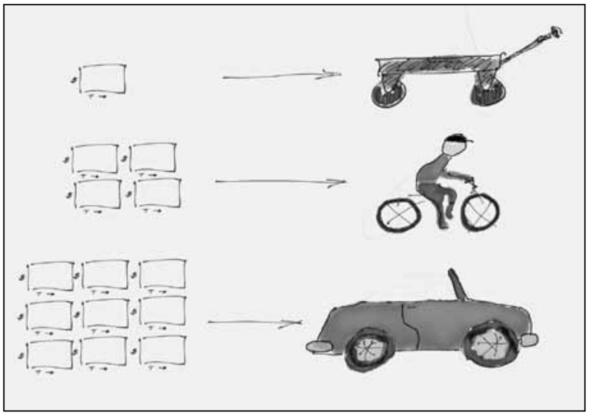
Scrum offers a "simple" fix for the Dark Scrum situation, and like most simple things, it isn't necessarily easy. Scrum requires that both stakeholders and development switch away from measuring things in a predictive fashion, using proxies like estimates and guesses. Instead, Scrum asks that we measure ourselves by directly examining what we have built so far.

Scrum demands that the team build a Product Increment, and centers all the key activities around the increment. We plan the next increment. We review it. We reflect on our difficulties in producing it. We refine our understanding of the shipped product by examining each Increment.



Management gives development teams a series of Time-Money Boxes. After each and every box, they examine the running, tested, fully-integrated software increments which the teams have produced. With tangible software in hand, management has the best possible sense of the future, the best possible material on which to base plans, the best possible basis for planning upcoming incrments.

As time and money add up in orderly boxes, the product matures. At each stage, management can pick the most important things to do next, ensuring the best possible product by any desired time or expenditure. And since every increment is running and tested, we're ready to ship on time, with the best possible combination of features.

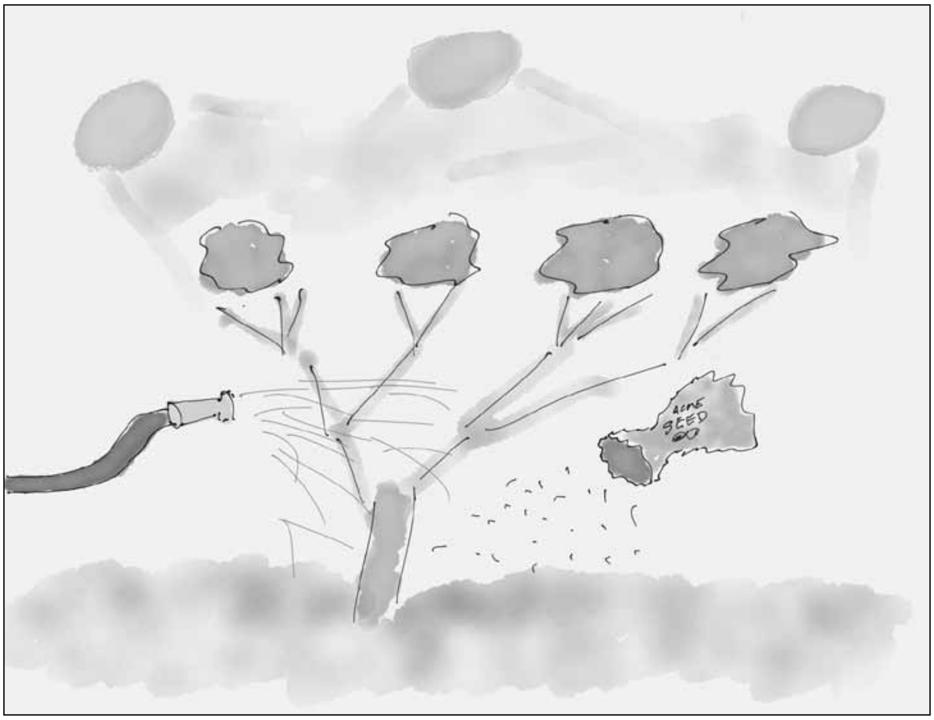


This is a new way of working, for most organizations. It requires that corporate stakeholders give up their old measures and questions, "When will you be done", "How much will it cost", "What will we have". Instead, we jointly look at what we actually have, the real product as it exists so far, and we jointly decide what to do next.

For this reason, all the current enterprise focus is a good thing. Over time, it can help the "Agile Mindset" to spread into the organization, providing the necessary support to the working Agile teams. We can think of it as a kind of fertilization of the trunk and branch, making them healthy so that they can and will provide the support that the teams really need.

This enterprise focus is based on a very simple truth: Agile ideas are not obvious and it is not obvious how to apply them. The conventional corporation, with the best of will, was created to control the "leaves", not to set them free. This inversion of the corporate hierarchy isn't just a nice metaphor: it's what really needs to happen.

And it is not enough. It is absolutely not enough. All the corporate focus, all the moistening and all the fertilizer are not enough to ensure effective product development at the leaves.



For stakeholders to have the opportunity to turn their attention to the Increment, and to turn around their management style, the developers need to be able to produce the Increment. This, too, is a new way of working. Developers are commonly used to building for a long time before anything works at all, and organizations expect long periods of testing and integrating at the end of a long and dreary project.

The Agile team produces a bright and cheerful, working, integrated, tested Product Increment every couple of weeks! And they have to learn how to do that.

Notice that our picture above shows the sun, shining on the leaves of our corporate tree, as well as care directed at the trunk and branches. For success in Agile development, the organization needs to provide that sun.

The trunk and branches don't automatically know how to manage in an Agile fashion. They don't know how to finance efforts in an Agile fashion. They don't know how to make product investment decisions, or how to specify products, in an Agile fashion. That's why all this enterprise focus is basically good: it helps the organization learn how to manage itself in an Agile-compatible way.

Management doesn't know how to operate in an Agile fashion. Why should development magically know?

The leaves of the tree, the product development teams who build whatever you're building with Scrum or Agile ideas, don't know how to do it either! They weren't born knowing how to do this any more than you were. They need to be shown, just as you do. And just as it takes time for people in the trunk and branches to see that this really works, it takes time for development teams as well.

Development teams do a different kind of thing from that of the stakeholders. The stakeholders help them plan an iteration or "Sprint", a couple of weeks of effort. Then the developers work for a couple of weeks and show the stakeholders what they've built. This "Increment", as it's called, is required to be a running, tested, integrated "product-so-far". It won't have every feature we hope it will have some day, but it will have every feature built so far, done, tested, working.

This isn't easy. (Development is never easy. If it were, stakeholders would do it themselves rather than put up with all those expensive weird-looking developers. But it's hard, so we hire people who can do it.)

Unfortunately, even though they look weird, developers aren't born knowing how to build in an Agile fashion any more than managers are born knowing how to manage in an Agile fashion. They have to learn.

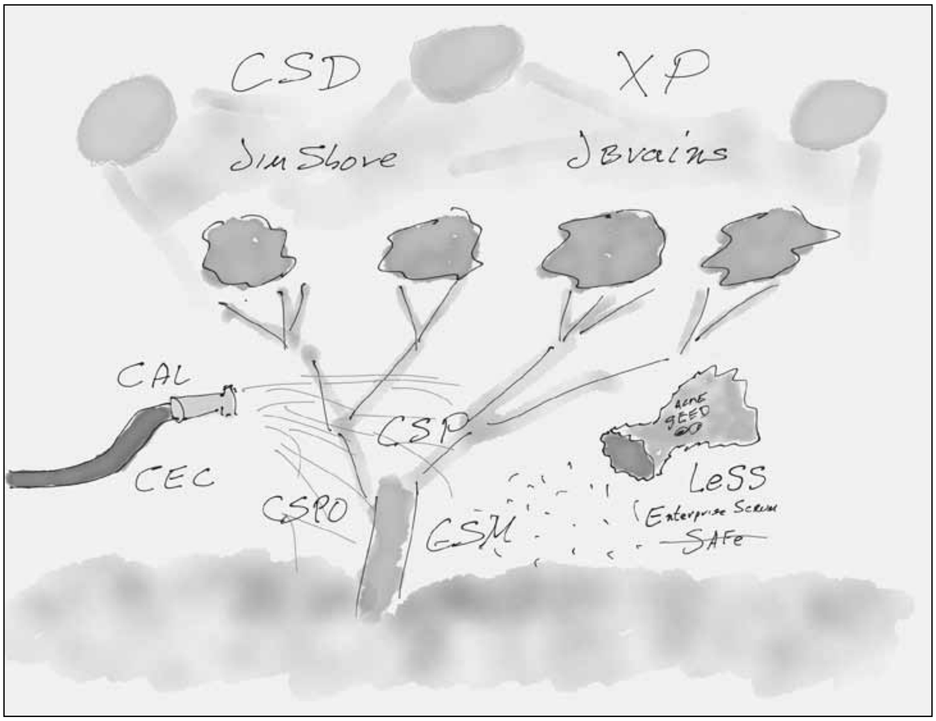
Developers have to learn to produce a running, integrated, tested Increment. Software development has historically been done with integration and testing at the end. That won't do for an Agile situation, so developers have to learn to do new things.

I have never seen a hyper-productive Scrum team that didn't use Extreme Programming development practices.

– Jeff Sutherland

The co-creator of Scrum, Jeff Sutherland, has made this explicit, as shown in the pull quote here. To be productive in an Agile software development situation, developers need to have new skills, including testing as they go, incremental design and development, and refactoring. They need to learn how to produce the Increment, and how to use it, jointly with their stakeholderss, to communicate what has been done and to decide what to do next.

Developer training and support is plentiful. Among the best alternatives are the Certified Scrum Developer program, which is based on the "XP Immersion" classes that Kent Beck, Bob Martin, and I created. And of course, Extreme Programming (XP) remains the repository of solid approaches to development in the Agile fashion. Web resources abound as well. We point here to only two, James Shore and J B Rainsberger, who are among the best.



Summing Up

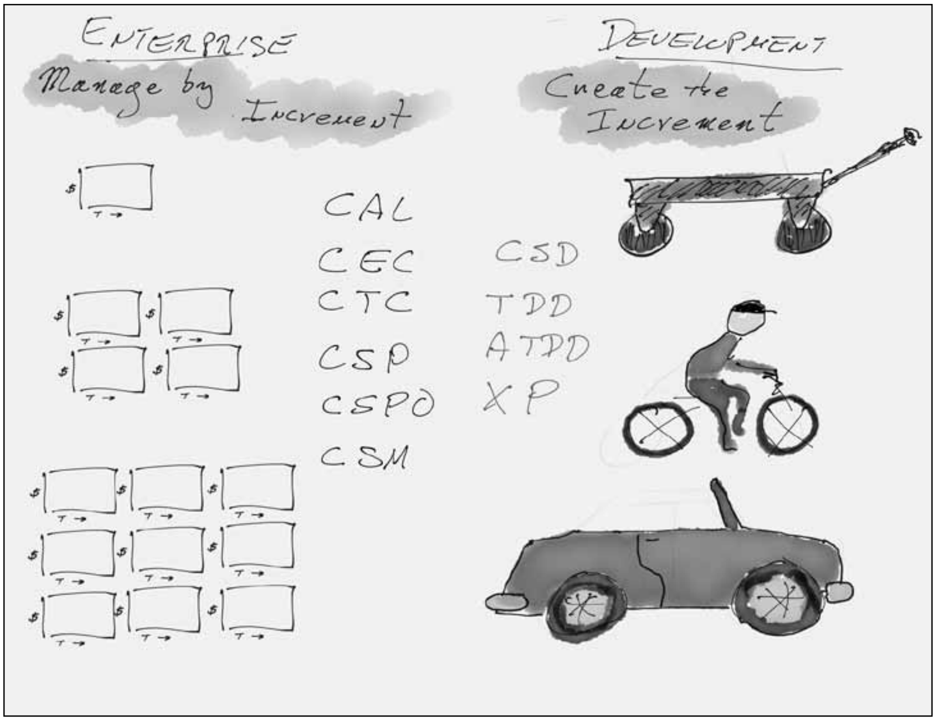
Management needs to invest using Time-Money Boxes and manage by examining the increment. Development needs to work within Time-Money Boxes to deliver running tested software increments.

There are support, coaching, training, and on-line resources available for both management and developers, and both managers and developers need support, coaching, training, and other resources.

Action Steps

1. The company's product stakeholders need to understand that just as they need some help and education to do their part in a move toward Agile ideas, the developers need help and education as well. A ScrumMaster course isn't enough to tell a developer how to do their part. Trainers and coaches need to be sure that all the stakeholders understand this.
2. Make sure that teams know that they can begin to turn around even the darkest Dark Scrum by producing a solid Product Increment, and focusing their planning and reviews on reality rather than proxy measurements and guesses. And make sure that they know how to do it.

If you focus on the Increment and bring everyone up to speed on creating and using the Increment, you'll get the most value from your Agile investment. This is the best way we know how to do it — in fact, it's the only way.



***

To read this article online with embedded hypertext links, go here:
https://ronjeffries.com/articles/017-02ff/gathering2017/

About Ron Jeffries

Ron has been developing software longer than most people have been alive. He holds advanced degrees in mathematics and computer science, both earned before negative integers had been invented. His teams have built operating systems, compilers, relational database systems, and a large range of applications. Ron's software products have produced revenue of over half a billion dollars, and he wonders why he didn't get any of it.

Does Your Coaching Build Roadblocks Instead of Relationships?

By Betsy Kaufmann

A big part of my role as an Agile coach is guiding clients through roadblocks. The roadblocks come in all shapes and sizes — organizational, team related and personal. I've encouraged reluctant executives, pacified anxious stakeholders, and coached old-school "waterfallers" into becoming agile advocates. But the one roadblock that continues to baffle me, comes from the most unexpected place — other agile coaches!

Almost every large organization has them—individuals that preach agile values both internally and externally but at the end of the day, let politics and paychecks get in the way of good practice.

The agile coach, more than any other role, should understand the critical importance of cooperation and collaboration. They should be mindful and espouse the agile manifesto and principles, which value customer collaboration, trust and transparency.

My plea to all agile coaches sounds something like this:

- **Partner with me.** Even if I'm a consultant, rather than an employee. Even if I'm a consultant from a competing firm. Even if my boss or client is your boss's or client's political nemesis. Even if I'm on the "wrong side" of the org chart. Even if I'm not on the org chart. Partner with me so we can build and design an alliance that is in the best interest of the customer as a whole.
- **Model collaboration.** The best way to help your team understand the benefits of agile collaboration is to model it yourself. Show your clients how to navigate politics, processes and hierarchies to most effectively serve the needs of the end users by modeling collaborative behavior. Coaches should develop shared key messages that keep all stakeholders focused on delivering value and promoting collaboration.
- **Be a resource.** Agile success is measured by how well we satisfy the customer through early and continuous delivery of value. Agile coaches, who get wrapped up in internal politics, refuse to leave their silos, or are only focused on growing their business, limit not only themselves, but also the organization's ability to deliver value. Agile coaches should

reach out, offer support and share best practices to aid other coaches and optimize the clients' transformation.

As agile coaches, we can't be roadblocks to each other. We need to be a united front on the road to agile transformation. That unity helps our stakeholders feel confident about the changes we are asking them to make. We need to set aside personal gain and politics for the sake of organizational success — united we stand, divided we fall.

⁂

To read this article online with embedded hypertext links, go here:
https://www.infoq.com/articles/escape-method-prson

About Betsy Kaufmann

Betsy is a passionate Organizational Coach and Trainer with more than 18 years' experience working with high performing teams. As an Organizational Coach, she is responsible for coaching, training, and implementing best practices at the executive, program and team levels for several Fortune 500 organizations. Betsy was selected by Agile Alliance to be one of seven authors for the *Agile Practice Guide* published in conjunction with the Project Management Institute (PMI)® in September 2017. Betsy is actively involved in the community and enjoys presenting on a range of topics regarding organizational agility, adaptive leadership and agile values. Betsy is Vice President of Software Education USA LLC (www.softed.com) and also President of Agile Pi Inc. (www.agilepi.com)

Myth: Scrum Events Take Too Much Time

By Jason Knight

[**Note:** grayed text indicates hypertext links in original article.]

Myth: Scrum Events Take Too Much Time

A refrain I hear often is that Scrum events are a passle of useless meetings that constantly go over their timeboxes and take valuable time away from the actual work of developing software. It's a bit like saying, a hammer is terrible for driving in screws and only ends up smashing your fingers.

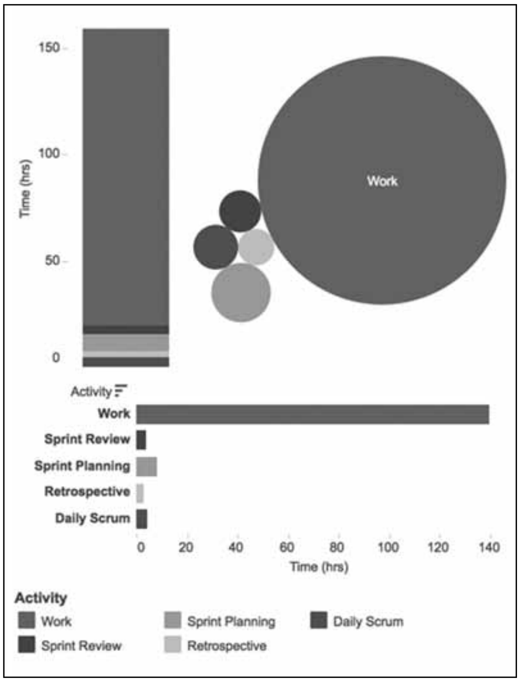
Hammers and Screws

A hammer is designed to drive nails. Scrum events are designed to exert a specific and focusing pressure on the goal of the time. In his book **Death by Meeting** Lencioni explains how different types of meetings have different uses. Trying to force a daily check-in to take the place of a weekly tactical or a monthly strategic to take the place of a quarterly off-site review is like using a hammer to drive in a screw. Don't get me started on the meeting stew where all four are slow simmered into the counter productive, life-sucking slop we all know and loathe.

Each event in Scrum has a carefully designed purpose, like a tool. Each is a precisely honed feedback loop designed to have a particular effect.

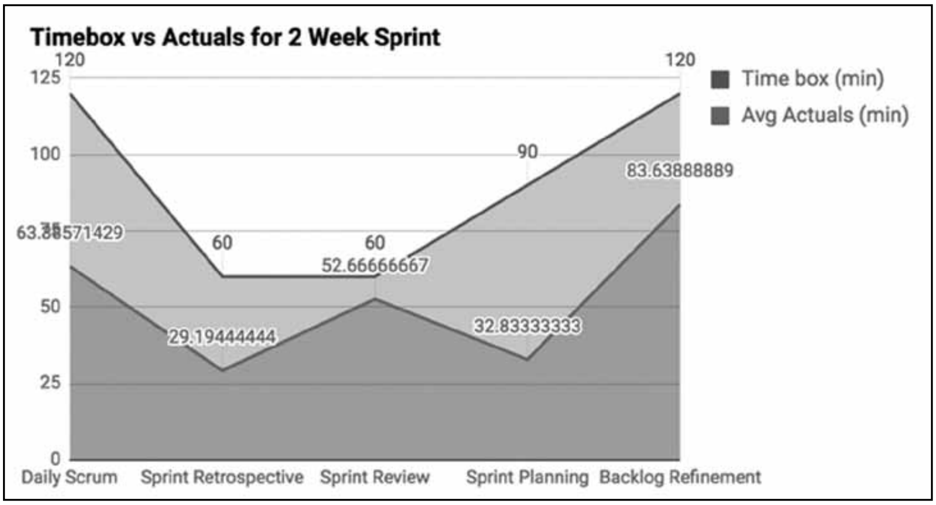
Smashed Fingers

Too much time spent in meetings can feel like an errant hammer blow to the thumb. Let's see how the time boxes of Scrum compare to the overall time available in a typical, 8-hour workday:



That huge chunk of work time is the *minimum* amount of time Scrum sets aside for designing, testing, coding, delivering etc. or in other words *developing software.* These numbers are based on a 30 day Sprint (counting 5, 8-hour work days per week for 4 weeks). They also assume the maximum timeboxes for each Scrum event. From these it's clear Scrum intends to take the as little time away from the work of producing working software as possible.

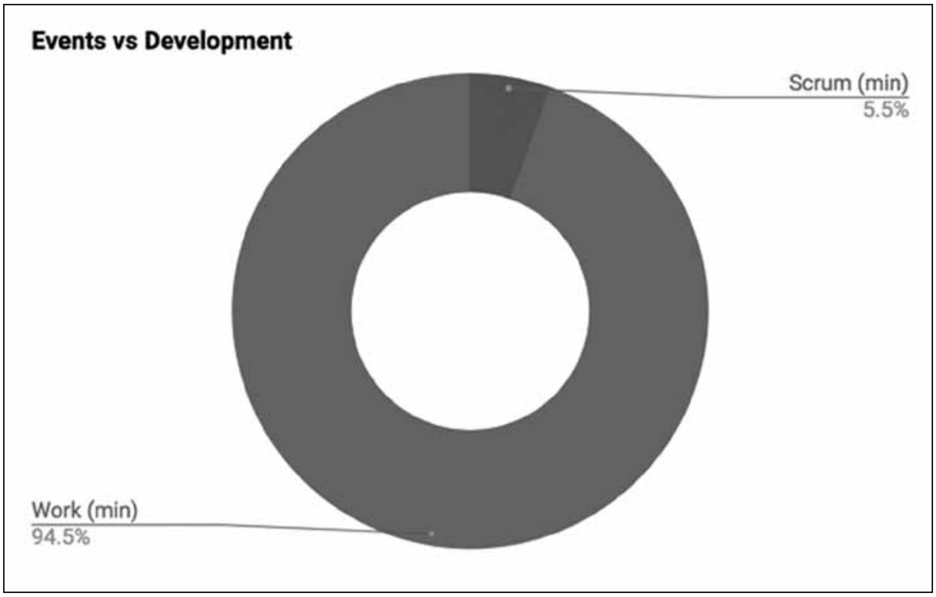
Even if the purpose of each Scrum event is well understood, executing them with precision can still be a challenge. Things like effective facilitation, good tooling and mutual respect among participants are necessary to execute each event expertly. Let me show you what we've been able to achieve where I work:



Important note: backlog refinement isn't one of the Scrum events, but we decided to track that time as well.

We've been able to squeeze the value out of each Scrum event without consuming the whole timebox…*for every event.* We can turn on a dime when most requirements or technology realities change, we quickly identify and solve many daily issues that would sink silo'ed teams, and we set aside time to have the hard conversations necessary to improve how we work.

Now for the big reveal, how much time does all this Scrum *idealism* cost us? About 5% of our total, average workday:



To be sure, we have administrative time costs, time lost at the coffee machine or around the ping pong table. We have thoughtless meetings here and there that sap our will to live. We waste time in new and creative ways occasionally, but Scrum don't care about that. Moreover, we can't honestly blame Scrum for that waste. Scrum calls us forth to operate as professionals in full command of our work time and with clear insight into the system around us.

Your Mileage May Vary…a Bit

Not everyone will become as effective at practicing the Scrum events as we have become. It wasn't instant. It took hard work over several years to get to our current level of dysfunction :). If you're faithful to study the purpose and perfect the execution of each event, you'll get there too.

⁂

To read this article online with embedded hypertext links, go here:
http://jasontknight.com/myth-scrum-events-take-too-much-time/

About Jason Knight

Jason is known around Tulsa, OK as the guy to talk to about Scrum and agility in general. He loves software development, the practice of agility and servant leadership. His journey has taken him from small development jobs to enterprise level coaching and teaching. Along the way, he's picked up many useful skills and a wealth of experience in the areas of professional software development, agile frameworks and methodologies, and leadership development through modeling, teaching and coaching.

These days, he works as a Scrum Master, lean coach and even a sort of internal consultant for his organization. He's learning more and more about how leaders can use sense-and-respond techniques to be effective in complex, adaptive environments. This interest has led him to join the Agile Leadership Institute and grow to be their Chief Accountability Officer. He still writes on his blog every now and then too :).

WIP Limits Must Die

By Klaus Leopold

A drastic title, but I really mean it. Some people have a fit when I say that you should limit the work in a Kanban system. The notion of limiting them, and the work, leaves an unpleasant aftertaste. At the implementation level, it sounds like, "You think I'm not capable of doing two things at once?" At higher levels, for instance in portfolio management, it sounds like, "We are rejecting customer orders."

In the world of working effectively, WIP limits are a core element. Their purpose is to simply prevent you from getting bogged down. This bogging down is most apparent when the only thing being discussed is starting initiatives, proposals and projects. Meanwhile, we know multitasking is a myth and companies are not successful because they start as many projects as possible, but rather when they finish as many projects as possible.

Nothing can fly where everything lands

Here's the thing: We do not want to restrict or constrain work with WIP limits. Rather, we want to get to the point where arrival and departure rates in the system are nearly equal. I like to compare this to an airport: When there are more airplanes landing than taking off, the entire area will be piled up with airplanes in very short order. It is absolutely logical that an airport has a certain capacity (WIP limit) and that arrivals and departures are planned based on this capacity (starting and completing work). If the airport is at capacity, airplanes must depart (work must be completed) before the next airplanes can land (new work can be started).

Most importantly, limiting the amount of work in a work system is a means to an end. There should not be more work started than can be finished. To prevent the system from becoming clogged, there can only be a certain amount of active work, and this amount is represented by the WIP limit. Even though my inherent enthusiasm for WIP limits will probably never waver, and from every possible practical and theoretical point of view they simply make sense, I find myself more and more often trying to avoid the term "limit". It prompts many people to make an incorrect association. But I am baffled at the moment how to phrase WIP limits differently.

Does anyone have an idea? I would be thankful for any suggestions.

***

To read this article online with embedded hypertext links, go here:
https://www.leanability.com/en/blog-en/2017/10/wip-limits-must-die/

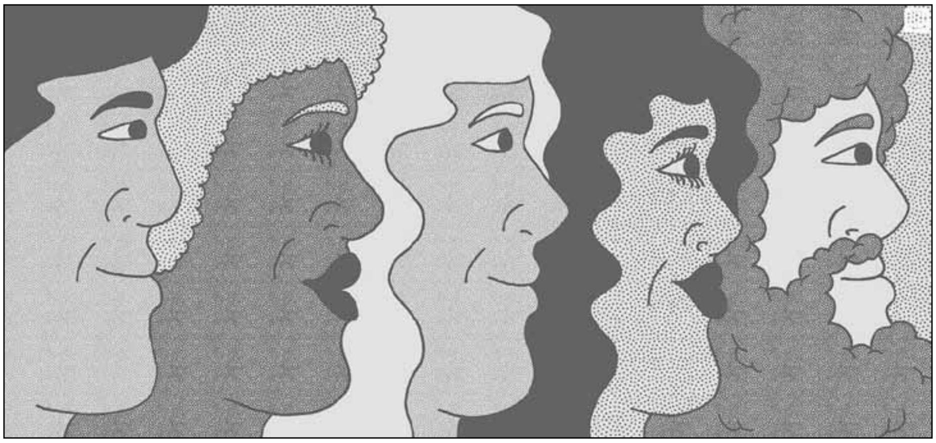
About Klaus Leopold

Klaus is computer scientist and Kanban pioneer with many years of experience in helping organizations from different industries on their improvement journey with Lean and Kanban. He is author of the book *Practical Kanban* (www.practicalkanban.com) and co-author of the book "Kanban Change Leadership". Klaus is one of the first Lean Kanban trainers and coaches worldwide. He was awarded with the Brickell Key Award for "outstanding achievement and leadership" within the Lean Kanban community in San Francisco, 2014. His main interest is establishing lean business agility by improving organizations beyond the team level, especially in large environments from 50 to 5000 people. Klaus speaks regularly at renowned Lean and Kanban conferences worldwide. He publishes his current thoughts on his blog www.LEANability.com and you can follow him on Twitter at @klausleopold.

Engineering a Culture of Psychological Safety

By John Loomey

[**Note:** grayed text indicates hypertext links in original article.]



When I worked for Google as a Site Reliability Engineer, I was lucky enough to travel around the world with a group called "Team Development". Our mission was to design and deliver team-building courses to teams who wanted to work better together.

Our findings were later **published as Project Aristotle.** The biggest finding was that the number-one indicator of a successful team wasn't tenure, seniority or salary levels, but **psychological safety.**

Think of a team you work with closely. How strongly do you agree with these five statements?

1. If I take a chance, and screw up, it will be held against me
2. Our team has a strong sense of culture that can be hard for new people to join.
3. My team is slow to offer help to people who are struggling.
4. Using my unique skills and talents come second to the objectives of the team.

5. It's uncomfortable to have open honest conversations about our team's sensitive issues.

Teams that score high on questions like these can be deemed to be "unsafe". Unsafe to innovate, unsafe to resolve conflict, unsafe to admit they need help. Unsafe teams can deliver for short periods of time, provided they can focus on goals and ignore interpersonal problems. But eventually, unsafe teams will break or underperform drastically because people can't introduce change.

Unsafe teams will break or underperform drastically because people can't introduce change.

Let's highlight the impact an unsafe team can have on your team's individuals, through the eyes of a recent, fresh-faced and enthusiastic graduate who finished top of their class.

Unsafe teams will break or underperform drastically because people can't introduce change.

This imaginary graduate, we'll call her Karen, was reading about an optimization that could reduce low-level locking in distributed databases, and realized it could be applied to the service her team worked on. She decided to test it out, it resulted in a 15% CPU saving on the test cluster, and in her excitement, decided to roll it out to production. Because it was a change to a database configuration file, it didn't go through the usual code-review process.

Unfortunately, it caused the database to hard-lock-up, causing a brief, but total outage of the website. Thankfully, her more experienced colleagues spotted the problem, and rolled back the change inside of 10 minutes. Being professionals, this incident was mentioned at the weekly "post-mortem" meeting.

1. "If I take a chance, and screw up, it'll be held against me"

At the meeting, the engineering director let everyone know that causing downtime by chasing small optimizations was unacceptable. Karen was described as "irresponsible" in front of the team, and the team suggested ways to ensure it wouldn't happen again. The engineering director forgot about this interaction quickly after. But Karen would never forget the exchange. She would never try to innovate without explicit permission again.

2. "Our team has a strong sense of culture, and it's hard for new people to join"

The impact on Karen was actually magnified because no one stood up for her. No one pointed out the lack of code reviews on the database configuration allowed this to happen. No one pointed out the difference between highlighting one irresponsible act and labelling someone "irresponsible". The team was so proud of their system's reliability, defending their reputation was more important than a new hire.

Karen learned that her team, and manager didn't have her back.

3. "My team is slow to offer help to people who are struggling"

As Karen was new to "production", she had no formal training in incident management, production hygiene, let alone troubleshooting distributed systems. As her team was mostly made up of people with decades of experience, they had never needed training, or new-hire documentation. There were no signals that it was OK for a new graduate to spend time learning these skills.

Karen developed **Imposter Syndrome.** She didn't understand how she passed the hiring process, and frequently wondered why she hadn't been fired yet.

4. "Using my unique skills and talents come second to the goals of the team"

Karen's background was in algorithms, data structures and distributed computing. She realized the existing system as a whole was suboptimal, and would never handle load spikes.

The team had always blamed the customers for going over their contracted rates, which is like blaming weathermen for rain during an Irish barbecue.

Karen proposed a new design, based on technology she'd used during her internship. Her co-workers were unfamiliar with the new technology and considered it too risky. Karen dropped her proposal without discussion. She wanted to write code and build systems, not have pointless arguments.

5. "It's uncomfortable to have open, honest conversations about our team's sensitive issues"

When a large customer traffic spike caused the product to be unavailable for a number of hours, the CEO demanded a meeting with the operations team. Many details were discussed, and Karen explained that the existing design meant it could never deal with such spikes, and mentioned her design. Her director reminded her that her design had already been turned down at an Engineering Review, and promised the CEO they could improve the existing design.

Karen discussed the meeting with one of her teammates afterwards. She expressed dismay that the Director couldn't see that his design was the root-cause of their problems. The teammate shrugged, and pointed out that they had delivered a really good service for the last five years, and had no interest talking about alternate designs with the director.

Karen decided to head home early, and look for a new job. When she left, the company didn't miss her. After all, she was "reckless, whiny and had a problem with authority". They never realized she had the design that could have saved the product from the customer exodus that follows repeated outages.

How to build psychological safety into your own team

So, what is special about Engineering that leads us to drive away so many promising engineers, and allow so many others to achieve less than their potential?

We need to balance respect for our culture, with an openness to change it as needed.

"We need to balance respect for our culture, with an openness to change it as needed."

We know that a strong sense of culture, shared understandings and common values are required to succeed. So we need to be able to balance that respect for our culture, with an openness to change it as needed. A team — initially happy to work from home — needs to change how they work if they take on some interns. A team — proud that every engineer is on-call for their service — may need to professionalize around a smaller team of operations-focused engineers as the potential production impact of an outage grows.

We need to be thoughtful about how we balance work people love, with work the company needs to get done. Good managers are proactive about moving on an engineer who is a poor fit for their team's workload. Great managers expand their team's remit to make better use of the engineers they have, so they feel their skills and talents are valued. Engineers whose skills go unused grow frustrated. Engineers given work they are ill-equipped to succeed, will feel setup to fail.

Make respect part of your team's culture

It's hard to give 100% if you spend mental energy pretending to be someone else. We need to make sure people can be themselves by ensuring we say something when we witness disrespect. David Morrison (Australia's Chief of the Army) captured this sentiment perfectly, in his **"the standard you walk past is the standard you accept"** speech.

Being thoughtless about people's feelings and experiences can shut them down. Some examples where I've personally intervened:

- Someone welcomes a new female project manager to the team, assumes they aren't technical and uses baby words to explain a service. I highlight the new PM has a PhD in CS. No harm was intended, and the speaker was mortified that their good-humored introduction was taken the wrong way.
- In a conversation about people's previous positions, someone mentioned they worked for a no-longer-successful company, and a teammate mocked them for being "brave enough" to admit it. I pointed out that mocking people is unprofessional and unwelcome, and everyone present understood a 'line' that hadn't been visible previously.
- A quiet, bright engineer consistently gets talked over by extroverts in meetings. I point out to the "loud" people that we were missing an important viewpoint by not ensuring everyone speaks up. Everyone becomes more self-aware.

It's essential to challenge lack of respect immediately, politely, and in front of everyone who heard the disrespect. It would have been wonderful had someone reminded Karen's director, in front of the group, that the outage wasn't a big deal, and the team should improve their test coverage.

Make space for people to take chances

Some companies talk of 20% time. Intercom has "buffer" weeks, in between some of **our 6-week sprints.** People often take that chance to scratch an itch that was bothering them, without impacting the external commitments the team has made. Creating an expectation that everyone on the team should think outside the box, and ensuring that the whole team can go off-piste at the same time, is a powerful message.

Be careful that "innovation time" isn't the only time people should take chances. One company in the transport industry considers "innovation time" to be 2:30 p.m. on Tuesdays.

Imagine how grateful Karen would have been, had a senior engineer at the Engineering Review offered to work on her design with her, so it was more acceptable to the team. Improve people's ideas, rather than discounting them.

Make it obvious when your team is doing well

I love how my team writes goals on Post-It notes at our daily standups and weekly goal meetings. These visible marks of success can be cheered as they are moved to the "done" pile.

"We can also celebrate glorious failure."

But we can also celebrate glorious failure. Many years ago, when I was running one of Google's storage SRE team, we were halfway through a three-year project to replace the old Google File System.

We can also celebrate glorious failure.

Through a confluence of bad batteries, bad firmware, poor tooling, untested software, an aggressive rollout schedule and two power cuts, we lost a whole storage cell for a number of hours, and though all services would have had storage in other availability zones, the team spent three long days, and three long nights rebuilding the zone. Once it was done, they — and I — were dejected. Demoralized. Defeated. An amazing manager who was visiting our office realized I was down, and pointed out that we'd just learned more about our new storage stack in those three days, than we had in the previous three months. He reckoned a celebration was in order.

I bought some cheap sparkling wine from the local supermarket, and with another manager, took over a big conference room for a few hours. Each time someone wrote something they learned on the whiteboard, we toasted them. The team that left that room was utterly different to the one that entered it.

I'm sure Karen would have loved appreciation for discovering the team's weak non-code test coverage, and their undocumented love of uptime-above-all-else.

Make your communication clear, and your expectations explicit

Rather than yelling at an engineering team each time they have an outage, help them build tools to measure what an outage is, a Service Level Objective that shows how they are doing, and a culture that means they use the space between their objective, and reality, to choose the work that will have the most impact.

Ask for a specific commitment, rather than assuming everyone agrees on its urgency.

"Ask for a specific commitment, rather than assuming everyone agrees on its urgency."

When discussing failures, people need to feel safe to share all relevant information, with the understanding that they will be judged not on how they fail, but how their handling of failures improved the team, their product and the organization as a whole. Teams with operational responsibilities need to come together and discuss outages and process failures. It's essential to approach these as fun learning opportunities, not root-cause obsessed witch-hunts.

I've seen a team paralyzed, trying to decide whether to ship an efficiency win that would increase end-user latency by 20%. A short conversation with the product team resulted in updates to the SLO, detailing "estimated customer attrition due to different latency levels", and the impact that would have on the company's bottom line. Anyone on the team could see in seconds that low-latency was far more important than hardware costs, and instead drastically over-provisioned.

If you expect someone to do something for you, ask for a specific commitment – "When might this be done?", rather than assuming everyone agrees on its urgency. Trust can be destroyed by missed commitments.

Karen would have enjoyed a manager who told her in advance that the team considered reliability sacred, and asked her to work on reliability improvements, rather than optimizations.

Make your team feel safe

If you are inspired to make your team feel more psychologically safe, there are a few things you can do today:

1. Give your your team a short survey, and share the results with your team
2. Discuss what "Safety" means to your team; see if they'll share when they felt "unsafe"
3. Build a culture of respect & clear communication, starting with your actions

Treat psychological safety as a key business metric, as important as revenue, cost of sales or uptime. This will feed into your team's effectiveness, productivity and staff retention and any other business metric you value.

⁂

To read this article online with embedded hypertext links, go here:
https://www.intercom.com/blog/psychological-safety/

About John Loomey

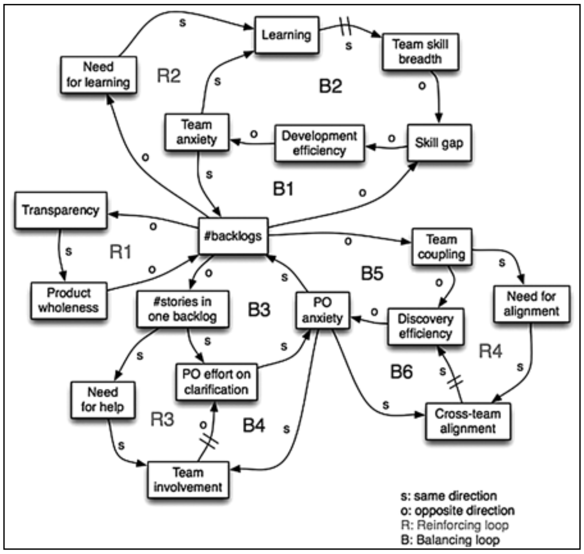
The biography and picture of this author/member of the Nominating Committee was not received in time for publication."

Seeing the system dynamic: 1 vs. n product backlogs

By Yi Lv

In a product organization with multiple teams, it raises a choice - whether to have one or many product backlogs. They usually start with one product backlog, either because they start with one pilot team, or because their product starts small from one team. Later, some organizations choose to have many product backlogs in response to more teams, while other organizations choose to keep one product backlog. When having many product backlogs, usually separate PO will be responsible for each backlog.

The below CLD illustrates the system dynamic around this topic.



Drive for one product backlog

As this is one product, it should be quite natural to think of one product backlog. The **R1-loop** reads like this.

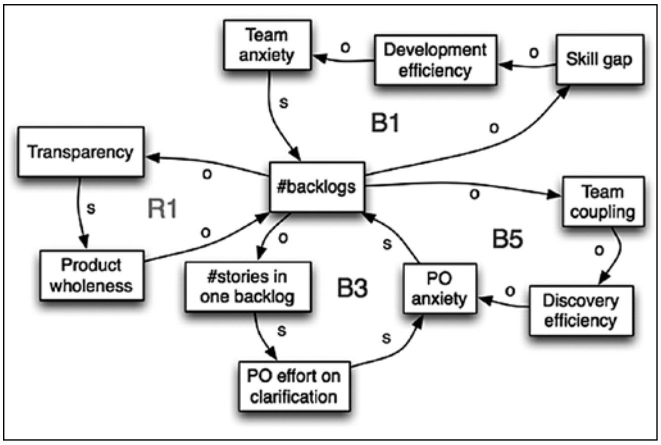
- the fewer product backlogs

- the more transparency
- the better product wholeness
- the fewer product backlogs

Potentially this could be a virtuous cycle, which eventually leads to one product backlog.

Why having many product backlogs?

Then, why do some organizations choose to have many product backlogs? There are three main balancing loops in play, which are **B1-loop, B3-loop** and **B5-loop**. Together with **R1-loop,** it creates "limits to growth" system archetype.



B1-loop reads like this:

- the fewer product backlogs
- the bigger skill gap
- the lower development efficiency
- the more anxious team gets
- the more product backlogs

B1-loop illustrates the limitation from team specialization. In order to make use of team's specialization for efficiency, product backlog essentially becomes team backlog to match their skills. This dynamic is similar as the one involved in having generic vs. specialized teams. However, there is fundamental solution, and we shall elaborate on it later.

B3-loop reads like this:

- the fewer product backlogs
- the more stories in each backlog
- the more effort by PO on clarification (assumption: PO does requirement

clarification)
- the more anxious PO gets
- the more product backlogs

B3-loop illustrates the limitation from requirement clarification. There is common misunderstanding about PO clarifying requirements for teams. If PO does all of that, it becomes a limiting factor for having one product backlog. However, there is fundamental solution, and we shall elaborate on it later.

B5-loop reads like this:

- the fewer product backlogs
- the more coupled among teams
- the less efficient in discovery and decision making
- the more anxious PO gets
- the more product backlogs

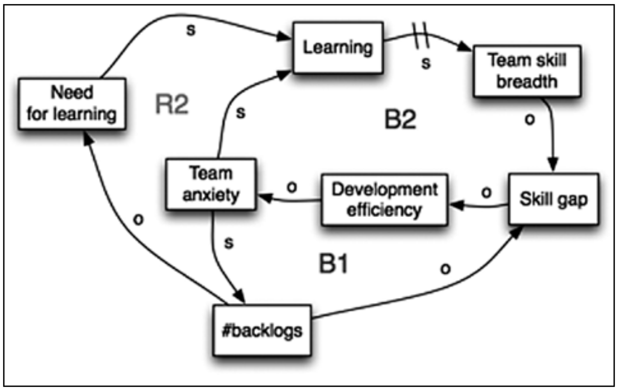
B5-loop illustrates the limitation from discovery and decision making. The assumption here is that every team has its own PO, and it is more efficient when PO could make decisions on his own. However, there is fundamental solution, and we shall elaborate on it later.

These are main restraining forces for having one product backlog. They limit **R1-loop** and damage the product wholeness. The leverage lies at weakening those forces by looking for fundamental solutions.

Look for fundamental solutions

Corresponding to **B1-loop, B3-loop** and **B5-loop,** there are alternative fundamental solutions shown as **B2-loop, B4-loop** and **B6-loop,** respectively. However, those solutions are with delay, thus, long-term. The short-term solution (i.e. having many product backlogs) shifts the focus on long-term solutions. That is essentially what "Shifting the burden" system archetype is about.

1. Team specialization



The fundamental solution is shown as **B2-loop,** which reads like this:

- the lower development efficiency
- the more anxious team gets
- the more learning
- the broader team skill gets (with delay)
- the smaller skill gap
- the higher development efficiency

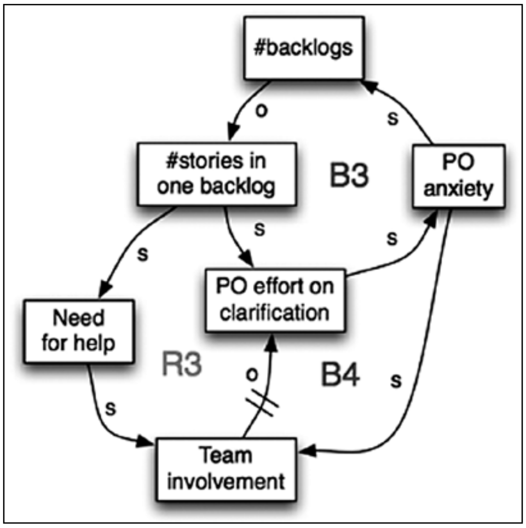
Instead of having many product backlogs to reduce skill gap for development efficiency, we focus on learning and expanding team skill breadth, eventually leading to higher development efficiency.

R2-loop is the addictive loop in "Shifting the burden", which reads like this:

- the more product backlogs
- the less perceived need for learning by team
- the less learning
- the narrower team skill gets (with delay)
- the bigger skill gap
- the lower development efficiency
- the more anxious team gets
- the more product backlogs

When having many product backlogs sort of fixes the development efficiency, we tend to focus less on learning and expanding team skill breadth, and become more addictive to having many product backlogs.

2. Requirement clarification



The fundamental solution is shown as **B4-loop,** which reads like this:

- the more effort by PO on clarification
- the more anxious PO gets
- the more involved team gets in requirement clarification
- the less effort by PO on clarification (with delay)

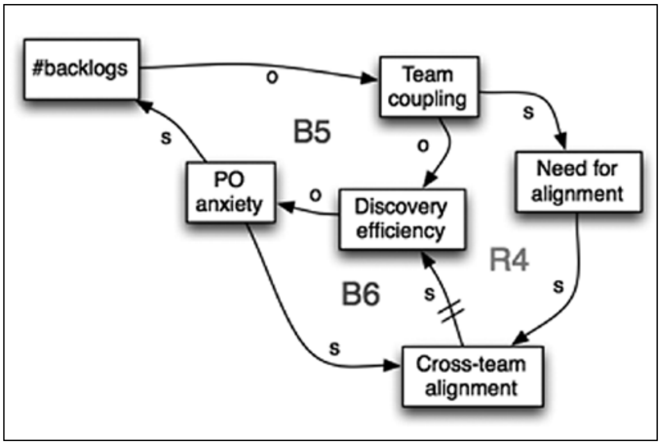
Instead of having many product backlogs to reduce PO effort, we focus on getting team involved in requirement clarification, eventually leading to reduced workload from PO side. The delay is caused by team having to learn how to work with users and the domain in order to do the proper requirement clarification.

R3-loop is the addictive loop in "Shifting the burden", which reads like this:

- the more product backlogs
- the fewer stories in each backlog
- the less perceived need for help by PO
- the less involved team gets in requirement clarification
- the more effort by PO on clarification (with delay)
- the more anxious PO gets
- the more product backlogs

When having many product backlogs sort of fixes PO effort problem, we tend to focus less on getting team involved in requirement clarification, and become more addictive to having many product backlogs.

3. Discovery and decision making



The fundamental solution is shown as **B6-loop,** which reads like this:

- the less efficient in discovery and decision making
- the more anxious PO gets

- the more alignment across teams
- the more efficient in discovery and decision making (with delay)

Instead of having many product backlogs to reduce team coupling for discovery efficiency, we focus on getting teams aligned and increasing the capability of group decision making, eventually leading to more efficient discovery with group of teams and POs. The delay is due to the time and effort necessary to create cross-team alignment and build group collaboration capability.

R4-loop is the addictive loop in "Shifting the burden", which reads like this:

- the more product backlogs
- the less coupled among teams
- the less perceived need for alignment across teams
- the less alignment across teams
- the less efficient in discovery and decision making (with delay)
- the more anxious PO gets
- the more product backlogs

When having many product backlogs sort of fixes discovery efficiency problem, we tend to focus less on creating alignment and building group collaboration capability, and become more addictive to having many product backlogs.

Summary

As we have one product, it is desirable to have one product backlog. We look at what prevents us from doing that. Those are barriers we need to overcome. There are three common reasons why having many product backlogs — team specialization, requirement clarification, discovery and decision making. We look at fundamental solutions for those, and how to avoid the traps associated with the quick fix, i.e., having many product backlogs.

***

To read this article online with embedded hypertext links, go here:
https://blog.odd-e.com/yilv/2017/01/seeing-the-system-dynamic-1-vs-n-product-backlogs.html

About Yv Li

Yi Lv (the correct *Pinyin* is Lü) lives in Hangzhou, China. He is an agile coach at Odd-e. He is the first Certified Scrum Trainer (2008) and the first Certified LeSS Trainer (2018) from China.

From late 2005, while working in Nokia Networks, he started to get acquainted with agile software development, in particular, Scrum. That turned out to be his first experience of LeSS adoption. He led a department inside that product organization and focused on developing teams and Scrum Masters to create sustainability. He left Nokia Networks in late 2010 and joined Odd-e till now.

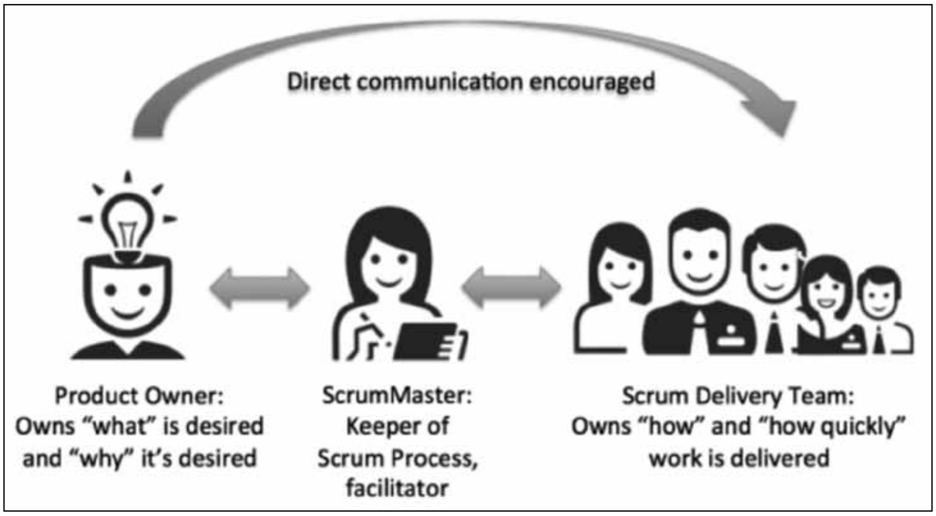
As a coach at Odd-e, he also worked in other industries such as Internet companies. His focus has been on large-scale product development, especially helping organizations benefit from LeSS and/or its adoption. He has been learning and practicing systems thinking since 2009. Over the past couple of years, he wrote a series of blogs to help see system dynamics in organizational design and change.

He can be reached at yi.lv@odd-e.com.

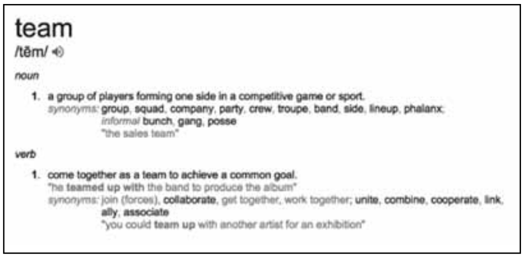
Why the idea of a scrum team is so powerful..

By Nirmaljeet Malhotra

[**Note:** grayed text indicates hypertext links in original article.]


Direct communication encouraged
Product Owner: Owns "what" is desired and "why" it's desired
ScrumMaster: Keeper of Scrum Process, facilitator
Scrum Delivery Team: Owns "how" and "how quickly" work is delivered

The idea of a team has evolved over the last decade. What started off with a group of people working together to achieve a vague goal under the control of a manager/ leader, has in some cases matured where teams are gradually getting more engaged and are aware of the business objectives and are being trusted to get to the finish line.



team
/tēm/

noun

1. a group of players forming one side in a competitive game or sport.
   synonyms: **group**, **squad**, **company**, **party**, **crew**, **troupe**, **band**, **side**, **lineup**, **phalanx**;
   informal **bunch**, **gang**, **posse**
   "the sales team"

verb

1. come together as a team to achieve a common goal.
   "he **teamed up** with the band to produce the album"
   synonyms: join (forces), **collaborate**, get together, work together; **unite**, **combine**, **cooperate**, **link**, **ally**, **associate**
   "you could **team up** with another artist for an exhibition"

The idea of a scrum team presented a new twist to the definition of a team, obviously with its share of discomforts. The thought of a team without a manager, attributes of self organization and self management and emphasis to build trust sounded great but had many heads shaking.

While some organizations have introduced structural changes to embrace 3 scrum roles (Scrum Master, Product Owner, Development team), most organizations are trying to fit the new roles in the context of their current organizational structure or are making a effort to somehow align existing roles to the new ones.

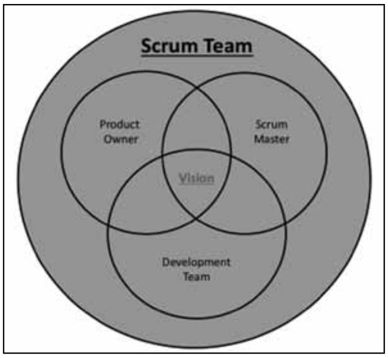
The thought that some existing roles may become redundant can be discomforting and lead to resistance. Some common questions/opinions are highlighted:

- What about the "other" roles like business analysts, architects, project managers etc..?
- These people have been with the organization for ever. We can't let them go.
- Our product owners are customer facing and have other responsibilities. They cannot be available to the team.
- A Scrum Master? Who is going to manage the team?
- Our teams are not mature enough to self organize.

The above questions are clearly indicative of the lack of understanding of the roles and the fact that the organization is focussed on individual roles and not the overarching impact of the roles.

The intent behind the idea of a scrum team was to bring all aspects of product development (business/product, engineering and process) together in order to realize the end goal. While the simplicity of the framework makes it acceptable, the roles continue to operate in isolation and be looked as "speciality driven". To simplify, Product Managers assume that the responsibility of development team is to implement their ideas only.

As I went around coaching many organizations, I have always made a focussed effort to communicate the attributes of a successful and high performing scrum team, and the attributes that make the idea of a scrum team so powerful. Here are some key attributes that distinguish the great scrum teams from the good ones:



Inclusiveness – Scrum teams works best in a inclusive environment. This means that while every individual might have a set of responsibilities that come with his/her role, what creates a big impact is how these roles come together and contribute to the overall success of the product. The idea that only Product Managers are responsible for product strategy, analysis and business decisions and development team implements the decisions made the manager defeats the purpose of a scrum team. In my experience, teams that have been able to achieve the highest level of productivity and created seriously innovative and disruptive products are the ones where these roles collaborate and engage on a day to day basis.

No culture can live if it attempts to be exclusive

Mahatma Gandhi

For example; the complexity and the time taken to implement a functionality can negate the value of the feature. This information from the development team can impact the priority of the items in the backlog and help the Product Manager make better decisions. So, the idea of a collaborative team that embraces the scrum practices as intended can have a positive impact on the business value produced and accelerate the time take to do so.

For a patient at a hospital going through a surgical procedure involving doctors from a variety of specializations, each doctor constantly provides inputs to others to make sure that every aspect of the patient's health is known to reduce risks and keep focus on patient's recovery. Each one is included to achieve the end goal.

Alignment – can go a long way in defining the interest of scrum team members. Often, team members have a very narrow focus on the immediate tasks at hand and lack clarity of the business goals and objectives. Creating alignment is a critical aspect for a scrum team.

Alignment is a practice, not a state.

Unknown

Alignment is critical both at the business and process level and the scrum framework provides practices to help create the alignment through the empirical process control. The scrum team exists so that product, engineering and process can tweak things to stay on course to achieve desired outcome.

Talking about alignment, US and India launched their respective missions to Mars about a year ago. A very big part of the journey to Mars that lasts about a year to complete is to adjust the trajectory of the space vehicle to aligned with the ultimate goal (red planet). This requires various teams handling a multitude of functions to work in complete collaboration and constantly align the vehicle to ensure that the vehicle does not go off course. Any kind of misalignment can have catastrophic results.

Passion – Alignment creates passion. Once every member of the team is aligned with the end goal of the product with clarity about what defines product success, they contribute in their unique way using their skills to make it big and successful.

A great leader's courage to fulfill his vision comes from passion, not position."

John Maxwell

Unfortunately, team members work in silos either unaware of the end goal to be achieved or are just not allowed to create impact outside their territory. There is no focused intent to leverage the team's creativity, skills or knowledge to drive decisions.

Time and again companies like Amazon and Google have shared instances where teams were able to come up with innovative solutions just by understanding a problem, doing some experimentation and adapting to feedback and these are the people who feel passionate about what they do. The intent of a scrum team is to create this combined passion for what is expected to be achieved.

Delight – The term delight is often associated with customers but it holds equal importance when it comes to the team we work with. The question one may ask "so how do we delight the team?". As humans we get a sense of delight from small gestures from people around us. These can include writing a note of gratitude for all they do for the team and the project, engaging in activities to familiarize with the ups and downs of their lives or by just acknowledging what they do as a member of the team.

When a team comes together to achieve a common purpose and hold each other accountable for the collective success, delight happens. Acts of support, trust, belief, respect, openness result in a overall delightful environment and experience.

There is no delight in owning anything unshared

Seneca the younger

Click **here** to read about an experiment conducted by Thalia Wheatley called impact design to evaluate a delightful experience.

Celebrate – A unique attribute of scrum teams is their ability to celebrate success and failure. The cause of a success or failure is never attributed an individual but the whole team.

The important aspect of celebration in this case is that the celebration should become part of the team culture. Celebrations should happen frequently, for the whole team and in a way such that it leaves a lasting impact of the team members.

"Each day offers a reason to celebrate. Find it and experience true bliss."

Amy Leigh Mercree

Conclusion: As organizations embrace the scrum team idea, the thought process needs to go beyond the need, skills and title of a role. Instead the focus needs to be towards creating an environment where unique skills are coming together to achieve a common goal in a inclusive environment where there is passion, alignment and celebrations and delight is not just for customer but for every member of the team.

***

To read this article online with embedded hypertext links, go here:
https://nirmaljeet.com/2017/08/31/why-the-idea-of-a-scrum-team-is-so-powerful/

About Nirmaljeet Malhotra

Nirmaljeet is a passionate agile, product, lean and leadership coach. He has been part of multiple small and large agile adoption and transformation journeys and has focused on continually improving his coaching skills to enable organizations succeed with change.

In his present role, Nirmal works for Improving in Dallas and a consultant enterprise coach with Intuit.

Nirmal holds a Master's degree in computer science and many certifications in agile, coaching and leadership. He likes to share his experiences and thoughts through his blog nirmaljeet.com. He is also a frequent speaker at conferences and meetups.

Nirmal can be reached at nirmaljeet.malhotra@gmail.com

20 Unagile Things to Avoid Saying and Some Better Alternatives

By Ian Mitchell

[**Note:** grayed text indicates hypertext links in original article.]

"See it all. See it fairly. Be truthful, be sensible and be careful with language"
- Henry Grunwald

In Scrum we care about the precise and considered use of language, since any obfuscation reduces transparency. When we try to implement Scrum, we can sometimes find that the pressure is on to change Scrum terms and their meaning, so that change may be "configured" or "customized" to fit the organization. Scrum terms of reference can become bent and twisted around those existing contours, and the way we even think about agile change can be tugged at and constrained by organizational gravity. The result of acquiescing to such pressure is that little change may actually happen, and there is surprise and disappointment amongst stakeholders when the expected benefits do not materialize.

We are nevertheless subject to those forces of organizational gravity, and no matter how rigorous or careful we try to be, we cannot entirely insulate ourselves from its effects. An important discipline we must therefore learn is to exert small corrections, early and often, before they build up and we face a crash. Here are twenty small things which you might be tempted to say or to silently agree with, and which are perhaps rather better to avoid.

1. Avoid describing a Sprint Backlog as a "commitment". It's a "plan" or "forecast" of work for meeting a Sprint Goal. Use those words instead. Remember that team members ought to commit to goals, not to forecasts.
2. Avoid language which suggests Story Points are "delivered", or in some way constitute value or otherwise proxy for value. The purpose of story pointing is to help a team forecast how much work it believes it can take on. In agile practice, value is only to be found in the delivered increment itself.
3. Avoid talking about an "ideal velocity" when making forecasts. Instead, talk about the ideal value which can be released in current and future

Sprints. Remember that an agile team does not consist of story point accountants. Speak of the work done in terms of **innovation accounting** instead.

4. Avoid talking about "Sprint Goals" when those supposed goals have not yet been planned and agreed by the team. If they are **tentative** Sprint Goals, call them that. During refinement, discuss how well they might align to features and Minimum Viable Products.
5. Avoid describing stages of work as "Sprints" unless they are time-boxed and produce an increment of functionality, however small it may be. "Special" sprints like "sprint zero", "integration sprint", "testing sprint" and so on are coded terms for stages or phases. If stages or phases are to be used, call them so honestly, and avoid devaluing agile terms of reference.
6. Avoid describing a Sprint Review as a "Show and Tell" or "Demo". A demonstration of work might very well form part of a Sprint Review. However, the essential purpose is to consider the work which has been done and which remains to be done, and to inspect and adapt the Product Backlog.
7. Avoid talking about a "Kanban" unless there is evidence of a closed economy of work. If there is merely evidence of a "to-do" list, call it that.
8. Avoid describing Acceptance Criteria as the "Definition of Done". They may represent a certain level of "Done" for certain Product Backlog items, but the Definition of Done, as an assertion of release quality, properly refers to the entire increment.
9. Avoid referring to a collection of people as a "team" unless there is evidence of their collaboration and teamwork. If those people are working in silos which are largely independent of each other, then there may instead be evidence of a "workgroup" engaged in craft production.
10. Avoid referring to an agile initiative in terms of its supporting tools. Achieving agile practice is not the same thing as "having Jira" or "using TFS" or indeed any other technology.
11. Avoid talking about "DevOps" as though it were distinct from agile practice and cultural change. If you are referring to technical practices such as automation or continuous integration and deployment, use those terms instead.
12. Avoid talking about "technical debt" when there is no plan to pay the accrued deficit back, or the liability incurred thus far is unmanaged and unknown. If they are in truth unquantified losses, call them that.
13. Avoid talking about a "Release Plan" if certain Sprints are not planned to result in a release at all. What you actually have is a plan for not releasing. In Scrum, each Sprint must yield an increment of value however small it may be. The decision to release or not to release ought to be made on a *Just in Time* basis. A true Release Plan should outline what is likely to be delivered, to whom and when...not *if* a release will happen.

14. Avoid talking about "bugs" or "defects" as if they are separate from other work which remains to be done. They must still be accounted for as work remaining, and planned and budgeted for. The urgency of the repair and the speed with which it is expedited does not obviate the need for this quality of transparency.
15. Avoid talking about "fixed scope" when a Product Backlog is subject to ongoing refinement, and distinct options might yet emerge. Instead, talk about each Sprint as the opportunity to deliver something of value from which useful things can be learned.
16. Avoid language such as "push to test" which suggests that anything other than a pull-driven flow of work is expected. Agile and lean practice is founded on *pull,* including the timely and efficient handling of work in response to clear demand signals.
17. Avoid referring to "distributed" teams. A team which is not co-located is a dislocated team. Call it that, and be transparent and open about the challenges and inefficiencies concerning teamwork which are likely to arise from such a model.
18. Avoid using the expression "being agile" as a euphemism for "being reactive" or doing work "faster and cheaper". An agile team exhibits full control over its work in progress and the work it chooses to take on. Any economies will be found in the team's ability to inspect and adapt, to evaluate outcomes empirically, and to reduce waste.
19. Avoid talking about "agile scaling" when the de-scaling of enterprise functions will be needed before even one team can achieve an agile way of working.
20. Avoid dehumanizing employees as "resources" or "work packages". If you contextualize people as inanimate objects, you will get less than the person has to offer. Employees are human beings with creative and innovative potential. Value them accordingly.

***

To read this article online with embedded hypertext links, go here:
https://www.scrum.org/resources/blog/20-unagile-things-avoid-saying-and-some-better-alternatives

About Ian Mitchell

Ian's experience in iterative and incremental delivery began with a PhD in object-oriented rapid prototyping in 1997. He then worked as a developer for several years where he witnessed the failure of many initiatives that were allegedly implementing Scrum. He realized that his true vocation was to help teams and organizations to better understand agile practice. Dr. Mitchell has a particular focus on evidence-based enterprise transformation and is the curator of a related site, agilepatterns.org. A Scrum advocate and accredited Professional Scrum Trainer, he is a veteran member of the scrum.org forums.

What Can You Do About Organizational Silence?

By Chris Murman

[**Note:** grayed text indicates hypertext links in original article.]

Regardless of your opinion of the president, I think many would agree that he speaks his mind on all topics. His tweets in the middle of the night **set policy.** The rally comments **set off protests.** Trump **speaks up.**

Only sometimes he speaks with silence.

When Puerto Rico was flooding, **he spoke of NFL players.** When white supremacists were demonstrating in Virginia, it took him several days to **respond only to take it back.** There are circumstances that even the most bombastic president ever speaks with silence.

Just like we do every day in our organizations.

Silence is **tacit agreement.** If you don't believe me, just **search on Twitter** for that phrase and you will see it strewn across many of our feeds. It's something we can imperatively get behind because it makes sense. We see something, but don't say something.

Mind you, I'm not discounting the ability of individuals to speak up with the moment requires it. Much of the history of social change in the US comes from the brave few with the courage to say "no more." I'm more referring to the collective level dynamics that plague office culture.

New York University researchers Elizabeth Morrison and Frances Milliken refer to this phenomenon as a culture of "organizational silence."

What are its origins?

In their paper **Organizational Silence: A Barrier to Change and Development in a Pluralistic World,** Morrison and Milliken show that although organizations may verbalize openness, most cultures send implicit and sometimes very explicit signals to employees that they should remain silent.

As with most organizational issues, it starts at the top. The writers state that a leadership group that positions itself apart from the workforce can create a barrier or air of superiority. Any cultural differences are magnified even more when this siloing of leadership occurs. It's not just about leadership, though. Environmental barriers such as a contingent workforce, external hiring of senior managers, and low-cost strategies can contribute to organizational silence.

The result will be poor implicit and explicit managerial practices as well as company policies that encourage silence.

Managers end up hiring people just like them. The workforce focuses on things like interdependence and job stability over innovation and welcoming change. When your top priority is to prove that you're all necessary and should stay exactly where you are, silence is the result. Thoughts and ideas only travel down, as opposed to both directions.

> *"It has been shown that when negative feedback comes from below rather than from above — from subordinates rather than bosses — it is seen as less accurate and legitimate, and as more threatening to one's power and credibility."*

This comes from the notion that a higher position equals higher respect. Better ideas must come from more elevated positions, so why would we challenge them? All it takes is one sly comment from a superior to make you think twice about speaking up the next time.

What are its effects?

> *"After my suggestions were ignored, the quality of my work was still there," an interviewee stated in the paper. "But I wasn't."*

Thanks to the research proving the validity of emotional intelligence in modern offices, we know feelings matter in the workplace. It would be irresponsible for a superior to ignore the feelings of the members of his or her team. And yet, when organizations create a culture of silence it disregards the feelings of employees.

Feeling disregarded leads to you offering fewer ideas.

If an individual manager doesn't value your ideas, and that person represents the company at large, then how trust the organization? You become an order taker from your boss and automate as much of your day as possible. Five o'clock on Friday becomes your ultimate goal.

Creating safe spaces for venting can have a short-term impact, according to the paper. It would only be short-lived, though. A harmful cognitive dissonance emerges when there is a stark difference between what employees can say in private as opposed to publicly. In a sense, organizational silence leads to a culture of harmful passivity.

Passivity leads to inaction in moments where it's most needed, and companies crumble underneath the pressure that is never released.

So what can we do?

Morrison and Milliken summarize that on the surface organizational silence can be a difficult culture to break. The destructive cycles that are outlined in the paper aren't easily observable, which make them difficult to prove to senior leadership. This means a change at the top is most likely necessary, and those types of sweeping changes are rare.

Even if it does come, the writers argue, it won't solely stop the culture of silence. New systems would need to be put in place to not only allow people to speak up but encourage it. This is why so many startups disrupt industries across the board globally. New companies don't have the excess baggage of existing structures that encourage silence.

Most of us don't have that luxury, though. While the paper has a somber tone to it, I wondered if it's possible to think positively. Surely there's something we can do starting today, right?

In the end, all we can do is look inward to our own teams or programs and try to start change there. We may not be able to change the overall structure of the entire organization, but we might be able to look at the few around us and decide to speak up to each other. Agile teams valuing transparency state that the only way we will improve our products and work lives is to say something.

By making our work visible, including our problems, we give voice to them. We prove they are a real thing and can rally around a possible solution. You can inspect and adapt your way out of organizational silence.

Will it change the organization overnight, or even over a long period of time? It might prove difficult. When others in the organization see what you and your teammates can accomplish, though, they will ask themselves what you are doing that they aren't.

Those are the seeds of true change.

⁂⁂⁂

To read this article online with embedded hypertext links, go here:
https://chrismurman.com/2017/09/26/what-can-you-do-about-organizational-silence/

About Chris Murman

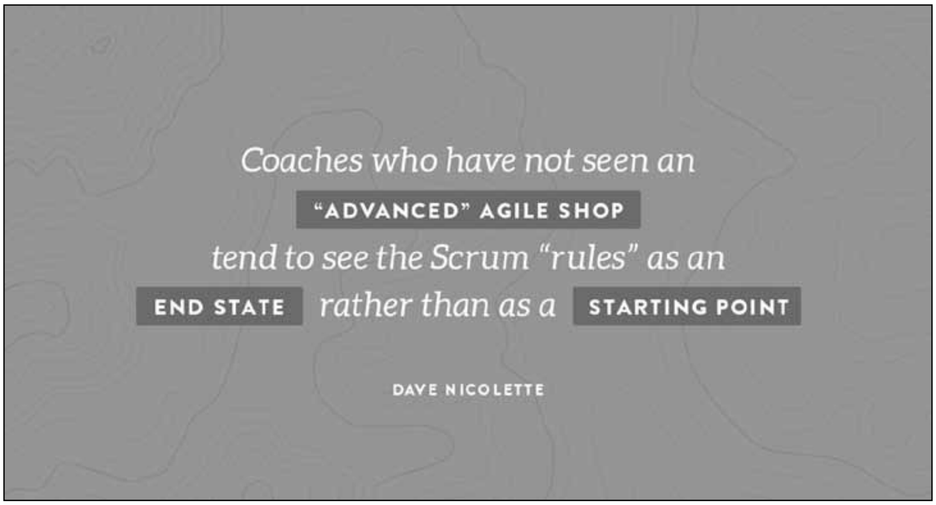
Chris' first job out of college was the weekend sports anchor at an NBC affiliate. If he had only known what was in store!

Interestingly enough, he still loves telling the stories of others every day. Each interaction is an opportunity to learn what made you unique, and understand where you came from. If we got to know each other more on a personal level, it would make the tough conversations easier to have.

Currently, he is a consultant for Solutions IQ focusing time on coaching, facilitating, training and mentoring organizations that want to change the way they work. In addition, he is a board member for the Agile Uprising Coalition, and writes about his work at chrismurman.com.

Zombie Scrum

By Dave Nicolette



Coaches who have not seen an "ADVANCED" AGILE SHOP tend to see the Scrum "rules" as an END STATE rather than as a STARTING POINT

DAVE NICOLETTE

A year ago, your organization adopted the Scrum framework. Scrum helped you break down functional silos, improve communication with stakeholders, increase collaboration on your team and across teams, and facilitate cross-disciplinary skill development among staff members.

It was exciting at first. Everyone was engaged and everyone was enthusiastic.

Stable teams were established, and work was allocated to teams as deliverables were completed. People began to spend their days in team spaces designed as collaborative work centers, rather than sequestered in cubicles like so many hermits.

In every Sprint, teams built trust with stakeholders and learned more about the business domain and the technologies in play than they had previously imagined possible. Delivery effectiveness, stakeholder satisfaction, and team morale improved over the next six months.

Laissez le bon temps rouler

Life was good.

And the clock ticked on. And on.

And on.

After learning all there was to know about the types of work their stable teams carried out, people began to wonder when or how they might get the chance to learn something new. Life took the shape of an endless series of User Stories, mostly comprising the same sorts of changes to the same parts of the same codebase over and over again.

And the clock ticked on.

What about collective ownership? Self-organization? Retrospectives?

Yeah, sure. Collective ownership of the same sorts of changes to the same parts of the same codebase over and over again. Self-organization regarding exactly how to carry out the same sorts of changes to the same parts of the codebase over and over again. Retrospectives in which the same issues are raised time and again, and all the remedies lie beyond the team's purview. As far as anyone on a delivery team can see, the root cause of the issues is Scrum itself, if not *life* itself.

Everyone is disengaged and everyone is apathetic.

Like zombies.

Management and business stakeholders don't really see the problem. After all, delivery is steady and predictable, and much more efficient than it was before Scrum was introduced.

Portfolio and Program teams (or their equivalent by any other name) don't really see the problem. After all, the delivery teams are accepting prioritized User Stories and delivering them on a steady basis.

If there were issues, the teams would report them, wouldn't they? That's the model, isn't it? The only reason a person would hesitate to report an issue is if they didn't believe any good would come of it. Scrum is good by definition, so that can't be the case.

But there's trouble in the trenches. If the technical staff can't change their organization, they'll change their organization.

Send in the coaches

It's a good thing you have agile coaches to help you! What do *they* have to say about the situation?

They say things like this:

You're a self-organizing team! You can figure it out!

- Bring up the issue at the next retrospective. Don't forget to create an action item!

- If you can't solve the problem at the team level, escalate!
- Agile people are *passionate* about their work. Look within and find your passion!
- Stare at this inspirational poster about Scrum until you feel better.

Well, okay, that last one was a little snarky. But only a little. The sad thing is it isn't too far from the truth. It's more-or-less the Scrum equivalent of the traditional mantra, "The beatings will continue until morale improves."

Certification overload

Informally I've been told more than 400,000 people have gone through a formal Certified ScrumMaster (CSM) course offered by the Scrum Alliance. On the Scrum.org website, that organization claims over 100,000 people hold Professional Scrum certifications. That's a lot of certifications.

Why, then, are there so few agile or Scrum coaches who can take it beyond the novice level? It may be a case of "too much of a good thing."

People who are interested in Scrum but not too sure what it's all about tend to look for certifications as a form of assurance that they're listening to the right people. ScrumAlliance and Scrum.org are happy to offer certification programs. Many agile coaches hold the basic certifications from one or both these organizations, and many have experience in getting teams and organizations started with rudimentary Scrum practices. Their career path consists of getting one organization started, then moving on to another organization to get *them* started, then moving on...well, you see where this is going. Most agile and Scrum coaches have never experienced a situation beyond the initial stages of *getting started* with agile in general or Scrum in particular. They have not seen the problems that normally occur after a long series of Sprints.

Is something wrong with the certification process? Well, not really. You have to start *somewhere*. But the base-level certification, Certified ScrumMaster (CSM) is surprisingly easy to obtain. A colleague told us he had his child take the exam, and he passed on the first try. The child had never taken a CSM course, and was only as interested in Things Agile as any other child would be (that is, not very).

The ease with which a person can obtain a CSM credential has led many individuals to take the CSM class who have absolutely no experience in IT. They've never written code, tested software, analyzed business problems, managed a project, administered servers or networks, worked with database systems, or worked in operations or production support. When they are coaching a delivery team, they have no way to relate to the practical challenges the team faces. Credit where due: They're pretty good at sticking things on the walls.

Zombie Scrum is a perfectly normal and predictable stage in an organization's development. Most coaches don't know how to advise zombie Scrum teams because they've never stayed with the program long enough to see the problem manifest.

And the Scrum literature doesn't talk about zombie teams. <understatement>It wouldn't be a very strong selling point.</understatement>

All problems are management problems

Lest you point the finger of blame at underqualified and overcertified Scrum coaches and ScrumMasters, consider the wisdom of the old saying, "All problems are management problems."

Scrum introduces a few unfamiliar concepts and roles. One of them is the role of ScrumMaster. Managers who have a traditional background often have difficulty with the concept of a ScrumMaster. When we simultaneously introduce the ScrumMaster role and reduce the importance of the traditional Project Manager role, it's only natural for people to fill in the gaps based on their own experience. They re-title the Project Manager as ScrumMaster.

This mangled ScrumMaster role has two responsibilities, and the two are in direct conflict. One is the responsibility to help the team use Scrum effectively and to support them in their continual improvement efforts. The other is direct responsibility for delivery. The freshly-minted ScrumMasters inherit the latter responsibility from the deprecated Project Manager role, which management still believes is necessary.

Even in the best case, it's hard to serve two masters. When you have the dual responsibility of delivery and team coaching, the delivery responsibility always prevails. Why? Because it's often necessary to present challenges to the team to create learning opportunities for them. If you have responsibility for delivery, you can't do that. You can't help the team improve. You, yourself, are measured on steady delivery. You have to keep the team running on the treadmill non-stop. You have no choice.

That isn't a coaching problem, it's a management problem. Your role is improperly defined.

Can zombies be awakened?

One of the top Scrum trainers and consultants in Europe, Joseph Pelrine, describes a model of team performance he calls the Cooking Model. When you're cooking, you don't want the temperature to drop so low that the food congeals into a flavorless mass. At the same time, you don't want the temperature to rise to high that the food burns. There's an optimal temperature for cooking. Similarly, there's an optimal "temperature" for a delivery team. Too much stress, and the team burns out. Too much boredom, and the team members turn into zombies.

Pelrine advises coaches to shake things up when they see the zombie team phenomenon starting to occur. Teams may miss delivery targets when this happens, but it's all to the good. We want the teams to deliver predictably over the long term. Sometimes, that means allowing short-term variation in delivery performance in the interest of improvement.

Another factor to keep in mind is that most of the guidelines and practices defined in Scrum are meant to be a *starting point* for teams. As teams progress in agile and lean thinking, they will require progressively less process-management overhead. Coaches who have not seen an "advanced" agile shop tend to see the Scrum "rules" as an end state rather than as a starting point. When teams have outgrown the need for the rules, and they are required to continue following the rules anyway, they check out. Coaches who haven't seen this won't know how to help teams overcome the perfectly normal Zombie Scrum stage.

Imagine, if you will

These "starting rules" include some of the holiest-of-the-holy dogma in beginner-level agile coachery. Team collocation. Stable teams. Story sizing. Others.

All these "rules" have a purpose. Remember that Scrum was created in the early 1990s based on work published in the mid-1980s. Recall the state of the corporate IT world at that time. Matrixed organizations. Individuals assigned to multiple projects concurrently. Functional silos. Isolated, solo work. Indirect communication methods.

And the results: Very long lead times. High defect levels. Low customer satisfaction.

Scrum addressed all those issues in a practical way. By following the Scrum "rules," 1980s-era organizations could break those nasty old habits and start to achieve better outcomes.

Do we really need *collocated teams?* Well, it's better than having individuals scattered all over the place, not collaborating, and communicating only indirectly through "tools" and such. But the goal isn't merely to sit in the same room together. The goal is *collaboration*. If people are interested in collaboration, they will find a way. We certainly have technologies today that support remote collaborative work. There's no longer a need to force everyone to sit together physically. That was never really the point, anyway. It was only a means to an end.

Do we really need *stable teams?* Back in the day when people were measured on individual performance, there was little sense of team membership or shared ownership of results. You had no incentive to collaborate. In fact, you had every incentive to make everyone else look bad, so you wouldn't be riffed in the next stack-ranked layoff sweep. The cure? Stable teams. People got used to working with each other and began to feel like a real team. What happens when people feel that way *throughout the organization?* Well, maybe there's no problem letting people work on different things. They're accustomed to smooth collaboration and transparency. There's no more "storming and norming." Collaborative work is natural for them. So, if someone wants to work on something other than the same sorts of changes to the same parts of the same codebase, it's no problem.

Is there a point to this?

Well, yes. The point is the Zombie Scrum phenomenon *may* be a signal that the organization is ready to move beyond beginner-level practices, shed some of the novice-level process-management overhead, and mindfully bend or break some of the "rules." Shake things up. Restore people's enthusiasm. Grow.

Just food for thought. Meanwhile, all this typing has made me feel a bit peckish. Please pass the brains.

⁂

To read this article online with embedded hypertext links, go here:
https://www.leadingagile.com/2017/06/zombie-scrum/

About Dave Nicolette

Dave has been involved in the software industry since 1977. He has worked in a variety of roles including programmer, analyst, tester, architect, DBA, system administrator, project manager, trainer, coach, consultant, and (like everyone else) ScrumMaster. He got involved with the agile movement in 2002 and never looked back, although his pragmatism casts him in the role of iconoclast more often than that of disciple. Currently, he helps clients understand and adopt sound technical practices and processes for effective software development and delivery. He pontificates on Twitter as @davenicolette, on the LeadingAgile blog at https://www.leadingagile.com/blog/, and in a book or article from time to time. He can sometimes be seen lurking around technical conferences looking for free food, and occasionally giving a talk.

4 Ways to Coach with the Scrum Values

By Stephanie Ockerman

[**Note:** grayed text indicates hypertext links in original article.]

Scrum is a framework that thrives on self-organizing teams. It gives you boundaries (e.g., time-box of a Sprint), clear accountabilities (e.g., Product Owner optimizes value), and goals (e.g., "Done" Increment). But it doesn't tell you exactly how to do the work. Every Scrum Team needs to figure out the strategy and tactics that work for their context in this moment.

But the Scrum Guide does give us the secret to maximizing the benefits of Scrum — the Scrum values.

We can use the Scrum values as a compass.

The Scrum values help guide us in how we are working as a collaborative team and how we are enabling the benefits of empiricism.

Note: *If you want to learn more about the Scrum values and read examples of how to use them in Scrum, check out these blog posts:* ***Focus, Openness, Courage, Commitment, Respect.***

As a Scrum Master, you demonstrate **servant leadership** by **coaching** with the Scrum values.

4 Ways to Coach with the Scrum Values

1. **Establish what the Scrum values mean to us as individuals and as a team.**

As human beings, we all have core values. Living our values help us feel in alignment and helps us show up as our most authentic selves. When we talk about teams having values, this doesn't just magically happen. Teams need to form an identity in order to be

Photo by Climate KIC on Unsplash

effective. One piece of identity is **understanding values.**

Team members need to work through what their values actually mean to them because individuals each have different interpretations of values, and different teams will have different interpretations of values.

Here are a few questions to use with individuals or teams to coach with the Scrum values:

- What is important about [Scrum value]?
- What does it look like to honor [Scrum value] in our daily work?
- What does it look like to ignore [Scrum value] in our daily work?
- How does it feel to be in alignment with [Scrum value]?

If we discover conflicts between individual values and the Scrum values, we need to explore that further. Negotiation may be necessary to enable teams to be effective and individuals to feel in alignment with what matters to them.

2. Use the Scrum values to help guide decision-making.

Once teams have established what the Scrum values mean to them, the values can be used as a tool to help guide decision-making. Scrum Masters can recognize when Scrum Teams or individual team members may feel stuck when facing a challenging decision. It could be about what to build, how to build something, processes, tools, relationships and interactions… anything.

Here are examples of questions to help you coach with the Scrum values:

- What would [Scrum value] tell us about this decision?
- Which Scrum value feels most important for this decision?

3. Observe and discuss outcomes and behaviors and refine what the Scrum values mean to us.

The Scrum values discussion is not a one-time thing. We will continuously encounter new situations that require more discovery of how we interpret and use the Scrum values. We will also fail to honor our team commitments regarding the Scrum values because, you know, we are human. So we need to be inspecting and adapting on how we are living the Scrum values. Sprint Retrospectives are a great opportunity for this.

Here are some reflective questions to coach with the Scrum values:

- How did we honor [Scrum value] this Sprint?
- Which Scrum values helped us achieve [outcome]?
- Which values do we want to honor more?
- What will be possible when we honor this value?
- In what situations does it feel most difficult to honor [Scrum value]?

4. Identify actions for improvement.

A key part of coaching is moving to commitment and accountability. This is what helps individuals and teams grow – taking action after we have enough learning and discovery.

Here are examples of commitment-focused coaching questions to use when coaching individuals and teams:

- When we/ you honor [Scrum value], what are we saying yes to?
- When we/you honor [Scrum value], what are we saying no to?
- Who do[we/ you need to be in order to do [actionable commitment]?

Remember, the Scrum Master does not own all of the actions for improvement. The individuals and the team own their coaching commitments. (Read: **Stop Being So Helpful**)

Want more coaching tips? Check out **6 Coaching Tips for Scrum Masters.**

***

To read this article online with embedded hypertext links, go here:
https://www.agilesocks.com/4-ways-to-coach-with-the-scrum-values/

About Stephanie Ockerman

Stephanie is a Scrum.org certified Professional Scrum Trainer (PST), Scrum Master, and Co-Active Coach. She is a Curriculum Steward for Scrum.org's Professional Scrum Master Program, working with the international trainer community and Scrum co-creator Ken Schwaber to carry forward the vision. She has over ten years experience delivering IT solutions and facilitating training for professionals.

Stephanie is a results-driven, compassionate person who enjoys working with people who believe in the art of the possible. She started her agile training and coaching business Agile Socks to fulfill this purpose. Stephanie combines training and coaching to help people level-up their skills and amplify their impact. She feels lucky to combine her passions for teaching, servant leadership, and experiencing the world by teaching and speaking internationally. You can read Stephanie's musings on Scrum, agility, and life on her blog at AgileSocks.com.

Feeling Safe?

By Tim Ottinger

I finally got around to watching *Frozen.*

I don't have any small children of my own and wasn't really interested in it for my own viewing pleasure, so it took a long time. I didn't know the songs, didn't know the characters, didn't know the storyline.

We were watching a friends' child last weekend, and the child really wanted to see *Frozen,* so we did.

Overall, it's cute and has nice jokes and beautiful animation. I can tell they spent a lot of money on the soundtrack. I probably won't watch it again, being well outside of the target audience.

There was one poignant moment that stood out to me, though. Anna, the red-headed sister of magical-powered Elsa, came to retrieve her (very dangerous) sister and bring her back to their town.

Elsa warned Anna that she was a danger to everyone. Anna said:

You don't have to protect me; I'm not afraid.

Boom. That line.

Feeling v. Being

It dawned on me that Anna thinks that *feeling unsafe* is the thing; actually *being unsafe* doesn't occur to her.

She's focused on the feeling instead of the reality.

When we say "make safety a prerequisite", people think we mean "feeling safe and unthreatened" which is not what I mean.

- Fragile people can remain fragile in a confrontation-free space, but this doesn't make them safer.
- Hiding problems can keep people from feeling afraid, but transparency gives them the ability to solve problems.

- Brutish people can “feel safer” when they’re allowed to run roughshod over others, but this does not make the group safer.

There is more to this than feelings.

Likewise, when we say “make people awesome,” some people think we mean “make people feel really great” instead of “give people the ability to do great things.” I mean the latter. If I meant “make people feel great” I would say “make people feel awesome” and would not bother with actually creating any enablement. Feelings matter, but there is more to this than feelings.

What Timing!

As I was pondering Anna’s “not needing protection” Elsa deals Anna a mortal blow which begins to freeze her heart, and in time may well kill her.

So much for not needing protection.

Eventually, Anna is frozen solid, essentially dead. This being a Disney movie, she’s restored to normal state and all is well at the end.

I wish it were the same for people who “feel safe” weaving through traffic on a motorcycle at 120mph with no helmet, or those who “feel safe” working with homemade explosives or modified firearms.

What Are We Really Doing?

The ideas of “make people awesome” and “make safety a prerequisite” are too important to me to have them confused and conflated with imparting (potentially deceptive) feelings of awesomeness and safety.

If we aren’t enabling greater accomplishment and reducing potential damage, then what are we doing? Just playing with people’s feelings?

⁂

To read this article online with embedded hypertext links, go here:
https://agileotter.blogspot.com/2017/12/feeling-safe.html

About Tim Ottinger

The biography and picture of this author/member of the Nominating Committee was not received in time for publication."

Myth 8: The Scrum Master is a Junior Agile Coach

Barry Overeem

[**Note:** grayed text indicates hypertext links in original article.]

Scrum is intended as ***a simple, yet sufficient framework for complex product delivery.*** *Scrum is not a one-size-fits-all solution, a silver bullet or a complete methodology. Instead, Scrum provides the minimal boundaries within which teams can self-organize to solve a complex problem using an empirical approach. This simplicity is its greatest strength, but also the source of many misinterpretations and myths surrounding Scrum. In this series of posts we — your 'mythbusters'* ***Christiaan Verwijs*** *&* ***Barry Overeem*** *— will address the most common myths and misunderstandings. PS: The great visuals are by* ***Thea Schukken.*** *Check out the previous episodes here (1, 2, 3, 4, 5, 6, and 7).*

Myth 8: The Scrum Master is a Junior Agile Coach

Are you a Scrum Master and ready for the next step as Agile Coach? Do you need an Agile Coach to help facilitate organizational change while Scrum Masters focus on the Scrum Teams? Do you have experience as a Scrum Master and want to become Agile Coach with a 3-day course? Ever considered changing your job title to 'Agile Coach' because it nets you a higher salary?

These statements exemplify the myth that we intend to bust today; the idea that the Scrum Master is a Junior Agile Coach. Or more simply; that the Agile Coach tends to larger organisational issues while Scrum Masters focus on Scrum Teams. In a way, busting this myth has been our mission over the past years. And one that we'll continue to pursue, considering just how tenacious it is. We've written several articles, spoken at seminars, provided trainings and facilitated workshops; all related to explaining the purpose of the Scrum Master. In this blog post we'll share our view on this topic, and why this is a myth that requires very much to be busted.

This myth concerns us for a number of reasons:

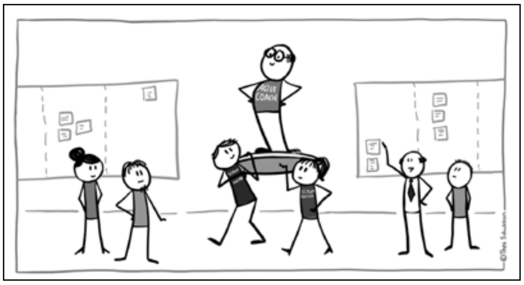
- It is based on a very poor and **incomplete understanding** of what it is that a Scrum Master actually does and should do according to the Scrum Framework;

- It positions the Agile Coach as being higher in a traditional hierarchical structure. Especially within organisations that are used to **'vertical growth paths'.** The Scrum Master as the junior, Agile Coach as the medior and the Enterprise Coach as the senior;
- Consultancy firms and training agencies encourage this way of thinking because it's easy to match with their increasing hourly rates and expensive training programs. **Notice the contradiction** with the services these organisations provide: advising clients to think in 'horizontal structures' that promote the self-organizing capabilities of the teams, yet promote a 'vertical structure' because it works well from a commercial- and marketing-perspective;

This myth leads to artificial boundaries between what Scrum Masters and Agile Coaches do. The Scrum Master is only "allowed" to act on team level. Therefore creating the necessary Scrum-friendly culture is far more difficult, causing the change for a successful Scrum adoption decrease. The Agile Coach is expected to "implement" the necessary organizational changes, but fails because of limited experiences "from the trenches" and not knowing how to deal with "outside in" change management.

Busting the Myth

Busting today's myth is actually remarkably easy, and requires only a simple reading of the **Scrum Guide.** As has been the case with every myth we've addressed so far. The Scrum Guide offers a clear description of the services that a **Scrum Master provides to the Development Team, the Product Owner and the *entire* organization.** This includes coaching the Development Team in self-organization and cross-functionality, helping the Product Owner find techniques for effective Product Backlog management and supporting the organization in delivering high-value products through the empirical process established through Scrum. To make this happen, the Scrum Master works with other Scrum Masters, Product Owners and people within the organization.



The 8 Stances of a Scrum Master

Another useful perspective on the role of the Scrum Master is offered in the white paper **"The 8 stances of a Scrum Master".** It captures the various responsibilities of the Scrum Master in eight stances that are closely linked to the Scrum Guide. The Scrum Master is ....

- An **Impediment Remover** that helps resolve issues that are blocking the team's progress, taking into account the self-organising capabilities of the Development Team;

- A **Facilitator** that sets the stage and provides clear boundaries in which the team can collaborate. This includes facilitation of the Scrum events to ensure they'll achieve the desired outcome and - most importantly - that the empirical process is optimized;
- A **Coach** that helps individuals and groups to continuously improve in how they deliver valuable outcomes as a team or as an organization;
- A **Teacher** that ensures that Scrum and relevant techniques are well-understood and enacted;
- A **Servant Leader** that creates environments where teams can work effectively with stakeholders to create valuable outcomes;
- A **Manager** that is responsible for managing (true) impediments, eliminating waste, managing the process, managing the team's health, managing the boundaries of self-organisation, and managing the culture;
- A **Change Agent** that helps to enable a culture in which Scrum Teams can flourish - on every level of the organization;
- A **Mentor** that transfers agile knowledge and experience to the team.

Scrum Masters should be aware of these stances and its diversity, knowing when and how to apply them, depending on situation and context. All with the purpose of helping people understand the spirit of Scrum.

Dealing with "senior" challenges

"A good Scrum Master helps a Scrum Team survive in an organisation's culture. A great Scrum Master helps change the culture so Scrum Teams can thrive."

- Geoff Watts

Both the Scrum Guide and the '8 Stances of the Scrum Master' inform us about the challenges of a Scrum Master:

- How to help people transition from plan-based approaches towards an empirical process that does more justice to the complexity of the work they do?
- How to facilitate transparency, inspection and adaptation in a traditional 'closed' organisation?
- How to coach organisations in truly collaborating with their Scrum Teams?
- How to manage the boundaries of self-organisation in control-driven organisations?
- How to offer a "safe to fail & learn" environment where experimentation?
- How to promote a culture where Scrum Teams can thrive?

Being a Scrum Master means dealing with these difficult challenges and influence the organisation's culture in such a way that…

- Team success is valued over individual success;

- Continuous improvement and experimentation are promoted;
- "Agile contracts" are encouraged;
- Stable team composition is supported;
- Behaviour is rewards, not individual achievements;

It's up to the Scrum Master to help create this Scrum-friendly culture. Thankfully, the Scrum Master is in a perfect position to do this, because (s)he can enable change from the inside out.

"The Scrum Master enables change from the inside out."

Being part of a Scrum Team, the Scrum Master knows exactly what needs to be changed and why this change is necessary. They help teams uncover the impediments that are holding them back and the other ways by which the organization can deliver (even) more value with Scrum. This puts them in an excellent position to work with **HR-departments** to find practices that are better aligned with Scrum. Or to help a **Sales-departments** move from 'fixed-price / fixed-scope'-contracts to contracts that are more Agile-friendly. Or to increase collaboration between Scrum Teams and **stakeholders**. Working with the **other Scrum Masters,** they ignite the necessary organisational changes by influencing the system from the inside out. From the perspective of the Scrum Team, the Scrum Master truly is a 'Change Facilitator'.

"The chances of successful Scrum adoption will increase drastically when you consider your Scrum Master as the true "inside out" change facilitators!"

When organizations choose to implement an empirical process primarily through Scrum, there should be no need for Agile Coaches. Instead, Scrum Masters should be enabled and supported to promote the empirical process on all levels of the organisation. If they can, and if they do, no other roles are necessary to help organizations generate valuable outcomes with Scrum.

"When organizations choose to work with Scrum, there should be no need for Agile Coaches."

Should we fire all Agile Coaches?

No, you shouldn't. By busting the myth that Scrum Masters are Junior Agile Coaches, we do not mean to say that Agile Coaches are of no value. We do mean to say that the need for Agile Coaches diminishes greatly when Scrum Masters are allowed to perform their intended role. We also mean to say that the hierarchical differences that we often see between Agile Coaches and Scrum Masters is based on a (very) poor understanding of Scrum.

Where Scrum Masters use an "inside out" approach, Agile Coaches use an "outside in" approach. Obviously we prefer the "inside out" approach to drive organisational change. But both can add value to the organisation from an organisational change point of view. They only have a different perspective on how to create a Scrum-friendly environment (if that's the goal of the Agile Coach).

Using an “outside in” approach can definitely work, but it’s incredibly difficult. It’s our experience that many (external) Agile Coaches offer little value in this regard. They are powerless to affect change and have a very superficial understanding of what goes on inside the Scrum Teams (where the value is being generated). They are not part of the team, lack the necessary support from management and don’t have the kind of extensive experience that is needed to drive change from “the outside in”. Furthermore, many Agile Coaches barely even have experience with Scrum or as a Scrum Master. Yet coaching Scrum Masters is frequently a part of their daily work.

“The reality is that most Agile Coaches are junior Scrum Masters.”

So our advice for organisations is:

- **Focus on enabling Scrum Masters** to facilitate change from “the inside out”. Support the Scrum Masters in creating great teams that build awesome products. Help them build the experience and the toolkit to do this, together.
- **Get rid of ‘Seagull Coaches’** that fly in, make a lot of noise, crap all over the place and fly on to a next customer, leaving a big mess behind;
- If you really want to hire an Agile Coach in addition to the Scrum Masters already present within the organization, make sure that they have **real, proven experience in affecting change “outside-in”.** Make sure they focus their efforts on helping the teams and the Scrum Masters drive change themselves. Don’t create the artificial distinction between “change on the management level” (by Agile Coaches) and “change on the team level” (by Scrum Masters);

What if we use Kanban/XP/DevOps?

Scrum is just one framework to improve organisational agility and to create engaging workplaces where people work with stakeholders to build awesome products. As Geoff Watts describes: “Scrum aims to harness the power of self-organising, autonomous, engaged teams who take responsibility for delivery and collaborate directly with their customers.”

Scrum is not a goal in itself. No matter what kind of framework or methodology you choose, it will involve organizational change to some degree. The people that are in the best position to effect this change are part of the teams that are doing the work. They may have titles like Scrum Master, Kanban God, XP Dude, DevOps Guru or no title at all: we don’t really care.

“Organisational change should be driven from the inside-out by people that are truly part of the teams.”

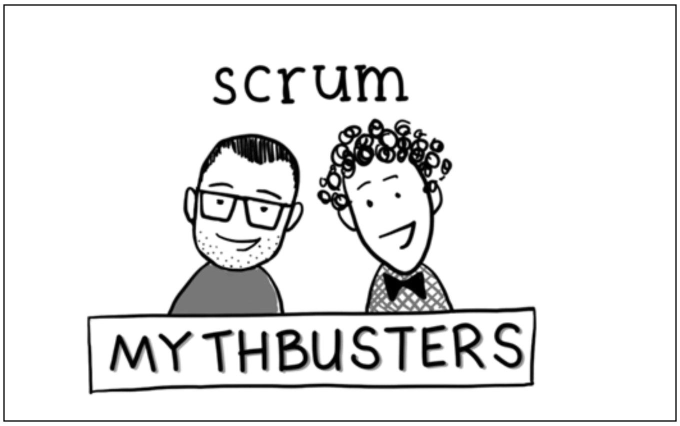
Closing

In this blog post we’ve busted the myth that “The Scrum Master is a junior Agile Coach”. Effective change is driven from “the inside-out”. The Scrum Master — being part of the Scrum Team — is in a better position to facilitate this change than an (external) Agile Coach. This is also how the Scrum Guide intended the role of the Scrum Master.

When organizations choose to implement an empirical process primarily through Scrum, there should be almost no need for Agile Coaches. Instead, Scrum Masters should be enabled and supported to promote the empirical process on all levels of the organisation. If they can, and if they do, no other roles are necessary to help organizations generate valuable outcomes through Scrum.

What do you think about this myth? Do you agree? What are your lessons learned?

*Want to separate Scrum from the myths? Join our **Professional Scrum Master** or Scrum Master Advanced courses (in Dutch or English). We guarantee a unique, eye-opening experience that is 100% free of PowerPoint, highly interactive and serious-but-fun. Check out our public courses (Dutch) or contact us for in-house or English courses. Check out the previous episodes here (**1, 2, 3, 4, 5, 6,** and **7**).*



***

To read this article online with embedded hypertext links, go here:
https://www.scrum.org/resources/blog/myth-8-scrum-master-junior-agile-coach?utm_source=newsletter&utm_medium=email

About Barry Overeem

As the co-founder of The Liberators, **Barry** liberates individuals, teams, and organisations from outdated modes of working. Bringing in fresh energy and creativity, he creates space for everyone to be involved in shaping the future and make a positive impact on their organisation. His mission is to unleash organisational superpowers by assisting the discovery of local solutions.

As a Professional Scrum Trainer and Scrum Master curriculum steward at Scrum.org Barry shares his insights and knowledge by providing Scrum training, facilitating workshops, speaking at conferences and writing blog posts. For fresh reading, visit the blog: https://medium.com/the-liberators.

Change is more like adding milk to coffee

By Niels Pflaeging

[**Note:** grayed text indicates hypertext links in original article.]

Change is not a journey. Never has been. *Trouble is: Change agents around the world have been imagining change as projects, programs, planned exercises to be "kicked off" and "implemented". We have interpreted change as difficult ventures, endlessly long hikes, and exhaustive trips. No more: Here are 5 key insights into the* ***true nature of change,*** *and into* **how to create profound, transformational** ***change,*** *effortlessly and fast. Sounds impossible? Then check out these concepts for a more constructive and robust alternative to change management, or* ***planned change,*** *as you know it.*

Insight 1. Change is not a journey — instead, it is *constant flipping*

The most widely used metaphors of change are related to that of a journey from the current state (often labeled 'status quo') to the desired state (a.k.a. 'vision'). The desired state, in this metaphor, is seen as a place out there in the future. Or as a north star - never quite to be reached. We tend to believe change-as-a-journey has to be long and arduous. That it is hard and dangerous. Consequently, armed with delusional maps, project plans, or blueprints, we embark on what we imagine will be a long and difficult journey. We start to foresee all sorts of obstacles - that don't actually exist, as we will see later in this article. But we find ourselves believing the milestones we invented are real, and get anxious when they don't appear on the horizon.

This approach misrepresents change as a "controllable process" composed of a sequence of discrete stages, phases or steps; and it deludes us into thinking we have to make a map for getting from the current state of affairs to the desired state. So this approach also trivializes change. We call this approach *Planned Change.* This is what we commonly think change management is all about: planning and controlling the change journey. The journey metaphor tricks us into ignoring the possibility that the desired change might be accomplished quickly, with little effort, right now, with existing resources and with minimal disruption. The metaphor itself makes change hard.

"Profound transformation never takes more than 2 years - independent if it´s about an organization with 20 people, or 200.000."

Now, spill a tiny bit of milk into coffee, and with this tiny nudge a new pattern is instantly being created. It's altogether different from the original one, pure coffee, and the change is permanent. there is no way of returning to the first pattern. This is much more similar to what change actually is than calling change a journey.

"Change is like adding milk to coffee."

This is a more helpful metaphor than the widespread notion of seeing change as a "journey from here to there". It means to see change as a something of a flip from Now (the current state) to New (the desired state). What is important: Both Now and New are in the present, not in the future. The New can be produced right here, right now. Profound change, different than problem solving, requires a sequence of flips. Or *many* flips.

"Profound change means sequenced flipping the system from Now to New - right here, right now. A thousand times or more."

Insight 2. There is no such thing as Resistance to Change — *only smart response to dumb method*

The man who invented *Resistance in Change* is Kurt Lewin, one of my **heroes.** Lewin, the brilliant founder of social psychology and of organizational change as such, introduced the term resistance as a systems concept: as a force affecting managers and employees equally. Unfortunately, only the terminology, but not the context, was popularized. We now cast resistance as a psychological, individualized issue, personalizing it as "employees versus managers".

In this mental model, it is always the others. Employees "resist", top management "isn´t committed". We judge others saying things like: "They have an interest in preserving the status quo." The *They* is very important, of course. The resistance assumption is implicitly arrogant. As long as we accept this mental model, it confuses our understanding of change dynamics, perpetuates the status quo and command-and-control organization. It´s better to let go of the term and embrace more helpful mental models for change.

So let´s give it a try:

"People don't resist change."

Can you say that to yourself, in your head? Now that is a start. But what is behind the behavior, then, that we are observing all the time, in change efforts, if it is *not* resistance to change? Take a step back and you will see that people act consciously and intelligently (overall), to other things than the change itself. They may resist loss of status and power — which is quite intelligent. They may resist injustice, stupidity and being changed. Which is also intelligent. The change may also cause need for learning that is not properly addressed. And these are the things that we have to deal with in change: power structures, status, injustice, consequence, our own stupidity, top-down command-and-control, and learning.

"The more resistance to change you observe, the more likely it is that your methods suck."

Instead of watching out for the possibility of resistance, we should watch out for common mistakes in implementing change and deal with the perfectly natural reactions to (our) poor interventions.

Let me be clear: The notion that people resist change is not held up by social sciences. It is actually completely opposed to our scientific knowledge about human capability to change (Alan Deutschman wrote a wonderful, summarizing book about this). But It is a fairy-tale that people resist change. There are symptoms of struggle with adaption and the new that should not be confused with resistance to the change itself. Once you start with kind of projection, the trouble really starts. We generally tend to have a hard time imagining future possibilities, though. This is why any change effort will have to deal with the need for imaginization, or visioning.

Insight 3. The problem is *in the system* — almost always

If resistance does not come from people, then where does it reside? Resistance is much more likely to be found elsewhere. Edwards W. Deming said: "94% of the problems in business are system-driven and only 6% are people-driven." Which means: If the problem is in the system, almost always, then change should mostly be about working the system.

Removing obstacles in the system to promote profound change is clearly easier than introducing entirely new features, rituals or memes within a system. This is what makes organizational hygiene such a compelling idea. But whether you are removing something, or introducing something new while flipping from Now to New: Making changes effectively in organizations requires specific, targeted action - not blaming. Which means: If the anticipated change will result in the loss of status by some employees, then we must develop strategies for dealing with the loss of status. Likewise, if the change will result in the loss of jobs, that issue must be dealt with. If the change will result in the need for learning, then let´s take care of that. If the change will come at a cost, then there should be space for emotions and mourning. Labeling these difficult, real-life problems as resistance to change only impedes the change effort. Resistance then becomes a self-fulfilling prophecy. Put differently:

"Change done well does not produce losers. Only consequences."

Power interests are also very real and often ignored by change „agents". And they shouldn´t. John Kotter, another one of my heroes, stated that individual resistance out of self-interest exists, but that it is "rare". More often, he said, the obstacle is in the organization's structure or in a *"performance appraisal system [that] makes people choose between the new vision and their own self-interest".* In other words:

"What we interpret as resistance to change is an intelligent response to inconsistencies between the organizational model and the desired state."

Change in this sense is successive re-negotiation of the organizational model - not revolution! **Kotter´s NoNo** has good reasons to oppose the change - reasons that are probably triggered by the current system, not the individual´s twisted psyche. Again: What we observe should ultimately be coined *lack of consequence,* not resistance to change.

Which all leads us back to the conclusion: In change-as-constant-flipping, we must work the system, not the people. Diverting from this path leads to blaming, and almost inevitable to self-induced failure of our change efforts.

Insight 4. Org change is *socially* dense — the technical side is (almost) trivial

The idea of Emergent Change, or continuous flipping from Now to New acknowledges that change happens within complex pattern that cannot be predicted or controlled - but only observed. One of the first to describe this kind of thinking on change coherently was John Kotter. His Leading Change approach neatly outlined profound change as dense, social movement: The collective, emergent side of change, so to say.

The element that was still be missing from this change approach is the individual side of change - the need for individual adaptation that members of an organization have to undergo to flip or when flipping. Adding the individual side of org change to the collective side, one starts perceiving change as two-dimensional. We call this the **double helix nature of change.**

Many change agents are enamored with their method of choice. Many of us like to believe that this method or tool is wonderful, effective and impactful. Change as flipping, however, is based on the assumption that

"Relationship is everything, method is secondary"

There are many decent or effective methods, but what really matters is creating different relationships within the system, and relationships of higher quality. Many methods can help doing that. In fact, the more complex the problem is, the more complex, or social, the method must be. Nothing is worse than crystallized method - or "dead" method, applied to living problems.

"Method must always be appropriately complex, and social."

We will explore this aspect of change and complexity-robust method in future articles.

Insight 5. The is no such thing as transformation — instead, *everything´s an intervention*

I am guilty. I am guilty of talking about transformation myself. A lot. And I liked it! I liked to say things like: **Organizations should transform from the organizational model of the industrial age ("Alpha") to a contemporary, complexity-robust one ("Beta").** I keep saying that kind of thing, occasionally, even though I know the term transformation is neither helpful, nor accurate. Sometimes I just can´t help it!

The truth is probably closer to: There is no transformation. Because:

"Constant flipping is the only thing there is in change."

This is consistent with the old adage "Everything is an intervention." Which is one of the most beautiful things that has ever been said about change (which probably is a rather misleading term, as well). That everything is an intervention does not mean, of course, that every intervention is good in itself. It just means that everything, really everything, influences, or potentially flips an organization.

Instead of change management, we should practice the craft of change as exercising constructive irritation - as we like to say in systems theory. According to systems theory, the only thing you can do is to irritate a system. Then observe the consequences and ripple effects. Then irritate again. Then observe. And so on. Any irritation can flip the system into the New state. If you are lucky and if the irritation was smart enough, the state is a form of desired state.

In any case: irritate again. This is never supposed to be over. It´s not a journey, remember? Welcome to the world of, well: Eternal flipping.

Here are a few other related articles by Niels: ***Org Physics: How a triad of structures allows companies to absorb complexity and Flat hierarchies: Just another step in the wrong direction.*** *The "flipping" and "Now to New" wording/idea from this article were inspired by* ***Jack Martin Leith.*** *and his wonderful writing. Some of the insights on resistance in this article were inspired by* ***Eric Dent's*** *beautiful* ***article*** *on the same matter.*

***

To read this article online with embedded hypertext links, go here:
https://www.linkedin.com/pulse/change-more-like-adding-milk-coffee-niels-pflaeging

About Neils Pflaeging

Niels is a passionate advocate for a "new breed" of leadership and profound change in organizations. He is an advisor and author, as well as founder of the international open source community BetaCodex Network. For five years, Niels was a director with the prestigious Beyond Budgeting Round Table BBRT. His second book, *Leading with flexible Targets. Beyond Budgeting in Practice* was awarded the German Best Business Book award, in 2006. His fourth book *Organize for Complexity* was lauded by critics and readers alike and became an international bestseller. His latest books are *Complexitools and OpenSpace Beta* — both co-authored with Silke Hermann. Together with Silke, Niels recently founded Red42, an innovative start-up on the fringe of organizational development and Learning & Development.

Starting an Agile Center of Excellence

by Allison Pollard

[**Note:** grayed text indicates hypertext links in original article.]

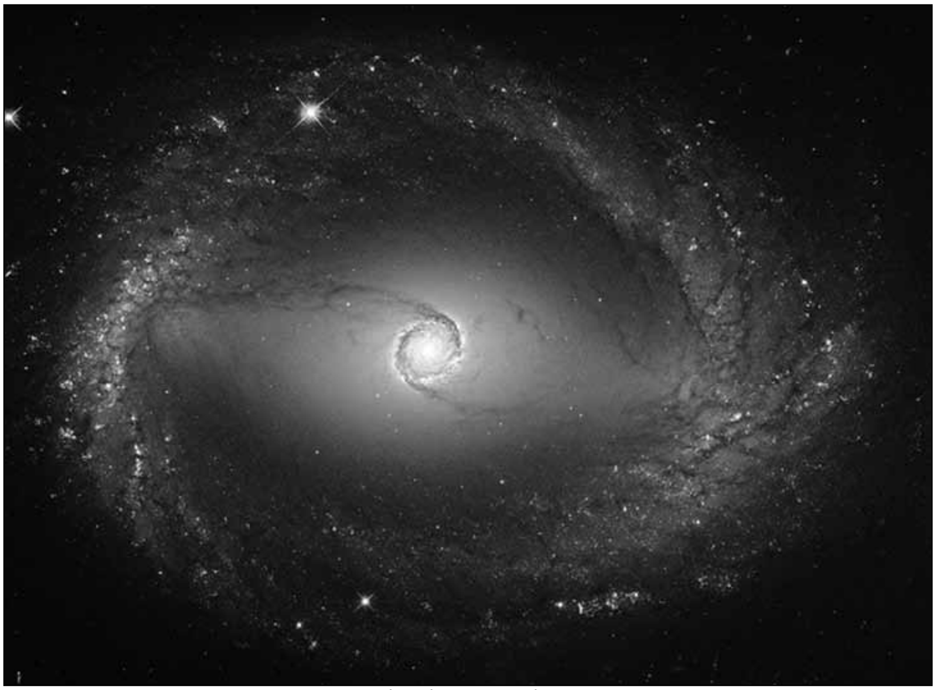


Photo by Stuart Rankin

Let me first say: I don't love the name "Center of Excellence." This is not about starting a group that has a monopoly on excellence or good ideas with an organization. Just the opposite--this is an entity that helps the organization become more excellent, which includes spotting internal excellence and promoting it.

Regardless of what you call it, an Agile Center of Excellence is meant to be a **helpful, consultative group.** Not a strict instrument of governance or compliance. While the group may help define mechanisms to promote transparency about product and team health, there is real danger in a COE becoming the internal compliance police.

Digging in further to the idea that an Agile Center of Excellence is a helpful, consultative group that helps an organization become more excellent, the vision of this entity is important. I've found it helpful to use an elevator statement format and Jason Little's **strategic change** canvas to gain alignment on the group's mission.

Another big challenge in starting an Agile COE is defining success criteria. What are the measurable results you are seeking? Why is this group being established? We often start thinking about the activities or services the COE will provide and how to measure them. I think of those services as the *how.* Measurements of these activities are our leading measures. I urge you to go deeper: what are the business outcomes wanted that are fostering the COE's genesis? The really important stuff that's probably harder to measure and will take longer to change: increased customer satisfaction, cost savings, more revenue, shorter time to market, etc. What is the reason for agile in the organization?

Why is it so important to define success criteria like this? It hinges on changes from people outside of the Agile Center of Excellence, which feels risky. And it is. Because it means that the Agile Center of Excellence is connected to the organization and must respond to its needs. The COE's success points to the why of the organization's change. I find that it enables—perhaps requires—the Center of Excellence to change, evolve, and pivot its offerings in order to continue helping the organization. It allows for agility by the group, which I think is important for those wishing to further enable agility. How cool would it be to see more Agile Centers of Excellence like that?

⁂

To read this article online with embedded hypertext links, go here:
http://www.allisonpollard.com/blog/2017/10/12/starting-an-agile-center-of-excellence

About Allison Pollard

Allison helps people discover their agile instincts and develop their coaching abilities. As an agile coach with Improving in Dallas, Allison enjoys mentoring others to become great Scrum Masters and fostering communities that provide sustainability for agile transformations. In her experience, applying agile methods improves delivery, strengthens relationships, and builds trust between business and IT. Allison is also a Certified Professional Co-Active Coach, a foodie, and proud glasses wearer. Visit her blog at www.allisonpollard.com to connect further.

The Cost Center Trap

By Mary Poppendieck

[**Note:** grayed text indicates hypertext links in original article.]

In the 1960's, IT was largely an in-house back-office function focused on process automation and cost reduction. Today, IT plays a significant strategic and revenue role in most companies, and is deeply integrated with business functions. By 2010, over 50% of firms' capital spending was going to IT, up from 10-15% in the 1960's.[1] But one thing hasn't changed since the 1960's: IT has always been considered a cost center. You are probably thinking "Why does this matter?" Trust me, cost center accounting can be a big trap.

Back in the mid 1980's Just-in-Time (JIT) was gaining traction in manufacturing companies. JIT always drove inventories down sharply, giving companies a much faster response time when demand changed. However, accounting systems count inventory as an asset, so and any significant reduction in inventory had a negative impact on the balance sheet. Balance sheet metrics made their way into senior management metrics, so successful JIT efforts tended to make senior managers look bad. Often senior management metrics made their way down into the metrics of manufacturing organizations, and when they did, efforts to reduce inventory were half-hearted at best. A generation of accountants had to retire before serious inventory reduction was widely accepted as a good thing.[2]

Returning to the present, being a cost center means that IT performance is judged — from an accounting perspective — solely on cost management. Frequently these accounting metrics make their way into the performance metrics of senior managers, while contributions to business performance tend to be deemphasized or absent. As the metrics of senior managers make their way down through the organization, a culture of cost control develops, with scant attention paid to improving overall business performance. Help in delivering business results is appreciated, of course, but rarely is it rewarded, and rarer still is the cost center that voluntarily accepts responsibility for business results.

Now let's add an Agile transformation to this cost center culture. Let's assume that the transformation is supposed to bring benefits such as faster time to market, more relevant products, better customer experiences. And let's assume that the cost center metrics do not change, or if they do change, process metrics such as number of agile teams and speed of deployment are added. I'll wager that very few of those agile teams are likely to focus on improving overall business performance. The incentives send a clear message: business performance is not the responsibility of a cost center.

Being in a cost center can be demoralizing. You aren't on the A team that brings in revenue, you're on the B team that consumes resources. No matter how well the business performs, you'll never get credit. Your budget is unlikely to increase when times are good, but when times are tight, it will be the first to be cut. Should you have a good idea, it had better not cost anything, because you can't spend money to make money. If you think that a bigger monitor would make you more efficient, good luck making your case. Yet if your colleagues in trading suggest larger monitors will help them generate more revenue, the big screens will show up in a flash.[3]

Let's face it, unless there are mitigating circumstances, IT departments that started out as cost centers are going to remain cost centers even when the company attempts a digital transformation. What kind of mitigating circumstances might help IT escape the cost center trap?

1. **There is serious competition from startups.** Startups develop their software in profit centers; they haven't learned about cost centers yet. And in a competitive battle, a profit center will beat a cost center every time.
2. **IT is recognized as a strategic business driver.** You would think that a digital transformation would be undertaken only after a company has come to realize the strategic value of digital technology, but this is not the case. IT has been treated as if it were an outside contractor for so long that it is difficult for company leaders to think of IT as a strategic business driver, integral to the company's success going forward.
3. **A serious IT failure has had a huge impact on business results.**

When it becomes clear exactly how dependent a profit center is on a so-called cost center, people in the profit center are often motivated to share their pain with IT. Smart IT departments will use this opportunity to share the gain also.

Many people in the Agile movement preach that teams should have responsibility for the outcomes they produce and the impact of those outcomes. But responsibility starts at the top and is passed down to teams. When IT is managed as a cost center with cost objectives passed down through the hierarchy, it is almost impossible for team members from IT to assume responsibility for the business outcomes of their work. When IT metrics focus on cost control, digital transformations tend to stall.

Every 'full stack team' working on a digital problem should have 'full stack responsibility' for results, and that responsibility should percolate up to the highest managers of every person on the team. Business results, not cost, should receive the focused attention of every member of the team, and every incentive that matters should be aimed at reinforcing this focus.

The Capitalization Dilemma

Let's return to the surprising assertion that in 2010, over 50% of firms' capital spending was going to IT.[4] One has to wonder what was being capitalized. Yes, there were plenty of big data centers that were no doubt capitalized, since the movement to the cloud was just beginning. But in addition to that, a whole lot of spending on software development was also being capitalized. And herein lies the seeds of another undue influence of accounting policies over IT practices.

Software development projects are normally capitalized until they are "done" — that is they reach "final operating capability" and are turned over to production and maintenance.[5] But when an organization adopts continuous delivery practices, the concept of final operating capability — not to mention maintenance — disappears. This creates a big dilemma because it's no longer clear when, or even if, software development should be capitalized. Moving expenditures from capitalized to expensed not only changes whose budget the money comes from, it can have tax consequences as well. And what happens when all that capitalized software (which, by the way, is an asset) vanishes? Just as in the days when JIT was young, continuous delivery has introduced a paradigm shift that messes up the balance sheet.

But the balance sheet problem is not the only issue; depreciation of capitalized software can wreck havoc as well. In manufacturing, the depreciation of a piece of process equipment is charged against the unit cost of products made on that equipment. The more products that are made on the equipment, the less cost each product has to bear. So there is strong incentive to keep machines running, flooding the plant with inventory that is not currently needed. In a similar manner, the depreciation of software makes it almost impossible to ignore its sunk cost, which often drives sub-optimal usage, maintenance and replacement decisions.

Capitalization of development creates a hidden bias toward large projects over incremental delivery, making it difficult to look favorably upon agile practices. Hopefully we don't have to wait for another generation of accountants to retire before delivering software rapidly, in small increments, is considered a good thing.

To summarize, the cost center trap and the capitalization dilemma both create a chain reaction:

1. **Accounting drives metrics.**
   ⇩
2. **Metrics drive culture.**
   ⇩
3. **Culture eats process for lunch.**

The best way to avoid this is to break the chain at the top – in step 1. Stop letting accounting drive metrics. Alternatively, if accounting metrics persist at the senior management level, then break the chain at step 2 – do not pass accounting metrics down the reporting chain; do not let them drive culture. When teams focus on improving the performance of the overall business, accounting metrics should move in the right direction on their own; if they don't then clearly something is wrong with the accounting metrics.

Beware of Proxies

This year Jeff Bezos's annual letter to Amazon shareholders[6] listed four essentials that help big companies preserve the vitality of a startup: customer obsession, a skeptical view of proxies, the eager adoption of external trends, and high-velocity decision making. These seem pretty clear, except maybe the second one: a skeptical view of proxies. Just what are proxies? Bezos explains:

> *"A common example is process as proxy. Good process serves you so you can serve customers. But if you're not watchful, the process can become the thing. This can happen very easily in large organizations. The process becomes the proxy for the result you want. You stop looking at outcomes and just make sure you're doing the process right. Gulp."*
>
> *"Another example: market research and customer surveys can become proxies for customers – something that's especially dangerous when you're inventing and designing products."*

Here are some common proxies we find in software development:

> Accounting metrics are proxies, and not very good ones at that, because they encourage local sub-optimization.
>
> Project metrics – cost, schedule, and scope — are proxies. Worse, these proxies are rarely validated against actual outcomes.

"The Business" is a proxy for customers. Generally speaking, so is the product owner.

Proxies should be resisted, Bezos argues, if you want a vibrant startup culture in your company. But without proxies, how do you manage the dynamic and increasingly important IT organization? You make a habit of measuring what really matters — skip the proxies and focus on outcomes and impact.

In his excellent book, *A Seat at the Table,*[7] Mark Schwartz proposes that IT governance and oversight should begin with strategic business objectives and produce investment themes that accomplish these objectives. IT leaders fund teams to produce desirable outcomes that will have impact on the strategic objectives. Note that these outcomes are not proxies, they are real, measurable progress toward the strategic objective. Regular reviews of teams' progress — quantified by these measurable outcomes — provides leaders with insight, flexibility and an appropriate level of control. At the same time, detailed decisions are made by the people closest to customers after careful investigation, experimentation and learning.

Schwartz concludes: "this approach can focus IT planning, reduce risk, eliminate waste, and provide a supportive environment for teams engaged in creating value."[8] What's not to like?

Endnotes:

[1] From ***What is Digital Intelligence*** **by Sunil Mithas and F. Warren McFarlan, IEEE Computing Edge, November 2017.** Pg.9.

[2] The 1962 book ***The Structure of Scientific Revolutions*** **by Thomas Kuhn** discussed how significant paradigm shifts in science do not take hold until a generation of scientists brought up with the old paradigm finally retire.

[3] Thanks to Nick Larsen. **Does Your Employer See Software Development as a Cost Center or a Profit Center**?

[4] ***What is Digital Intelligence,*** ibid

[5] ***What is Digital Intelligence,*** ibid

[6] **Jeff Bezos – Letter to Shareholders– April 12, 2017**

[7] ***A Seat at the Table*** **by Mark Schwartz**

[8] ***A Seat at the Table,*** ibid

⁂

To read this article online with embedded hypertext links, go here:
http://www.leanessays.com/2017/11/the-cost-center-trap.html

About Mary Poppendieck

Mary started her career as a process control programmer, moved on to manage the IT department of a manufacturing plant, and then ended up in product development, where she was both product manager and department manager. After Mary left the corporate world in 1998, she found herself managing a government software project where she first encountered the word "waterfall." When Mary compared her experience in successful software and product development to the prevailing opinions about how to manage software projects, she decided the time had come for a new paradigm. She wrote the award-winning book *Lean Software Development* to explain how the lean principles from manufacturing offer a better approach to software development. Over the past several years, Mary has found retirement elusive as she lectures, teaches classes, and writes books with her husband Tom. They are co-authors of three books on Lean Software Development and a fourth book, *The Lean Mindset,* about product development. A popular writer and speaker, Mary continues to bring fresh perspectives to the world of developing software-intensive products.

Facilitating Squadification for a SAFe Agile Release Train

By Em Campbell-Pretty

[**Note:** grayed text indicates hypertext links in original article.]

The squadification day had arrived! We had **management buy in to allowing people to self-select into teams** and a **structure for our new new Agile Release Train** (ART). I turned up with my **Time Timer** in tow ready to facilitate what I hoped would be a great beginning for this brand new ART.

Over the course of the week leading up to the self self-selection, the thought of launching a new ART with no experienced Scrum Masters had been on my mind. How we would find the right people for those Scrum Master roles? I tend to choose what I read based on what is on my mind, so I had picked up my copy of **Geoff Watts's Scrum Mastery** and reread a few chapters on my flights to and from Sydney that week.

I took two bright ideas away from this: (1) we had to reinforce the message at self-selection that the Scrum Master role "holds no authority", and (2) when asked to nominate a Scrum Master teams tend to know instinctively who will be the right fit. Inspired by this my first task on the day of the self-selection event was to track down the **Release Train Engineers (RTEs)** and suggest that rather than letting individuals self-select into the Scrum Master role we let the teams nominate their Scrum Master after the squads had been formed. They were agreeable so that became the new plan.

Now we had to get organised. Flip charts were drawn up for each squad and a Product Owner's photo added. Everyone else's photos were laid out on a trestle table at the front of the room. By 9:15am we had almost full house, so we decided to kick off. I opened with a quick run through of the agenda for the day, followed by the lead RTE who set the scene for why they had chosen to use self-selection as the ap-

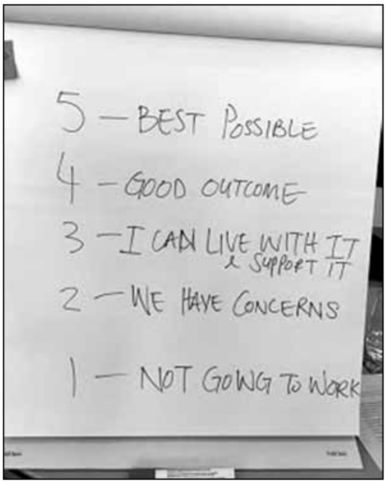
proach to forming teams. Then it was back to me to run through the logistics for the morning.

First everyone needed to collect their photo from the front of the room, or have one taken if somehow they had managed to avoid being photographed during the week! Next we heard from each of the Product Owners about their features and why people should choose their squad. One product owner was quick to offer up food and wine as an incentive to join his squad!

Then it was time for the self-selection to begin. Some people moved quickly, almost running to the squad they wanted to join. Others were more cautious. At the end of the 10 minute time box for Round 1 we were faced with a few unexpected outcomes. First, no one had remembered to brief the interns, so they formed their own team! Secondly, no one was without a home. Thirdly, adherence to the "rules" was sketchy at best.

One of the recommendations **Sandy Mamoli** and Dave Mole make in **Creating Great Teams** is to minimise the constraints. For this self-selection we had come up with three rules: (1) do what is best for the company, (2) teams needed to be made up of 8 or 9 people and (3) each team should have a least one person from each of the functional groups. At the end of Round One we had a number of teams of 9 and some teams of 5 or 6. We also had teams that were completely lacking in some skill sets.

Round 2 was marginally better. There was some movement but also some very stubborn participants and the teams still varied greatly in size. Something just wasn't quite right but I couldn't put my finger on it. Each team played back to the room their overs and unders and then we took a morning tea break, during which we reminded everyone of the number one rule — do what is best for the company.


5 — BEST POSSIBLE
4 — GOOD OUTCOME
3 — I CAN LIVE WITH IT & SUPPORT IT
2 — WE HAVE CONCERNS
1 — NOT GOING TO WORK

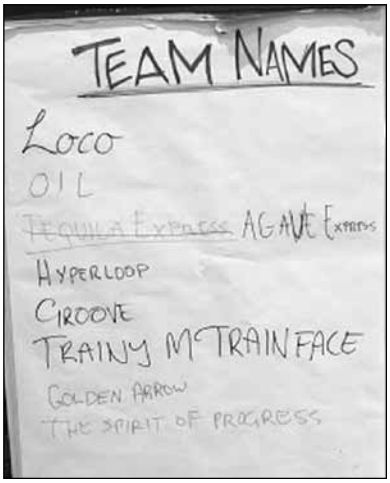
At the end of Round 3, we introduced confidence voting. Using a "fist of five" we asked each team their confidence that their team could deliver on its mission. Where squads responded with a 1 or a 2 we asked what they needed in order to increase their level of confidence. This helped the teams get far more specific about what skill sets they were missing. We also asked the RTE and the department head to vote, which helped maintain focus on the big picture and doing what is best for the company, In the final round, a couple of people were nudged by management to moved teams, in the best interests of the company and the ART. I found this uncomfortable however with the clock ticking and the **rest of the QuickStart commencing Monday,** it felt like the only way we were going to get to an viable outcome. Despite the management interference, when it came to the final confidence vote all the squads voted confidence of three or above. It was a wrap.

As much as I should have been thrilled at this point, I could not shake the feeling that something was not quite right. We ended up with six squads - two teams of nine, two teams of eight, one team of seven and one team of six. Not exactly evenly matched feature teams!

The three squads without dedicated Scrum Masters nominated Scrum Masters. That was also more difficult than anticipated. One squad essentially had a volunteer so that was easy. One squad voted and the nominee said "I'm too busy!". When they re-voted the next nominee was quite rightly concerned that he also did not have the time! The third squad nominee was about to go on extended leave. Not exactly the magic answer I had been hoping for, but we had Scrum Masters.

We closed the morning with a lightweight retrospective. While there had clearly been some challenges with the process, I think it would be fair to call the event a success.

We used the afternoon for some team kick off activities. The new teams were given an hour to come up with team names and build a Team Product Box. Over the prior few weeks the department had nominated theme for the train and then voted to decide between them. After a very close battle between Game of Thrones and trains, the train theme won out. Strangely, of all the trains I have been involved with, this is only the second time that the train has had a train theme for team names. (In this instance this choice has ended up creating some confusion as newcomers have understood each team to be its own Agile Release Train!)

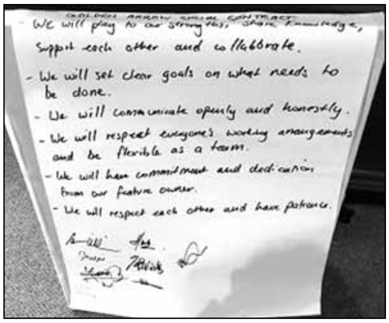


TEAM NAMES

Loco
O'L
~~TEQUILA EXPRESS~~ AGAVE Express
HYPERLOOP
GROOVE
TRAINY M^c TRAINFACE
GOLDEN ARROW
THE SPIRIT OF PROGRESS

The creativity of the teams with both creating their product boxes and naming their teams was inspiring. And of course it would not be a team naming ceremony it one or two names did not have to be vetoed by leadership. At the end of the hour the teams introduced themselves to the train and showcased their product boxes. The energy in the room was nothing short of amazing.

The other kick off activity for the afternoon was the creation of team charters. For this we used a variation on **Edwin Dando's *How to make a social contract and build better teams.*** While the teams were working on their charters, the "aha" moment I had



Golden Arrow Social Contract

- We will play to our strengths, share knowledge, support each other and collabbrate.
- We will set clear goals on what needs to be done.
- We will communicate openly and honestly.
- We will respect everyone's working arrangements and be flexible as a team.
- We will have commitment and dedication from our feature owner.
- We will respect each other and have patience.

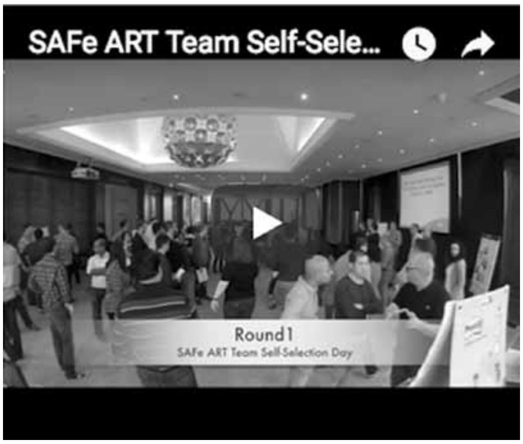
[illegible]

been waiting for occurred. The four offshore developers that we had thought would be joining us for the quick start had been unable to arrange travel at short notice. This meant that we were four people short but we did not adjust the constraints for the team-selection. The maximum size for a team should have been eight not nine! That was why the teams were so unbalanced. It was time to confess.

I pulled aside the RTE and filled him in on my thinking. I also expressed concerns about communication challenges the nine person teams were likely to encounter. I was keen to rebalance sooner rather than later, but when would be a good time?! After some debate about the pros and cons of making the change immediately, we decided to leave it be. Take the weekend to think it over and revisit the topic on Monday — day one of the QuickStart and SAFe for Teams training.

Once the teams finished up their team agreements, we did a quick walk through of the run sheet for next week's quick start and called it a day. One day down and five to go!

Time Lapse Video from the Self-Selection Day



https://youtu.be/cKNbvBpXbJY

Reflecting on the self-selection event there were a few lessons learned:

Don't assume everyone knows everyone

One of the things I discovered after the self-selection event, was that there were not a lot of existing relationships between the functional teams. Given they were a co-located team of teams, I had just assumed they all knew each other. Seriously I surprise myself sometimes! I have told the story of the beginnings of the EDW Agile Release Train countless times, always explaining that there were circa 100 people that had worked together for years mostly collocated over a couple of floors in the one building that did not know each other's names. Why did I think this team would be any different?!

Be crystal clear on your expectations

In *Creating Great Teams,* Sandy and David recommend minimising constraints. I completely agree with this - however, I would temper this advice by suggesting you also need to be clear about your expectations. If the constraints and your expectations aren't aligned you are sure to end up disappointed. In this case we wanted even matched feature teams — ideally with two people from each competency, but we didn't tell anyone that!

This did end up being resolved after Day 1 of the SAFe for Teams training, when the RTE shared our concerns regarding team size and balance with the ART. In an attempt to minimise disruption we asked the over and under size teams to stay and work through a solution, with goal being to reduce the nine person teams to eight person teams and add a person to the six and seven person teams. We asked the smaller teams to nominate the skills they were short of and then asked them to work with the nine person team that had the most people with that skills set. The intent was to find volunteers to move, which of course proved more challenging than anticipated.

It was interesting to observe the very active role the product owners who were also line managers played in this horse trading. Their sense of "ownership" over their new teams made me nervous. While not something to solve for that day I noted this as something to watch for as we moved into execution mode. After about an hour of rather emotional and uncomfortable discussion, the moves were agreed. We had fairly evenly matched feature teams — at last — but I fear the cost of the last minute changes could take some time to surface.

It was this event that crystallised for me how much the features allocated to each team had influenced the team shape. At the end of each round of the self-selection process we had asked the squads "Do you have all the skills to deliver on your mission?" What we should have asked is: "Do you have balanced representation of all the A&I skillsets?"

When using self-selection for a feature team ART perhaps don't seed the teams with missions

As you already know we chose to follow Sandy and David's guidance and seed each team with a mission. We did this by pre-allocating features to product owners and using these features as a proxy for the team mission when seeding the teams. In an effort to avoid teams being too theme centric, when it came to providing the teams temporary names for the purpose of the self-selection event we went with Product Owner names not themes. In hindsight this was an abject failure.

First, it created the impression that the product owner's owned the teams. This coupled with the fact that most of the product owners were the most senior person on their team. This created a strange power dynamic, that is taking some time to breakdown.

Secondly, people tended to choose teams based on the work anyway! This was different to the patterns that Sandy and David have observed where people tended to choose a team based on who they want to work with. The weird part of this was that teams were not going to be changed for at least 6 months but the features only represented 10 weeks worth of work. This choice set an expectation we would move people to the work instead of work to the people. This was contra to our goal of creating a world in which teams would "pull" in the work they wanted each PI. While not catastrophic this did mean we had to manage expectations as we moved into PI2.

I think if I had it over, I would try and structure the event so that the newly formed teams pulled down the features that they wanted after the self-selection event!

Communicate earlier

This was simply a miss. There were lots of good reasons why we did not communicate earlier but I do think it hurt us on the day. At a minimum I would like to have communicated the problem we were trying to solve and the constraints before the event. This provide an opportunity to flush out any flaws with the thought process and gives people more time to make considered choices.

The good news is none of these challenges had a catastrophic impact on the ART. In fact 5 months later this challenges have paled into the background as the ART has hit the ground running, with a momentum that has the whole building talking about the marked changed in the department since the 6-day quick start!

Stay tuned over the next few weeks to learn about how we tackled **just-in time training at scale** and **the ART's first PI Planning event.**

Read what happened when we "re-squadified" after three Program Increments.

***

To read this article online with embedded hypertext links, go here:
http://blog.prettyagile.com.au/2017/01/facilitating-team-self-selection-safe-art.html

About Em Campbell-Pretty

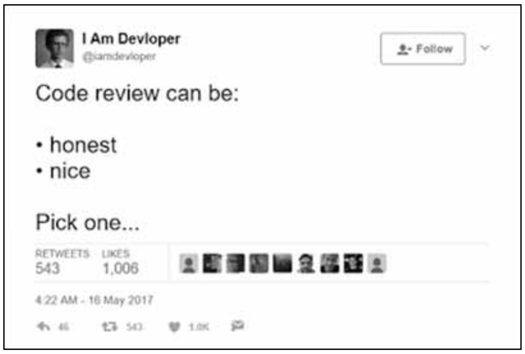
After close to 20 years in business management roles within multinational blue chip corporations, **Em** discovered Agile and became passionate about the opportunity it provides to align business and IT around the delivery of value. She is an internationally acclaimed business strategist, keynote speaker and one of Australia's leading Enterprise Agile consultants. At the heart of Em's success is her passion for creating cultures of transparency, lean leadership, learning, innovation and fun.

Today Em is a SAFe Fellow and the Managing Director of Pretty Agile Pty Ltd, a boutique consulting and training company focused on helping organisations achieve business agility leveraging Em's unique, culture first approach to the Scaled Agile Framework. Em is also the author of the popular blog about scaling agile: prettygile.com and the Amazon #1 Best Seller: *Tribal Unity: Getting from Teams to Tribes by Creating a One Team Culture.* Em can be reached at em@prettyagile.com.

Honest or Nice

By Jane Prusakova

[**Note:** grayed text indicates hypertext links in original article.]


I Am Devloper
@iamdevloper

Follow

Code review can be:

• honest
• nice

Pick one...

RETWEETS 543 LIKES 1,006

4:22 AM - 16 May 2017

46 543 1.0K

We have a lot of conversations how we could do better. Deliver more functionality and prettier UI, cause fewer bugs, have more fun while we put in more hours. Write better code, and pay back the technical debt.

At **200OK** web professional's conference we were talking about **code review** — the part of software development process where developers and architects get together to consider other people's code with the purpose of offering critique.

Having one's code subject to review is terrifying for many people, and liberating for others. It is also necessary, for most of the code outside of personal projects. But how we are going about doing the review can make a huge difference for all involved.

There comes a scary idea of being nice to the people one works with. Genuinely, authentically nice — actually wish them to be successful, be willing to put effort in helping them, and talking about that.

Here are some suggestions:

- Start code reviews with saying to your fellow developers that the work they have done toward the team's goal is noticed and appreciated.
- Call out good code — clearly expressed logic, fitting patterns, relevant abstractions, meaningful naming.
- Notice good intentions, as expressed in code, even if the result is less than perfect. That includes error handling, code broken down into smaller modules, attempts at unit tests.

- Finally, suggest changes as you would to a very senior, very experienced colleague — share knowledge while acknowledging their wisdom and understanding. Everyone needs to learn, and everyone deserves to learn in a respectful, cooperative environment.

Being actively and explicitly nice, yet honest, to one's teammates does a few interesting things to overall team dynamic. More people speak up and offer ideas. Jerks become more visible. More conflict bubbles up and out, and leads to a healthy discussion.

It may even lead to delivering more value, while enjoying working together more.

***

To read this article online with embedded hypertext links, go here:
http://blog.prettyagile.com.au/2017/01/facilitating-team-self-selection-safe-art.html

About Jane Prusakova

Jane has been a software developer for over 20 years. She has worked for fast-growing startups, large established companies, and everything in between. As a developer, Jane is focused on building quality software, whether it is a large-scale distributed system utilizing the latest technology, or a quirky solution for a small business with unusual needs.

Jane believes that the best part of being in software development industry is participating in a continuous learning experiment — how do we do what we do better, easier, faster? One answer to that question that has been shown to work is to use Agile methodologies. Another answer is to pay attention to technical quality in the earliest stages of design and implementation.

Jane shares her thoughts on software craftsmanship and improving software development in an online blog at http://softwareandotherthings.blogspot.com.

Don’t Limit the Role of the Scrum Master

By Paulo Rebelo

[**Note:** grayed text indicates hypertext links in original article.]

I’ve heard quite a few questions about the necessity and value of having a Scrum Master in an organization. Sometimes, the role of the Scrum Master is combined into the engineering manager role or any other technical role which would create a conflict of interest. These questions are born either from a poor knowledge of Scrum or a misunderstanding of the principles and values behind it.

However, there are some people who limit the role of the Scrum Master and follow exactly what is written in relevant literature. That’s another problem that affects the community of Scrum Masters in general: the **Scrum Guide** tells you what to do in overall terms, and it’s your responsibility as a Scrum Master to go beyond that and bring value to both the company and the team. There is never a one-size-fits-all recipe; every team and company has different missions, people, values and cultures.

In this post, I'm going to share some tips and practices to help other Scrum Masters excel in their role and add real value to their organization.

Challenge the Process

As an enthusiastic Scrum Master, challenge the process of the entire company and lead the organization to embrace the agile mindset, principles and values (such as commitment, focus, openness, respect and courage).

Push not only toward continuous improvement, but also toward continuous adaptation. The Scrum Master is accountable for adapting the process in respect of the business strategy and mission.

In certain occasions, perhaps for operations/infrastructure/maintenance teams when it makes sense and is appropriate, the Scrum Master can streamline the process and turn it into a Kanban method. The Scrum Master will benefit from other agile methods, tools and practices as well. Continuous learning is essential.

Coach the Engineering Practices

If you are a Scrum Master who has a good technical background in software engineering, or if you develop some code in your free time, try to foster and coach technical practices within the team, such as continuous integration, continuous delivery, pair programming, test-driven development (TDD), automated acceptance testing and refactoring.

For instance, continuous delivery is a very valuable practice that will allow the team to deliver features into production more often and get better feedback from the stakeholders. Your team code will obtain both quality and performance with the introduction of these engineering practices.

Communicate With Stakeholders

Be the conduit among business and technical stakeholders, make sure the communication is fluid and well understood and spread the knowledge of Scrum. Engaging the stakeholders and making sure they are all on the same page is important for the health of the product. Teach them how to maximize return on investment (ROI) and meet their objectives.

Peer With the Engineering Managers

Help the engineering managers with the team's performance reviews as necessary.

Encourage them to empower the team and provide any type of support needed. The Scrum Master can also help to expedite the hiring process by finding candidates who would be a great fit for the team.

Educate new engineering managers, product owners and team members on-boarded in the company. Whenever a new person is hired, the team will need to be restructured again to be balanced and continue working efficiently. The Scrum Master is needed to provide training and coaching during that restructuring period.

Scale Scrum and Agile

Scale up large products into multiple agile teams, integrate the pieces all together and promote training to spread the knowledge of Scrum and agile. Help business people to understand the framework. Get them involved and obtain feedback frequently, listen closely and take immediate action.

Create and foster communities of practice for product owners, Scrum Masters, engineering managers and development teams. Agile is all about collaborating with other teams and people, sharing good practices and learning lessons.

Learn new practices continuously and experiment with them with the teams. There is always a new technique to apply from the Scrum experts, so don't hesitate to be curious.

Promote hackathons and dojos to drive innovation and excellence. Both the company and the teams will gain significant benefits from hackathons. People will have an open time to expose their ideas, and the company will have the opportunity to hear those ideas and possibly implement them.

On the other hand, a dojo is a powerful technique to improve your skills, grow up and be more efficient. For example, you can facilitate a coding dojo for new hires and bring experienced software engineers to hook up with them. The newbies will get to see how coding is done in the company while simultaneously bonding with senior-level engineers. It also enables the transfer of knowledge.

Nurture Scrum structures with high-level managers and executives. Show the benefits of Scrum to them and speak up. Don't be intimidated by them; if you are seriously passionate about Scrum, you can influence and lead the entire organization to the next level.

Work Closely With Product Owners

While helping the product owner to enhance the product backlog, why not contribute to the backlog and innovate with some new insights and ideas? A great Scrum Master takes an opportunity to understand the business context and drive every decision based on it. Each team and each product require different strategies and, again, no one recipe is a good fit for every team.

Moreover, a great Scrum Master focuses on improving the agile approach the product owner is taking, thereby providing templates and tools for the product roadmap, business-driven development (BDD), Business Model Canvas and mockups/drawings.

Streamline the Scrum Ceremonies

Improve the way retrospectives are held depending on the situation and phase of the product/project. There are several different types of retrospectives that can be conducted to extract the right pain points and identify action items to be tackled.

Call each ceremony a "conversation," since development teams typically prefer conversations rather than meetings. Remove waste in each conversation and invite people with an effective purpose and clear agenda. If possible, take the team to demonstrate the user stories as soon as they are done by the team. Coordinate with the product owner on what is proposed to be brought during the sprint planning and do your best to make it efficient.

Assist the Development Team

To remove waste, bureaucracy and unnecessary work assigned to the team, take ownership. In some cases, they need someone to take accountability of the change management control due to compliance regulations or something similar. I once faced a situation in which a company needed to create some documents and present the changes planned to be released into production. It was not rocket science, but it took time and focus from the team.

Run team-building exercises to connect with each other in a positive atmosphere, cultivate happiness across the team members, celebrate each milestone reached and resolve conflicts in a collaborative way.

Collaborate With Other Areas and Departments

There are some companies that have a PMO (project management office), and the Scrum Master can help turn it into an Agile PMO with KPIs and metrics that make sense in terms of deliverables.

Moreover, the Scrum Master can conduct process improvements on the business side and influence them to implement Scrum as well. Scrum is not only for software; it can also be applied in marketing, finance, accounting, HR and many other places. Spread it across the organization.

Conclusion

We used to limit the Scrum Master role according to the parameters provided by literature. However, this is not enough: the world is evolving and demands flexible frameworks adapted to the circumstances of the company and the market in general. One particular method might fit very well with one team, but may not work for another team, it really depends on several factors, such as product, people, technology, company and market. The basis is taught, then from there, you will adapt, experiment, learn and evolve.

From the topics depicted above, there is a common sense that the Scrum Master role is a long-term assignment and will never end. The success of the Scrum Master relates to the value that it brings to the organization as a whole, as well as the level of happiness the team is experiencing: they are learning, delivering, feeling safe and awesome, working at a sustainable pace and not disturbing their personal lives. Even the best high-performance team can benefit from the Scrum Master role.

After reading all these tips, do you still feel that the role of Scrum Master can be replaced or discontinued? Do you believe that the success of a Scrum Master occurs when the team doesn't need him/her anymore? There are tons of different ideas to be accomplished by a Scrum Master. This post listed only a few, and I guarantee that the work will never be done. As always, be passionate about serving others and delivering value.

If you have any doubts, questions or contributions related to the role of the Scrum Master, feel free to let me know in the comments section below.

⁂

To read this article online with embedded hypertext links, go here:
https://www.frontrowagile.com/blog/posts/130-don-t-limit-the-role-of-the-scrum-master

About Paulo Rebelo

Paulo helps companies to improve using agile and lean principles and methods like Scrum, Lean Startup, XP, and Kanban. He currently works at Blackhawk Network in the U.S., helping teams succeed by building great products. He is teaching programming to kids (Python) as a volunteer and for fun.

His background is a developer, Scrum Master, product owner, project manager, program manager, and agile coach. He is comfortable with most dimensions of software product development — both technical and business sides.

Paulo is a CSP-SM and CSPO from the Scrum Alliance. A PMP from the PMI and has been certified as a Management 3.0 consultant from Happy Melly. He is also Java Certified.

Paulo can be reached at https://www.linkedin.com/in/paulohenriquefonsecarebelo/.

Empowering a new culture to emerge in organizations

By Chelsea Robinson

(*illustrations by the author)*

[**Note:** grayed text indicates hypertext links in original article.]

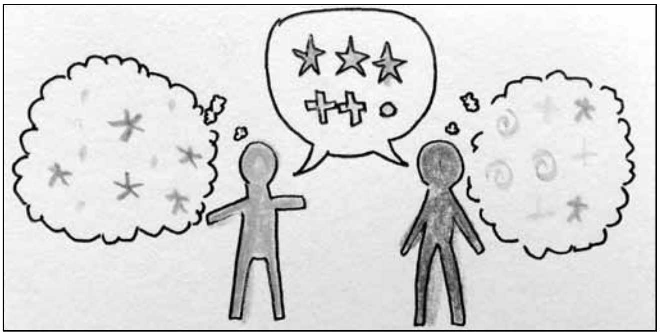
> *"Yes we need the whole organisation to become more collaborative, agile and innovative, but I want our executives to make the final decision on who gets budget to try things."*

I've consulted with a few different organisations in the past couple of years. Some government department teams, some small start-up sized teams, and some global organisations. Everywhere I go around the world, people want to learn about design thinking skills, collaboration & consensus building skills, non-hierarchical management, the ability to navigate and lead in complexity with authenticity. These are fantastic skills to build in this century. We will face unprecedented changes as a global human community in the years to come, so I'm delighted to help some people transition their behaviours, attitudes and skills towards resilience & responsiveness.

This is not straight forward work. People resist change within our institutions as much as they resist changes in our society, if not more. Compare communication through heavily designed slide decks replaced by quick chat feeds, or a detailed planning timeline versus a well noted initial hypothesis to iterate from — these cultural differences may as well be on the other side of a mountain range. Mostly we are all unaware of our resistance to change. We have every intention to evolve, but learning is hard. Especially when it necessitates recognising you might have once been right, but now your hard earned expertise is the very thing that's letting you down. So, I have compassion for every person pushing their own boundaries as they navigate and lead change. It is these inner battles that draw lines of irony in our behaviour. Lines of irony look like co-creation in one meeting and coercion in another because in one space you feel threatened and the other you don't. This is the territory of culture change work. This means you cannot simply come into an organisation knowing the answer or knowing the best course of action. "Best" is context sensitive, and context awareness teaches you how to get good results.

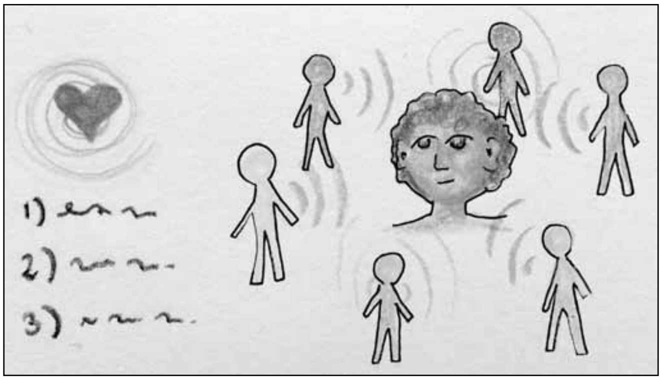
After an online call recently, I made a cup of tea and thought about what I had just been experiencing. I've never worked in a large consulting company where they train you in "The Way" to problem solve for clients. I've mostly built organisations, worked in social-outcome focused companies with friends, and focussed on creating as much positive social & environmental impact I could manage. So, in my consulting I am providing input from a place of lived experience, from trial and error, and the best practices I've discovered as breakthroughs. I am weaving participatory design into community development, sculpting innovation work streams into teams running careful experiments, and hosting groups through Theory U. This feels genuine to me, and although I am a structured thinker, there is no single approach for each engagement. Or is there?

Standing in my kitchen and thinking it through, I noticed that regardless of whether I'm running training workshops, introducing new technology platforms, coaching new leadership styles, recommending specific process improvements or facilitating whole workflows in groups, there are some phases which reappear each time. These phases are relational. They describe the type of interactions that take place over time and how the role of the consultant/ facilitator/culture catalyst changes and evolves. The phases do not address content or ideas, but rather where to put your attention. Here is my expression of these phases. I believe that if I skip one, the rest will be much less effective. I want to share this to support others who I know are working hard to help amazing organisations transform their culture, structure and practices. *Let's learn together how to best serve brave organisations that want to change.*



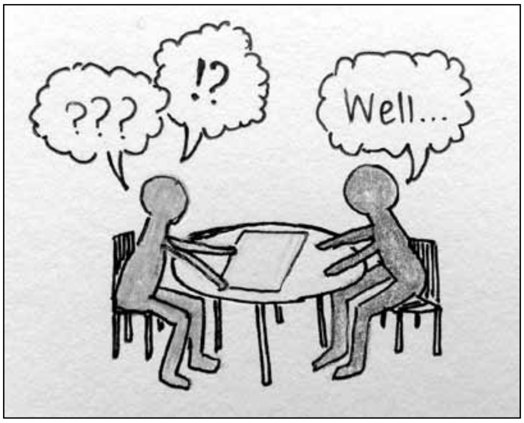
(0) Expectation setting & mutual understanding

The sales process itself is part of the work. From the outset, get focussed on coming into shared understanding. What does the client need, rather than what you want to sell them? What do they actually mean when they say "more innovative" and is there some deeper need they're hinting at? This is the moment to talk about what's hurting, what the biggest possibilities are and how a consultant can help. It's also the most creative part of the process — you can think big together. You can generate options and articulate a scope that brings the best out. It's also the hardest part because it's a negotiation and you want to build a relationship and not let numbers get in the way, yet you have to be really clear about timelines and money and hold that gracefully. What will you do? What will you not do? Decide together.



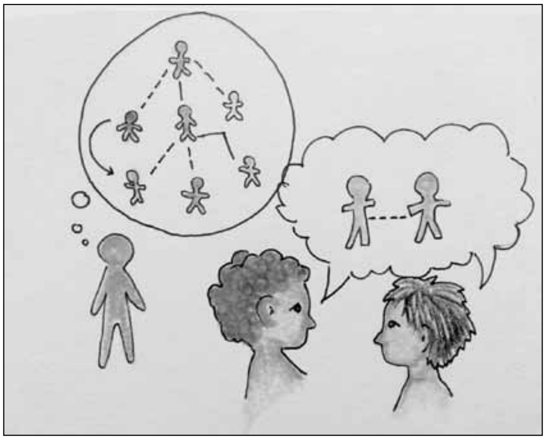
(1) Trust and relational understanding

When you get involved in the work, and all systems are go, it feels like time to prove yourself and take action. But, it's actually time to listen well and help people feel heard. Let go of the fear of underperforming. You will underperform (compared to your potential) if you get into solutions too soon. Listen to every view point. Listen to the person who hired you, their senior and their junior and everyone else who touches what you're working on. Better yet, get permission for this to be explicitly a listening/researching phase. **There are two things to listen for, and two goals: how people are feeling, and what actually needs to be done.** You want most people you're interconnected to in the organisation to feel heard by you and trust that your comments & thinking will incorporate their perspective going forward. The time invested in one-on-one coffees, calls and also group check-ins about the initiative/issue will contribute to the development of a trusting and exciting atmosphere. Slowly, you will build a three-dimensional view of what this organisation is ready to do together.



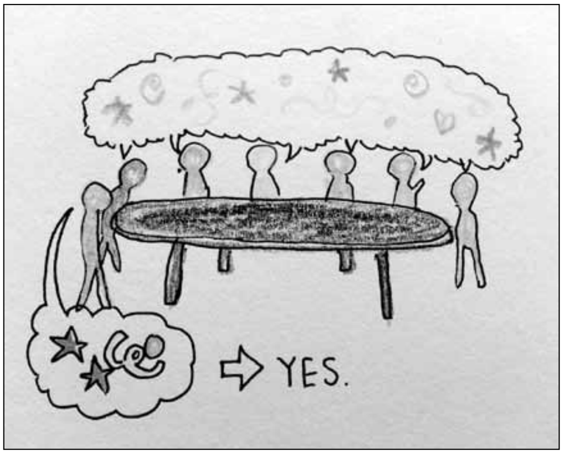
(2) Institutional context / system awareness

Through your listening, you probably built a foggy mental picture of "how things work around here". This can be worth drawing or writing down to help build this picture with others. But don't solidify it — gain an understanding of why things are working like this. What happened a year ago that made the organisational structure take this shape? What was the personality of the founding team and how does that still show up today in the way leadership relates to staff? With your new social capital you can also test the edges of the conversation. Bring up something you're noticing that you think is a key issue that no ones' talking about and see what the reaction is. Do they see it too? Or is it not appropriate (yet) to work with that edge? Initially, the questions you'll ask in your workflow will be basic. Over time, try to find the most powerful questions you can ask. "Who makes decisions about hiring, firing & promotions?" "Tell me a story of how a conflict was handled recently" "How are we holding ourselves accountable to work by the values we're espousing to others?"



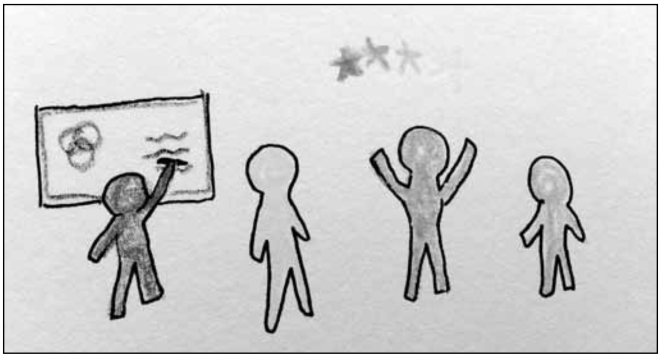
(3) Power, influence flows and barriers

When ever I hear all the voices in a system, I am always confronted by the pain and frustration I hear from those who don't have the power to change the barriers/ceilings upon their work. This is not always the same as who has less authority in the organisational structure diagram. Simultaneously, people with more power are juggling very real confusion about who they can delegate to or trust with more power, without having to micromanage. Some people in organisations can use their intuition every day, no matter who it affects when they change course. And others are stuck catching dropped balls or re-orienting their workflow again to cater to the latest insight. Often there is a reason some people are more senior than others, and sometimes despite their deservedness of authority, the way they conduct their form of leadership can be punishing to those around them. As you learn about how people feel and about what people need to achieve, pay attention to **who is doing emotional labour and interpretive labour.** Who has to guess what someone else is thinking in order to do their work? Who doesn't feel valued? Who is the person said to be showing initiative when they're following their nose, when other's might be considered to be off task? I have begun to incorporate a phase of this work which is explicitly about working with those with less power to build their confidence and identify new strategies for contributing their best gifts to the teams they're part of. These conversations focus on how they can choose to interact with their colleagues differently, and kick start a new dynamic. Change processes require leadership from everyone, so this focus on igniting the agency of people affected by change is a key success factor. Importantly, remember who has authority in the organisation and maintain an open and clear communication channel with them throughout. Working with power issues can make the people with power feel betrayed or undermined, so find a neutral identity to help you speak with integrity to people in all the different situations.



(4) Collaboratively generating actionable interventions

With trust, ideas, empowered agents of change and a timeline to meet, it's time to get a group together. Help the group see the issues and opportunities ahead and around them. Take people through mini-version of the journey you have been through yourself. Reflect back to the group what they have brought up within the process so far. Summarise, without coming to a conclusion yourself. Create a space to allow the stakeholder groups in the organisation to articulate with freshness some key changes they're ready to make based on their new understanding of what needs to be done. This should feel enlivening and the only force you need to apply to this moment is pressure to crystallise and clearly describe what it is that as a group we're going to try to change together. And how we will start & continue to develop that together over time.



(5) Pushing a new concept forward from a place of understanding

As you move into action, start pushing your insights into the conversation more actively. The group is activated, they trust you, they're in motion. It's okay to bring boundary pushing concepts to the table. You have done the research, you know the vision and the group has mandated themselves to realise it. You can be an inspiration to them if you can bring fresh thinking, actionable optimism and pragmatic ambition into the room continuously, even as it gets challenging and their attempts to change things get slowed by emerging issues. Work with the willingness of the group to make strong recommendations based on what you think should change. Re-iterate the most powerful suggestions that the group has made to each other in the past and remind them that those ideas are possible now. As you move forward, continue to work on the process that you're all working in. Keep talking about barriers to adopting new ways of working and hold the change agents inside the organisation fiercely accountable to their vision.

(6) Get alongside the implementation of the changes

You are not just a facilitator here. Once people have momentum — get in motion with them. Take responsibility for the delivery of some of the new work that has emerged. Change processes generate a huge new work load — people have to deliver the same outcomes they always had to, and do their change process work around the edges. Take some of the load — its what you're here to do. Scope the type of details you get into, as you cannot change it all. Join or create a working group which focusses on changing one specific part of the organisation. At the same time, maintain high-level alignment between all change agents. Hold regular spaces to talk through and refine or continuously improve what everyone is trying out.

A simple process in a bigger picture

I like to think that this approach is not only how we might shift the cultural ecosystems within organisations, but also towns, sectors and other groups of people. I'm not saying this is the most elegant model for social innovation — I'm saying it's a blueprint for organisational change phases which hints at the same ingredients needed for kickstarting wider systems change. As we support organised groups to shift they way they're organising, it will help us build the necessary skills to facilitate a wider cultural shift in society. Together, let's strengthen our ability to help people make that shift.

***

To read this article online with embedded hypertext links, go here:
https://medium.com/enspiral-tales/empowering-a-new-culture-to-emerge-in-organisations-9fe35131d59b

About Chelsea Robinson

Working across climate change, public-private partnership programs, and the future of work for over 10 years, **Chelsea** is a facilitator, entrepreneur, program designer and venture coach. Chelsea works with many organisations including World Wildlife Foundation International, the Chan Zuckerberg Initiative, the Good Work Institute, Enspiral, and more to turn bold, provocative strategies for impact into into something tangible, pragmatic and actionable. She takes every opportunity to codesign initiatives with the people they are supposed to benefit, even when it means building new strategies from first-principles. Impact is her compass.

chelsearobinson.me

Agile Approaches Require Management Cultural Change

By Johanna Rothmann

[**Note:** grayed text indicates hypertext links in original article.]

Ron Jeffries, Matt Barcomb, and several other people wrote an interesting thread about **prescriptive and non-prescriptive approaches** to team-based agile. The issues are nuanced and for me, don't lend themselves to a Twitter discussion. (Learning how to write short and coherently is a different post.)

If you don't want to read the entire thread, here is a summary: People often need help with their agile approach. Some people start with Scrum in its entirety. Some people use a combination and build up.

In some ways, Scrum is a prescriptive approach: it defines roles, it defines a timebox of work, and the minimum times to plan and reflect. It's a framework, not a fully defined process. And, that's part of the problem. To use Scrum effectively—or any other agile approach—team members need to think themselves about what agile approaches mean to them personally, and as a team.

That's why we have the agile **values** and **principles.** Too often, I meet people who haven't internalized the values and haven't read the principles. (And, if they're supposedly using Scrum, they haven't read the **Scrum Guide.** Argh!!)

Gil Broza has a terrific video about why people don't realize the mindset is a critical part of an agile transformation. **See Practice Does not Make Perfect: Why Agile Transformations Fail (50-min video**).

Andy Hunt (along with the late Jared Richardson) started the **Grows Method.** The idea is you start with small experiments, and proceed to more complex ideas as you master the necessary project "hygiene": work on one thing at a time, use continuous integration, work in rank order, etc.

I wrote about the history of agile approaches in the first chapter of **Create Your Successful Agile Project** and what people might need to consider for their agile project.

I am sure that Ron (and Chet) teach understanding, not just "do this practice" because they are terrific teachers and explainers.

There are many potential problems with an agile transformation. The biggest one I see is the difference between **Theory X and Theory Y management:** the idea that people are resources who need to be pushed to work, or the idea that people want to do a great job for the company.

Agile approaches challenge the management mindset and therefore the corporate culture.

Culture expresses what managers value. Culture (according to Edgar Schein) is what people can discuss, how people treat each other, and what we reward. If we reward hero work, multitasking people and (excessive) planning instead of throughput, and no or insufficient feedback about everything, our agile transformation cannot succeed. It doesn't matter what approach we use, we can't succeed.

Prescriptive frameworks, such as Scrum can help everyone see the culture is closer to Theory Y rather than Theory X. In addition, Scrum makes the culture clash visible.

However, using skills or prescriptions as a way to transform the organization fails in these ways:

- We don't see how to change the culture of management, which drives the culture of the entire organization.
- A prescriptive approach doesn't help the team members, teams, and managers see the culture and know what to do to change it. There is a difference between team-based work where people are interdependent and workgroup work where people work independently. Iterations don't work for management and other workgroups. Standups don't work for workgroups because standups are about micro-commitments between people, not status reporting. (I wrote about this in **Create Your Successful Agile Project**).
- And, not every framework is useful for your project. You might need a more frequent **cadence** of planning and reflection than your approach suggests.

I don't buy the **Shu-Ha-Ri approach** to agile transformation because it assumes that by changing behavior, we can change culture. That might be true for a project. I have yet to see it be true for management. Even though I prefer the **Dreyfus model of skill acquisition** (because it's more nuanced), it's often not quite enough.

We need to address the culture changes for agile with small experiments. (This is why I like Cynefin so much and used it to explain many of the issues in Agile and Lean Program Management.)

For me, the question is how can we help managers move from a plan-driven, resource-efficiency mindset to an adaptive, **flow-efficiency,** feedback-driven mindset? (Yes, I am thinking/starting to write that book, too.)

We don't need managers to change *first.* However, for any agile approach or a transformation to work, we need the management culture to change. For me, that's the difference between iterations of waterfalls and a real agile culture.

⁂

To read this article online with embedded hypertext links, go here: **https://www.jrothman.com/mpd/agile/2017/12/agile-approaches-require-management-cultural-change/**

With Agile, No Warnings Needed

By Johanna Rothmann

Have you ever worked on a project where the management and/or sponsors felt it necessary to provide you warnings: "This release better do this or have that. Otherwise, you're toast."

I have, once. That's when I started to use release criteria and check with the sponsors/management to make sure they agreed.

I happen to like release criteria. Even better is when you use agile on your projects. You might get feedback before the release. Here's what a client did on a recent project:

- They had release criteria and the sponsors agreed to the criteria.
- They released internally every two weeks and asked people to come to the demos.
- They asked the product managers and product owners to review the finished work and to make sure the managers/sponsors liked where the roadmap was going.
- The team worked in ways that promoted technical excellence, so they could (relatively) easily change the code base when people changed their minds.

The project didn't fulfill all the wishes that managers and sponsors wanted. Those folks wanted the proverbial 15 pounds of project into a 5 pound bag. On the other hand, the team is on the verge of delivering a terrific product. (They have one more week to finish.) They are all proud of their effort and the way they've worked.

This morning, the project manager emailed me. "I'm so angry I could spit," she said. "One of our sponsors, who couldn't be bothered to see any demos just told me that if he doesn't like it, he's going to send us back to the drawing board. Do you have time for a quick call so I don't get myself fired?"

This is a culture clash between the agile project's transparency and request for frequent feedback vs. the controlling desires of management.

We spoke. She realized it was a difference in expectations and culture that will take a while to go away. There are probably reasons for it, and that doesn't make it any easier for the team.

These kinds of situations are why I recommend new agile teams have a servant leader. I don't care if you call that person an agile project manager or some other term, but the person's role is to run interference between the two cultures.

The worst part? With the project's transparency and interim delivery of value, no one needed to warn anyone about anything. The data this guy was looking for was in the demos, in the meeting minutes and was easily accessible.

I don't know why people think they need to provide dire warnings. It's not clear what effect they want to create. Dire warnings make even less sense when the team uses agile and provides interim value and demos.

If you're using agile approaches, and you see this happening, decide what you want from this relationship. If you think you'll have to work with this person again and again, it might make sense to have a conversation and see what they really want. What are their concerns? What are their pressures? Can you help them with information at other times instead of a week before the end of the project?

Don't be surprised if you see this kind of a culture clash in your organization as teams start their transformation. Managers have a lot to do with culture (you might say they are the holders of the culture) and we're asking them to use different measurements and act differently. A huge change. (Yes, after the agile project book, I'm writing an agile management book. I know, you're not surprised.)

***

To read this article online with embedded hypertext links, go here:
https://www.jrothman.com/mpd/agile/2017/05/with-agile-no-warnings-needed/

Power, Management, and Harassment: It's a Cultural Problem

By Johanna Rothmann

[**Note:** grayed text indicates hypertext links in original article.]

You may think the #MeToo tag doesn't have anything to do with product development. Not so fast. When people behave badly (more often it's men than women, but it can be either), the people suffer. When the people suffer, the product suffers. It suffers in development and it suffers in release.

The issue is this: Behavior like this (sexual harrassment and discrimination) is an abuse of power. It is a cultural problem in society and in our organizations.

Edgar Schein defines culture as what you can discuss, how people treat each other and what you reward. (See **Organizational Culture and Leadership.**)

When a culture allows discrimination, harassment, or abuse, the organization says, "We won't talk about that." When people treat each other according to their role in the hierarchy, they say, "It's okay to treat other people badly." When the managers in charge get promoted, the organization actually rewards that behavior.

Abuse of power is a cultural problem.

You have heard this quote:

"The culture of any organisation is shaped by the worst behaviour the leader is willing to tolerate."

–Gruenert and Whitaker

In the case of Weinstein and Company, the worst behavior was quite bad. I have worked in places where it was almost as bad.

Some people in the agile community say, "We don't have this problem." Not so fast. I have coached and mentored other women in the past two or three years about how to deal with behavior based on this power dynamic.

When managers (anyone, but I mostly see this in managers) abuse their title-based power, they destroy the necessary social contract and the working behaviors that create a reasonable workplace.

In any workplace, abusing power becomes a disaster. In a supposedly agile environment, the people stop collaborating. Often, they stop the transparency around the work. They stop the agile behaviors that create value, and delivery. A team that was producing no longer produces. And, no one "knows" why.

A team sees the effects immediately: people withdraw from collaboration and certain social situations. Teams may not know what happened, but they know something occurred. And, the people in the situation know exactly what happened.

How do you manage an abuse of power?

Expose it. Don't reinforce it.

I see too much hiring that reinforces power abuse in an organization. Here are some of my hiring suggestions:

- Hire people who are not just like you: **How to Hire for Cultural Fit Without Becoming Insular and Mediocre**
- **Hire for Cultural Fit: It's Time to Add Women, Pt 1.** (Part 2 is about hiring people who are no longer young.)
- Understand what cultural fit really is: **How NOT to Look for Cultural Fit**
- Hiring managers for integrity over all else: **Hiring Managers: Asking About Integrity**

Also, consider reading **Hiring Geeks That Fit** because many of the ideas in there will help you assess your culture and your hiring practices. Read Behind Closed Doors to see what great managers do.

In addition, I have suggestions about **feedback** and, **women in management, one-on-ones** and how to **build career ladders and "manage" performance** so people can learn to build their interpersonal skills.

Here's the most important thing you can do: Expose the power dynamic and anyone's behavior that's not appropriate. Be a whistleblower on the abuse of power.

I actually mentioned some discrimination on the **Shift-M Podcast Posted About Hiring.** (We recorded it before the scandal broke.)

My points:

- Sexual harassment, discrimination, and abuse is about power. It's not about hormones. It's about power.
- The organization's culture reinforces this abuse of power.
- Decide what you want to reinforce in your culture and expose the abuses.
- Creating a culture that enhances collaboration will also enhance your product development. Reinforcing a culture of abuse makes it more difficult to create and release great products.

Oh, and #MeToo. I don't know a working woman who has not dealt with an abuse of power. My first experience was when I was 19. It has continued every decade of my working career. It's time to stop.

⁂

To read this article online with embedded hypertext links, go here:
https://www.jrothman.com/mpd/management/2017/10/power-management-and-harassment-its-a-cultural-problem/

About Johanna Rothman

Johanna, known as the "Pragmatic Manager," provides frank advice for your tough problems. She helps leaders and teams see problems, resolve risks, and manage their product development.

Johanna was the Agile 2009 conference chair and was the co-chair of the first edition of the Agile Practice Guide. Johanna is the author of 14 books that range from hiring, to project management, program management, project portfolio management, and management. Her most recent books are *From Chaos to Successful Distributed Agile Teams* (with Mark Kilby) and *Create Your Successful Agile Project: Collaborate, Measure, Estimate, Deliver.*

Read her blogs, email newsletter, and more information about her books at www,jrothman.com.

The Burger House: A Tale of Systems Thinking, Bottlenecks and Cross-Functionality

By Rafael Sabbagh

Key Takeaways

- Understanding the Theory of Constraints and Systems Thinking helps teams stop starting, and start finishing work
- Addressing bottlenecks is the most important activity to improve throughput in a system
- Local optimization frequently results in sub-optimizing the overall system
- Cross functional teams where individuals can step out of their specialist roles deliver better results

A few years ago, a small burger house opened on a narrow street in the business district of Rio de Janeiro, my hometown. A lot was going on there in the wake of the World Cup 2014 and Summer Olympics 2016. The idea was good: few, but very high-quality ingredients, and an open, visible kitchen in a small, cozy place.

As we used to run some of our training classes nearby, there was a day when my business partner Rodrigo de Toledo invited me to try their burger. We got there, I stood in the cashier line and waited just a little bit. The first thing I noticed was their system was optimized so ordering would be very efficient: the options were very visible and it very clear what I had to do. And then, when it was my turn to order, I had to choose: hamburger or cheeseburger? Add some bacon? Tomato, lettuce, onions, pickles, ketchup, mustard, house sauce? Add soda and fries? I made my choices, paid for your burger, stepped to the side and then… the whole experience went awry.

A bunch of people was standing disorderly at the waiting area, hoping to get their food while other people behind a counter were struggling to assemble the burgers and put together a large number of orders to deliver them to the hungry customers. Waiting a long time, fighting to grab your order just to find out you got the wrong burger or with unexpected ingredients was not unusual.

A few weeks later, Rodrigo went to this place again for a burger, and one of the two cashier positions was closed. The guy who was supposed to be there called sick that morning. Can you guess what happened? Chaos would you say? Not at all.

Here's what he experienced: he got to the store, has stood in line for the only available cashier and, of course, waited a little longer than usual. And then he had to choose from the few options available. A couple of individuals in the back were searing some burgers and frying potatoes ahead of time. He paid for his order and stepped aside, where surprisingly a few people were orderly standing waiting for their burgers. On the other side of the balcony, he could see people assembling the burgers and putting the orders together at a synchronized good pace. In a minute, he grabbed his burger in a tray and moved to the back where he could sit and enjoy his lunch. Cool, huh?

Yes, you got it well: the experience had improved! Way better, and one person shorter at the store. Any idea how this is possible? Please let me explain the "magic".

In the regular days, the assembling station — where a couple of line cooks assemble the burgers and put the orders together — is a bottleneck. The two cashiers, along with their optimized ordering system, are throwing a lot more work to be done than what is supported by this station, leading up to growing inventory, that is, orders to be delivered. Which puts pressure on the people working at that station: "Oh my god, we've got a huge number of orders to prepare!".

Those customers standing there waiting for their burgers - the next step in the flow — add to the pressure on that bottleneck. The cooks want to keep up, and they speed their pace over their capacity, which makes quality to fall. Lower quality means delivering burgers to the wrong people, and with the incorrect ingredients. Customers will return the food, which ends up lowering the throughput. As a result, they are selling fewer burgers.

What happened when the cashier didn't show up for work that day? A lower rate of orders, which relieved the pressure on the bottleneck. Luckily, the order inventory went down and stabilized to just the sufficient amount, so the people assembling the burgers could do the work, just in time. And they started doing it efficiently, in a sustainable pace, with fewer errors in preparation and delivering the food to the right people. Therefore, fewer people were standing there waiting for their orders. Yes, there was a longer wait at the cashier line. But not only the whole experience was a lot smoother for the consumer, as I can bet system throughput - the number of burgers sold — was higher that day (though nobody actually measured it).

But you know what? This causality is so counter-intuitive that the restaurant manager got the second cashier back the next day. And… the problems were back.

Just a few months ago, I was at an airport — as usual — in Porto Alegre, a southern Brazil city. If you like great meat, that's the city you want to go. But this time, I was just watching one of these international fast food chain stores at the food court. No real meat involved, then.

They had like five cashiers open. I could identify a few other stations, such as burger frying, burger assembly, potato frying and salad assembly. The cashier herself pours soda and puts the order together.

Suddenly, the store manager yelled something at the back, and one employee closed her cashier, washed her hands and went to help at the burger frying station. It took me a few minutes to understand the beauty of what happened there: the two guys at that burger station were in trouble, and the burgers weren't coming out as fast as needed to supply the orders. Therefore, orders started to accumulate. With the rising order inventory, the manager understood they were about to get deeper and deeper into trouble. Clearly, at that moment, that station was the system bottleneck.

Maybe that employee was a wizard at the cashier, but… where was she more valuable at that moment? Throwing more orders at the other stations? That would not only not increase the number of burgers sold, as it could, in fact, decrease it! So she could just stop doing her work, which would be better. But there was way more value on having her assisting burger frying, in any way she could possibly do to help speed it up.

Of course, for that to be possible, that employee needed to know more than her single specialty - being a cashier. She needed to know something about frying a burger. Instead of professionals who know a lot about one single thing, people who can also do a couple of other things well are needed. It doesn't matter whether she would only be able to be of some help, any help, or if she could do the whole act, as long as if with sufficient quality. Any of that would bring value to the system, where working at the cashier at that moment would not.

And what if, in another moment, the bottleneck laid on another station, like assembling the salads? She or someone else would need to know how to help there as well. Maybe another cashier?

The point is, given all activities needed in a system to produce something end-to-end (in that case, to deliver an order), the more of it team people know how to do, the better it is for the throughput of the system. Having cross-functional individuals is the key there.

It is not hard to figure out that whatever your system produces, whether it is burgers, software or anything else, its throughput will always be limited by its bottleneck. It will never deliver faster than what the bottleneck allows it.

Not onIy that, but if you optimize any stage of your system before the bottleneck, there will be a rising inventory, which has its cost, and will possibly also increase the pressure over the bottleneck and lead it to sacrifice quality. If you optimize any stage after the bottleneck, it will starve and keep asking for more, thus increasing the pressure on the bottleneck as well. Both may help reduce the throughput. In other words, optimizing any stage without looking at the system as a whole - what we call "local optimization" - will possibly lead to system sub-optimization.

It is important to notice that, in that case, they used a manager to tell them when and how to act in those situations. But, with a self-organizing team, its members themselves would find ways to understand when their work in progress had reached a reasonable limit and then take action.

Now picture this pretty usual software development team. The DBA is busy creating and updating the database tables. The back-end developer is busy building the API to access them. The business logic person is busy creating the classes. The front-end developer is busy creating the dynamic HTML. The designer is busy creating the design. And the tester, at this moment, is idle waiting for something to test. The manager is happy because he is keeping almost everyone "productive." But at what rate are they creating anything end-to-end, anything that works?

They do get a lot started, but not so much done. What if they would work in priority order, starting with the most important things? What if they would limit the amount of work they could have in progress, so they wouldn't start anything new before finishing more important stuff? What if they would break those silos or, at least, blur those borders and start sharing responsibility and helping each other?

Impossible? Well, as with the second burger house story, they would get a lot more done! Once testing is the bottleneck, it is more important that the DBA guy helps in that task than if he creates one more table. Or if the bottleneck is the front-end, the back-end guy would need to give a hand! As a benefit, one would learn from each other, and they would grow as a team!

In this new scenario of better throughput, quality, and motivation, titles lose importance and having different knowledge and skills becomes key. No matter what the team members used to call themselves, they will now have different jobs with their teams: one is doing whatever each do best, but only when it is necessary. The second is teaching others, which is not just a gift to others, but to the team as a whole and thus, back to them. And three, getting out of the comfort zone, learning from others in the team and taking on work in different areas whenever there is a bottleneck.

Fourth would be working with their managers, so they can help or, at least, get out of the way. As counter-intuitive as this all is, there is no surprise in learning that too many managers today simply don't get it. They worry about people doing their jobs in their silos. They worry about keeping people busy — "busyness" is their key metric. They worry about people having a lot started, but they forget that finishing stuff is what really matters.

As we often hear from practitioners of Kanban, a framework for knowledge work inspired in the Toyota Production System and the Theory of Constraints, the crucial thing is to "stop starting and start finishing!"

***

To read this article online with embedded hypertext links, go here:
https://www.infoq.com/articles/burger-systems-thinking

About Rafael Sabbagh

Rafael is the co-founder of Knowledge21, where he has been working with agile transformations of companies ranging from startups to multinationals in the last few years. He has over two decades of experience with enterprise IT projects, product development and software development methods and techniques. He has been working with Agile and Scrum since the mid 2000's.

Rafael is a Certified Scrum Trainer (CST) and was a member of the Scrum Alliance Board of Directors from 2015 to 2017. He has a master's in business and a Computer Engineering degree.

Rafael has done training and coaching in over 20 countries in the past. He speaks English, Spanish and Portuguese fluently. Rafael spoke in several events throughout the globe, including over a dozen Scrum Gatherings and major Agile events worldwide.

He wrote and published, in 2013, the first book about Scrum written in Portuguese. It is today a well known reference in Brazil.

Rafael can be reached at rsabbagh@knowledge21.com.

Consciously Approaching Agile for Lasting High Performance

By Michael Sahota

[**Note:** grayed text indicates hypertext links in original article.]

In this post, I share how to Consciously Approach Agile so we build lasting high-performance in our organizations. **It's a proven framework for creating success with Agile.**

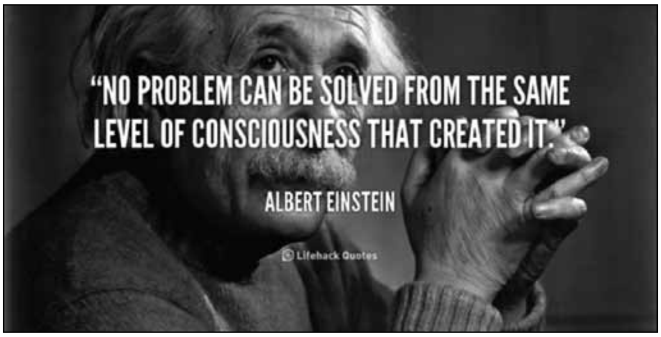
Why we need a new Approach

It is well understood in our industry that Agile is failing due to lack of attention to the organizational system and culture in particular.

- In 2012, I started publishing **industry research showing Agile failure and culture challenges**
- In 2016, we continue to see **80% of organizations having culture challenges**
- In 2107, this continues to be a **severe Agile industry challenge**

Doing things the same way from the same way won't work.

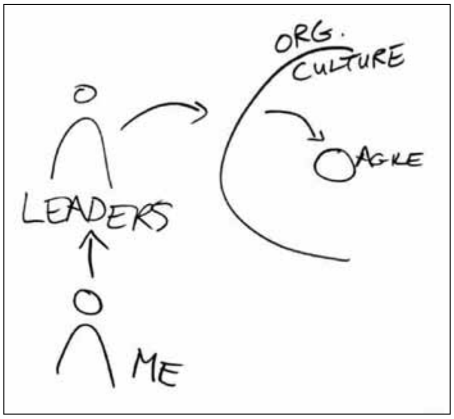
We need a new paradigm.


"NO PROBLEM CAN BE SOLVED FROM THE SAME LEVEL OF CONSCIOUSNESS THAT CREATED IT."
ALBERT EINSTEIN
© Lifehack Quotes

Consciously Approaching Agile for Lasting High Performance

I introduce an approach that I have proven through years of development and experimentation. Hundreds of students of my **Certified Agile Leadership Training** all over the world have validated this.

The diagram below shows how to approach Agile from a different consciousness. I will walk you through step by step how to approach change in a way that supports lasting success.



Let's walk through the steps …

Step 1. Start with a desire for Agile (or Innovation, Digital, DevOps, Engaged Workers, etc.)

Let's say that you want the full benefits of Agile — you want Agile to produce faster delivery, better products, or increase operational effectiveness. If you have an Agile Transformation or some Agile Initiative, this is a good starting place — there is desire and interest in improving the organization.

Note: Everything I am sharing here fully applies to Innovation, Digital, DevOps & other approaches that require a shift in mindset and culture. I am using Agile as an example since that is where I spend most of my time.

Step 2. Create an organizational and cultural context suitable for Agile

The first thing we do here is to drop the "Agile blinders" that only see things from an Agile perspective. To stop seeing things from a just a team perspective.

Instead, we look from an organizational and cultural perspective. We know Agile, Digital, etc. will flourish when we have the right organizational and cultural context.

In order to do this, we Drop Agile as goal and focus on org goals. Please refer to **Agile is a Means not a Goal** for a detailed explanation of why this is needed and how we may do this.

"Culture Eats Strategy for Breakfast."

– Peter Drucker

When we look at the Agile Manifesto, we see that Agile is actually pointing to a culture system that supports high performance.

- In 2011, I used the Schneider culture model to articulate that **Agile is about collaboration and learning (cultivation) culture.**
- In 2015, I use the Laloux culture model to highlight that **Agile is about engaged workers and high performance.**

Success with Agile requires us to focus on culture. There is no other way. (Unless you are in one of 5% or organizations that already have an amazing culture).

3. Leaders go First

Who is responsible for creating and shifting the organizational and cultural context? Leaders! **Organizational Behaviour Follows Leadership Behaviour.** Culture change requires leaders who model the new behaviours and ways of working. Leaders who lead.

Most Agile initiatives have tell-tale signs that there are challenges around this. It's not enough when we can say "We have leadership buy-in" or "We have leadership support". That is sufficient for adoption of Agile practices. However, the culture change needed for high performance requires highly invested leaders. Places where people say "We have leaders who are inspiring us." **We need Leadership Leadership.**

4. It starts with Us

In high performance organizations we see leaders at all levels. We see leaders who build other leaders around them. That where we all can play our part. Regardless of your role, lasting change starts with us. We need to examine our own behaviour and take a serious look at ourselves to see how we are shaping culture. We can only lead others when we model the new ways of working. Success requires that we model Agile at a personal level. And not just a new technique or concept.

What is required is that we actually live the Agile values. We model excellent listening, respect, collaboration, courage, etc. I have started speaking about this as Wave 2 of Agile: Living Agile and plan to write about it soon.

Do This Now

The *most critical and practical thing* to start Consciously Approaching Agile is to Conduct a **WHY Workshop to discover the organizational drivers for Agile.**

Here is a health check list so you can do a reality check. Do you and your leadership team:

1. Trust each other deeply?
2. Admit to making mistakes?
3. Ask for help or admit limitations?
4. Challenge each other to get great outcomes?
5. Show they are deeply committed?
6. Hold themselves and each other responsible?
7. Focus on shared outcomes and not their department?
8. Create leaders at all levels of the organization?

If you answered answered no for some of these, then your leadership and organizational culture would benefit from investment and focus. Another resource you can use is an earlier **organizational transformation checklist.**

Where to Learn More

Read my blog or join me worldwide for my unique **Certified Agile Leadership Experience (CAL1)** to learn a detailed playbook for how to deliver high performance in your organization.

⁂

To read this article online with embedded hypertext links, go here:
http://agilitrix.com/2017/10/consciously-approaching-agile/art.html

About Michael Sahota

Michael trains and guides leaders how to create high-performance organizations. His highly accoladed "Agile" Culture & Leadership (CAL1) Training gives gives managers and coaches worldwide the mindset shift needed for lasting success. In 2012, he published the ground-breaking book *An Agile Adoption and Transformation Survival Guide: Working with Organizational Culture.*

Stop Wasting $$$ Building So Much Crap!

By Reese Schmit

[**Note:** grayed text indicates hypertext links in original article.]

So many teams have a list of projects laid out on a roadmap sometimes months or years out, without a clear idea of how success is measured. Are they being measured based on the number of projects completed? Getting them done “on time”? High quality? Team utilization? Are any of these things helping meet the company objectives?

When did we stop experimenting and start believing we were always right?

Why are we spending so much money building things that may or may not have any real value? How are we even determining what we build?

We have spent years calling ourselves Lean or Agile, as we optimize the delivery of the highest priority items in our backlogs. That is making the big assumption that we're building the right things. What we are probably doing, though, is building the wrong things, faster.

Building The Right Product

Before we move onto the next idea that will "make us a million bucks" or that's sat long enough on a wishlist that it has nagged it's way to priority, let's get our acts in gear and I ask ourselves a few questions:

1. What are our goals?
2. What are our hypotheses around how to hit those goals?
3. What are the tiniest experiments we can do to prove or disprove our assumptions around our hypotheses?
4. How can we validate our experiments with our customers?
5. What's the next big hypothesis?
6. Rinse and repeat.

Picture this: a team has the quarterly goal to increase user adoption of the mobile platform and enable a segment of users to become completely independent of the website, doing all business via the mobile app. In collaboration with Product, Marketing, UX and Customer Service, they start with a brainstorming session looking at where the numbers are now and come up with a few hypotheses on where barriers to entry are for users who fall out.

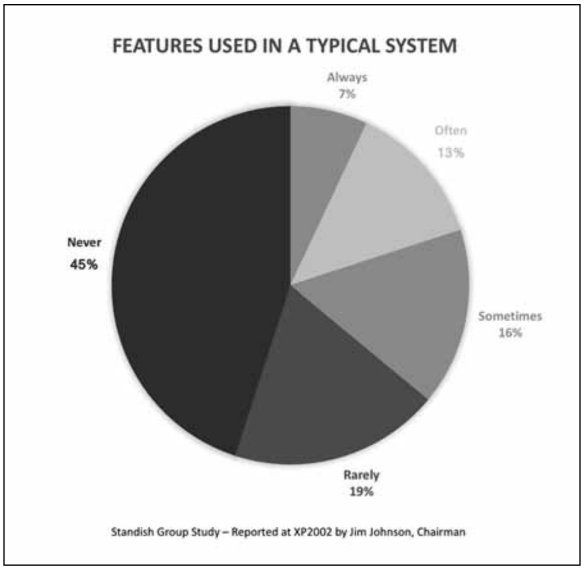
They determine what are the riskiest assumptions made, then come up with small experiments that will quickly help them determine if their assumptions are right or wrong. Once they determine what they should build from those experiments, they get that out in front of customers as quickly as possible to continue learning. Only after they truly see how and if the customers are using the product do they flesh out past the bare bones.

Sounds fantastic, right? So why aren't we doing it?

Building Only What We Need

If you are already doing the above, congratulations! You might be building the right thing, but you're likely still building WAAAAY too much. Too much of a good thing is still too much. When do you stop?

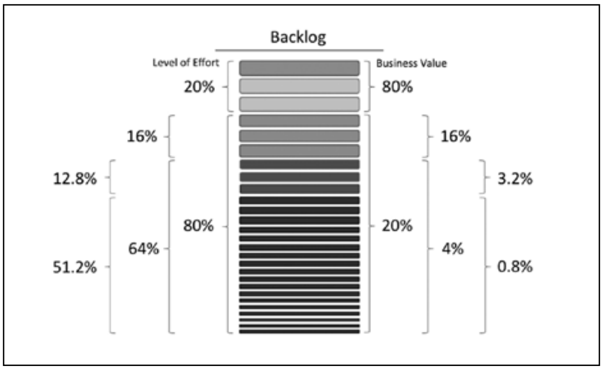
You have taken that idea and broken it down into the Epics or Features that make up the high-level backbone of your project. You've prioritized those by value and you may even have broken down the top of the backlog into roughly sprintable stories. You are ready to bring in the team and fill them in on your vision and walk through the stories that will be their guides for the next few months to build out your gorgeous product. Heck, you might even have trimmed it down to an MVP. But is it really what your users want? Will they really use it? Will it solve their problems? How do you know?



You've likely seen the above graphic of the findings of a Standish Group Study. They discovered that 64% of software features are rarely or never used. 64%! That is a lot of wasted time. That is a lot of wasted money. That is 64% more value we could have been adding somewhere else. Why do we keep building when we aren't adding value?

Jim York did an Open Space session at Scrum Gathering earlier this year about Awesome Product Ownership. He challenged us to apply the Pareto Principle (also known as the 80/20 Rule) to a typical backlog. The logic goes as follows. You get 80% of the value from 20% of the effort. As you move down the backlog you apply the 80/20 Rule to what's left. The first 20% of the backlog nets you 80% of your business value. That's the easy calculation. Now it's math time!

After knocking out that first 20%, 20% of the remaining 80% is 16%. This 16% of effort delivers 16% more value (80% of the remaining 20% = 16%). The next jump is another 12.8% effort delivering only 3.2% more value. That leaves 0.8% of the value of the backlog items taking up almost 50% of the effort. Where should we stop building? After that initial 20%? Into the next 16%?



If we compare the backlog to the segments of the Standish report below, we see a correlation. Always and Often represent 20% of the features, while Sometimes makes up the next 16%. So we can deliver 96% of the value by **building only the top third of our backlog.** WHY ARE WE STILL BUILDING EVERYTHING AND THE KITCHEN SINK!?!?!

When the business value dips below the level of effort percentage, we should stop. Yes, this requires ruthless prioritization. It requires talking to your users, testing hypotheses via experiments with validated learning and really figuring out what they want.

It is also not good enough to just apply the 80/20 rule to features or stories. You will need to dig into the stories themselves and apply an 80/20 on the acceptance criteria. York asked us who on the team would know what the most valuable thing to build in each story would be. It took a minute, but then someone had the "ah-ha" moment and exclaimed, "The QA Analyst!" "What is it?", Jim questioned. "The Happy Path!!"

The cheapest, most valuable thing to build is the Happy Path. If you aren't familiar with the term, it is the path the user takes through the product when everything is going as planned. This is such a high percentage of the time, yet we build out the 13 other paths they could take complete with error handling. Why??! This adds a ton of time and complexity to the code and doesn't necessarily get us back the ROI. Some alternate paths might be necessary to avoid crashes or error conditions for "productization" sake, but the key is to assess which alternate paths deliver the most business value, and discard or delay the others. Perhaps apply the 20% rule again, this time on the Alternate paths as well. We are drowning in a sea of "what ifs" and our users are suffering because of it.

We need to stop spending so much money building so much crap. Set goals, hypothesize how we will hit those goals, run the smallest experiment we can to prove or disprove our assumptions, build the happy path, and then stop. As we learn we can choose to continue investing in the current project or allow ourselves to move on to the next experiment.

***

To read this article online with embedded hypertext links, go here:
https://agilevelocity.com/agile/stop-wasting-building-much-crap/

About Reese Schmit

Reese is an Agile Coach at Agile Velocity. Over the past 15 years, she's done just about every job in the software industry, from User Experience Designer to Product Manager, QA Engineer to Scrum Master. This varied experience has taught her to frame problems from different perspectives and drive change through empathy. As an Agile Coach, she has worked to help organizations increase customer value, transparency, and collaboration, and to motivate teams to continuously improve.

When she's not coaching, Reese is Program Chairing for Agile Austin, or helping build the program for the Global Scrum Gathering and the Agile Alliance's annual conference. Otherwise, she's either chasing her tiny human girl or two canine boys around, hanging with her amazing husband, brewing beer, or, if by some miracle she gets to sit down, knitting.

Scrum is simple, just use it as is!!

By Ken Schwaber

[**Note:** grayed text indicates hypertext links in original article.]

Scrum is a mindset, an approach to turning complex, chaotic problems into something that can be used. Jeff Sutherland and I based it on these pillars:

1. Small, self-organizing, self-managing teams;
2. Lean principles; and,
3. Empiricism, using frequent inspection and adaptation to guide the work of the teams to the most successful outcome possible.

The Scrum Guide is a body of knowledge that explicitly defines what Scrum is (and, by default, what it isn't). The Scrum Guide doesn't tell you how to use Scrum, how to implement Scrum, or how to build products with Scrum.

People learned what Scrum was and how to use it by going to courses, conferences, reading books and blogs, etc., but primarily by trying to create useful things from visions, concepts, and desires using their understanding of Scrum. As they went at it, Scrum started to make sense. Scrum helped them manage outcomes, But… When people tried to use Scrum, they learned that the difficulty of Scrum was getting a shared understanding of what was desired, what was possible, what their skills would allow them to create, and to work together to do their best.

In 2009, I recognized we had broken the waterfall mold. People understood — largely — that our "agile, lightweight" approach worked and was appropriate for the emerging complexity in the world. However, just like the telephone-tag game, there were many interpretations of Scrum… Sometimes this was because poor communications, inadequate mentoring, and other commercial reasons. People who felt that Scrum would tell them how to build a product to solve their needs felt that Scrum was weak because Scrum didn't explicitly tell them how.

Exactly. As I've often said, Scrum is easy. Solving problems with Scrum is very hard.

So …. In 2009, when I founded Scrum.org I wrote a definition of Scrum. This was short, but retained all of Jeff's and my important thinking and learnings. I made sure that it retained its identify as a framework and eschewed inclusion of opinions, context-dependent practices, and anything that restrained it to only certain applications. A framework, not a methodology.

This was the first Scrum Guide, and it was the definitive body of knowledge. Anything not in the Guide, or contrary to the guide was not Scrum.

I created some assessments that helped people test their understanding of Scrum anonymously and for free. The initial results were scores below 40 percent correct. As people went back to the Scrum Guide and studied, these scores rapidly improved.

I wrote the Scrum Guide, and Jeff Sutherland then joined me to refine, sustain, and maintain it, so that:

1. Courses could be developed based on what Scrum was, not something else.
2. People who taught courses would have a solid foundation to stand on.
3. We could develop assessments to test whether a person knew Scrum and how to use it to solve their problems.
4. Anyone could evaluate their understanding of Scrum and whether what they had been taught or told conformed.

etc.

The Scrum Guide was created from Jeff's and my work, and the work of everyone else that had tried to use Scrum. It has been adjusted by us since then. The Scrum Guide has no commercial purpose other than to offer a litmus test of what Scrum is.

Jeff and I maintain the Scrum Guide at **https://www.scrumguides.org.** We are indebted to the people who have translated the Guide and to those who help us sustain it.

REMEMBER: Scrum is simple. Stop worrying about polishing it up so it is perfect, because it never will be. Anyway, there are far too many complex, chaotic situations in our world that you are skilled to help others address. We do not need to waste our time staring at our belly-buttons.

As the Kingston Trio famously sang:

The Merry Minuet

They're rioting in Africa
They're starving in Spain
There's hurricanes in Florida
And Texas needs rain
The whole world is festering with unhappy souls
The French hate the Germans, the Germans hate the Poles
Italians hate Yugoslavs, South Africans hate the Dutch
And I don't like anybody very much!!

But we can be tranquil and thankful and proud
For man's been endowed with a mushroom-shaped cloud
And we know for certain that some lovely day
Someone will set the spark off
And we will all be blown away!!

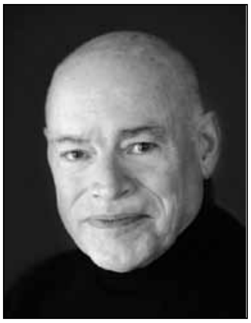
They're rioting in Africa
There's strife in Iran
What nature doesn't do to us
Will be done by our fellow man

Scrum On … Ken

⁂

To read this article online with embedded hypertext links, go here:
https://kenschwaber.wordpress.com/2017/11/11/scrum-is-simple-just-use-it-as-is/

About Ken Schwaber



Ken co-developed the Scrum framework with Jeff Sutherland in the early 1990s to help organizations struggling with complex development projects. One of the signatories to the Agile Manifesto in 2001, he subsequently founded the Agile Alliance and Scrum Alliance. He founded Scrum.org in 2009 in order to execute on his mission of improving the profession of software delivery. A 30-year veteran of the software development industry (from bottle washer to boss), he has written four books about Scrum: Agile Software Development with Scrum, Agile Project Management with Scrum, The Enterprise and Scrum, and Software in 30 Days. He lives in Lexington, Massachusetts.

Managing Culture Risk: A Matter of FLOW

By Hadyn Shaughnessy

[**Note:** grayed text indicates hypertext links in original article.]

Amid accelerated, technology-driven change, the key to good conduct and continuous performance improvement is more cultural than technological.

Today's enterprises need to innovate more quickly than ever, but they need to do so in a more delegated manner. These new facets, speed and deeper delegation, introduce more risk to firms as they transition to digital work. A recent Deloitte white paper, **Managing Conduct Risk,** identified some of the challenges of this faster innovation cadence.

Eight potential sources of conduct risk were identified. Among them: innovation and product development is not guided by customer needs; performance is not being judged in a balanced way, or individuals are not held accountable for poor conduct. And the authors propose a variety of technologies that could help to establish proper oversight.

An assumption behind conduct risk is that it arises out of dysfunctional conditions, or processes that are not functioning well. While that is probably the case, there is a problem with trying to address it through technology. The very rapid cadence of change we are seeing calls for a deep cultural change anyway. It cannot be achieved within traditional corporate cultures. What that means is that it entails quite a different kind of conduct to begin with, very different relationships between people, new forms of leadership and new work processes.

My argument here will be that the focus of attention should be on techniques that form appropriate conduct, rather than technologies that compensate for poor process and culture.

Innovation and Behavior Risk

For rapid innovation to take place, people need to feel as though they are in a supportive environment where failure-risk is tolerated. An over-regulated and over-supervised environment cuts against the grain of what we know about high performance. Management literature is now replete with **references to the right to fail.**

Less prominent in the literature, but well known in IT circles, is that innovation is accelerating to levels unimagined only five years ago. Excellence in IT would set the bar for innovation delivery at 20+ times per day, that is, a minimum of 20 updates per day to key platforms, as opposed to 20 updates per year. Some companies now talk about automating innovation to the point where thousands of updates take place each day.

One reason for this new pace of change is the move away from large monolithic software to what are called microservices, small software packages that communicate with each other. Microservices can be added or pulled out of a loosely coupled architecture at any point in time. They introduce a new flexibility to what a business can plan to do.

This change is accompanied by a shift away from sequential handovers in software. Why is this important? Development projects used to begin with a requirements document that was passed to developer teams and onto testing teams and then to operational teams. In the new "DevOps" paradigm, as much as possible in software development takes place within multidisciplinary teams with no handover. Handover risk is reduced, and so too the old risks associated with software integration.

The result of these changes is that Innovation has become continuous, hence the terms *continuous delivery* and *continuous integration.* Along with that, the need for *continuous learning* and *continuous process design* is also emerging.

The trend towards integrating teams in IT is extending to integration across the business. So not just holistic IT teams, but also teams that incorporate staff from data analytics, customer resource management, billing, and sales.

New Culture of Work

There is a new work culture that goes along with this more holistic, continuous innovation credo. New work cultures are visible at companies such as Netflix, which pioneered much of the process innovation; ETSY, the craft selling platform; and SkyScanner, the travel platform. And of course, companies like Google and Facebook embrace fast paced innovation.

You might say, well, these are all high-tech platform companies and they behave differently. Rapid change is in their DNA. Yes, they are, and yes, it is. They are also companies that have an eye on scale economics (pushing beyond the concept of diminishing marginal returns). But it might surprise you to know that one of the pioneers of this new culture is Aviva, a 300-year-old insurance company.

In a forthcoming book, FLOW, Aviva's international chief information officer **Fin Goulding** and I describe the cultural underpinnings of continuous innovation and how it can be captured by any company.

The reason we used the word FLOW to describe this culture needs elucidating. FLOW reflects the fact that these companies have a culture that drives continuous change. There is not a before and after of digital transformation; there is no end goal where these companies reach a digital culture and can put their feet up.

They continuously move new ideas through the organization, teasing out the value for customers, prioritizing the ones that should go into production first, meanwhile making every effort to ensure that every project delivers new value quickly.

Total Involvement

The idea of FLOW is not just that a mass of work flows through the organization without disruptive handovers, or that the size of workpackage makes daily delivery of innovation possible. It also reflects the fact that high paced innovation cannot be managed by a leadership team. Everybody engaged in the work has to be a participant in managing the process by bringing their own unique knowledge and experience to daily judgments about value and technique.

Until FLOW-like cultures began to develop, there was a feeling among business strategists that the modern economy left very few avenues for gaining and sustaining competitive advantage. One writer even titled her book **The End of Competitive Advantage.**

Continuous innovation is a competitive advantage, but it calls for significant cultural change. In a globally competitive economy, anything less is risky. Customers have simple switching mechanisms and don't tolerate poor service for long. Conversely, finding ways to improve customer satisfaction is a never-ending challenge for a very simple reason: the power of segmentation.

Start-up Thinking and Market Segmentation

Since the dawn of social media, many firms have been able to augment their market segmentation (traditionally based on demographics, salary levels and geography) with more cultural characteristics drawn from social behavior online. They developed customer personas, in effect caricatures of different cultural types that they were serving.

The reality is that companies like Amazon have long demonstrated the need for extreme segmentation, or what Chris Anderson has called **The Long Tail.** Companies need what I have **called elsewhere new economies of scope.** If you cannot provide scope, then a start-up somewhere, or a nimbler competitor, will spot the holes in your offer and move in with a segment buster.

In financial services, this has happened in remittances, working capital and payments, and even in the basic notion of what the term "currency" actually signifies.

What is common to financial services challengers is that they have identified a market segment that they can serve more cheaply and efficiently than the incumbent. The business models of these new companies are far from complete and often not very exacting, but that's an entrepreneurial failing. The potential to displace financial services companies is very much feasible because incumbents are weak at granular market segmentation and economies of scope (and often also lack the right kind of scale).

Innovating in the FLOW

In many firms, there is a sense that the agility they need in order to compete with more tech-oriented companies could come from Agile methods that began in software development.

Agile, ostensibly, promises a faster way to get things done. The trouble is that Agile does not necessarily get the right things done. It can easily over-commit teams to inappropriate projects and can take on the form of a continuous inquisition into who is performing well and badly, in some cases actually causing precisely the issues Deloitte draws attention to.

In other words, poor conduct is often a consequence of inappropriate process. However much you may wish to monitor and supervise that, it makes more sense to tackle the root of the problem, rather than add an overhead cost to it.

There is a technical reason why Agile actually works against agility and can also promote poor conduct. Agile delivery cycles can vary from 20 to 80 days. Agile projects are also divided among different teams. That can mean, say, five teams, on a three- to 12-week development cycle, each delivering at different times. This is partly a "cycle time" problem, and it has all kinds of consequences.

In FLOW projects, the software delivery cycle tends to be around 24 hours, whatever the project. That means each day, a project is developing multiple packages of finished, tested code, ready for deployment. The same discipline can be developed in any area of business of course.

This kind of innovation cadence is going to be the norm, as more companies switch their focus to real value creation.

FLOW represents work processes that continuously clarify and respond to issues of value, performance and accountability. In place of technology it draws on a different tradition – the use of visible work in knowledge-rich environments.

Visible Work

Less than a decade ago, a small group of analysts drew attention to one of the primary risks of knowledge work. Prior to digitization (and work-from-home), **most work was visible or observable.** With digital knowledge work, work becomes invisible. It resides in people's heads, occasionally to be shared in meeting rooms.

To produce something as simple as a presentation used to be a visible, shared process. A scientist or engineer might sketch out some initial ideas and pass these to a graphic artist who would create a first pass of a slide deck. The two would iterate back and forth, sharing ideas and background, drawing in a colleague or two for advice, drawing out meaning together as they built a good way to communicate intricate knowledge. The process was interactive, iterative and visible. Therein are some clues to good culture.

Today it is more likely that somebody creating a deck will just do it in PowerPoint. The whole process of sharing information and background in order to clarify and express ideas is skipped over. Because it is so isolating, nobody learns from the process. The bulk of knowledge stays in one person's head.

Visible work was all too briefly an Internet meme, along with "working out loud" and "radical transparency". But these ideas are now infused into FLOW-like processes.

FLOW requires every element of work to be drawn out on a series of walls. These walls document the core objectives of the firm or department, the requirements of customers, the projects that will deliver value, the risks and issues arising with each project, their interdependencies, the work breakdown that makes tasks easily deliverable within 24 hours, appraisals of colleagues left in open view.

So far we have seen Customer Walls, Customer Feedback Walls, Executive Portfolio Walls, Project Walls, Team Kanban Walls, Risks and Issues Walls and many more. Here is an example of an Executive Portfolio Wall:

When all work is visualized, good conduct becomes good culture and vice versa. The peer group itself becomes the monitoring and supervisory mechanism.

That is not to say it diminishes the role of risk management. In fact, understanding the interdependencies of projects within a firm is a key enterprise risk skillset. So too are tasks like ensuring all work is broken down according to some value-metric that the firm buys into; making sure that Risks and Issues visualizations are fully fleshed out; monitoring responses to customer issues.

Conclusion

The idea of FLOW is to create the conversations that create a good culture, one that constantly aspires to improvement.

It is difficult to argue for culture as a solution (everybody sees it as a problem!). However, the alternative of layering on technology costs risks alienating staff and adding to cost. It makes behavioral monitoring the objective when the objective should be good a culture that aspires to do better and is self-policing because everyone's objective is customer value.

No technology will create good behavior. Talking more, working out loud, making everything visible, will.

***

To read this article online with embedded hypertext links, go here:
http://blog.prettyagile.com.au/2017/01/facilitating-team-self-selection-safe-art.html

About Hadyn Shaughnessy

Haydn has a thirty year background in product development, innovation and strategy where his passion has been to help people "think different". He wants to help developers think like business people and business people to create value from code without developer intervention. He co-authored *Flow: A Handbook For Change-Makers* with Fin Goulding; and *12 Steps to Flow* and developed the first accredited Flow Academy training course. The objective is to create truly multidisciplinary people for multidisciplinary teams. Flow is beginning to gain traction globally as a way to capture the real spirit of agile and to cultivate an agile business. Haydn and Fin take a holistic view of roles, believing we should all be able to adapt to new roles quickly and be capable of devising processes in real time to get work done in the best possible way. Originally from a documentary background he has a long term interest in the use of storyboards to visualise projects.

Innovation: Best Practice for Product Leaders

By Salma El-Shurafa

[**Note:** grayed text indicates hypertext links in original article.]

In a world where the pace of change is constantly increasing and becoming more complex, can you really afford to stay still?

Living and working in the United Arab Emirates, with its blindingly rich cities and unbelievably luxe lifestyles, I see that the value of innovation is obvious. After all, how does this crazy-wealthy region plan for the future? By investing in innovation as a matter of both visionary thinking and as a survival tactic.

Other Arab nations may have simply banked on their rich oil reserves, but Abu Dhabi and Dubai diversified to **international tourism, business conferences and global trade.** The government launched a National Innovation Strategy, a seven-year plan developed to make the UAE among the most innovative nations in the world. Innovation is seen as key to meaningful **social and economic development** in the region. According to UAE Prime Minister Sheikh Mohammed bin Rashid Al Maktoum: "The competitiveness race demands a constant flow of new ideas, as well as innovative leadership using different methods and tools to direct the change."

You don't have to be a national leader to weigh up the challenge and opportunities of innovative thinking and doing. Project managers and business leaders are increasingly seeing innovation as an absolute need and not just a nice add-on.

But change – which is inevitable with innovation – is never easy. How can project managers and leaders make innovation integral to their corporate ecosystem? Based on our coaching work, the following are some **leadership best practices** that can help strengthen a team's capacity for innovation, which is a non-negotiable in effective product management today.

You are a Change Instigator

Innovative leadership means making it your duty to act proactively as opposed to simply reacting to events. This approach is not sustainable, no matter how correct or smart your responses may be. In this uncertain world, merely being great at adapting to change is not sufficient anymore. Instead, you want to be able to shape the future so you can lead and create that change.

You Champion Collaboration

Even the most creative ideas won't pave the way to innovation when it's being driven only by an individual or two. For innovation to happen, the entire team must work together with a common direction and sense of purpose. And this is possible only when the leader knows how to establish the spirit of collaboration and cooperation rather than individual competition.

You Strive for a Strategic and Purposeful Approach

Change per se is meaningless if it lacks purpose and direction. Many make the mistake that if they are liberal and open to new ideas, then they're already being innovative. But innovation requires strategy and a system. One solid solution that is aligned with your overarching objectives and relevant to your project or business as a whole is better than a number of brilliant but random ideas.

You Value Diversity

Innovative leaders don't just accept or embrace diversity; they understand its great value. After all, the workforce today has become more diverse in generation, location, culture and beliefs than any other time in history. Having varied perspectives in your team is a huge challenge, but also one of your greatest assets for building innovation. If you know how to establish unified and collaborative work behaviours amidst diversity, then your team is all set for the future.

You Work to Sharpen Your Self-Awareness

As clichéd as it may sound, it really and truly all starts with you. How can you instil the significance of change and failure when your employees see your aversion to risk and fear of failure? How can people grow and work together on their ideas if they don't have the time, space and resources to do it?

Taking the Leap

Admittedly, it's incredibly hard — even the most courageous leader will have hesitations and misgivings whenever they take a leap toward something new and untested, no matter how promising it may be.

When you're in charge of a project (or an entire business venture), you're not just in charge of a mission. You're in charge of people. Your people. The more you care about their development, careers and livelihood, the more it can be unnerving to drive them toward unpaved paths, which is what innovation needs you to do. And that's where the value of self-awareness comes in. When you have a solid sense of self and clarity in your purpose, taking a leap doesn't just become less scary. The act also makes much more sense and meaning.

***

To read this article online with embedded hypertext links, go here:
https://www.mindtheproduct.com/2017/02/leading-the-way-to-innovation-2/

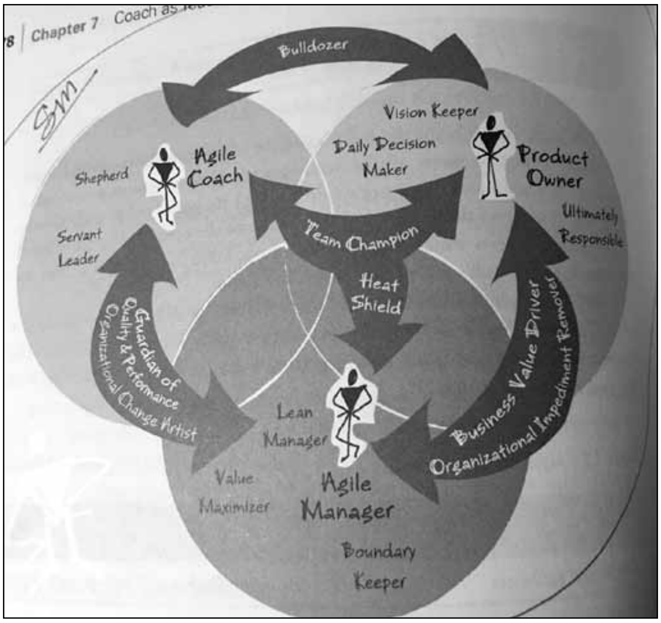
About Salma El-Shurafa

Salma supports leaders on their journey to self-discovery and reaching their full potential. She has coached hundreds of professionals, teams and groups across the Middle East, Asia and Europe.. She worked with Fortune 500 companies, government agencies, and leading organizations in the region. Salma's passion is to empower and challenge leaders by creating a safe space for them to break their own boundaries and facilitate the discovery of their inner leader and the path to get there. Salma's multi-cultural background helps her capitalize on a global understanding of professional & leadership development. She designs programs and interventions for organizations to develop and support leaders in channeling their energy towards creating transformational change on an individual and collective level, while moving towards a common vision that creates a positive impact on the wider organization. Salma is a Professional Certified Coach by ICF and a Certified Professional Co-active Coach from the Coaching Training Institute (CTI). She a Faculty Member at CTI, where she trains and supervises future generations of coaches.

The Power of Interlocking Roles

by Cherie Silas

[**Note:** grayed text indicates hypertext links in original article.]



So, now you know two things about me. I write in my books and I can't draw. I snapped a shot of this image from the **Coaching Agile Teams** book — Chapter 7, (**Lyssa Adkins)** because it is an amazing way to portray how the role of the scrum master, product owner, and agile manager work together. Too often I see coaches running off managers and basically telling them that they no longer have a job. Managers are seen as the enemy of Agile. It doesn't have to be that way. It shouldn't be that way. We should be teaching managers what their new, even more powerful, role is!

This picture really stirred me up because it put into writing questions I have often had explaining to the scrum master and agile manager. There are also pieces of this that validate things I've always instinctively known but didn't have anything but my gut to tell me it was true.

I love that the overlap in these roles is there by design. It's not a place to struggle for control — it's a place to partner for power! The scrum master intentionally shares the bulldozer of impediments function with the product owner because the person with the most influence is more effective depending on the actual impediment.

The scrum master AND the manager are the guardian of quality and performance and partner also on organizational change. When I read this my first thought was, "There it is! I have a tool to help scrum masters understand that THEY are a guardian of quality and performance." This means that the scrum master very clearly has a role in ensuring that the team is getting better at delivering quality code. It's not only about ensuring they collaborate and sing Kumbaya — if they aren't improving on delivery of quality code and satisfying their customer's needs it really doesn't matter that they are collaborating and enjoying one another's company. The purpose of adopting scrum is because companies want to deliver. We've done so much focus on the people side of the scrum master that along the way we have lost the part where we must deliver. Accountability. Responsibility. These are not dirty words. They are the signs of a maturing team.

The manager is a value maximizer and a partner with the product owner in driving business value. We can't tell managers to step back and be completely uninvolved with the team. They are a partner to help ensure that organizational impediments aren't hindering the teams and to ensure that the team is delivering business value. There it is again ... delivering. I'm not sure why everyone has forgotten the importance of delivery. We (the industry / agile coaches) have been so focused on creating self organization that we have stepped to the other edge to a place where responsibility and accountability are foreign to agile teams. They believe they should be left alone to do whatever they want and no one can tell them anything because managers aren't supposed to be managers. How interesting how this anti-pattern has developed over the past 15 years. Ken and Jeff must be a bit frustrated with "agile coaches" and the damage that has come from those who really don't now what they are doing.

The last thing on this that was so powerful to me was that triple responsibility of being a champion for the team's success and being a heat shield keeping things that distract them from DELIVERING from creating churn with the team. The reason this resonated with me so much is because I have seen scrum masters take the "protect the team" message as, "protect the team from managers" "protect the team from responsibility for their own actions or lack of action" "protect the team from the product owner." The truth is that if we focused more on protecting the team from themselves and stopped colluding with them they would become high performing much sooner.

Wow, there's a lot of passion in here. It's true. I'm passionate about this because I want to see people become successful. People who do not willingly take responsibility and stand accountable for their actions rarely experience true or lasting success. As a coach, I love my clients too much to leave them that way.

Happy trails …

⁂

To read this article online with embedded hypertext links, go here:
https://tandemcoachingacademy.com/2017/05/20/the-power-of-interlocking-roles/

About Cherie Silas

Cherie is a Scrum Alliance, Certified Enterprise Coach (CEC) and an ICF Professional Certified Coach (PCC). She has a strong desire to help people arrive at the place they define as success in both personal and professional life. Her goal is to invest the experience and talents she has gathered through years of learning, often times the hard way, into people whom she hopes will become greater than she can ever dream to be.

Her main focus is culture transformation and development of people in Agile roles. When not coaching agile organizations, Cherie can be found coaching individuals, executives, employees, and rising leaders of non-profit organizations. She also teaches professional coaching to people in Agile careers who want to earn ICF credentials and add coaching to their toolkit. Cherie has background training from CTI and ORSC and has a passion to bringing professional coaching into the Agile world.

Cherie's life mission that drives every interaction with every individual she encounters is simply this: To leave you better than I found you with each encounter

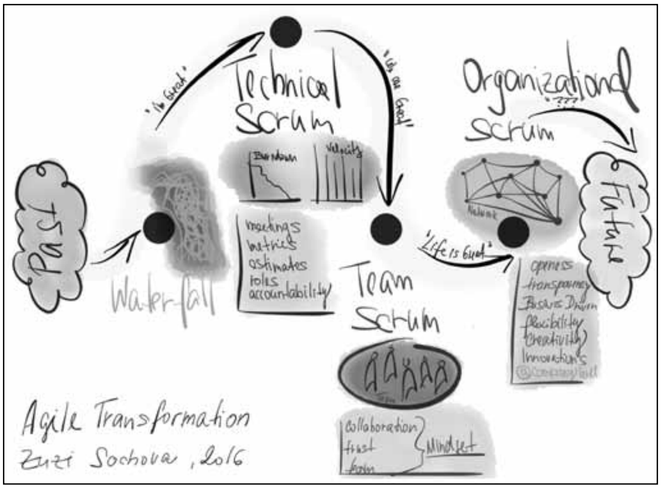
https://tandemcoachingacademy.com

Scrum Transformation Journey

By Zuzi Sochova

As one of the CSTs — Certified Scrum Trainers — I've got a unique opportunity to travel around the world during the last two years and teach Scrum at a variety of businesses, organizational environments, and very different cultures. I must admit that Scrum is awesome as it is universal. You can apply it to software, hardware, marketing, HR, executive teams and be rapidly successful, significantly better, change the way of work and become the best of the greatest. The flip side of the coin is, that despite the easy way how Scrum is defined, there are still companies, teams and individuals completely failing to understand what Scrum is and therefore failing to implement it.

I draw this picture to illustrate that becoming Scrum is a journey. You can't just do Scrum, you have to embrace it. You have to become Scrum yourself first. It's often not that straightforward as we've been got used to the traditional processes throughout the history, but at the same time, this is the very best strength of Scrum. Once you master it, it becomes the part of your life. It's not just a process, it is a way of living.



Technical Scrum

First, let me say explicitly that "Technical Scrum" is not Scrum. It only pretends to be Scrum. It's a camouflage. However, it might be the necessary first step in certain organizations to move to the real Scrum. How do you recognize Technical Scrum? People "do" Scrum. They are looking for ways how to remain the same as they used to be. They are eager to get checklists of practices which need to be done, in order to do proper Scrum. Therefore Technical Scrum is all about estimations techniques, burn-downs, measuring velocity. The very important metric would be individual utilization, so they usually insist on time task estimates, capacity calculations, and timesheets to be filled. They have identified new roles, but in reality, they just renamed the traditional roles and didn't change the behavior. Scrum meetings are usually long and felt redundant. Managers use Scrum to micromanage. The overall team focus is on "how". The team is not any team but a group of individuals working on similar items. The individual accountability matters. They are looking how to split responsibilities instead of how to collaborate to achieve the goal. Product Backlog is usually a to-do list where most things have to be done.

Scrum Mindset

In the real Scrum, your team understands the mindset and they are "living" Scrum. They take it as the way how to focus on customers, how to innovate, how to collaborate. The estimates, efficiency, and utilization become quite unimportant, as they focus on delivering value to the customer and overall long term results. The first step here is usually "Team Scrum" where the development team becomes a real self-organized and cross-functional team which works together. The team creation process produces a huge trust internally among the team members but also externally to the organization. It's the first tiny 'snowball' which afterward starts the whole transformation and creates forces to change how we run our business and how the organization itself is structured.

The "Organizational Scrum" builds on top of the values we experienced at team level — openness, transparency, and trust which leads the organization to be more business driven, flexible, and open to innovations. The business slowly starts to be picking up and the organization has to follow the rest. At this time, the snowball is big enough to attract the rest of the organization. At that time, you are truly Agile.

Such transformation can take years. It's not uncommon that companies are falling back and restarting the whole initiative again. It's hard. To succeed you need a good reason for change and courage. Eventually, every company has to change as the word is getting more complex and fast. The same way as industry revolution changed the way we were hundred years ago, the complexity of our current life is changing us now. To succeed in a long term, we have to be more flexible and dynamic — more Agile.

⁂

To read this article online with embedded hypertext links, go here:
https://agile-scrum.com/2017/02/23/scrum-transformation-journey/

About Zuzi Šochová

Zuzana “Zuzi” is an independent Agile coach and trainer and a Certified Scrum Trainer (CST) with more than fifteen years of experience in the IT industry. She started with agile and Scrum back in 2005, when she was implementing agile methods in the USA. From that time, she has been credited with agile transformation and implementation for many companies and teams around the world. By creating and sustaining agile leadership, Zuzi believes the worlds of work and life can be made happier and more successful.

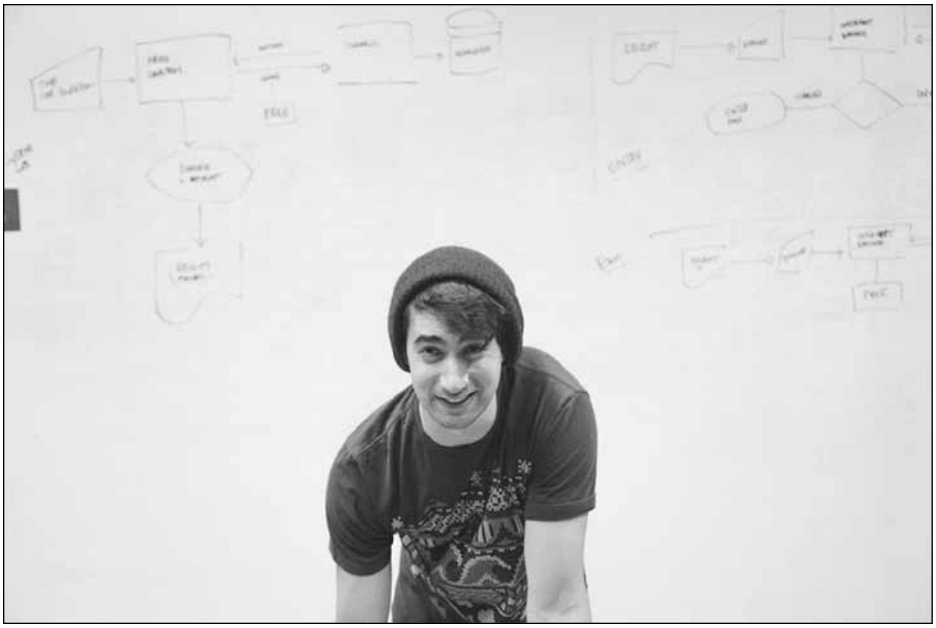
Zuzi is a founder of the Czech Agile Association, and the world famous Agile Prague conference.

Zuzi was elected to the Scrum Alliance Board of Directors in 2017. She is an author of *The Great Scrum Master: #ScrumMasterWay* book (Addison-Wesley Signature Series (Cohn)).

Her website is sochova.com and blogsite is agile-scrum.com.

The Scrum Task Board and the Self-Managing Team

By James Sywilok



In the early days of Scrum, the quickest way to locate a Scrum team's work area was to look for the task board, which was usually mounted on a nearby wall. Work was managed using index cards, sharpies and spreadsheets, and the task board served as a tool for tracking work as well as an information radiator.

Anybody walking by could simply look at the task board and see the team's progress at that point in time without having to ask a single question.

However, what inevitably happens in nearly every field is that new technology and tools are developed over time with the intention of "making it easier" to manage work, and the world of agile is no different. Some tools were built from the ground up to manage agile project work, while others were developed as add-ons to existing tools.

When an agile project is just beginning, it seems like the first question asked is always “What agile tool are we going to use?” Let’s face it, we in the IT industry love our tools, and I am no exception.

However, the technology we perceive as progress can sometimes have unintended consequences. Take, for instance, society’s extensive use of social media, texting, and other technological forms of communication. They were originally created to save time and effort, but we are only now discovering that these tools can lead to a sense of social isolation in certain segments of the population.

High-Tech Tools: More Harm Than Help?

So, what does this have to with Scrum teams? A Scrum team’s success is all about collaboration, which in turn is all about co-location and face-to-face communication. While technology can certainly enhance a distributed Scrum team’s collaboration, it also has the potential to hinder a co-located team: if the team relies too heavily on technology, it can start to act as an inadequate substitute for face-to-face communication and collaboration.

For example, I was working with two Scrum teams over the course of many sprints and, while all their information was readily available in a high-tech agile tool, I rarely saw it displayed on anyone’s screen. I also noticed that their stand-ups were functioning as more of a status report than an opportunity for the team to share information and level-set the team’s progress in the sprint.

Although the team reported a high level of confidence in completing stories during the mid-sprint, I could see from the story point burn-down chart that they were scrambling to complete stories in the later stages of the sprint. I knew that all the team members were solid professionals, so their work ethic clearly wasn’t the problem.

Eventually, I realized that, while they may have been focused as individuals, they weren’t focused as a team. I also realized that the unintended consequence of technology was that the team’s most crucial information was buried in a tool that no one bothered to access.

A Low-Tech Solution

Since I didn’t have two 70-inch monitors to put in the team rooms, I decided to go old-school. So, the next day I came in with painter’s tape and put a task board on the wall. I then printed out the stories and tasks from our agile tool and recreated the task board to reflect the status of the sprint.

I told the team that, during the sprint stand-up, each team member would go to the task board to address the team. I also told them to focus on the team and ignore anybody else in the room, and that each time they spoke about a specific piece of work they would need to move the corresponding tasks on the board to the appropriate columns as well.

It took some time for them to get comfortable with doing the stand-up in this way, but the result was that the task board started to provide them with the focus they needed as a team. It had a constant presence, easily showed the team’s progress and gave each team member the satisfaction of physically moving their work across the board from the "to-do" column to the "done" column.

During the mid-sprint checks, the accuracy of the team's confidence level vote increased dramatically. And, when a mid-sprint check indicated that the team might have a problem, they used the task board to determine how to resolve the problem and re-allocate resources accordingly. For these teams, as well as many others, the task board quickly became their primary tool for self-managing.

The Value of Planning

I always tell my teams that the most important aspect of sprint planning is not the plan itself but the fact that they engaged in the act of planning in the first place. This is because the act of planning gives the team a shared understanding of what must be accomplished.

And, given that things rarely go according to plan, we must constantly re-plan "in light of what we know now," and every team member should be fully aware of the changes in the revised plan. With the help of a humble task board, teams can easily collaborate, re-plan and focus for the duration of a sprint, and that's the sign of a truly effective agile tool.

***

To read this article online with embedded hypertext links, go here:
https://www.frontrowagile.com/blog/posts/105-the-scrum-task-board-and-the-self-managing-teamart.html

About James Sywilok

As an IT professional with over 25 years of progressive technical and business experience, **James** has had the rare opportunity to work on both sides of the IT industry prior to becoming a Certified Agile Coach. This has given James a unique insight into my chosen industry. James has worked on or managed several infrastructure and software development projects over the years and has seen firsthand the subtle disconnect within companies between their technical side and the business side which has often led to frustration and mistrust on both sides.

For this reason, James believes that Agile methodologies such as XP, Scrum, Kanban, SAFe, and others when implemented properly can provide a means to better align the technical-business relationship within the company and company-client relationships.

Given today's pace of business and technological improvements, James believes we really have no alternative but to adapt the way we work to account for these rapid changes in our environment or fall to the wayside.

3 Skills for an ACE ScrumMaster

By Christine Thompson

[**Note:** grayed text indicates hypertext links in original article.]

As a ScrumMaster, I must unconditionally support my team members, but what transferrable skills can I borrow from elsewhere to help me do this?

For some of my teams, when we hold a retrospective, I display the **Retrospective Prime Directive** to remind them about the mindset from which we are approaching our discussion:

Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.

While looking at this statement one day, it struck me that this sentiment applies to more than just a retrospective. A key role for the ScrumMaster, in conjunction with the line manager, is the pastoral care of the members of the team. As a ScrumMaster, I must unconditionally support my team members. This does not mean that I agree with everything that they do or support any kind of behavior they might display, but it means that I unconditionally support them to grow and develop as people and professionals. Everyone makes mistakes (especially me!) and everyone has room for improvement, *but I have to truly believe that everyone does the best job they can, given what they know, and their skills and abilities at that time.*

The parallels with parenting are obvious. I support my children unconditionally. I truly believe that they want to do well and I want to support them in a positive and constructive way. This reminds me of a parenting course that I went on and the skills that they advocated using to better support children. The course was entitled Attachment Focused Parenting with **PACE** and taught four key skills for parenting: Playfulness, Acceptance, Curiosity, and Empathy. The latter three of these are surely transferrable to our relationships at work.

Acceptance

A ScrumMaster must accept people as they are and accept the way that they feel, including all the ups and downs. As with remembering the Retrospective Prime Directive, a ScrumMaster must always assume that a team member is doing the best they can given *their skills and abilities, the resources available, and the situation at hand.* This acceptance must be true and unconditional. This doesn't mean accepting inappropriate actions or behaviors, but it does mean accepting the person and truly believing that they are doing their best given the circumstances. From this position, the ScrumMaster can support the individual to grow as they need to, by understanding, coaching, and encouraging them.

Curiosity

A ScrumMaster must always be asking, "Why?" But it must be a nonjudgmental "Why?" A curious "Why?" Thinking about what's behind things, looking for reasons and causes and not just taking things at face value. By understanding the feelings and reasons behind a reaction, a ScrumMaster can better help the individual take responsibility for their feelings and actions and to inspect and adapt themselves appropriately. The team needs to know that their ScrumMaster is on their side, and this needs to be handled delicately, never with finger-pointing.

Empathy

This requires the ScrumMaster to put him- or herself in the person's shoes and feel what they are feeling, so they can better understand where the person is coming from. It's essential that a ScrumMaster spend time acknowledging and reflecting with the individual to let them know that they understand and empathize with them and want to help them. This goes back to the point of unconditional support. Whatever is going on for them, their ScrumMaster understands and wants to help them move forward.

Here's a scenario that I experienced recently: I had a team member contact me to say that she didn't feel able to attend the retrospective the following day. My first thought was to tell her that we needed her there as part of the team and to ask her to please attend. But then I stopped for a moment and, instead of making an immediate judgement, I thought about what she was saying. She didn't feel able to come. I needed to accept that this was how she felt and not just try to talk her out of it. Once I'd accepted this, I could try to understand why. I was curious about the reasons behind this and asked her what had happened to make her feel like this. That gave me the opportunity to understand how she was feeling and empathize with her. Again, I accepted her concerns and what had happened to make her feel this way and reassured her of my ongoing support. At no point did I tell her she had to come to the retrospective, I only accepted what she had said, asked why she felt that way, listened to her answer, and told her that I understood how she felt. The following day, she came to the retrospective, entirely of her own accord.

So, as well as having a Prime Directive for retrospectives, I like to think about a Prime Directive for pastoral care, which says:

Regardless of the behaviors I see, I understand and truly believe that everyone *is doing the best job they can, given their skills and abilities, the resources available, and the situation at hand.*

And knowing this, as ScrumMaster, I will back my team members unconditionally. I will accept them, be curious about what is going on for them, and show empathy for them. I will support them to understand themselves better and to grow and develop, both individually and professionally. I hope that will help me on my journey to become an ACE ScrumMaster.

⁂

To read this article online with embedded hypertext links, go here:
https://www.scrumalliance.org/community/articles/2017/april/3-skills-for-an-ace-scrummaster

Building Trust Safely at Work

By Christine Thompson

What is the downside of trust?

In November 2014, I attended an Agile conference in London at which I heard one of the speakers talk about bringing your whole self to work. This was an entirely new concept for me. She had suffered from a mental illness but had found it relieved her stress as she returned to work when she shared this information with her colleagues. She suggested that letting people know about your whole self means that you do not have to put on an act at work, and that enables people to make allowances for you as necessary.

I decided to try this approach. After all, trust is commonly listed among the core principles of Agile. Until then, I had always kept myself very private from my colleagues at work and never shared very much about my home life or my issues and problems. I had felt that being at work meant being professional, which meant holding back anything personal from my colleagues. This newfound freedom enabled me to let people know what issues I had and certainly meant that they could be sympathetic and make allowances for me whenever it was needed. I liked it.

Some years later, I found out the hard way that I had shared so much information with my colleagues that I had actually begun to make myself vulnerable to them, especially where I had misjudged the safety of my relationships. It was at this point that I learned about vulnerable trust.

Depending on your dictionary, trust is defined more or less as, "The belief that someone is good and honest and will not harm you, or that something is safe and reliable."

Vulnerability is therefore inherent in trust. Trust requires us to become vulnerable to others. If you believe that someone will not harm you, or that something is safe, then you make yourself vulnerable as soon as you rely on this expectation. There is always that possibility that you may be let down. If you misplace your trust, then you may be hurt by the person or thing that you relied upon and you experience the manifestation of that vulnerability.

Back to my example of sharing personal information with colleagues and opening yourself up to them at work: Every time you trust someone, you make yourself a little more vulnerable. With each new snippet of information that you share about yourself, you are extending that trust and that vulnerability. Share the entire contents of your heart and mind, and you have taken a very great risk and made yourself very vulnerable indeed.

I experienced this myself, firsthand, with a colleague with whom I wanted to develop a close and positive working relationship. There were some tensions between us, but each time I hit one of these I believed that opening myself up and trusting further was the way to ensure that we finally reached the nirvana that I was looking for. Unfortunately, in doing so I had made myself so vulnerable to that person that when we hit a major professional disagreement one day, it had catastrophic effects. I felt betrayed and badly hurt in the extreme, because of the level of trust I'd assumed and the vulnerability that accompanied this.

I was very fortunate at this time to be given some wise words from an astute professional who understood well what I had done. She gave me a wonderful analogy to follow, which clearly demonstrated the mistake I had made. This is what I have learned about vulnerable trust and how to build trust more safely.

Think of yourself in the central stronghold of a castle. The castle has a number of outer walls with gates. Only you can control who is allowed in through these gates. Each time you trust someone, you let them through a gate into a more central part of the castle. Keep letting people through and they will eventually be in your most private and safe confines, where you will become completely vulnerable to them. Keeping them in outer courtyards of the castle limits their closeness to you and, equally, limits your vulnerability to them.

Trusting someone with something — be it a possession or some personal information — is letting them in through the gate of the next castle wall. If you don't feel that someone can be trusted, do not let them come through. Do not share that item or that thought or feeling with them. If you let them through and they demonstrate that you were right to trust them, great. You can consider letting them through the next gate. If they let you down, kick them out of the castle walls until they prove they can be trusted again.

I think of this as layered levels of trust and checkpoints that people must go through that protect you from becoming too vulnerable, unless you are sure that your trust is well placed. Of course, your partner in life has been allowed right into your inner sanctum because you have chosen to trust them completely in your relationship. Your best friend is probably there too. But where are your work colleagues? They are distributed throughout the layers of the castle courtyards, depending on the level of trust you have given them and the level of vulnerability you have risked in doing so. Letting people through your castle gates without being certain of their trustworthiness is a risky thing to do and is how I ended up so badly hurt. My trust was based solely on hope and not on the reality and experience of the relationship I wanted to improve.

Trust is a good thing. It builds relationships and it builds teams. However, it needs to be measured and verified so that it can be built without the level of risk that can cause catastrophic results if it fails. I'd taken the advice of the conference presenter to the extreme, without testing the safety of the relationships I had in place. Now the castle gates are there as checkpoints to validate that the risk is appropriate at each stage. Be ready to move people in and of out those castle gates one by one, not on a blind hope that it will be fine in the end but based on your experience of their trustworthiness to you.

***

To read this article online with embedded hypertext links, go here:
https://www.scrumalliance.org/community/articles/2017/june/building-trust-safely-at-work

Scrum Chums: The Product Owner and Scrum Master Partnership

By Christine Thompson

First published on Scrum Alliance Community Articles on 2nd Feb 2017.

During my 7+ years as a Scrum Master, I've worked with a number of different Product Owners. As I look back over them all, I notice how I considered most of them to be more than just colleagues. They became friends. How lucky I was to have been working with friends. But is this coincidence? On reflection, it strikes me what an important partnership the Scrum Master and Product Owner must form. The level that our relationship reached was merely indicative of the need to work so closely together with each other and the investment that we both made in this relationship.

The Scrum Master (SM) and Product Owner (PO) fulfill two key roles for the team. The SM nurtures the team and helps them to grow and become independent. The PO gives guidance to the team on the purpose and value of what they are doing. With this input from the two of them together, the team has the #support and guidance they need to apply their skills appropriately, produce positive results and to develop as individuals and as a group. I've seen the two roles aptly described as "leadership partners". To this end, then, the SM and PO need a solid, supportive relationship and a united intent for the team. How do they achieve this?

Constant collaboration

The PO and SM must have open and honest interactions. They need to have regular conversations about their concerns for the team. They need a shared view on progress, process and the needs of the team. They need to be regularly available to each other, as much as to the other members of the team.

Shared goals

The PO shares the vision and strategy for the product whereas the SM promotes the vision and strategy for the team. Without both of these, the team will not be effective and therefore these "leadership partners" must share their respective visions and encourage each other in the support of the team achieving its goals.

Mutual respect

The PO and SM must understand, respect and value the unique contribution that each makes to the team. They must know the purpose and value of each other's role and understand the unique way in which they deliver this. They must get to know each other and the strengths and weaknesses that each has. They should celebrate each other's capabilities and show appreciation for each other on a regular and ongoing basis.

Mutual support

The PO and SM must support each other. They need to recognise when the other needs help or encouragement and they need to challenge each other to be even better and to grow in their respective roles. The Scrum Guide describes the "Scrum Master Service to the Product Owner". Indeed, the SM supports the PO and supports the Team but nowhere does it describe who is supporting the SM! The SM needs support and encouragement too. The PO should reciprocate the support that the SM role offers to them so there is a triangular support network available across all the members of the team.

For two individuals working in such close and supportive collaboration it's not surprising that this relationship can lead to a true and valuable partnership on which the team can rely and from which the team can truly benefit.

***

To read this article online with embedded hypertext links, go here:
https://community.alfresco.com/people/cthompson2/blog/2018/09/24/scrum-chums-the-product-owner-and-scrum-master-partnership

About Christine Thompson

Christine is an Agile Coach with over 15 years' experience of Agile methodologies, including DSDM, XP, KanBan and Scrum. As a Scrum Alliance Certified Scrum Professional (CSP), she has several years' experience of Scrum Mastery, working with teams spanning the whole range of Agile maturity, both in Engineering and in non-technical arenas. She also manages, coaches and mentors Scrum Masters with varying levels of experience, leading by example to embody a true Agile mindset. Christine has a clear focus on people, coaching them to be the best they can and forming teams which are both effective and fun. She drives an Agile strategy of constant improvement to achieve reliable and predictable delivery in an inclusive environment which challenges all its members.

8 Keys to Transforming into a High-Performance Agile Team

By Uday Varma

Summary:

Following an agile process alone will not guarantee your teams will be high performers. Teams undergo various challenges while transforming into a highly productive team. This article looks at the areas where teams generally struggle in adopting agile principles and the typical root causes for those struggles, as well as eight behaviors that can help drive teams toward greater success.

In this age of digital transformation, every organization is working to build teams that produce predictable outcomes and deliver software that meets user demands and timelines. Following agile methodologies and practices has become the norm for such teams to meet these business requirements. Every business stakeholder expects their teams to exhibit high performance and frequently release working software to production.

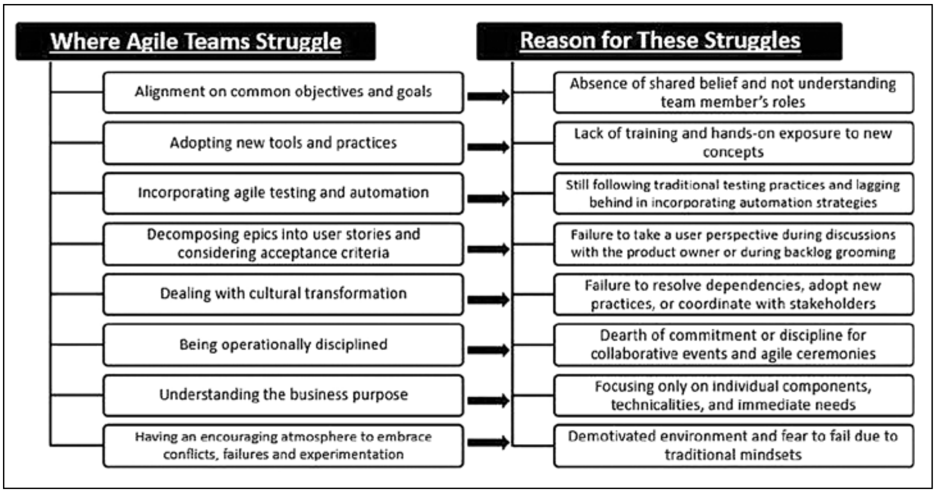
But following an agile process alone will not guarantee your teams will be high performers. Teams undergo various challenges while transforming into a highly productive team. Let's look at the areas where teams generally struggle in adopting agile principles and the typical root causes for those struggles, as well as the behaviors that can help drive teams toward greater success.

Where and Why Agile Teams Struggle

There are many areas where agile teams struggle while working to become high-performance teams. Issues with a change in culture, effectively utilizing individuals' expertise and experiences, adopting to new ways of working, socializing and collaborating with stakeholders, and understanding the business can all be challenging.

The following table depicts the typical areas where teams struggle in their journey.

(see graphic at top of next page)


Where Agile Teams Struggle
Alignment on common objectives and goals
Adopting new tools and practices
Incorporating agile testing and automation
Decomposing epics into user stories and considering acceptance criteria
Dealing with cultural transformation
Being operationally disciplined
Understanding the business purpose
Having an encouraging atmosphere to embrace conflicts, failures and experimentation
Reason for These Struggles
Absence of shared belief and not understanding team member's roles
Lack of training and hands-on exposure to new concepts
Still following traditional testing practices and lagging behind in incorporating automation strategies
Failure to take a user perspective during discussions with the product owner or during backlog grooming
Failure to resolve dependencies, adopt new practices, or coordinate with stakeholders
Dearth of commitment or discipline for collaborative events and agile ceremonies
Focusing only on individual components, technicalities, and immediate needs
Demotivated environment and fear to fail due to traditional mindsets

Let's look into each of these areas in more detail.

Alignment on Common Objectives and Goals

Absence of a shared belief in the team and a lack of understanding of each team member's role will hamper the team's speed. Roles that are still aligned with traditional functional silos will often not be on the same page as other team members and will not be effective.

Adopting New Tools and Practices

With the evolution of agility in software development activities, new tools and practices are necessary for efficiency. Application lifecycle management (ALM) tools such as JIRA and Rally, continuous integration (CI) tools such as Jenkins and Bamboo, distributed software configuration management (CM) tools such as Git and GitHub, and lightweight test automation tools such as Selenium and JUnit all become critical.

New practices such as test-driven development (TDD), behavior-driven development (BDD), and DevOps also are introduced during this transformation. A lack of formal training and hands-on experience with these agile tools and practices will often result in a team that struggles to meet its commitments and to reach its full potential.

Incorporating Agile Testing and Automation

Teams often struggle with how to build and test software in concert, as they are used to following traditional testing practices that often start testing after code is frozen and automation is only done as an afterthought. But following these traditional practices results in delayed feedback to the developers about the quality of their code, as testing gets deferred to subsequent iterations. It can also lead to delays in deploying tested features into downstream test environments.

Decomposing Epics into User Stories and Considering Acceptance Criteria

One of the most important activities in agile planning is properly decomposing epics into user stories and estimating their size. Teams often lack the focus and ability to scrutinize requirements from a user-centric perspective, resulting in ambiguous user stories that are difficult to properly implement and test within the time the team planned for.

Dealing with Cultural Transformation

While teams are doing their best to transform to agile, there are organizational and cultural aspects that impact their performance, such as resolving dependencies with other teams in the same program or portfolio, new release management processes that must be followed, coordination with multiple stakeholders on priorities and feedback, and learning new communication and interaction channels. Not addressing these cultural challenges often results in pseudo-agile behavior where agile principles are followed in name only.

Being Operationally Disciplined

Being operationally disciplined means adhering to a set of well-defined, proven, and well-thought-out processes and consistently performing them correctly. In agile, this means conducting agile ceremonies diligently, such as having periodic meetings and discussions with stakeholders for planning, user acceptance and sprint reviews, sprint retrospectives, team brainstorming sessions, and daily Scrum meetings. These collaborative activities demand a lot of commitment and discipline from the team members in order for them to be productive.

Understanding the Business Purpose

It is very important for every member of the team to understand the business purpose of what they are working on and what impact new features will have on the users of their software. Often, the tendency of a new agile team is to focus only on their individual software component or feature, the technical details in developing it, or the immediate delivery need, while ignoring the bigger picture of the project. This results in teams that veer off track, away from customer value and needs.

Having an Encouraging Atmosphere

Agile is not only about following certain practices and ceremonies or using automated tools and technologies to speed software releases. It also demands that teams have no fear of failure, can deal with lots of unknowns, and can manage and embrace conflicts. It is also about the ability to try out innovative ideas, experiment frequently, and fail fast if failure is going to happen. Lack of having a safe environment will lead to demotivated individuals who are afraid to try new practices and processes and will not produce innovative solutions.

Becoming a High-Performance Team

While coaching helps get teams on the right path, it's the team's responsibility to embrace agile principles and sustain the efficiency in their activities and effectiveness in their outcome. I have found that the following eight practices help a team become high performers.

1. Aligning with Leadership Regularly

Agile teams should have regular interactions with the program sponsors or leadership and have a common understanding of project goals. Teams should understand their role in addressing the business objectives, and the entire team should speak with a "one-voice" approach when communicating with stakeholders.

If leadership asks the team to act in a way that does not align with agile principles, it is incumbent upon the team to respond in a unified voice that what is being asked isn't acceptable agile behavior.

2. Sharing Knowledge and Experiences

Sharing new knowledge and experiences across all teams is critical to getting your entire organization up to speed as fast as possible. By actively participating in team product demonstrations, showcases, and established agile communities of practice, organizational knowledge grows much quicker than if each team attempts to learn everything on its own. Sharing experiences frequently also builds relationships among teams and increases the likelihood of effective collaboration.

3. Adopting Test-Driven Development and Behavior-Driven Development

TDD is a development practice in which low-level unit tests are used to drive successful software implementation. BDD and ATDD (acceptance test-driven development) are similar practices for specifying expected software behavior for stories and use cases using tests. All allow the business, testers, and developers to collaborate on understanding the requirements and properly building and testing the right functionality. Embedding these practices into day-to-day activities of the team not only fortifies the quality of deliverables, but helps the team reduce rework and communicate what needs to be done more clearly.

4. Defining User Stories and Requirements

The effective decomposition of epics into appropriate user stories is one of the most important activities for agile development. This not only helps provide clarity to the agile team on the requirements, but also aids them in estimating their work properly.

A proven practice for effectively breaking down epics is to use a Three Amigos approach, where representatives from the business, development, and testing have collective conversations on deriving the behavioral aspects and acceptance criteria for every user story. Your entire team should also participate in backlog grooming sessions to share their ideas and define the guidelines for a definition of "done" to determine when a user story is ready to be picked up for development.

5. Participating in Organizational Change Management

When an organization is undergoing transformational change, it is not only the responsibility of the management, but also the individual teams, to contribute positively to the process. Teams should consistently demonstrate their commitment toward achieving business goals through continuous collaboration with business stakeholders while helping to instill a high-performing agile culture in their team and overall organization. A key aspect of this commitment is delivery of promised functionality during each sprint.

6. Practicing Good Collaboration and Communication

Achieving high performance within the team and software delivery process without strong communication and collaboration will be very difficult. Team must exhibit the behavioral aspects of discipline, close collaboration, and commitment with stakeholders during iteration planning, and be open to feedback during review and retrospective meetings. Availability of high-end infrastructure, such as video conferencing, messaging systems, and other collaboration tools, at the team's workplace will help distributed teams effectively collaborate and communicate.

7. Having Systems Thinking and Mindfulness

It's very important that each team has a complete picture of the project and program within which they are working. To achieve this, teams should develop a deep understanding of business domain, business rules, enterprise architecture, and applications of client organization and align this knowledge with the software modules they are working on. As much as possible, teams should not focus on optimizing their specific aspects of a larger program, but help the entire program optimize its efficiency.

8. Generating a Positive and Energizing Work Culture within the Team

If team members are not open with each other and with their stakeholders, there will be very little trust. Team members that trust others, are open-minded to feedback and suggestions, are cheerful, and encourage others will make it easier for all to express and articulate new ideas. These attributes can be spread among team members through activities such as discussions without agendas (e.g., lean coffee), celebrating small achievements, and constant inspiration from leadership.

Making Agile Really Work for Your Team

When transitioning to agile, teams undergo training on whatever methodology they'll be adopting, such as Scrum, kanban, or Extreme Programming. However, they are often not given as much help understanding the interpersonal dynamics necessary for agile to be successful. This is how teams fall into the habit of agile antipatterns.

To become a successful, high-performance agile team, it's important to identify and act on any interpersonal or cultural issues that may be standing in the way of true agility. By adopting the eight practices outlined above, your agile team can realize the benefits of improved communication, more frequent software releases, happier customers, and overall higher performance.

***

To read this article online with embedded hypertext links, go here:
https://www.agileconnection.com/article/8-keys-transforming-high-performance-agile-team

About Uday Varma

Uday has varied years of experience in IT industry and has been part of Organizational transformational initiatives for more than 10 yrs. primarily helping clients in Enterprise Agility. He has provided Agile coaching and Devops consulting services for many Banking and Retail client programs of UK, Germany, Australia for adopting Agile methodology and DevOps capabilities. He has experience in defining, promoting, and implementing agility at enterprise level, and identifying and resolving organisational and system impediments through collaboration and facilitation with stakeholders.

Myths and Misconceptions About Trust

By Marjan Venema

[**Note:** grayed text indicates hypertext links in original article.]

When I was a teenager, I was bitten by a German Shephard out of the blue, or so it seemed to me at the time. I never trusted that dog again, mostly because we never met again. I'm sure that if we had, we might have become friends and I would have become much better at reading him. The incident certainly hasn't stopped me from loving dogs.

The romantic in me would like to say that this incident is what sparked my interest in trust. It didn't, but it's a nice story, and it illustrates a couple myths and misconceptions about trust.

Myth: Trust Is All or Nothing

For a long time, I believed that if I trusted someone not to bash my head in, it meant that I completely trusted them. You either trust someone or you don't, right?

Not exactly.

I like to drive convertibles at high speeds up and down mountain roads. Roads like the one in the above picture. When that picture was taken, I was not at the wheel. Does that take trust? You bet. Would I trust the driver with anything? No way.

Would you trust the driver of a getaway car with your wallet? Can you trust a very competent rally driver to look out for you in difficult personal circumstances? Can you trust a co-worker that would drive you up or down this mountain safely not to go behind your back to reap the fruits of your labor?

Trusting someone for one thing does not necessarily mean that you trust them for everything. It's quite possible to trust a co-worker not to gossip about you, yet have no trust for them at all about the way they handle their own or your mistakes.

Myth: Trust Is a Matter of Life and Death

It certainly can be, and in many professions it is, whether between teammates or between the person performing a service and the person undergoing it. However, most of us don't work in the fire department or have medical jobs. Most us have office jobs, where life and death situations are few and far between.

In those circumstances, trust revolves more around emotional safety. Can you make a promise because you can trust someone to deliver so you won't have to face another's wrath? Can you trust someone not to judge you? Can you trust someone to walk their talk?

Myth: Trust Is About Competence

According to some, competence inspires trust. While I don't disagree, I don't entirely agree, either. Competence, or being good at something, is more about inspiring confidence. Specifically, confidence that someone is the right person to get a job done. On the other hand, trust is more about being able to rely on them using that ability to do the job right and get the best result possible.

Take the co-pilot of Germanwings flight 9525, trained to the T and perfectly competent to fly an aircraft to its destination. His ability also made him perfectly competent to fly it into the ground. Which, unfortunately, is exactly what he did on 24 March 2015. Despite his competence, he should not have been trusted to fly on that fateful day.

This next example hits a bit closer to home for me. The people I wouldn't want behind the wheel on that mountain road fall into two categories. One group simply lacks the ability to safely get me to the top and down again. I have zero confidence in their competence.

Another group consists of the people that *can* complete the trip safely, but I am uncertain if they *will*. Not because they may have a death-wish like the German co-pilot, but maybe because they tend to drink a little too much in the evenings, or because they are easily distracted and tend not to be focused on the job at hand.

Myth: Trust Arrives on Foot and Leaves on Horseback

Google "trust quotes" and this comes up a lot. It's a widespread belief.

However, trust isn't slow to arrive. Yes, some people will not trust anyone unless and until they have "proven" their trustworthiness. Most people, however, function the other way around. They will trust until they are proven wrong.

Further, being proven wrong once isn't enough to warrant a complete lack of trust. Yes, it will make you more cautious, but your trust for them doesn't simply fly out the window. People are quite capable of distinguishing between intention and effect, and are generally willing to give someone the benefit of the doubt.

Even when proven wrong a couple of times, that doesn't mean that all trust is gone, because trust is not all or nothing. Oh sure, after banging your head against the wall several times, you will not trust X to deliver on time anymore, but you can still rely on X to deliver high-quality work. And what if X were to deliver quality goods on time several times in a row?

Regardless of the initial level of trust for others that you operate from, that level of trust is in constant flux. Brené Brown uses a marble jar analogy to illustrate this.

Your marble jar for Peter starts with an initial number of marbles in it. Every action by Peter either adds a marble or takes one away. For example, Peter remembering your mother's name, inquiring about your recent exam, delivering quality work on time or being discrete about something you told him in confidence will add marbles. On the other hand, delivering something late or of inferior quality, being harsh to you or someone else or always saying yes but doing no will remove marbles from his jar.

So, where does the popular belief that trust leaves on horseback come from?

I have no idea. My guess is that we are often unaware of the effects of other people's words and actions on our level of trust for them, perhaps because questioning your trust for someone feels like a betrayal in itself. When you've finally had enough of someone's failures to act trustworthy, it feels like emptying the jar in one fell swoop, when in reality it has been running on fumes for some time already.

Myth: Trust Happens Automagically

Everything I've read about agile and high-performance teams stresses the importance of trust. Trust between team members, trust between teams and trust between teams and their stakeholders. And yet, none of the agile frameworks or methodologies I've seen go any further than that. We are all apparently expected to "get" it and get on with it. "Trust is important. Now go forth and trust each other."

Coaches and facilitators do get a bit more training and can find a lot more resources on trust building exercises. Unfortunately, most trust building exercises I have had the pleasure of reading or being subjected to are not about building trust. They are about building a connection. Connectedness is a lubricant that makes trust easier: you are far more likely to trust someone with whom you have broken bread, played or exchanged personal information. But a connection is not trust itself.

What everybody also seems to forget is that trust or team building exercises and activities can just as easily destroy trust when the person you previously thought was pretty nice turns out to be an utterly unreliable partner in the exercise.

What's more, just like training a dog doesn't just happen during obedience classes, trust doesn't just grow or erode during exercises and events intended to build it. Trust levels wax and wane with every observed word and action. If you want trust levels to improve, you will have to work at it all the time. This doesn't mean you can't slip up, just that when you do, you have to acknowledge it and make amends.

Myth: Trust Must Be Earned

This myth is one of my pet peeves. I am firmly in the "trust until proven wrong" camp, even though it may sometimes be with a lot of trepidation.

That doesn't mean I trust everyone for anything in every situation. For example, I am very much in favor of assessments and tests during the hiring process. I could say that is because of cognitive biases that make people believe they are better than they actually are. While that plays a part, the true reason is that hiring is a process with a lot of conflicting interests, and assessments may add some much-needed objectivity.

But, don't put people through a wringer just to gauge their trustworthiness as a human being. Doing so is a clear signal of distrust that is clearly heard by the person on the receiving end. Distrust begets distrust. They may tolerate it if it's the way to gain a prize they want, but, if anything, it will lower their trust (and respect) for their "testers."

Hearing an agile coach utter this myth really got me on my **high horse.**

Besides, it is futile. If you are not willing to trust me, there is nothing I can do or that I can give you that will make you trust me. Because, as we're about to discuss, trust can't be built.

Myth: Trust Can Be Built

Now there's a bummer. Bet you didn't expect that one, especially with all the trust building exercises, ice breakers, team building activities and what have you that all intend to build trust levels in teams.

Unfortunately, it is true. There is nothing I or anyone else can do to make you trust us.

It's not like I can stack packets of trust on you and that will increase your trust for me. All anyone can do is speak and act in ways that will facilitate your trust for them to grow. However, whether it has the desired effect or not depends entirely on whether you allow it and are willing to trust.

Trust can't be built or earned, it is given and it grows.

So, are all those exercises, ice breakers and games in vain?

Absolutely not.

They are essential for trust to have a chance. They create conditions and get people to interact in ways that are conducive for trust to grow. You need to take people out of their “normal” work — the transactions of day to day business — and put them in situations where they can interact as people.

Interacting as human beings, with as little interference from formal hierarchies as possible, is what makes people more comfortable with each other and what will allow them to interact more easily and with less trepidation in their “transactional” work. It allows trust to grow and to be given.

Myth: Trust Is Optional for the Bottom Line

Just like “leaving your emotions at the door” is wishful thinking, hoping that you can make do without trust is daydreaming at best. Sure, companies where distrust reigns supreme can be successful and make a profit. I just wonder how much more profitable they could be if they worked on increasing trust and happiness levels. Read Patrick Lencioni’s **“The Five Dysfunctions of a Team”** to see where an organization can leak money left, right and center when trust issues run rampant.

And, if you want the people in your company to innovate, to be creative and to be open to change, then you need them to be willing to make themselves vulnerable to the words and actions of their co-workers. That takes trust. Loads and loads of trust.

So, What Is Trust?

Trust is multi-faceted.

Trust is feeling emotionally safe.

Trust is knowing that someone will use their abilities appropriately.

Trust is resilient.

Trust is a two-way street.

Trust is essential for smooth collaboration so innovation, creativity and change can flourish.

Trust is in constant flux, it waxes and wanes with every interaction.

Trust is a verb. It needs to be worked on and you need to be aware of the effect of words and actions on trust levels.

Trust is not built or earned. It grows and is given.

⁂

To read this article online with embedded hypertext links, go here:
https://www.frontrowagile.com/blog/posts/114-myths-and-misconceptions-about-trust

About Marjan Venema

Marjan is a developer of 30+ years, turned coach and facilitator.

Having struggled with the demoralizing effect of low trust between team members and with management, Marjan is on a mission to bury the carrot and the stick and turn workplaces into high trust havens where collaboration, innovation and creativity thrive.

As part of that mission, Marjan helps software teams and their stakeholders, as well as individual professionals, to clarify, prioritize and achieve their goals with confidence. Self-trust and mutual trust will grow, allowing collaboration, innovation and creativity to thrive. High performance is then an almost inevitable outcome.

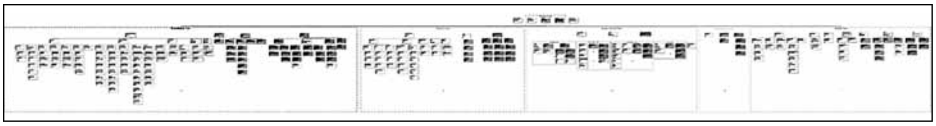
Join Marjan at www.marjanvenema.com for inspiration to turn your environment around and achieve more by doing less.

The shadow org chart

By Henry Ward

[**Note:** grayed text indicates hypertext links in original article.]

Like every company, we have a hierarchical org chart with me at the top. It looks something like this:



I have long felt there is a shadow org chart, much like a shadow economy, where employees trade ideas, give direction, offer help, and spread culture. This shadow org chart is built bottom up by employees and is very different from the top down hierarchical org chart set by me.

I wanted to map this shadow org chart and find employees who have disproportionate levels of influence relative to their hierarchical position. I also wanted to see the influence centers and decision makers, and the directional current between them and the rest of the company.

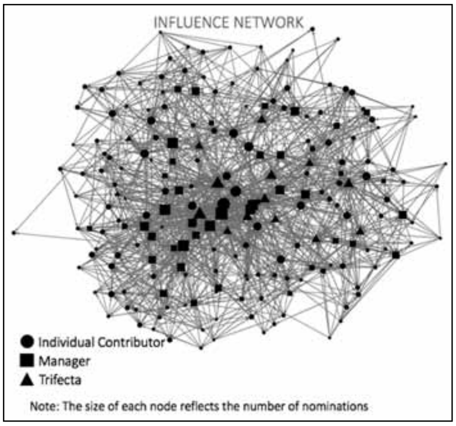
Top down org charts are trees. But bottom up influence charts are network graphs. We used **Innovisor** to map the network graph for us. To do this we asked employees three questions:

1. Who energizes you at work? (list 4 or more people)
2. Who do you go to for help and advice? (list 4 or more people)
3. Who do you go to when a decision needs to be made? (list 4 or more people)

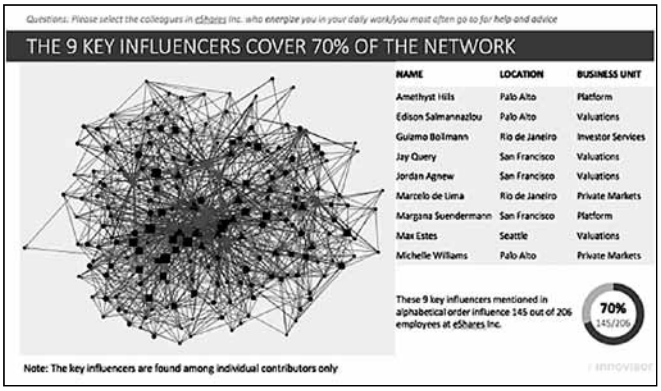
Every time an individual was listed counted as a nomination. We connected the nominators to the nominees in a directional graph. The result is the influence network below. A couple of notes:

1. Trifecta is our term for executives.
2. If you look closely you can see the directional arrows of influence

3. The larger nodes cluster in the middle


INFLUENCE NETWORK
● Individual Contributor
■ Manager
▲ Trifecta
Note: The size of each node reflects the number of nominations

On our first pass of understanding the network we filtered by individual contributors. In red is the smallest group of ten employees that influence the most employees. This is different than the top ten nominated employees. This is a union of ten employees that influences the most employees. This group influences 70% of our 250 employees. That means if we wanted to spread a meme, this is the ten we would start with.



Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice

THE 9 KEY INFLUENCERS COVER 70% OF THE NETWORK

NAME	LOCATION	BUSINESS UNIT
Amethyst Hills	Palo Alto	Platform
Edison Salmannazlou	Palo Alto	Valuations
Guizmo Bollmann	Rio de Janeiro	Investor Services
Jay Query	San Francisco	Valuations
Jordan Agnew	San Francisco	Valuations
Marcelo de Lima	Rio de Janeiro	Private Markets
Margana Suendermann	San Francisco	Platform
Max Estes	Seattle	Valuations
Michelle Williams	Palo Alto	Private Markets

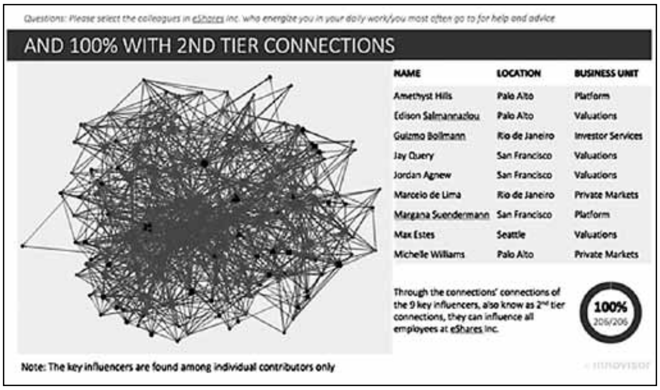
These 9 key influencers mentioned in alphabetical order influence 145 out of 206 employees at eShares Inc.

70%
145/206

Note: The key influencers are found among individual contributors only

innovisor

When we looked two degrees removed from the ten influencers (employees influenced by employees influenced by the core group) we were able to influence all 250 of our employees.



Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice

AND 100% WITH 2ND TIER CONNECTIONS

NAME	LOCATION	BUSINESS UNIT
Amethyst Hills	Palo Alto	Platform
Edison Salmannazlou	Palo Alto	Valuations
Guizmo Bollmann	Rio de Janeiro	Investor Services
Jay Query	San Francisco	Valuations
Jordan Agnew	San Francisco	Valuations
Marcelo de Lima	Rio de Janeiro	Private Markets
Margana Suendermann	San Francisco	Platform
Max Estes	Seattle	Valuations
Michelle Williams	Palo Alto	Private Markets

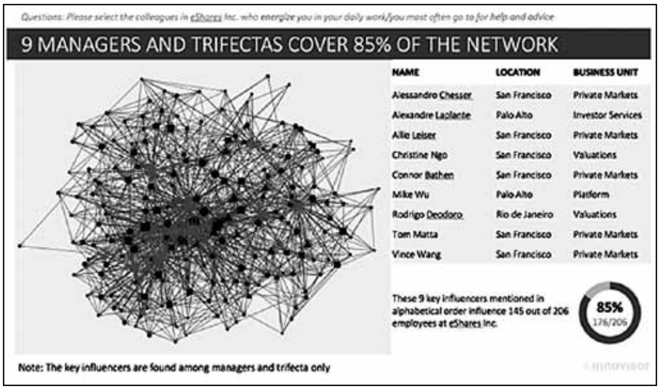
Through the connections' connections of the 9 key influencers, also know as 2nd tier connections, they can influence all employees at eShares Inc.

100% 206/206

Note: The key influencers are found among individual contributors only

© innovisor

We did the same exercise with managers and executives to find the nine core influencers. They influenced more of our employee base than the individual contributors, but not by as much as we expected. This strengthened the thesis that the undercurrents of interpersonal influence are as significant as the institutional management hierarchy.



Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice

9 MANAGERS AND TRIFECTAS COVER 85% OF THE NETWORK

NAME	LOCATION	BUSINESS UNIT
Alessandro Chesser	San Francisco	Private Markets
Alexandre Laplante	Palo Alto	Investor Services
Allie Leiser	San Francisco	Private Markets
Christine Ngo	San Francisco	Valuations
Connor Bathen	San Francisco	Private Markets
Mike Wu	Palo Alto	Platform
Rodrigo Deodoro	Rio de Janeiro	Valuations
Tom Matta	San Francisco	Private Markets
Vince Wang	San Francisco	Private Markets

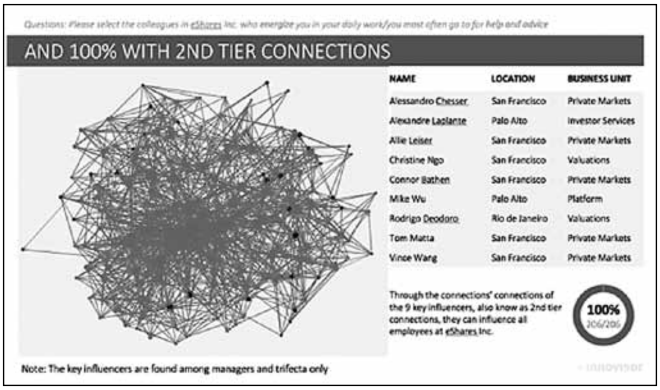
These 9 key influencers mentioned in alphabetical order influence 145 out of 206 employees at eShares Inc.

85% 176/206

Note: The key influencers are found among managers and trifecta only

© innovisor

Unsuprisingly, two degrees removed from these core managers we could influence the entire population of employees.



Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice

AND 100% WITH 2ND TIER CONNECTIONS

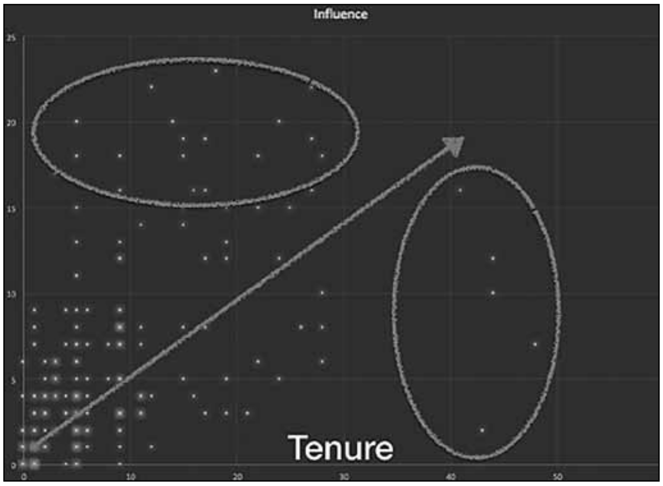
NAME	LOCATION	BUSINESS UNIT
Alessandro Chesser	San Francisco	Private Markets
Alexandre Laplante	Palo Alto	Investor Services
Allie Leiser	San Francisco	Private Markets
Christine Ngo	San Francisco	Valuations
Connor Bathen	San Francisco	Private Markets
Mike Wu	Palo Alto	Platform
Rodrigo Deodoro	Rio de Janeiro	Valuations
Tom Matta	San Francisco	Private Markets
Vince Wang	San Francisco	Private Markets

Through the connections' connections of the 9 key influencers, also know as 2nd tier connections, they can influence all employees at eShares Inc.

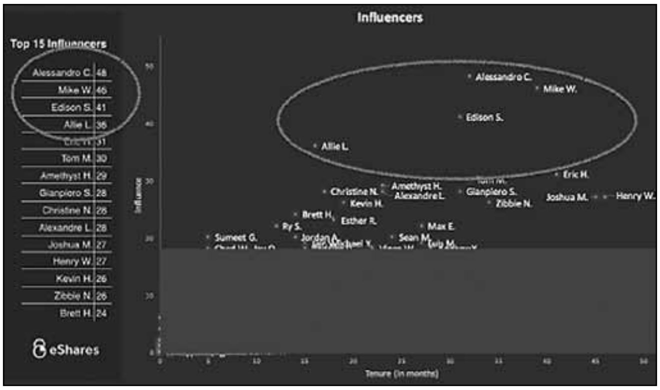
100%
206/206

Note: The key influencers are found among managers and trifecta only

We also looked at a scatter plot of the most nominated employees (plotted against tenure). There is a correlation of influence to tenure, but it is a looser correlation than expected. There are outliers on both sides. Many employees (the top cluster) have an outsized influence in the organization relative to their tenure. On the right are a handful of long-term employees that never reached their influence potential.



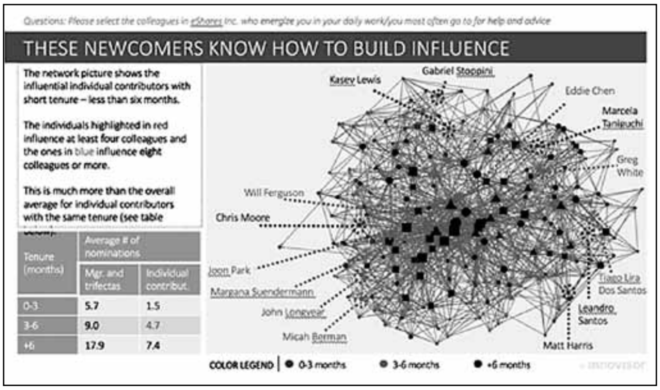
Below are our top 20 influencers with nominations. Impressively, the top four influencers had a combined 171 nominations. Both fascinating and humbling, I am the ceo and didn't make the top 10.



Top 15 Influencers

Alessandro C.	48
Mike W.	46
Edison S.	41
Allie L.	36
Eric H.	31
Tom M.	30
Amethyst H.	29
Gianpiero S.	28
Christine N.	28
Alexandre L.	28
Joshua M.	27
Henry W.	27
Kevin H.	26
Zibbie N.	26
Brett H.	24

We hire approximately twenty employees per month. One third of our company has been with us less than 6 months. Since tenured employees have an advantage, we did a new filter to look at new employees. Below are the most influential employees who have been here less than six months.



Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice

THESE NEWCOMERS KNOW HOW TO BUILD INFLUENCE

The network picture shows the influential individual contributors with short tenure – less than six months.

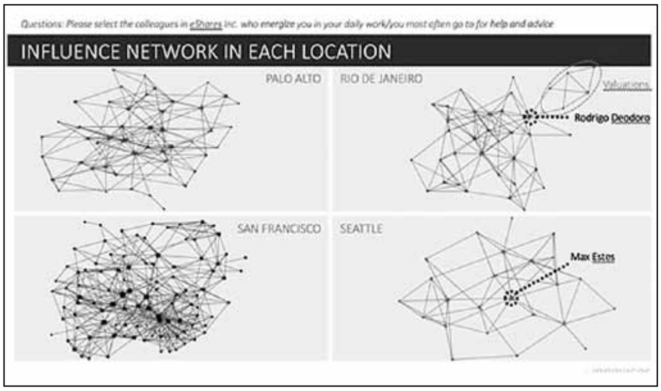
The individuals highlighted in red influence at least four colleagues and the ones in blue influence eight colleagues or more.

This is much more than the overall average for individual contributors with the same tenure (see table below).

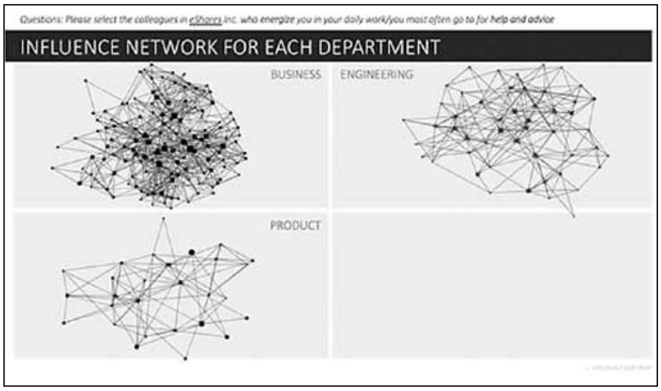
Tenure (months)	Average # of nominations	
	Mgr. and trifectas	Individual contribut.
0-3	5.7	1.5
3-6	9.0	4.7
+6	17.9	7.4

These new employees did something to rocket into high influence positions. The first half of this exercise is to identify our most influential employees. The second half is also to understand how they did it so we can manufacture more of these high influencers.

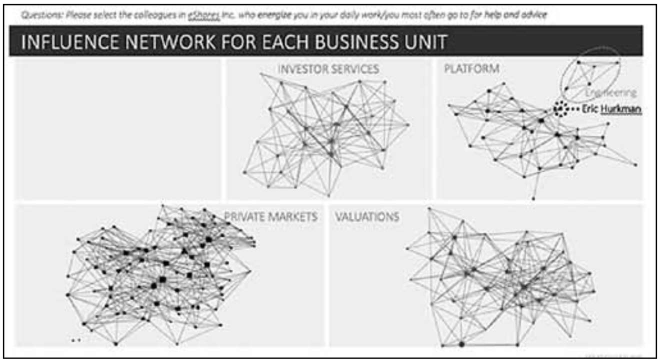
We also mapped the networks by office...


Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice
INFLUENCE NETWORK IN EACH LOCATION
PALO ALTO
RIO DE JANEIRO
Valuations
Rodrigo Deodoro
SAN FRANCISCO
SEATTLE
Max Estes

...by function (business, product, engineering)...


Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice
INFLUENCE NETWORK FOR EACH DEPARTMENT
BUSINESS
ENGINEERING
PRODUCT

…and by business unit (Private Markets, Investor Services, Valuations, Capital Markets).


Questions: Please select the colleagues in eShares Inc. who energize you in your daily work/you most often go to for help and advice
INFLUENCE NETWORK FOR EACH BUSINESS UNIT
INVESTOR SERVICES
PLATFORM
Engineering
Eric Hurkman
PRIVATE MARKETS
VALUATIONS

We had a few takeaways from the department, function, and geographic slices.

- Some business units and teams are siloed. For example, one cluster of Valuations engineers is completely detached from the Rio office. They are only connected through a single node. This happens to be our newer mobile team.
- Smaller offices have more key influencers relative to the size of the office. For example, one of our first product designers, is connected to almost every node in the Seattle office.
- **Dunbar's number** is much fewer than the conventionally accepted 150 people. Our offices become increasingly disconnected after approximately 50 people. Smaller offices are better.
- Each office develops their own culture and shadow org chart. The shadow org chart crosses business units and functions but is tightest through geography. Tribes form most consistently through physical proximity.

There were a number of other conclusions including why employees work at eShares, how they find help, and how decisions are made. I'll leave those for a future post. But if you'd like to dig in you can access our Innovisor final presentation **here**.

⁂

To read this article online with embedded hypertext links, go here:
https://medium.com/@henrysward/the-shadow-org-chart-cfcdd644575f

About Henry Ward

Henry is the CEO and co-founder of Carta, a software platform for founders, investors, and employees to manage equity and ownership. Since its inception in 2012, Carta has managed hundreds of billions of dollars in equity at more than ten thousand companies, helping companies like Slack, Coinbase, Flexport, and August Capital manage their cap tables, valuations, portfolio investments, and equity plans. Prior to Carta, Henry was founder and CEO of Secondsight, a portfolio optimization platform for retail investors. He also held leadership positions at software companies including Reddwerks Inc. and BetweenMarkets. Henry graduated from University of Michigan with a BGS in Mathematics and Computer Science and holds a MSC in Market Finance from EDHEC Business School.

Am I a Good Scrum Master?

By Tanner Wortham

[**Note:** grayed text indicates hypertext links in original article.]

Much of the role of the Scrum Master is intangible. We don't write elegant code, we don't craft beautiful designs, and we better not be creating Gantt charts. Instead, we're masters at soft skills, but how can such a thing be quantified? And if not quantified, how can I know if I'm a good Scrum Master? How can I honestly assess myself in the spirit of continuous improvement? With respect to a Scrum Master's service to a team, it begins by asking six questions.

- **Am I valuing outcomes over outputs?** I hope so. Let me explain the difference. An output is a sprint backlog. An outcome is a customer pleased with our latest deploy. In this case, our output (the sprint backlog) led to a positive outcome (a pleased customer), but that won't always occur. Do we talk often of what we're about to accomplish, but it never seems to come to fruition? Or can the team tell stories about how and where we contributed to the team's success or helped them learn a valuable lesson? It's the latter that we want.
- **How does the team look different than it did 4 to 8 sprints ago?** I don't mean do the teams have different team members, or are they doing different work. I mean how are they interacting? How do the things they do differ then versus now? Has the team adopted any new engineering practices? Lean and XP are great places to look for inspiration. After all, we should be inspiring a culture of experimentation and always challenging the status quo. Doing so encourages learning, and fostering a learning mindset is one of our primary responsibilities.
- **What data interests the team?** Data can be dangerous in the wrong hands, but if used wisely, it can greatly benefit the team. (Check out **An Appropriate use of Metrics** by Martin Fowler for an explanation.) Have we sought and encouraged useful metrics that interest and benefit the team? We're not talking about a burn down chart, which is largely a **vanity metric.** While useful, that's just scratching the surface. We're

talking about things like cycle time, team or customer happiness, and many more. I cover more ground on this topic **here, here,** and especially **here.**

- **Does the team see me as an asset or impediment?** I'll admit. I sometimes get in the way of my teams. The culprit is usually the same:

  "It's better to go slow in the right direction than fast in the wrong direction."

  When I get in the way of teams, I'll take as much time as necessary to explain why. This why makes the difference between being viewed as an asset or an impediment. In fact, let's talk more about why in our next question.
- **Can I explain why?** Talking about what we do is easy. It's why we do what we do that's interesting. Knowing why helps us find new and innovative ways of working. In fact, I explain my own whys for each Scrum ceremony here. However, it doesn't stop at the whys for ceremonies. Why estimate? Why groom a backlog at all? Why do we need to work as a cross-functional team when I can't understand a damn thing the designer is talking about? We should be prepared for these questions and more.
- **How much of my work does the team do?** I've said it often, but I'll say it again:

  "Every Scrum Master should always be trying to put him or herself out of a job."

  Am I facilitating all the ceremonies? Am I removing all the impediments? Am I chasing down information for the team? I certainly hope not. Instead, we should be looking for ways to nudge the team to do so. Take a two-week vacation and don't answer any emails. How'd the teams do in our absence? My point is this. Never derive value from feeling needed. It should be derived by imbuing in others the mindset that good enough never is.

After answering these six questions, do you think you're a good Scrum Master? What additional questions would you ask yourself as you introspect? Finally, thanks for stopping by and thanks to **Manjari** for inspiring me to think through this topic. Until next time.

⁂

To read this article online with embedded hypertext links, go here:
https://www.spikesandstories.com/good-scrum-master/

The Map Is Not The Territory

By Tanner Wortham

[**Note:** grayed text indicates hypertext links in original article.]

Too often, we emphasize the name of a thing over the potential outcome of the thing. We assume that when we exercise our understanding of the agile terrain via terminology that others have the same, rich mental models we do. I'm also sure many of us feel that using agile terminology lends us a bit of credibility. However, it's not knowledge or buzz words that makes us credible. That's earned by helping those around us solve challenging problems. To that end, let's try a different approach:

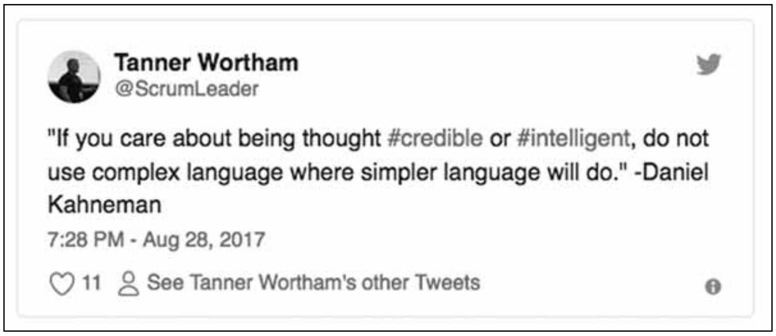
"Stop looking down at the map for answers. Instead, look up at your surroundings and talk to the natives."

To emphasize my point, have you ever told a team that we're going to give Scrum a go and heard something like this?

"We tried Scrum in my last organization, and it went miserably. I don't want to live through that crap again."

Like me, I'd venture to guess you began asking more questions. As I did, I usually discovered they weren't talking about Scrum at all but a cargo cult or possibly Dark Scrum. No wonder they hated it. Based on their stories, I would too!

In fact, I've been talking around this topic on Twitter lately.

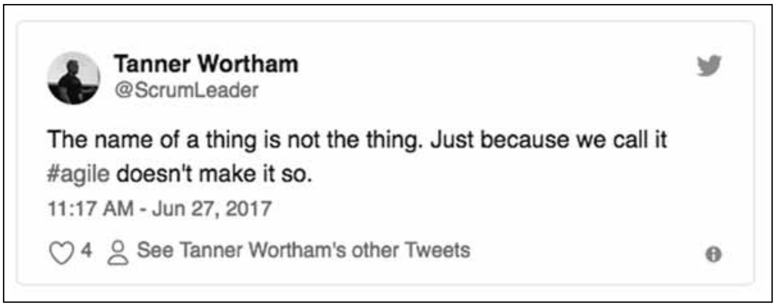


Tanner Wortham
@ScrumLeader

"If you care about being thought #credible or #intelligent, do not use complex language where simpler language will do." -Daniel Kahneman

7:28 PM - Aug 28, 2017

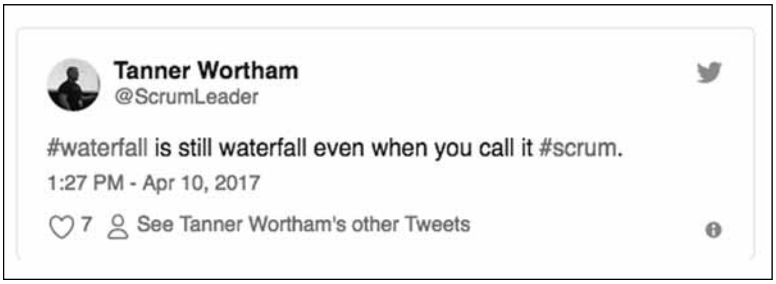
11 See Tanner Wortham's other Tweets



Tanner Wortham
@ScrumLeader

The name of a thing is not the thing. Just because we call it #agile doesn't make it so.
11:17 AM - Jun 27, 2017

4 See Tanner Wortham's other Tweets



Tanner Wortham
@ScrumLeader

#waterfall is still waterfall even when you call it #scrum.
1:27 PM - Apr 10, 2017

7 See Tanner Wortham's other Tweets

I think it's about time we put agile terminology aside — avoid it even — and simply begin helping others. With that in mind, I'd like to take two common terms and demonstrate how to weave them into conversation with organizations and with teams without using the words themselves.

Servant Leadership

This is one of those terms I see thrown around the community like a sailor throws around four letter words, and I think the concept is largely misunderstood. It is also the impetus behind this blog post, and it looks like this in conversation:

- "What can I do to make your job easier?" (This is often the last thing I say after talking with team members, my own reports, and even the founder of our company.)
- "Grant, you've been kicking ass getting the deploy out today. Want me to grab you something from the kitchen?"
- "Look. You can expect I'll pull you in a room and tell you if I see something that concerns me. I'll probably be blunt about it so I hope you return the favor if you see me slacking. I'm human just like the rest of us."
- "The best ideas are never be mine. You live and breathe this every day. You all just keep me around because I sometimes ask some interesting questions."

T-Shaped People

T-shaped means that a team member excels at one activity but is also capable of performing other activities that benefit the team. That's not to say that everyone on the team can excel at every activity equally. Instead, it means that there's a great deal of overlap in skills across the team to mitigate bottlenecks and to foster a common language. Below are a few ways to weave this concept into conversation:

- “Frank seems like he holds a rather pivotal role on our team. What happens if he’s sick or wants to take some time off?”
- “Linda, I know how much you enjoy designing, but have you thought about helping the team test by beating up on the implementation of your designs? That way, it’ll give us time to shore up any problems before we go live.”
- “Can we stop talking about how we should just focus on the stuff we’re good at? Look at all the confusion last sprint. Am I the only one who thinks hand offs are expensive? How could we quantify that cost?”
- “You’re looking for something to do today? Are you interested in pairing with Jack? It sounds like he was looking for a second set of eyes.”
- “Bill, I don’t want you checking email while you’re on vacation. You earned some time to relax and recuperate. We got this, and even if we don’t, maybe we should feel it. How’s that sound?”

The idea of avoiding agile terminology isn’t new. In fact, many have begun to call this notion **Undercover Agile.** Take a moment with me and appreciate the irony of naming a thing that’s adverse to talking about the names of things. Let me know in the comments below if the break down of the two terms above was useful. If so, I’ll do a series of similar posts.

Until next time.

⁂

To read this article online with embedded hypertext links, go here:
https://www.spikesandstories.com/agile-terminology/

Scrum Is Easy. People Are Hard.

By Tanner Wortham

[**Note:** grayed text indicates hypertext links in original article.]

I often compare Scrum to the frame of a house. We can look at the frame to know the size and shape of our rooms and how the house joins together. However, it's up to us to figure out what color to paint the walls and where to place the couch. The saying goes, "Scrum is easy to understand but difficult to master." I agree, but I prefer a different slant:

"Scrum is easy. People are hard."

But why? How can the Scrum Guide explain Scrum in 17 pages, but when it comes to its implementation, it can sometimes feel like herding cats? Let's talk about that today.

Rational Or Not, Here I Come

While working on my MBA, I remember sitting in a portfolio management class. My professor told us a story about how a **chimpanzee picked better stocks** than the world's highest paid money managers. That couldn't be right. Humans are rational, and that should be doubly true when our money is at stake. Shouldn't these money managers be experts in understanding human rationality and apply it to picking stocks that return better than the market? It turns out I was wrong. We humans aren't nearly as rational as we pretend to be, and I've since learned to provide latitude for irrationality that inevitably arises from the fear of organizational change. While a bit harsh, I mostly agree with **Will Smith:**

"Human beings are not creatures of logic; we are creatures of emotions. And we do not care what's true. We care how it feels."

Simplicity Is Rarely Simple

Complex problems are rarely solved by complex solutions. It's often one small, simple solution after another that succeeds. However, it's easy to lose sight of simplicity when the environment or issue is chock-full of nuisance and noise. Learn to explain the solution in two sentences or less. If that's impossible, keep working. Finally, it's unnecessary to create solutions that accounts for everything. Have a vision of what perfect looks like, account only for the variables that matter most, tweak once implemented and until satisfied, and manage the exceptions through conversation and collaboration.

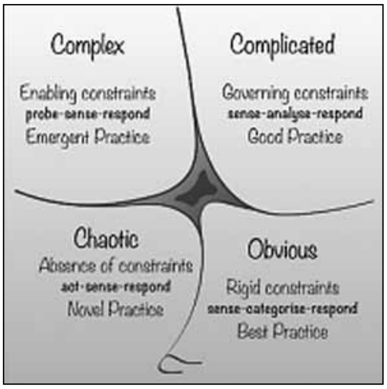
Value Values

Take a moment and enumerate all Scrum ceremonies and artifacts. For many of us, it won't be hard; we talk about them almost daily. Now take another moment. What are the Scrum values? There's five. Did you remember them all? I'd imagine many didn't. (By the way, here's those **values.**) Many team members want to be left undisturbed and don't wish to collaborate with the team. Do these members value openness? What do they value more? Other teams deal with sprints that frequently get blown up by interruptions and distractions. Does the organization value focus? What does it value more? It may be time to sit down with those around us to understand what is valued and why.

"A failed Scrum adoption can often be traced back to a misalignment of values between team, management, and Scrum."

Humans Are Not Created Equally

Let's take the term "best practice." It insinuates that a person is a cog and can be exchanged for another. We know this assumption is wrong, yet we continue to operate within this and other **Tayloristic** constructs. Additionally, "best" assumes there's no better way making continuous improvement pointless. This is also wrong, and it is in stark contrast with our agile mindset where good enough never is. As **Cynefin** tells us, Scrum does not operate in the obvious domain so abandon the term best practice and replace it with good or emergent practice. We should allow teams to find their own way with our guidance. We should encourage teams to experiment with what has worked for others and determine if it works for them. However, we should never impose our solutions on others. Allow them to succeed or fail based in the merit of their own ideas and actions. If we don't, we rob them of the opportunity to learn from their mistakes.



The Cost of Fighting Inertia

The system within an organization is often unkind to Scrum adoptions. For example, where Scrum is risk absorbent, an organization is often risk adverse. Put differently, organization wish to minimize risk while agilists wish to fail in small, yet substantial, ways in the spirit of learning about themselves and their customers. Just as traditional organizations feel planning gives them control over their future (which it doesn't), these same organizations believe avoiding risk creates a better, more resilient company.

But back to the system. How does the organization reward what it values? What kinds of themes are present on employees' performance reviews? Consider **diagramming and discussing parts of the system** to have a clearer view of where it fits together and how it can be influenced. After all, it was **Deming** who teaches us a powerful lesson:

"A bad system will beat a good person every time."

Much more comes to mind on this topic, but I'll stop here for today. Also, I'd like to give a special thanks to the people in the **Hands-On Agile** slack channel for brainstorming this topic with me. What do you think? Do you find the human element of Scrum easy?

⁂

To read this article online with embedded hypertext links, go here:
https://www.spikesandstories.com/scrum-is-easy-people-are-hard/

About Tanner Wortham

I'm the product of my experiences, education, and perspectives. Then again, I think that holds true for all of us. But I've been lucky. My background is a paradoxical mixed bag, and I believe it gives me a unique style.

Military veteran. Technically trained. Business savvy.

I'm currently an agile coach with LinkedIn and am the author of an agile blog at www.spikesandstories.com. I've helped companies in varying sectors — from gaming to edtech — as they rethink their approach to problem solving and customer satisfaction. I've been accused that my military training would mold me into a rigid, unmoving agilist, but nothing could be further from the truth. What civilians call agile, the Corps calls leading Marines, and it's through my experiences as a Marine that I derive most of my insights. It is by way of my software engineer beginnings that that help me relate to developers. Finally, it's through my experiences as a business professional that helps me get inside the minds of executives.

Key Best Practices for Using Customer Feedback

By Daniel Zacarias

[**Note:** grayed text indicates hypertext links in original article.]

As Product Managers, we perfectly understand the need to generate and use customer feedback. What isn't so often clear is how to do this on a day-to-day basis, when we're not as experienced or when we deal with "less than ideal" products and organizations.

This led me to reach out to 14 leading Product Managers and talk with them about how they use customer feedback in their own companies and teams. You can find the full audio recordings, along with transcripts and highlights **in this resource.** There's a ton of useful information throughout those conversations. In this post, I wanted to share with you some of the key takeaways I got from them.

1. Feedback is only relevant vs. a goal and user context

Understand where it's coming from

A piece of feedback usually comes to us in the form of "users are asking for X" or "customer Y and Z are telling us this". By itself, that's absolutely meaningless. The first step to figure out if something is relevant or not, is to know where it's coming from — and since we're dealing with products and markets, this isn't about knowing which specific users are giving the feedback, but about which segments they belong to. That will provide the necessary context for us to understand the motivation and problem they might be facing.

Just like a cake, there are many ways in which we can slice our customer and user base, and there isn't one true way to do it. **It all depends** on what we need to do, the stage and type of the product. Different Product Managers opt to think of segmentation along the dimensions that are most effective for their particular goals. In particular, they most often group their users and customers along their:

Characteristics and Behaviors

Traditional market segmentation is **typically done** around observable characteristics and behaviors for customers and prospects. First, we have ***demographics*** — statistical characteristics of the population, such as age, gender, income, etc. Then, there's also ***psychographics*** — which classifies people according to their attitudes, aspirations, and other psychological criteria. However, these kinds of segmentation are mostly useful for Marketing purposes, but not so much for PMs.

Finally, and although there are many potential issues around how these are defined and used, **roles and personas** are a staple of many teams' workflows for designing new features, and thus are also frequently used to think about different segments of the user base.

Needs

Frameworks like **Jobs-to-be-Done** are extremely helpful in determining exactly what the product is supposed to be doing for its customers — that is, the needs it serves. The same product may be used by people with quite different needs and under a wide range of contexts. This means that a product's suitability will not just depend on the person and her characteristics, it will actually depend on the product's usage context and the goals for the task at hand.

By segmenting our user base in terms of the jobs they're looking to get done, and not just their role or descriptive characteristics, we'll have an essential piece of context that provides much more clarity in how to seek and interpret the feedback we get from them. A classic example illustrating this point is that customers don't actually need an 1/8-inch drill bit; what they need is an 1/8 inch hole in their wall.

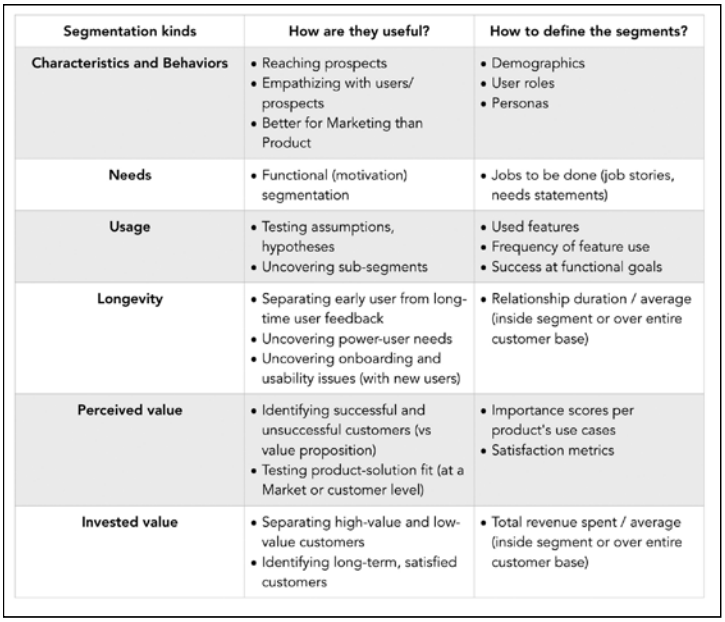
Relationship with the product (over time and over value)

Another way around which to segment customers is how they relate to the product over multiple dimensions–most commonly: their usage level, how long they've been users, the value they get from it and what they pay (or have paid so far) for it. These dimensions are cross-cutting (and complementary) to other types of segmentation and can be very useful in understanding why people in what should be the "same group" are giving different answers.

Let's go a bit more into each dimension and the sort of questions they answer:

- ***Usage*** — Each role or needs-based segment will have some assumptions about the features that will be used and how frequently we expect them to be used. If the data shows different feature-use and frequency clusters, we can go into a lot of interesting questions with those specific users — Why are they using it more/less than expected? Are our assumptions about needs or role-based segmentation wrong? Are they getting what they need from the product?

- ***Longevity*** — Where the customer is in his relationship with the product is very important to classify unsolicited feedback and knowing which kinds of questions to ask them. With new customers, we're looking for product fit, usability feedback, indications of continued use in the future, motivations behind the purchase/usage decision. With older customers, we're typically interested in satisfaction, power-user and early-testing feedback and pain points that the product doesn't solve.
- ***Perceived value*** — A set of customers can have the same underlying need and motivation to use the product, but the value they get from it is different. Their particular pain points might be the same, but the intensity isn't homogenous. We're looking to have a clear view of *"What is the customer getting out of the product?"* and *"How important is that problem for them?"*. By understanding where they fit within this gradient, we can get much more insight into their feedback.
- ***Invested value*** — The amount of money customers have spent on the product, relative to other customers is also telling of the kind of relationship they have with it and a proxy for their satisfaction, perceived value and importance. This of course varies widely and depends on each product's characteristics; however, it is an easy metric to use as guideline.



Segmentation kinds	How are they useful?	How to define the segments?
Characteristics and Behaviors	• Reaching prospects • Empathizing with users/ prospects • Better for Marketing than Product	• Demographics • User roles • Personas
Needs	• Functional (motivation) segmentation	• Jobs to be done (job stories, needs statements)
Usage	• Testing assumptions, hypotheses • Uncovering sub-segments	• Used features • Frequency of feature use • Success at functional goals
Longevity	• Separating early user from long-time user feedback • Uncovering power-user needs • Uncovering onboarding and usability issues (with new users)	• Relationship duration / average (inside segment or over entire customer base)
Perceived value	• Identifying successful and unsuccessful customers (vs value proposition) • Testing product-solution fit (at a Market or customer level)	• Importance scores per product's use cases • Satisfaction metrics
Invested value	• Separating high-value and low-value customers • Identifying long-term, satisfied customers	• Total revenue spent / average (inside segment or over entire customer base)

Uses and definition of different kinds of segmentation

You need to know where you're headed

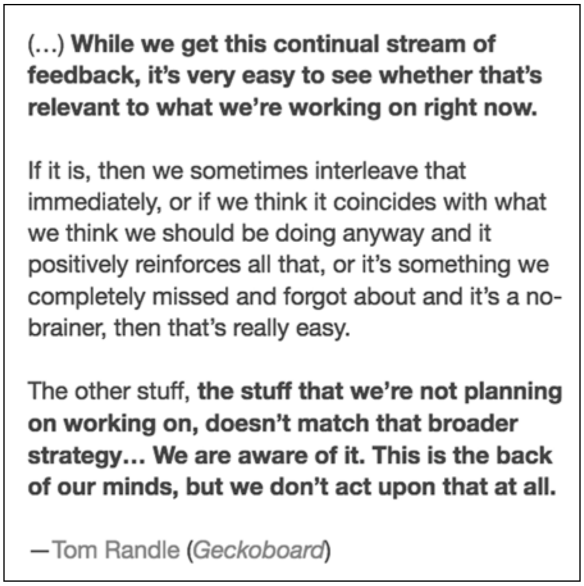
Yet, having a good segmentation model and being aware of where the piece of feedback is coming from (and the context and motivation behind it) is not enough.

The only way to have a clear answer to: *"is this bit of feedback relevant?"* is by considering both the user context and our current product and business goals. If our current goal is expand our MRR by up-selling to customers on paid plan A to plan A+, then feedback from users on the Free plan will not be as relevant. It might be, if we were looking to increase retention or improve satisfaction, for instance.

It's a two-level processing system:

1. Do we know "Who" is giving us this feedback and why?
2. Is this something that we want to focus on right now?

If the answer to the latter is No, then you can safely move on to whatever else might move your needle — there are never enough resources, so you might as well focus on what matters to your goals. When the answer is Yes, you can proceed with a clearer definition of **how to evaluate success.**



(…) **While we get this continual stream of feedback, it's very easy to see whether that's relevant to what we're working on right now.**

If it is, then we sometimes interleave that immediately, or if we think it coincides with what we think we should be doing anyway and it positively reinforces all that, or it's something we completely missed and forgot about and it's a no-brainer, then that's really easy.

The other stuff, **the stuff that we're not planning on working on, doesn't match that broader strategy… We are aware of it. This is the back of our minds, but we don't act upon that at all.**

—Tom Randle (*Geckoboard*)

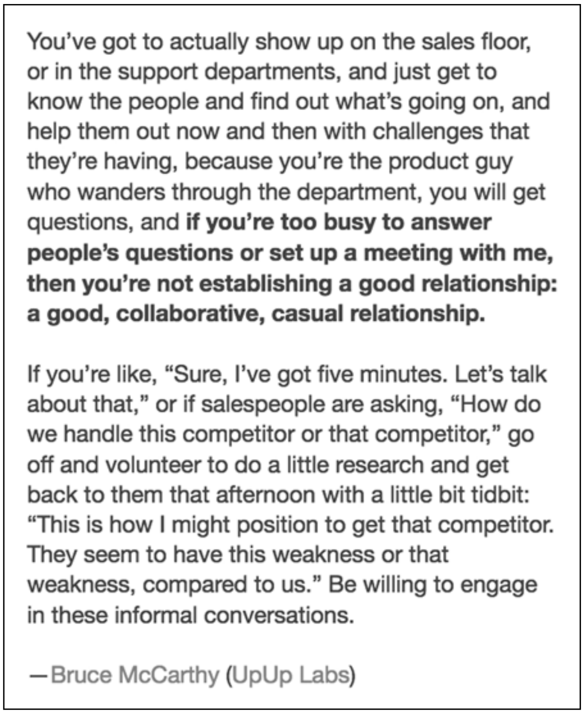
2. Getting quality feedback is a cross-functional effort

Insert yourself into customer touch-points

Organizational silos exist for many reasons, but they particularly affect Product Managers because they are the engines of the cross-functional process that defines and ships products. So, it's on us to break those barriers down.

One way to do it, is to find ways to help other customer-facing teams (like Sales and Support) do their job. **Meet with them,** be available for questions, go through their concerns and explain future plans or workarounds. This effort to reach out will signal other teams that they should come to you with customer issues or questions of their own.

A further step into this is to actually be part of those teams sometimes. Join the support team and answer a few tickets yourself. Ask them to send you summaries of top problems every week or so. Go on sales calls and listen. Understand what customers ask for and object to (and what salespeople are telling them). Later, you can debrief them about your plans and how to focus their message so they sell what you have or what you're sure you'll have (and not some random feature idea).



You've got to actually show up on the sales floor, or in the support departments, and just get to know the people and find out what's going on, and help them out now and then with challenges that they're having, because you're the product guy who wanders through the department, you will get questions, and **if you're too busy to answer people's questions or set up a meeting with me, then you're not establishing a good relationship: a good, collaborative, casual relationship.**

If you're like, "Sure, I've got five minutes. Let's talk about that," or if salespeople are asking, "How do we handle this competitor or that competitor," go off and volunteer to do a little research and get back to them that afternoon with a little bit tidbit: "This is how I might position to get that competitor. They seem to have this weakness or that weakness, compared to us." Be willing to engage in these informal conversations.

—Bruce McCarthy (UpUp Labs)

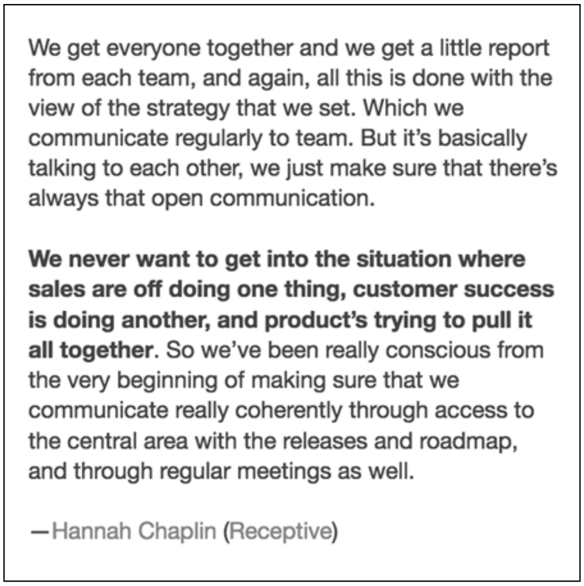
Get everyone to think like a PM

Since you can't always be there for other teams, coach them so they give you data that's closer to what you need. **Help them think like a PM,** so that in their interactions with customers they dig further to understand the problem, **and they don't come to you with solutions.**

Feedback is most valued (and valuable) when shared

If your organization doesn't see the value in customer feedback, find a way to get some, and **show it to people in leadership positions.** It's amazing the impact in empathy and understanding that comes from this.

Also, if you set up a **regular check-in with your cross-functional team** to gather and share what you (and they) are learning from customers, you'll be aligning and empowering everyone to understand the problems you're facing, and contribute to the design of solutions.



We get everyone together and we get a little report from each team, and again, all this is done with the view of the strategy that we set. Which we communicate regularly to team. But it's basically talking to each other, we just make sure that there's always that open communication.

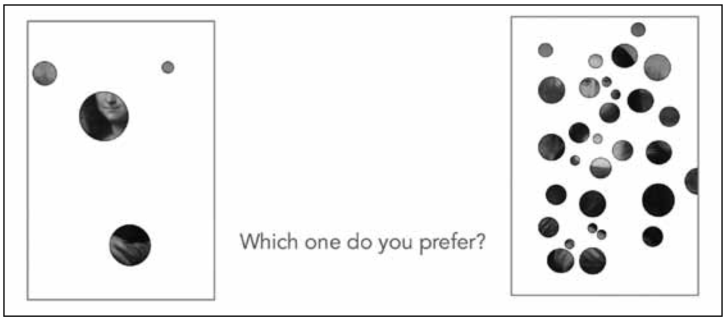
We never want to get into the situation where sales are off doing one thing, customer success is doing another, and product's trying to pull it all together. So we've been really conscious from the very beginning of making sure that we communicate really coherently through access to the central area with the releases and roadmap, and through regular meetings as well.

—Hannah Chaplin (Receptive)

3. Think of customer feedback as asystem

When talking about customer feedback, people usually think about a particular type of tool — it might be surveys, user tests, feature requests or others. But in reality, it's a system of tools and techniques, **combined.**

The goal of customer feedback is to understand **whether we're hitting our and customers' goals or not.** It's about getting **customer input throughout the development cycle and getting a more complete picture of their needs.**

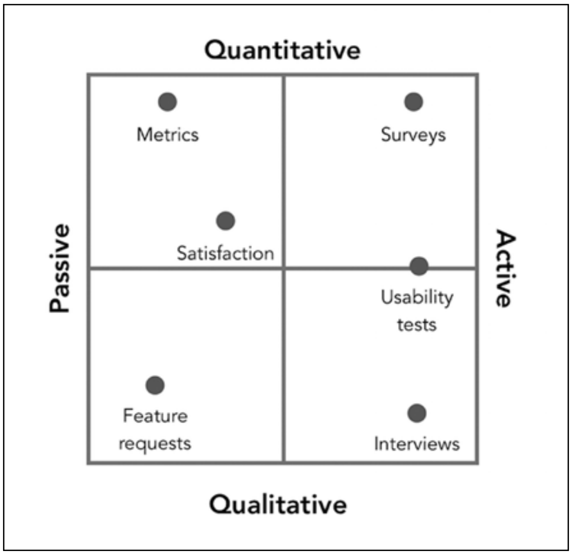


Visually, you can think of it as poking holes through a curtain, trying to see what's behind it. Each tool lets you uncover different areas, and that is why you need to use many of them.

The most immediate way to think about customer feedback methods is in terms of the type of data they produce–they can be either **quantitative** or **qualitative.**

Another way to think about them is how the feedback is triggered — **are we "actively" probing for something or are we "passively" listening and monitoring what**

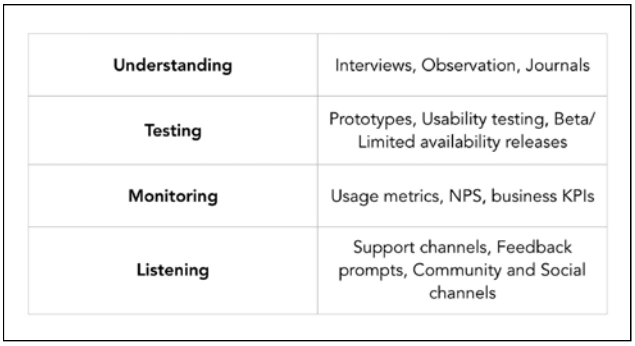
comes in? Combining these two dimensions produces a matrix like the one below.



However, a much more interesting way of looking at this is to consider the purpose of the feedback method. In other words: what does it help us solve? Using that perspective, we can divide feedback methods into four major categories:

1. **Understanding** — methods that let us understand what customers need, find valuable, and the reasons why things work or don't work for them.
2. **Testing** — methods that help us test and validate if a concrete idea, feature or value proposition matches our expectations or not.
3. **Monitoring** — methods that work as "thermometers" to track over time if some feature, release and the product in general are truly matching our expectations or not.
4. **Listening** — open feedback channels for customers to reach us for support, questions, requests, or general feedback.

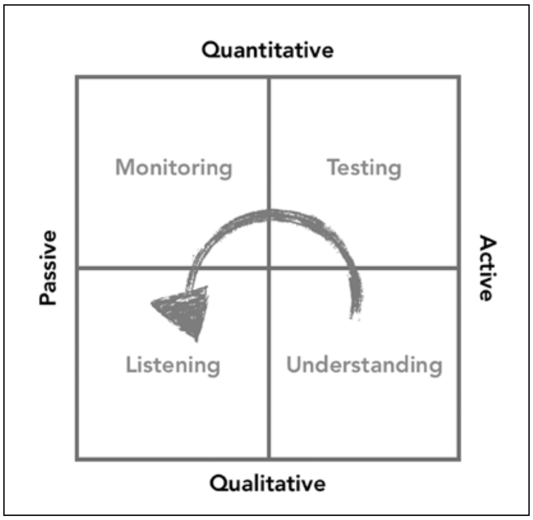
Let's have a look at how these groupings line up with commonly used feedback methods:



Understanding	Interviews, Observation, Journals
Testing	Prototypes, Usability testing, Beta/ Limited availability releases
Monitoring	Usage metrics, NPS, business KPIs
Listening	Support channels, Feedback prompts, Community and Social channels

Reviewing these categories we can see how they correspond with the product development cycle and also how they mostly match the Quantitative/Qualitative and Passive/Active matrix.

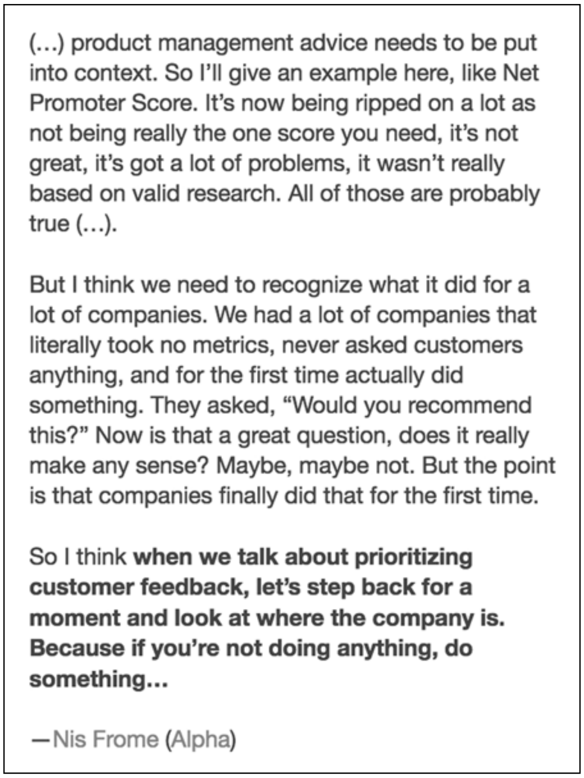
This classification shows the value that comes from every kind of customer feedback and provides for a structured approach when it comes to putting these methods together.



Something is better than nothing

The good news is that we don't need to do everything to get valuable insights — at least not right away. We usually fret about whether or not we're *doing the right things* and whether we're *doing them right*. This is a great instinct to have as PMs — we should be thinking about how to improve our processes and work. On the other hand, this can also lead to inaction (*"what if I send this survey to the wrong audience?", "what if I'm not asking the right questions?", "which tool should I use for this?",* and so on).

As long as we're aware that whatever process we follow isn't going to be perfect, then **it's much better to do something, than nothing at all.** It's very likely that we'll be working off of imperfect inputs, but being conscious of it is key: this way, we'll have something to question, research further, and test. At least we're starting from something that came from our customers, rather than our own heads.



(…) product management advice needs to be put into context. So I'll give an example here, like Net Promoter Score. It's now being ripped on a lot as not being really the one score you need, it's not great, it's got a lot of problems, it wasn't really based on valid research. All of those are probably true (…).

But I think we need to recognize what it did for a lot of companies. We had a lot of companies that literally took no metrics, never asked customers anything, and for the first time actually did something. They asked, "Would you recommend this?" Now is that a great question, does it really make any sense? Maybe, maybe not. But the point is that companies finally did that for the first time.

So I think **when we talk about prioritizing customer feedback, let's step back for a moment and look at where the company is. Because if you're not doing anything, do something…**

—Nis Frome (Alpha)

⁂

To read this article online with embedded hypertext links, go here:
https://foldingburritos.com/articles/2017/12/08/best-practices-customer-feedback/

About Daniel Zacarias

The biography and picture of this author/member of the Nominating Committee was not received in time for publication."

Nominating Committee

About Dimitar Bakardzhiev

Dimitar is an expert in driving successful and cost-effective technology development. As a LKU Accredited Kanban Trainer (AKT) and Kanban Coaching Professional (KCP) Dimitar puts lean principles to work every day when managing complex software projects. As an Agile consultant, his major engagements have been with Unum and Bosch. Dimitar has been one of the evangelists of Kanban in Bulgaria and has published David Anderson's Kanban book as well as books by Goldratt and Deming in the local language. He can be reached at dimitar.bakardzhiev@modernmanagement.bg.

About Georg Fasching

Unfulfilled potential is abundant; be it food wasted, or people in teams whose environments lock up their genius. **Georg** cares deeply about resolving these issues.

Georg is a coach and lifelong learner. Tracking a conventional product management path 2000-2010, Georg enjoyed the experience of the cultural nuances of team collaboration around the globe, whilst struggling with the confines of traditional IT project delivery. Since 2010 Georg has been on a new route on his journey, deeply exploring how people work and work together.

As enterprise coach, Georg guides organisational growth to help digital agencies unlock their teams' genius. As executive coach, Georg helps leaders & leadership teams embrace

Continued on page 326

their potential. Georg, as trainer & mentor-coach, also helps others further their skills and knowledge so that even more teams can have their team genius unlocked.

Georg invites you to find educational videos and posts at www.georgfasching.com, where you can also contact him.

About Jon Jorgensen

Jon is President of Needle Hop, Inc., an Accredited Kanban Trainer, and Certified Team Coach.

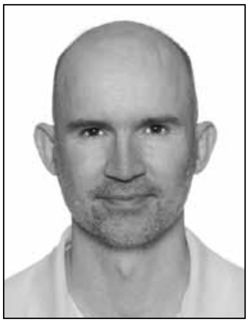
With 7,000+ hours coaching and mentoring 100+ teams across 30+ years, in the Agile industry, Jon is a leader in the wider group that is pioneering an extended inquiry into the necessity of employee engagement in any legitimate and lasting organizational change. See: OpenSpace Agility

Jon leads the Audacious Agile Conversations Meetup, the Lean Kanban, Agile & Self-Organization Meetup, Agile Open San Diego, Agile Open Arizona and Agile Coach Camp US West. He is a contributor to the book *Agile Coaching: Wisdom from Practitioners.* Jon co-created a framework for corporate executives called Leadership Scrum.

He also volunteers time and donates to a non-profit called 5 Saturdays, which teaches kids how to be Agile, toward achieving their life goals.

Jon's forthcoming book is entitled *The Business Agility Kernel* which he is authoring iteratively and releasing incrementally.

About Michael de la Maza



Michael is a Scrum Alliance Certified Enterprise Coach (CEC). As an Agile consultant, his major engagements have been with Paypal, State Street, edX, Carbonite, Unum, and Symantec. He is the co-editor of *Agile Coaching: Wisdom from Practitioners* and co-author of *Why Agile Works: The Values Behind The Results.* He holds a PhD in Computer Science from MIT. He can be reached at michael.delamaza@hearthealthyhuman.com and his website is hearthealthy-scrum.com.

About Angie Pate

Angie has led and transformed teams and leadership during times of change, chaos and calm for more than 12 years. She has helped teams and leadership to adopt an agile mindset and develop agile practices that continually aid in their growth and success. Angie has held many different roles throughout her career such as Software Engineer, Business Analyst, Agile Leader, Product Manager/Owner and Scrum Master/Coach. She is a certified Agile Coach and Scrum Master and currently consulting with Toyota North America and Toyota Connected. Her goal is to help you show up differently today than you did yesterday. Visit her blog at angiepate.com to connect and learn more about her.

About Allison Pollard

Allison helps people discover their agile instincts and develop their coaching abilities. As an agile coach with Improving in Dallas, Allison enjoys mentoring others to become great Scrum Masters and fostering communities that provide sustainability for agile transformations. In her experience, applying agile methods improves delivery, strengthens relationships, and builds trust between business and IT. Allison is also a Certified Professional Co-Active Coach, a foodie, and proud glasses wearer. Visit her blog at www.allisonpollard.com to connect further.

About Cherie Silas

Cherie is a Scrum Alliance, Certified Enterprise Coach (CEC) and an ICF Professional Certified Coach (PCC). She has a strong desire to help people arrive at the place they define as success in both personal and professional life. Her goal is to invest the experience and talents she has gathered through years of learning, often times the hard way, into people whom she hopes will become greater than she can ever dream to be.

Her main focus is culture transformation and development of people in Agile roles. When not coaching agile organizations, Cherie can be found coaching individuals, executives, employees, and rising leaders of non-profit organizations.

Continued on page 328

She also teaches professional coaching to people in Agile careers who want to earn ICF credentials and add coaching to their toolkit. Cherie has background training from CTI and ORSC and has a passion to bringing professional coaching into the Agile world.

Cherie's life mission that drives every interaction with every individual she encounters is simply this: To leave you better than I found you with each encounter

https://tandemcoachingacademy.com

About Ted Wallace

Ted is a Certified Scrum Professional (CSP) from the Scrum Alliance who is in the process of working towards being a Certified Team Coach (CTC). Ted has worked in both startup and corporate environments in developing their organizational and technical structures to be able to incorporate the Agile mindset. He loves seeing people, teams, and companies grow and gain the benefits of increased Agility. Ted holds masters degrees in Computer Science and Physiology from Maharishi University of Management. He can be reached at tedtalktoday@gmail.com.

***

Co-Editors

About Cherie Silas

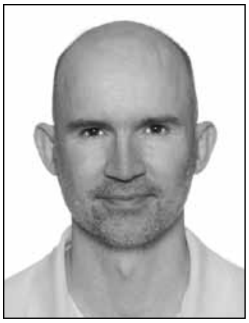
Cherie is a Scrum Alliance, Certified Enterprise Coach (CEC) and an ICF Professional Certified Coach (PCC). She has a strong desire to help people arrive at the place they define as success in both personal and professional life. Her goal is to invest the experience and talents she has gathered through years of learning, often times the hard way, into people whom she hopes will become greater than she can ever dream to be.

Her main focus is culture transformation and development of people in Agile roles. When not coaching agile organizations, Cherie can be found coaching individuals, executives, employees, and rising leaders of non-profit organizations. She also teaches professional coaching to people in Agile careers who want to earn ICF credentials and add coaching to their toolkit. Cherie has background training from CTI and ORSC and has a passion to bringing professional coaching into the Agile world.

Cherie's life mission that drives every interaction with every individual she encounters is simply this: To leave you better than I found you with each encounter

https://tandemcoachingllc.com

About Michael de la Maza



Michael is a Scrum Alliance Certified Enterprise Coach (CEC). As an Agile consultant, his major engagements have been with Paypal, State Street, edX, Carbonite, Unum, and Symantec. He is the co-editor of *Agile Coaching: Wisdom from Practitioners* and co-author of *Why Agile Works: The Values Behind The Results.* He holds a PhD in Computer Science from MIT. He can be reached at michael.delamaza@hearthealthyhuman.com and his website is hearthealthy-scrum.com.

***

PROJECT SPONSORED BY:

Scrum Alliance®

Special Thanks to the Nominating Committee

Jon Jorgensen
AKT

Cherie Silas
CEC

Angie Pate
ICP-ACC

Dimitar Bakardjiev
KCP

Ted Wallace
CSP

Michael de la Maza
CEC

Georg Fasching
ICP-CAT

Allison Pollard
CPCC