Dynamic Reteaming: How We Thrive by Rebuilding Teams

HEIDI HELFAND, AppFolio, Inc.

Who says you need stable teams in order to build a successful software company? We've thrived through dynamic reteaming - the act of moving team members around teams in different ways. While the addition or removal of one person from a team means you have a "new team", there is a myth that when your team compositions change you're doing it wrong. Reteaming helps improve your organization, your code and most importantly your people. For the past 9 years, from startup to public company, reteaming has been critical to our success. Reteaming has propelled our engineers' learning. In this experience report, I share the what, why and how about reteaming.

1. INTRODUCTION

In the Tuckman Model for group development (1965) Bruce Tuckman asserted that teams go through: Forming, Storming, Norming, Performing and Adjourning. (1) It's a little known fact that Tuckman forgot a stage in his model. It's the stage called Stagnating, or when you let your teams continue on "forever" and don't change them. I didn't make up this concept, it emerged from the study I did at my company AppFolio, Inc.:

"You just get different perspectives working with new people...unless you actively have a source of new input to your team, like you're reading books together or something, it's going to stagnate over time..." -Bryce, PhD, Senior Software Engineer

"If the team stays stagnant, the abilities you have stay stagnant. We have people on the global engineering team for a reason. They're good at different things. They're good team members. And so mixing it up all the time is important." -Comron, Principal Software Engineer

According to one of AppFolio's founders, Klaus Schauser, PhD, and computer science professor at the University of California, Santa Barbara, our company was started with the vision that we would build an organization that would continually learn and adapt. Klaus's cofounder, Jon Walker, shares the vision and refers to the company as being built on feedback loops starting with test driven development and continuous integration. So how do we create a learning organization? We need to nurture and develop our people and teams.

At the individual level, we learn how to find the right team "fit" for our engineers continuously via feedback loops like in one-on-ones with managers, periodic surveys and through retrospectives. These activities aim to help our engineers find fulfilling work so that they are always learning. As Peter Senge says, "Organizations learn only through individuals that learn." (2)

At the team level, we are on the relentless pursuit to find the *optimum* collection of individuals working together to achieve a common goal. We want our teams to thrive and deliver incredible value to our customers while having a meaningful and enjoyable experience. One way we seek to find that chemistry is through reteaming.

I define reteaming as: when you change your team composition. Reteaming could be as simple as the addition of one team member, or the removal of one team member. (3) As we will see, it could even be "radical", with pulling team members off of multiple teams to form a new team. Reteaming is the opposite of keeping your teams the same.

Adding just one person to a team can make a huge impact to the learning present on the team, since this person brings the collection of prior experiences they have had developing software with others. Besides impacting the learning, the addition of just one person can impact the culture of the team because people bring their interests and hobbies with them.

Author's address: Heidi Helfand, AppFolio, Inc., 50 Castilian Drive, Goleta, CA 93117; email: Heidi.helfand@gmail.com

But wait, you might have heard people say that you should keep your teams the same. You might have heard that reteaming is costly. Or that for predictability, stability is encouraged. There is, in fact, a Scrum pattern that says, "Keep teams stable and avoid shuffling people around between teams. Stable teams tend to get to know their capacity, which makes it possible for the business to have some predictability." (4)

That stance is way too narrow. The fact is, if teams aren't producing incredible value and learning continuously, and if the team chemistry is off, it's to everyone's benefit to reteam. Let's look through the lens of fulfillment, or as software development coach (Michael) GeePaw Hill calls it, Geek Joy. We want to have inspired engineers who have choice in their work and in their teams. We want to help them thrive and feel excited to come to work every day. Reteaming helps with this.

Possible Misconceptions about Reteaming

I'm not saying don't retain people. If I look back at our earliest company photos, I can see that I still work with the majority of the core founding team. These team members have been learning and adapting together for 9 years.

I'm not saying reteam at the fastest, most dynamic speed all the time. You can dial up or dial down your reteaming to meet human needs for change and learning, and to meet business needs for growth and challenges. Some individuals might want more change, while some might want less. The business can choose to ramp up or slow down hiring. These are all deliberate decisions. Step on the "gas pedal" accordingly..

2. BACKGROUND

AppFolio, Inc. is a SAAS company with products for running your property management business, or your law firm. We were voted as one of the best places to work by our employees in 2016 by GlassDoor. We went public in June 2015. In terms of product development and engineering, we've thrived through reteaming. And, we've been Agile since our first team - our roots are in XP.

In general, our teams are comprised of a nucleus of roughly 3-4 engineers and 1 QA. Product Managers, and UX are typically at a 2:1 ratio with teams. Agile coaches might be with 3-5 teams each, assigned to "Colleges," which might be similar but not identical to Spotify's Tribe concept. (5) Each college has an engineering director and possibly a tech lead. Engineers report to either of those roles.

Our teams are primarily feature teams that can work on any part of our applications. We have collective code ownership, test-driven development, continuous integration and weekly releases. We also have specialized teams that build and maintain our data centers and that do tech support.

Traditionally work has been assigned or "pushed" to the teams. In the past year this has shifted, and it is becoming more common for teams to "pull" work from our backlogs.

To better understand and express what we do, I decided to do a qualitative study with a basic grounded theory approach. (6) I interviewed 16 engineers, resulting in 16 hours of data. Each participant told me stories - how we've formed and reformed teams over the years. The initial findings I presented at Agile and Beyond in Detroit in May, 2016.

More than 27 reteaming patterns emerged out of the study. I break them into three general categories based on why we did the reteaming: 1) for business reasons, 2) for code reasons and 3) for human reasons. Due to space constraints, I share a sampling of the patterns. Further reading on this topic is offered in my Reteaming book. (7).

3. THE BUSINESS NEED TO RETEAM

From growing a startup to creating new products, reteaming has been a successful strategy for AppFolio to thrive. Here are patterns related to business needs.

3.1 Growing the Company

3.1.1 Put a new engineer on a team with existing engineers.

This has been the most common pattern in the last 9 years. A new person is hired, is assigned a mentor on an existing team. Their mentor is in charge of their onboarding, and serves as their "first pair." As Andrew our Chief Scientist says, "If a new engineer joins a team and they have the perfect mentor and the situation with that mentor can give them a lot of interaction and a lot of learning and feedback, then a year later...two years later...they are going to be a much better engineer than they would have been if they just joined any random team, a team that didn't have time for them or a team where there wasn't enough mentorship."

3.1.2 Cell division: Grow a team "big," and split it in half.

Our first team grew to about 15 people. Events such as sprint planning started to take forever. So we split the team in half, and things became more manageable. The downside was less variety in pairing, so in hindsight it was a threat to learning. The rotation pattern mentioned in section 5.2.1 accounts for this. Nevertheless, this cell division pattern has been common since our first team. Mike Cohn in Succeeding with Agile talks about spreading Scrum using a similar pattern that he calls the grow-and-split pattern. (8)

3.1.3 Higher-level cell division. Grow a college "big" and spawn a new college with one team.

This is a management pattern and is a deliberate way to form communities that know each other. We cluster groups of teams together into Colleges. Colleges sit together in the same area, separated into teams. Engineers in each college are managed by an Engineering Director, or a hands-on Tech Lead within the college.

As the teams multiply, we grow or hire engineering directors who first join teams as regular contributors. Team members know that they are on their way to becoming directors of a new college. When ready, that director and team break away to spawn a new college. Other teams will emerge through time, most likely seeded with existing engineers from the original team.

Colleges facilitate localized culture building. It's a way to get the feel of being at a smaller company as your company grows in size. It's a proactive way to combat the Dunbar number. (9) People get to know others in the college, which facilitates future reteaming.

3.2 New Product Development

3.2.1 Take engineers out of current teams. Put them into a new team. Isolate them from other teams. Tell people not to bother them. Give them complete process freedom.

After we created our property management software, we applied this reteaming pattern to create our second product which was a secure online data room for mergers and acquisitions, called SecureDocs. Comron recalls: "...we had a goal which was outside the day to day of the engineering team. ...It was kind of a blue sky project. We were building this whole new product. We did market validation. We experimented a ton....And on that team not only were we completely separate but we totally changed the process. Instead of sprint planning and Scrum and two week or four week sprints...we said listen, we can't predict what we will be doing two days from now, sprint planning doesn't work in that world, stories go from being 8 points to 1 point overnight because we learned something brand new...and so it doesn't necessarily work planning that far out in the future so we switched to more of a flow based process where we had a backlog, we kept it in priority order, and we just took stuff off the top and we just constantly iterated. ...We couldn't necessarily commit even one week at a time. And so we went down to basically one hour sprints. And we would take a story off the list and we would learn something by doing that story … and so it was just a completely different process than the rest of the team. ...we didn't necessarily know how it was going to work, and so splitting it off to a separate team had its own feedback loop outside the bigger team allowed us to experiment more."

SecureDocs is a successful product that became a separate company that exists today. This is one example that dispels the myth that in order to succeed you need to have stable teams that don't change their composition.

3.3 Spreading around the "high performance" anti-pattern

3.3.1 Move existing team members out of a "high performance" team in order to spread out the "high performance" among the other teams.

Our CTO and co-founder Jon tells the story of when we had three teams. One of the three teams stood out as being highly productive and delivering a lot of customer value compared to the other two. Jon decided to split up this high performing team with the thought that he could spread that performance out among the other teams. The result was that he got three "OK" teams instead. In his words, "I broke up that team, and I regretted it afterwards."

Finding high performance - the synergy and the feel of an incredible team is a magical thing. When it happens, and you are on the team, you know it's there. Jon describes it as a team that says, "Hey, there's new stuff that we're excited about, and can learn here...what if we did this? What if we did that?" He further says that he noticed a high performance team sitting outside his office…"they're always noisy and celebrating stuff and talking with each other...you can feel it and it feels really different." Jon compares how it feels when you have a team that is mediocre, "The energy feels low... If you're in meetings with them it feels like, 'Ah, we've got to get through this.'"

Why doesn't high performance happen with all teams that come together? What we learned from Jon's story is that when there's incredible performance and energy, it's best to keep that team together, or you could destroy the magic.

So what if it's not there? What if you have a team together for a while and their output is mediocre and the energy is low? You might explore splitting up that team. Keeping a non-resonant combination of people together could lead to misery and stagnation. As (Michael) GeePaw Hill says, "the quest for geek joy **is** the quest for high performance. there might be such a thing as a happy team that doesn't produce, but there's definitely no such thing as an unhappy team that performs at a high level."

4. THE CODE'S NEED TO RETEAM

Over the years we have reteamed to meet the needs of our code. This could be viewed as the inverse of Conway's Law. (10)

As Michael Feathers said, "...different areas of code are going to need different skillsets at different times, and for that reason we need to be able to organize and galvanize the people to go and work on those particular areas at different times, and then if the code needs something else, then different people go to different areas to work on different things. None of this is basically static. It's very fluid. And if you have the ability to reteam, then you are able to go do this sort of thing consistently." (11)

Here are a few examples of code-driven reteamings that we have experienced.

4.1 Focus on technical debt and sustainability

4.1.1 Take engineers out of standard teams, put them in an infrastructure team, give them process freedom. Let them work on what the code needs.

With our early teams, work would get assigned to each team on a relatively random basis. We built the essential features for our property management customers and as we grew there continued to be many hands shaping the application in its entirety. The result was that it had become increasingly hard for any one person to have the entire system in their head. Donnie is a Software Architect and was one of the proponents for forming our infrastructure teams. As he puts it, "...you work on this feature and two weeks later someone else works on it so over time what happens is that not one person has the whole thing in their head. It takes extra work. You can't really expect any new person to come in and make a change in this thing for the better. They need to understand all the moving parts. You need a team that has the time and skills to understand a complicated system like that and make changes that are better holistically for the system."

4.2 Address challenges posed by the code

4.2.1 Take team members off of existing teams to solve a short term challenge. Isolate the team and give them process freedom. Dissolve the team when the challenge is accomplished.

Before releasing our first product, some team members were pulled out of existing teams into a new team to identify performance issues. This team isolated themselves, identified and solved key challenges, and then dissolved.

The reteaming here is to accommodate different levels of uncertainty in the work at hand. The higher the uncertainty or unknowns, the smaller the feedback loops required. In other words, the performance team required daily to hourly iteration on their work tasks. The teams they came from used larger feedback loops of two weeks. If both types of work are done in the same team, the chances of becoming a "prisoner" in someone else's planning and standups is high when you combine work like this together. Splitting off into a separate team worked here. According to Principal Engineer Comron, "...sometimes you have problems that don't necessarily fit the process that you have. And it's easier to change the process in a small team than to do it in the context of a bigger team where you have to explain what you're doing and it doesn't necessarily fit into the daily routine that everybody else has…"

5. THE HUMAN NEED TO RETEAM

Human needs for reteaming include the need to find fulfillment, to learn, and to liberate others from undesirable situations.

5.1 Reteaming to Find Fulfillment

5.1.1 Switch an engineer to another team in order to work with the people or content they want.

We reteam with the aim of helping developers find fulfillment in their jobs. It's finding opportunities to work on the things that excite them and bring out the best in them. It is the opposite of feeling bored or unmotivated. The practice of having one-on-one walks with a manager is one way that we tap into needs. As our Chief Scientist Andrew Mutz put it, "[An] engineering manager has a continuous conversation with the

engineers that he works with about what they enjoy, what they don't enjoy, what their ambitions are in life, where they are trying to educate themselves, what they are passionate about, all these sorts of things…and it's not always the case that a given group of engineers working on something can be matched up with the perfect project to satisfy everyone. Sometimes the right way to maximize global happiness, global morale ...is to adjust the team composition….so matching engineers with what they are passionate about is a really common reason that we would adjust these team compositions."

5.1.2 Hack Days: Temporarily break up all teams for 24 hours. Facilitate activities to derive content and team formation.

We reteam for a special event twice per year that we call Hack Day. For this event, we invented brainstorming activities to derive topics, as well as a "marketplace" activity where people cluster around topics that are of interest to them to form teams themselves. Teams result which stay together for 24 hours to build the project they dreamed up. At times people join multiple teams. After Hack Day these teams dissolve and people go back to their regular teams. We learned the basics of this from our friends at Atlassian who call it "ShipIt Day." Our Hack Days are highly successful events, leading to feature enhancements, internal tools and an abundance of relationship building that makes future reteaming easier.

5.2 Reteaming to Learn

Another human reason that we reteam is to encourage learning from the people around us. If we have access to more minds, we will learn more. In this section I share three patterns related to learning.

5.2.1 On a regular basis, rotate an engineer from one team to the next in order to stimulate learning and camaraderie.

In our early days with our first team, we created our property management application from scratch. Our first team grew to about 15 people and then split in half. We hired more engineers, and created a third team comprised of existing and new team members. This three way split limited the pairing variety. And so began the practice of rotating team members from one team to the next in a loop. As Principal Engineer Comron described it, "It was really good because you had that momentum with your team and you knew what everyone was working on and we would sit right next to each other and communication was super easy but then every few weeks you would get new blood you would get new ideas, you would get new faces and these are people you would see every day in the office obviously but I was on a team [and] I really liked working with Donnie, but Donnie was on another team but I knew that in a couple weeks he might be on my team and we could do something new together. "

5.2.2 Switch engineer from a specialized team like tech support onto another team in order to gain knowledge to bring back to their team.

Our tech support team is comprised of software engineers that handle escalated customer service requests. If we have enough engineers staffed on that team, as a growth opportunity, an engineer can rotate to a feature team. These software engineers are able to keep up with development practices through visiting a team. They also can get a change of pace, and develop empathy for the feature developers.

We have moved tech support software engineers over to feature teams while critical features were being developed (e.g. payment features) so that when these engineers returned to tech support their knowledge of the feature would be very in-depth, making the new feature easier to support.

It's happened in the past that our tech support engineers love the feature teams so much that they don't come back to the tech support team. It's really a negotiated "trade" between the directors of both groups. If there are enough tech support people staffed, and if there is a need and fit on the other team, this sort of thing can happen.

5.2.3 Switch engineer from an existing team over to another team for a short term in order to share specialized knowledge with the other team

We have redundancy in our feature knowledge to an extent through pair programming. We have built in redundancy at the team level by having multiple teams understand certain key features. It works like this. An engineer will join another team for a few weeks as the feature expert and will pair with people on the other team as they are building onto the existing feature. This expert engineer will teach via pairing and answering questions that the team has regarding the feature. They transfer knowledge. When they feel like it's time, they leave and go back to their regular team.

5.3 Reteaming to Liberate

Besides fulfillment and learning, liberating people from undesirable situations is another reason for reteaming. This requires being present and active with teams to notice behaviors like silence or work among team

members that is not discussed at standard team meetings. It relates to reading the social and emotional field of teams.

5.3.1 Acknowledge that you have one team embedded within another team. Break them into a separate team and "reset" the team to help them find their process.

I had a retrospective one day with an infrastructure team and it came up that a developer's work was not being discussed in any of the standups or planning meetings. He was essentially "along for the ride" despite the fact that he was working with a consultant on a specific type of work. As a result of that retrospective, we decided that he would form a separate team with the consultant, and together they determined the appropriate process that worked for them for discussing, tracking and making their work visible to the organization. That was the seed of a team that has now grown.

Another moment of liberation was the beginning of our tech support team. Starting with one to two engineers, they were embedded within our web operations team simply due to the fact that they had the same manager. A similar thing happened to them. The standups and planning meetings were dominated by the discussions of some "other" work that had nothing to do with their daily work or goals. They were just present at those team events without active participation. Accordingly, they were liberated out of that team, and then found the process and daily mechanics that helped them meet their own objectives.

I don't believe that it is deliberate or intentional to have silenced teams within teams. What I've noticed is that, instead, one or two people are serving as the "seeds" of a new team within another team, and as more people get added, teams just emerge and then they split off. We can make that reteaming easier. We can get good at spotting and facilitating this kind of reteaming if we read the social and emotional field while present with those teams. It's a type of Anzeneering. (12)

6. CONCLUSION

A key takeaway I have regarding reteaming is that as an industry, we need to think critically about dogma like "keep teams stable and unchanged." If the chemistry is "off" or is negative on a team, try reteaming. If a team member is completely demotivated and is unsatisfied working with a particular team, find them a better team fit. If a team is highly productive and they love working with each other keep them together. It's not always clear cut, however, and that's where reading the emotional field of the team and paying attention to the humans is imperative. Strive for Geek Joy.

"Reteaming is inevitable. You might as well get good at it." That's what my friend Lean/Agile coach Nayan Hajratwala said to me recently, and I agree strongly. The fact is, people will come and go from your company and your teams. Life happens. We've all experienced having a new person join, and we've experienced the loss of a team member. We can tap into the people in our companies and nurture them to grow and thrive. Reteaming is a natural result - it just might happen at different rates for different people. We can do deliberate practices to make reteaming easier - I've written about that in my Reteaming book. (7)

In this experience report I've shared a sampling of reteaming patterns that have emerged at my company. Have you experienced any of them? Have you discovered any others? Want to connect with others who have reteaming patterns like you have in your company? I invite you to email me at heidi.helfand@gmail.com. Check out my web site for my Reteaming book and for future work in this area. http://www.heidihelfand.com

7. ACKNOWLEDGEMENTS

I'd like to thank the following people for their thoughtful comments on drafts of this paper: Josh Kerievsky, Mark Kilby, (Michael) GeePaw Hill, Sandy Mamoli, Rahul Sawhney, Michael Feathers Rene Rosendhal and Regina Rodwell. I'd also like to thank AppFolio for being supportive of this research. I would also like to thank the participants in this study who helped me learn what reteaming is all about.

REFERENCES

(1) Tuckman's Stages of Group Development: https://en.wikipedia.org/wiki/Tuckman%27s_stages_of_group_development
(2) Senge, Peter M. "The Fifth Discipline: The Art and Practice of the Learning Organization." Deckle Edge, 2006.
(3) My colleague Paul Tevis would always say this. So does Organizational Relationship Systems Coaching (ORSC).
(4) Scrum PLoP Stable Teams Pattern https://sites.google.com/a/scrumplop.org/published-patterns/product-organization-pattern-language/development-team/stable-teams
(5) https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/
(6) Brown, Brene. Daring Greatly: How the Courage to Be Vulnerable Transforms the Way We Live, Love, Parent, and Lead. Avery, 2015. Appendix - Trust in Emergence: Grounded Theory and My Research Process. P. 251.
(7) Helfand, Heidi. Reteaming book. See heidihelfand.com for further information.
(8) Cohn, Mike. Succeeding with Agile. Addison-Wesley: 2010.

(9) Dunbar's number: https://en.wikipedia.org/wiki/Dunbar%27s_number
(10) Conway's law: http://www.melconway.com/Home/Conways_Law.html
(11) Feathers, Michael. "A Technical Keynote", Video: https://www.youtube.com/watch?v=3AMzRAjA0-o
(12) Anzeneering, Joshua Kerievsky, Industrial Logic. https://www.industriallogic.com/blog/anzeneering/