# Self-Organization Eats Agile at Scale for Breakfast

RON QUARTEL, Fluid Scaling Technology

In 2016 my department, inside a highly traditional health insurance company, decided to try something unconventional: An experiment in Agile scaling no one had done before. We threw the work on a wall and let a tribe of 50ish people self-organize into teams, repeating this exercise every two days! For almost two years we continued to evolve the system, and it worked extremely well. The experiment continued until some key environmental factors changed, ultimately killing it. This paper outlines what we tried, what we learned, and what I would recommend to others, should they be interested in creating a scaled, self-organized, complex- adaptive system.

## 1. INTRODUCTION

May 2014, New Orleans, an Agile conference with around 800 attendees, and an unexpected Eureka moment! The last day of this three-day event was an unconference, a.k.a. Open Space event. In 20 minutes, I witnessed 800 people self-organize and create an agenda, and then flawlessly execute a one-day conference. It struck me just how powerful, elegant, and capable this simple system was that relied solely on self-organization and scaled to boot! Perhaps it was because the theme of the conference was agile scaling, that I made a conceptual leap—why are we not using Open Space Technology (OST)[1] to solve agile scaling? All the efforts to date felt too command and control for my liking, but Open Space revealed an approach that felt much more compatible with agile values and seemed full of potential.

Fast forward two years to Spring of 2016, USA Pacific NW, a health insurance company—this Eureka idea returns. I find myself faced with a scaling challenge, and I run the idea of using OST by the team. It was well-received, and we agreed to run it as an experiment. An experiment we ended up running for almost two years. A host of surprising truths about corporate culture and human nature revealed themselves. As did some exciting new tools and bleeding-edge approaches to agile and agile scaling that are worthy of sharing with the community. We were truly pioneering. A new language, new tools and a new process formed. (I have highlighted new language terms in bold.) It changed me as a person, opened me up to the human movement, and changed my views on many now common agile assumptions and best practices.

## 2. BACKGROUND

Having worked at this insurance company previously as a consultant, I was familiar with the culture. I could not imagine wanting to work there again. So, in 2016 when a friend contacted me and suggested that I should come and join him on his team at said company, it sounded like a bad idea. He assured me that this group was different. It had a strong leader, autonomy, some exciting products to build, a commitment to Extreme Programming (XP)[2] and would be run more like a startup or skunkworks inside the company. Once assured all this was true; I joined a small team of six as a software developer.

## 3. THE STORY BEGINS—HOW TO SCALE THAT WOULD BE FAIR?

We were hiring fast and needed a scrum master. I had many years of experience in agile as a developer, manager, coach, trainer, instructor, and consultant. My experience taught me that if we got the hiring for the Scrum Master position wrong, things would not go well. So, I volunteered to fill in while we continued to look for a good match. Looking for a scrum master ended up taking some time, so instead of keep searching, I stayed in the position and changed the title to an Agile Coach.

From the start, we knew our Digital Platforms department was going to own two products with the view of adding more later. One product was fast becoming a legacy system, and we had not yet started on the other—our mobile platform. Mobile was our new and sexy platform that would give developers a chance to learn new skills. Here was my dilemma: who works on mobile, and who stays behind on the legacy product if we split the team?

Ron Quartel ron.quartel@fastagile.io

What would be fair? Why should the people that know most about the legacy system be shackled to it? That felt particularly wrong because some of those developers were long time employees. If anything, they should be rewarded for their time and given a chance to gain new skills if they desired a change. Which is when the idea hit me—what if we stayed together as one cohesive unit? A tribe that could dynamically form teams to match changing and fluid priorities? The idea from years back on using OST to scale agile returned as a possible solution. If the team went with this idea, we would be able to stay together and not have to split!

I gathered everyone and ran the idea by them. Why don't we have a marketplace of work where individuals self-select into teams around volunteer leaders who have identified work items? This system would allow people to move between products and give the organization the ability to flex effort between products as needed! It was a win/win. If everything that needed to be done got done, what did it matter who did it? The tribe would pull work from a story map and form teams (versus having work pushed into static teams). It was choreography over orchestration! See FIGURE 1 below.

The idea won out with the understanding that if it did not fly after two months, we could easily roll back to a scrum approach and process. I could do that with my eyes closed if needed as I had done it so often.

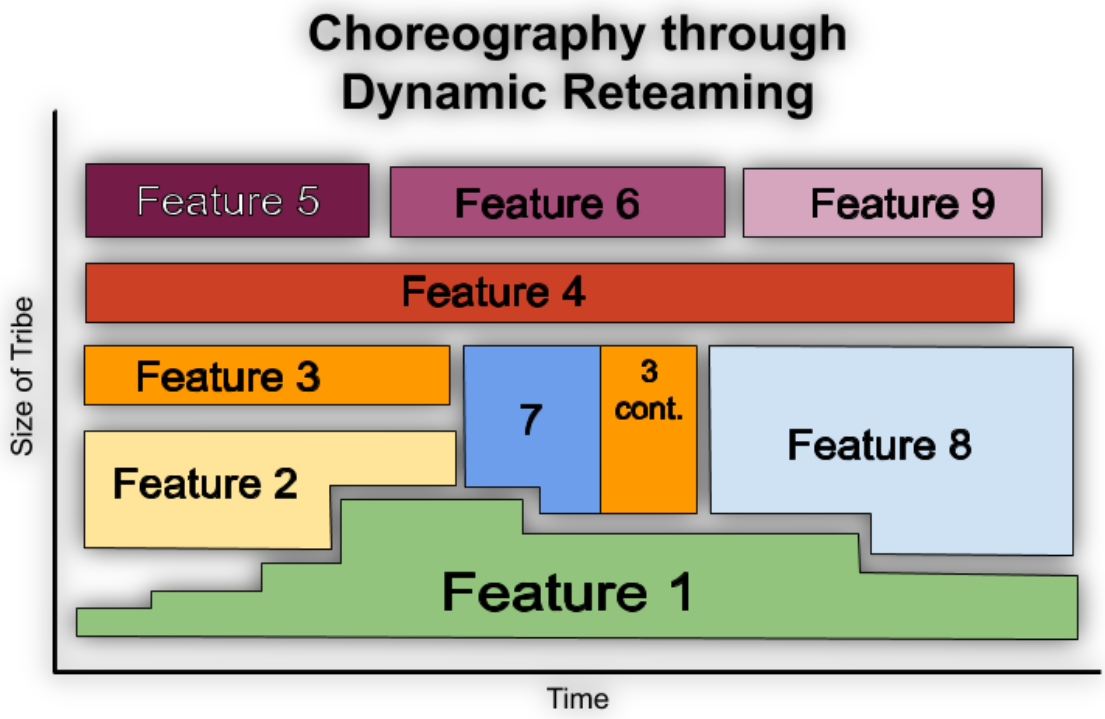


Figure 1. Choreography of Tribe Work Over Time

## 4. THE CHALLENGES WE ENCOUNTERED IN OUR JOURNEY TO SELF-ORGANIZATION AT SCALE

### 4.1 Backlog management—feature mapping & feature stewards

When it came to how to manage the backlog, there was something in the idea of a story map[3] that was attractive. It would give the tribe an ability to see the work in its entirety and understand the context between the various work items. But in practice, we soon found maintaining a story map tricky as the splintering of work was rapid and voluminous as the tribe grew. (We tried both physical and electronic formats.)

The solution came when we decided to restrict the map to show just the high-level features. We renamed the story map a release map. Finer-grained work items we tracked on their own individual board, assigning a board per feature. These boards became known as a feature map or feature tree due to the branching nature of how work items split recursively from the root feature node. Interestingly, the notion of a story did not always fit the sub-nodes and we ended up calling everything a work item instead. This new system of backlog management we called Feature Mapping[4].

Somewhat related, we also came up with the role of a Feature Steward, which solved an issue we had when the business didn't know who to ask questions of in the tribe (due to dynamic reteaming).

We were developing a new agile lingo, process, and roles within our tribe. It was an exciting time!

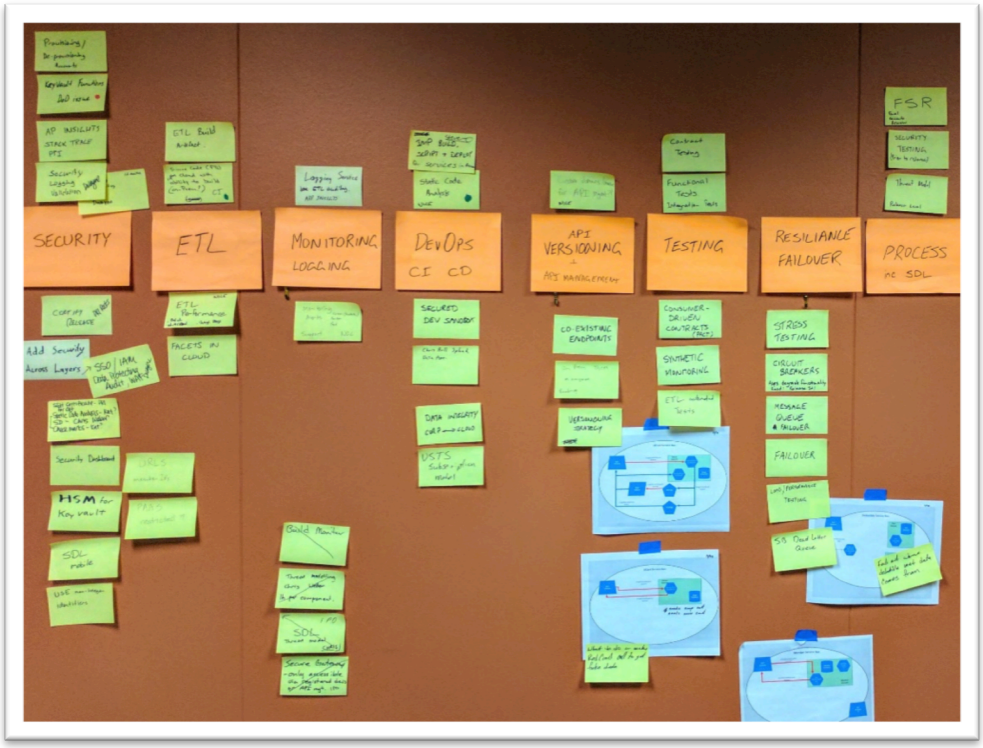


Figure 2. Release Map Example

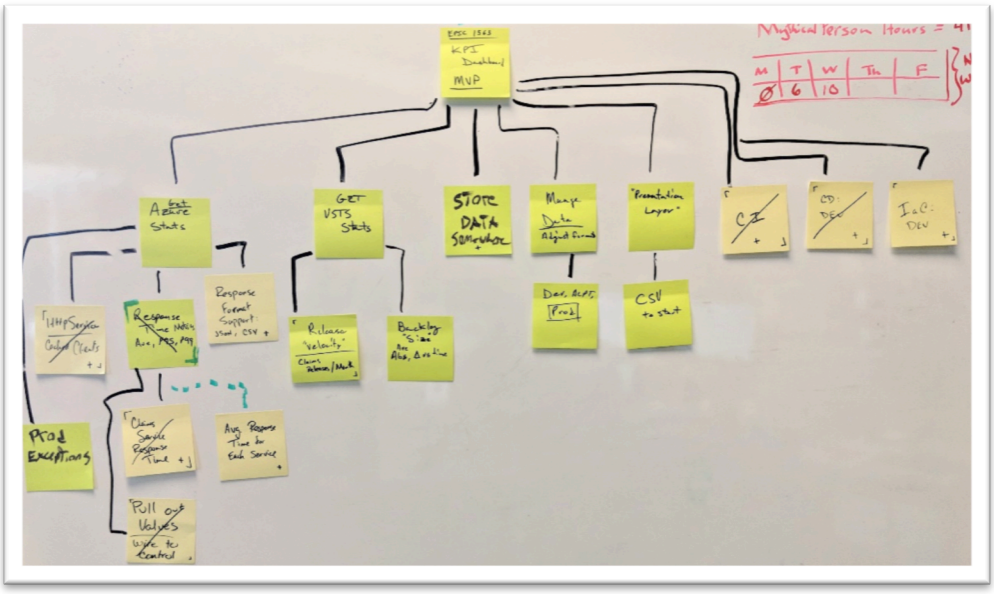


Figure 3. Feature Tree Example

### 4.2 Forecasting and estimation when teams are not static—the wisdom of crowds

The next challenge was forecasting and estimation. Scrum recommends that the team that does the work estimates the work. But in our environment of Dynamic Reteaming[5], there was no knowing who exactly would be working on which work item ahead of time.

While pondering this question, I also happened to be reading the Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations by J. Surowiecki[6]. Eureka! Let's try wisdom of crowds estimation. Wisdom of crowds forecasting works by collecting a guess from a large population, and then averaging them for a final estimate. I later learned that Dave Snowden, the father of the Cynefin framework[7], suggests that for problems in the complex domain (such as software delivery), the best way to estimate is to use the wisdom of crowds[8]. My hunches were right!

Using the wisdom of crowds, we could estimate the effort needed to complete each feature on the release map! Sum these estimates and you can calculate a target release date. To facilitate this process, we created a lightweight web tool. In a matter of minutes (versus hours or days of traditional agile estimation methods), we could forecast a feature or entire release!

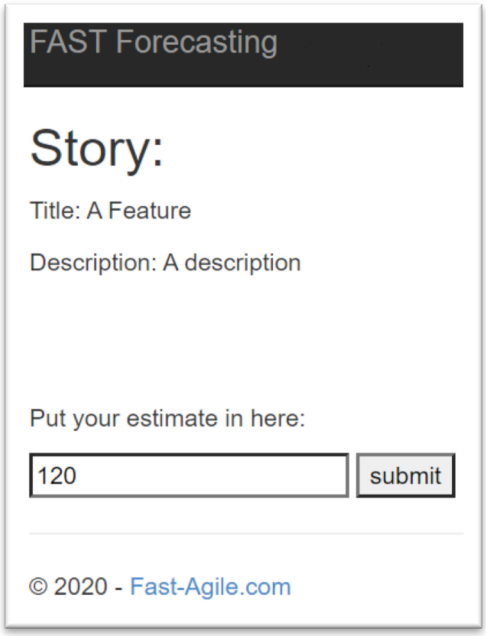


Figure 4. Screen Grab of Forecast Tool

### 4.3 Continuous Improvement at Scale—an Open Space Retrospective Event and Process Improvement Guild

*At regular intervals, the team reflects on how to become more effective,*
*then tunes and adjusts its behavior accordingly.*
*~ Principle behind the Agile Manifesto*

Next challenge: In a dynamic teaming environment, what to do for reflection? Team retrospectives were of little use as our team composition was fluid. We had heard about a practice of "turn up the good" at Hunter Industries[9] and decided to give it a go. We would start each iteration boundary meeting by highlighting which experiments in the last iteration went well and why. While there was value in this practice of turn up the good, it still felt more was needed.

The next experiment was forming a **guild for process improvement.** The guild turned out to be a great idea. At guild meetings, anyone could come and share observations and propose experiments. All experiments went through a ratification process by the entire tribe and we paced ourselves through these (not changing everything at once). And yet we felt something was still missing. There were unheard voices and grievances.

Eventually, we tried taking an entire day for the tribe as a **reflection event using Open Space** to facilitate. This event felt like it finally addressed what was missing, and I planned on running this event at some regular cadence going forward e.g. once a month. The theme for the Open Space event was - reflect and improve.

At an Open Space event, anyone feeling strongly about a topic can facilitate a session to discuss it with others interested and together might come up with action items or experiments. Or not, sometimes just a conversation is all that is needed. This approach was very compatible with our core value of self-organization.

### 4.4 Too Much Work in Progress (WIP) —Mob Programming and WIP limits

In the beginning, there was a tendency for a team pulling in work interpreted as a pair working on a story. The issue was partly my fault in sharing the analogy of an XP team able to complete a story in two days when we were deciding on iteration length. I talk more about iteration length in 5.2. The result was too many work items in progress. The **iteration board** (that we would fill during the marketplace phase of the iteration boundary meeting) was going wide, and progress felt slow and hard to track. Surprisingly, the fix for this came as a side effect of our ongoing commitment to mastery (of our craft). We had decided to get training on mob programming[10] and invited Woody Zuill in (as the inventor and expert). Mobbing became an instant hit and

most of the teams from that point on were mobs of 3-6 developers. A team was now more than a pair (though that was still allowed), work was larger in scope, and WIP reduced.

## 5. FINDINGS

### 5.1 Extreme Programming (XP) Supports Dynamic Team Formation

An enabling factor to our success with dynamic team formation was that in our **tribe agreement,** we specified all coding work completed using XP disciplines (pair programming, merciless refactoring, test-driven development etc.).

XP was a success factor to dynamic reteaming for these reasons:

- Pair Programming – XP developers are accustomed to switching pairing partners and contexts often. They sync up quickly and deliver value.
- Collective code ownership - makes jumping into any/all areas of the code as needed a usual practice.
- Coding Standards - as the style and syntax of the code become consistent throughout the code base, developers are very quickly able to read and understand the code throughout.

In *Dynamic Reteaming: The Art and Wisdom of Changing Teams* by Heidi Helfand [5], the author dedicated a chapter to Constraints and Enablers (of Dynamic Reteaming). Helfand here similarly noted an enabling relationship between XP Practices to Dynamic Reteaming.

### 5.2 Short Iterations Preferable to Long—The Two-day Iteration and Unshackling from Scrum Think

When we started and were contemplating cadence, I wanted to see just how short a timescale we could achieve. I thought back to my XP days when it was typical for a pair to complete a story in about two days. So, why not try two days?

We started with two-day iterations, but after a while, there was grumbling. I put it down to people still thinking in Scrum and its typical two-week or month-long iterations. So, we experimented with a one-week iteration and were surprised to find that after a few months of running such, the consensus was that two-day iterations were preferable, and we switched back!

We had to unlearn thinking in Scrum, and once we did, the agile landscape looked very different. It was now open and full of potential.

### 5.3 We Were Not Iterative—We Had Inadvertently Created a Flow System

The meeting we had at each iteration boundary would follow these steps:

- close out the last iteration with each team reporting a **show and tell** to highlight the delta to the system that was important for the rest of the tribe to know
- clear the iteration board
- **Product Director** addresses the tribe on what is important and has changed since the last meeting
- self-organize into teams around work using Open Space Technology

The iteration boundary meeting (**FAST meeting**) is the only meeting in the process and functions as a synchronization point for the tribe. (Typical length was 15-30 minutes.) There was no concept of getting a predetermined set of work done. Rather, the focus was on continuously delivering value and keeping the tribe in sync of the current state of the system as it changed. In this way, this synchronization meeting was closer to a huddle or scrum than a sprint boundary, if you were to compare our process to Scrum. We had set out to create an iterative system, but what emerged was we closer to Kanban or a flow system to our surprise.

### 5.4 Self-organization is Not for Everyone—The Need for an Offboarding Process

To my surprise and sadness, it turns out that not everyone yearns for autonomy. It broke my heart to think about what our schooling, society, and organizations have done to us, to have broken the spirit of some in this way. It became my life purpose from that moment on to "Untether the human spirit in the workplace." I do not recommend forcing autonomy on anyone. That is just as cruel as the opposite. I recognized the need to gracefully offboard anyone that wanted off or was not ready for working in a self-directed environment.

## 6. HOW AND WHY OUR SELF-ORGANIZATION AT SCALE EXPERIMENT ENDED

The experiment ultimately came to an end due to a handful of internal and external factors.

External factors. Our part of the organization reported to the CTO who granted us autonomy and protection from the broader company culture and ways of working. From the beginning, this created tension and it felt like we were under continuous scrutiny. When the CTO left, it was not long before there were demands for doing things in the same ways as the rest of the company. Gone was our skunkworks and startup environment.

Internal factors. As the tribe grew, there was a budget to hire middle managers with a ratio of 8:1 of individual contributor to manager. Some of the middle managers that we hired seemed still locked into a command and control mindset. Actions from these individuals soon affected morale and began eroding at our self-organization principles. For example, these middle managers pushed for static teams and team leads. Compare this to our self-organizing system of **natural leadership** and dynamic team formation, where individuals would volunteer to lead a **swarm** (our name for a team) and others voluntarily join.

Around the same time, we absorbed a team into the tribe from the larger organization when one of our products would sunset theirs. Many from this team struggled with self-organization and extreme programming. There entered a sense of subterfuge to collapse the system. We began seeing untested code and black-ops projects.

The combination of these factors resulted in my losing influence and interest in continuing. I had become disheartened and decided to leave the company as the impediments for autonomy became insurmountable. Enough of the battles were lost to know the war would soon also be lost. I left while on good terms.

My understanding is that the tribe stayed in place for some time afterward, with only part of the system. They have since been subjected to an enterprise agile transformation initiative and to the uniformity that comes with such initiatives.

## 7. LESSONS LEARNED

### 7.1 The Wins—Why I Would Do It Again

Resilience—The tribe made continuous progress regardless of vacation, sickness, and staff changes.
Adaptability—We were able to move effort to where it was most needed.
Employee Engagement— "I never want to work any other way," "This is my preferred agile method going forward" were some comments I heard during the experiment. I put this down to the intrinsic motivation that comes from autonomy at work. According to Daniel Pink in his book *Drive: The Surprising Truth About What Motivates Us*[11], autonomy is one of the key motivators of why people work.
It worked! —We were delivering continuous value.
Pioneering—Local businesses were coming to do tours of our process as word got out.
Lightweight—I was determined to keep the process meeting light. There was only one meeting, and we kept that short, under 30mins. (A common complaint about scrum is that it's meeting heavy.)
Natural leadership—This system allowed for anyone to grow their leadership skills if so inclined.
Low cost of ownership—We did not need scrum masters and had only one agile coach for the entire tribe.
(My theory is that a tribe's upper limit in size is Dunbar's number—150 people.)
Choreography over Orchestration—Having fixed teams requires an orchestration approach to work due to dependency management. By having fluid teams, we moved to a pattern where dependencies fell away as teams would move to other work when blocked or merge teams to remove dependencies. Fluid teaming also allows for limited skills to be shared across a tribe e.g. UX and design.
Common sense over explicit rules—For example, we did not need a definition of done.
Freeing the human spirit—Autonomy and self-organizing systems are a better way to treat adults in the workplace, particularly with knowledge workers. See 7.3

### 7.2 A New Scaling Framework is Born—Fluid Scaling Technology (FAST)

As we went along, I tried to codify the system, and we named the system **Fluid Scaling Technology** with the acronym **FAST**. You can read more about FAST at http://fastagile.io. (As of this writing, the site is in a minimum viable state.)

### 7.3 Ushering in a New Era—The Reinventing Work/Humanizing Work/Teal Movement

This experiment was my doorway into an exciting movement. It has set me on my next adventure beyond agile and is where my passion lies. This movement that has many names:

- Teal [12]
- Humanizing Work
- Human Movement
- Self-management Movement
- Next Phase Organizations
- Reinventing Work
- Responsive Org [13]

Perhaps one day, a group of pioneers will meet in a ski lodge and come up with a manifesto and agree on one name for this movement!

Because of the self-management aspects of FAST, I discovered that FAST is as much a Teal framework as it is an Agile Scaling framework.

## 8. ACKNOWLEDGMENTS

I want to acknowledge Paige Watson, Quinn Gil, and Steve Kuo, who each went above and beyond in helping make the experiment a success and joy. Acknowledgments are due to the whole tribe for sharing in my vision of a new way to scale agile and for people to work. Big thanks to my manager who shouldered the accountability and for believing in me, supporting the vision for as long as you could, and creating the starting conditions we needed to make this happen.

These two books deserve mention as that heavily influenced my thinking:

- *Turn the Ship Around!: A True Story of Turning Followers into Leaders* by L. David Marquet,
- *Team of Teams: New Rules of Engagement for a Complex World* by Gen. Stanley McChrystal.

Other key inspirations and supporters. Woody Zuill for giving the world Mob Programming and just being you. I want to be you when I grow up. James Shore, Alan Dayley, Kat Daugherty and Lance Kind for your pioneering spirit and getting the word out. Harrison Owen for Open Space. Margaret Wheatley for ushering in a new age. Kent Beck and the XP founders—it is the right way to code. Frédéric Laloux for introducing me to a movement I was previously unaware. Ricardo Semler for planting the first seed of self-management in my mind in 2002 when I read *The Seven Day Weekend*. Niels Pflaeging for the notion of decentralized control and decision making. Doug Kirkpatrick for sharing your story with the world, showing us that self-management at scale is feasible and viable. Heidi Helfand for Dynamic Reteaming and being a voice of reason in the agile community. Finally, the Open Space Institute and the broader Open Space community, I have found my tribe and my passion. Thank you.

REFERENCES

[1] OpenSpaceWorld.ORG. *What Is Open Space Technology?* [Online]. Available at: https://openspaceworld.org/wp2/what-is/. [Accessed 20 June 2020].
[2] K. Beck. *eXtreme Programming eXplained: Embrace Change*, Addison-Wesley. 1999.
[3] "What is Story Mapping?," Agile Alliance, [Online]. Available: https://www.agilealliance.org/glossary/storymap/. [Accessed 20 June 2020].
[4] Q. Gil, "Feature Mapping: A Simple and Natural Way of Story Creation, Project Tracking and Shared Ownership," [Online]. Available: https://quinngil.com/2020/02/03/feature-mapping/. [Accessed 20 June 2020].
[5] H. Helfand, *Dynamic Reteaming: The Art and Wisdom of Changing Teams*, O'Reilly. 2020.
[6] J Surowiecki, *Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, Anchor.2005.
[7] "Cynefin framework," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Cynefin_framework. [Accessed 20 June 2020].
[8] D. Snowden, "Applying Cynefin to Kanban," 2019. [Video]. Available: https://youtu.be/GbF5OT32B-8. [Accessed 20 June 2020].
[9] W. Zuill, "Mob Programming Experience Report," 2015. [Online]. Available: https://www.agilealliance.org/wp-content/uploads/2015/12/ExperienceReport.2014.Zuill_.pdf. [Accessed 20 June 2020].
[10] W. Zuill, *Mob Programming: A whole Team Approach*, Leanpub. 2020.
[11] D. Pink. *Drive: The Surprising Truth About What Motivates Us*, Riverhead Books. 2011.
[12] F. Laloux. *Reinventing Organizations: A Guide to Creating Organizations Inspired by the Next Stage of Human Consciousness*, Nelson Parker. 2014.
[13] "Responsive Org Manifesto," [Online]. Available: https://www.responsive.org/manifesto. [Accessed 20 June 2020].