June 2019

Scrum.org | Whitepapers

Enable Scrum Teams to Embrace Team Composition Changes

By Roland Flemm

In the Scrum and agile world, we commonly agree that the pattern of changing team composition frequently is undesirable because it leads to performance degradation (Scrum PLoP! [1]). The Scrum Guide elaborates on team composition and enumerates team design characteristics:

"The team model in Scrum is designed to optimize flexibility, creativity, and productivity."
- The Scrum Guide (Sutherland and Schwaber)

Designing a team for productivity implicitly suggests we aim to keep a team's composition as stable as possible. The Large Scale Scrum (LeSS) framework is more explicit about this and states as its primary rules:

"Structure the organization using real teams as the basic organizational building blocks. Each team is (1) self-managing, (2) cross-functional, (3) co-located, and (4) long-lived."
-LeSS Guide (Vodde and Larman)

The concept that teams need to be long-lived to become performant originated a couple of decades ago. Numerous scientific studies and research on team dynamics and performance learn that it takes teams about 1.7 years for teams to become performant and that they can remain performant for about 4 years under the condition that the team remains unchanged (Katz) (Wang, Ge and Feng). However, in today's time and in our industry it is uncommon for teams to remain unchanged (not replacing a single team member) for such a period of time. More recent studies investigate consequences of team stability in new product development teams.

Observing that constant change in team composition is a fact, while knowing stable teams are more performant, we need to discuss how we can enable the teams to mitigate the performance degradation caused by team composition changes.

When is a Team "Stable"?

A team is a group of people with a collective responsibility where members are interdependent and share a common goal. A stable team has a few additional characteristics:

- Individuals on the team only belong to one team.
- The team stays together for a longer period of time.

What is "a longer period of time"?

Various researches conducted since the 1980s that are considered as the industry standard state that performance of teams is highest in teams that were together for 2-4 years (Katz) (Wang, Ge and Feng). In other words, performance measured in relation to the churn-rate shows teams perform best between 2 and 4 years of stability. In reality, we don't see many teams celebrating their third year of unchanged existence: People change jobs, contractors get hired and fired, teams get disbanded or reorganized due to organizational changes.

Why Do We Want Stable Teams?

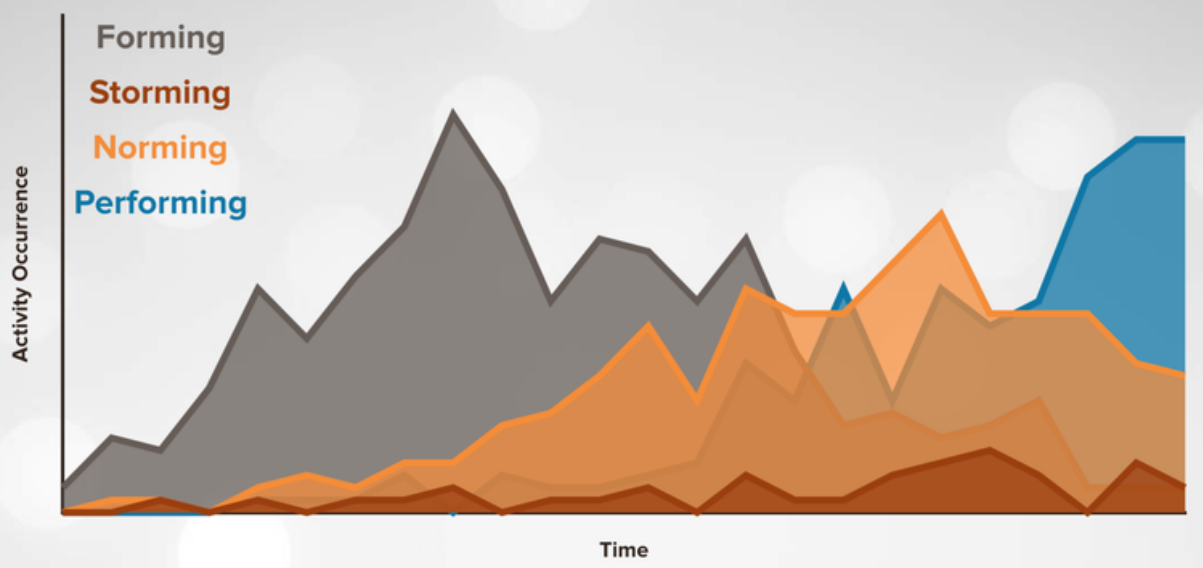
There are various reasons why we prefer stable Development Teams. A few of them are:

- Stable teams enable agility in your organization because the members stay long enough together to become multi-functional specialists
- Stability in team composition reduces variance in team predictions which makes them more predictable towards stakeholders
- A stable team builds a collective identity that can be a foundation of shared sense of pride both in the product and in belonging to the team
- Agile teams need stability to flourish and realise their full potential. Stable teams are happy teams
- Stable teams have higher throughput

What Is the Problem with Changing Teams?

Changing team composition can impact performance. The effects can be both negative or positive (Akgün and Lynn).

The idea that changing teams is bad for performance comes from models like the one Tuckman (Tuckman) created. In his model for team maturity, Tuckman suggests that teams move through the stages of forming, norming and storming before being able to perform. Considering the speed at which team compositions change, our teams would be in constant norming and storming phases according to Tuckman. In a 2007 research study it is stated to not consider Tuckman as a linear model but rather a continual process (Knight). Team stages norming, storming and performing are fluid and alternate, which later models such as "TEAM" describe (Morgan, Salas and Glickman). It may very well be that small team changes like changing a single team member provide exactly these team dynamics. In my personal experience, people sometimes just gel immediately and other times they don't (even after working years together). Tuckman's model is useful to understand team dynamics, but it's not an immutable law of team evolution.



[Image 1] Storming is not a stage, but a consistent occurrence (Norton)

In practice, we see that changing teams can be damaging when done without a clear purpose, mechanistically, without good communication and without the input of the people involved.

Reteaming as a structural model such as matrix organizations for sourcing projects increases the effort and cost of resource management and does not generally create high performing teams due to related effects like task switching. Also consider expensive budgeting and cost accounting practices related to these approaches.

Reteaming is often used by management as a quick fix to solve team problems. Using reteaming in this context is often hiding the real problem instead of revealing it and is harmful because it creates an unsafe environment. Moreover, when people do not decide for themselves what product they want to work on or people they want to work with autonomy is limited and intrinsic motivation decreases.

However, changing the team composition can be used as a conscious strategy to boost team performance. Image the case of a misfit of personalities in a team. Replacing a person might stop the team from wasting energy on unhealthy conflict and works as a liberation. Reteaming is considered good from the perspective that it decreases the knowledge silos by creating redundancy of knowledge, hence reteaming creates more flexibility and reduces risk.

We might also argue that reteaming reduces team member attrition by providing career growth opportunities. Teams that remain in relative isolation for long might develop tunnel vision, settle in patterns they don't see, are not triggered with new technology and insights and develop groupthink. However, Hackman (Hackman) (Coutu) states on this:

"Perhaps the most common misperception about teams, though, is that at some point team members become so comfortable and familiar with one another that they start accepting one another's foibles, and as a result performance falls off. Except for one special type of team, I have not been able to find a shred of evidence to support that premise. There is a study that shows that R&D teams do need an influx of new talent to maintain creativity and freshness – but only at the rate of one person every three to four years."

Recommendations to Retain Team Performance While Reteaming

Things nowadays in the IT industry are very different than back in the 1980s when most studies were conducted upon which we base our current practices. People change jobs more frequently, knowledge and skill requirements are more demanding, the way we work in teams in agile environments provides different dynamics, etc. The result is that long-lived stable teams are non-existent, therefore we need to prevent our teams from suffering from performance degradation due to reteaming.

To create optimal circumstances that limit the negative impact of reteaming, consider the following recommendations:

- **Keep teams small.** Research consistently shows that large teams underperform, despite all the extra resources they have. That's because problems with coordination and motivation typically chip away at the benefits of collaboration. Optimal team size is around 6 members.

- **Equip the team with a vision, mission and goal,** give them a shared, compelling direction. Goal clarity and stability is positively associated with team stability (Akgün and Lynn). In my experience I see that teams gel better when they have a clear common purpose. In many, such teams have ownership for some component. Although these teams are stable and cohesive through this ownership, this kind of ownership has downsides. I have discovered that it is worth the effort to replace component ownership by product level ownership to increase organizational flexibility and to avoid local optimisations.

- **Ensure safety** for all teams (Duhigg).

- **Provide a product-oriented organizational design** with product teams instead of project teams. Although a project gives a clear goal and provides focus to teams, the lifespan of projects is relatively short. Creating and disbanding teams with the life cycle of projects is undesirable and costly. I have seen teams with high stability in near-shore setups, where a single team works on different short-lived customer projects. However, these teams remain stable and productive because their organization does not break them apart when a project is over. They remain together and work on another initiative. The organization enables these teams to invest in a long-term inter-team relationship, broadening their abilities and skills and domain knowledge.

- **Teach teams how communication works** so that a team understands communication basics among each other. Most developers are not great communicators. When I work with teams as a Scrum Master, I experience that investing in soft skills by teaching the team members the power of communication reduces stress and builds a safe environment (Wikipedia).

- **Teach teams how to absorb new team members.** We all share experiences of being assigned to a new team. Teams can use those experiences and combine their experiences into an assimilation routine. Scrum Masters can encourage them to make a concerted effort to onboard new team members. Through conscious practice, they can become good at absorbing new colleagues.

- **Startup teams properly and create team charters.** In order to have teams gel, let them formulate their norms of conduct (Scrum PLoP! [2]) and make them gel by discussing the

knowledge each brings to the table from the start. This changes the criterion for power from social influence to informational influence. It makes sense to share and align these charters across the company to create a tangible representation of the company do's and don'ts, call it culture manifest (Adkins).

- **Make team norms actionable.** After starting up, give the norms a prominent place to continuously help the team improve their collaboration. A good example of this I once encountered was a team that had "Stay curious" in their charter, and selected new team members on this criterium. In some industries like aviation, norms of conduct have been formalized into strict protocols and to such a degree, that the possible impact of performance degradation through miscommunication is almost eliminated. Airline companies like KLM do this because it is impossible for logistical reasons to create stable cockpit crew composition, and hence they need to optimize speed of "Team gelling" without performance degradation leading to safety risks.

- **Carefully choose team composition** to provide a healthy balance in diversity and people with the traits that create successful agile teams (Aghina, Handscomb and Ludolph).

- **Promote team culture throughout the organization:** Align HR and product teams to reward team behavior instead of focusing on individual rewarding and promotion strategies.

- **Develop reteaming strategies** using reteaming patterns to support company growth or change so that new teams can be formed gradually, contain original team DNA and are already accustomed to being together (Helfand).

- **Temporary reteam.** Do not avoid reteaming for temporary purposes, but rather stimulate it purposefully and communicate its temporary nature explicitly. For example, use the "Volunteer Fire Department" team model where teams volunteer members to create a temporary task team to solve a key issue and then re-join their team. I have seen many cross-team issues being solved through this approach. It is a morale stimulator, productivity booster and promotes learning and cross-team bonding.

- **Apply high-collaboration practices.** Mob-programming is a great way to share learning by making software development a true team activity (AgiliX). When this practice is applied regularly or when it is the team's default way of working, they will be able to absorb new team members fast. I have seen mob-programming increase the team togetherness already after a single day experiment.

- **Create communities** to actively spread knowledge across teams to prevent knowledge silos that hinder team adaptivity. This strategy includes communities of practice (such as chapters and guilds), Trading Places or Travelers (concepts used in LeSS) where teams temporarily exchange team members to share learning across the organization.

- **Frequently retrospect the effects of reteaming** to enable inspection and adaptation of the reteaming activities.

Conclusion

Being agile means adapting to change. To limit negative performance impact of team changes, we need to apply the "adapting to change" concept to our team composition as well. There are numerous ways to equip our teams with the skills to better absorb team member changes. We can also put our efforts into creating an environment where our teams can flourish and thrive so that they are stimulated to stay together and the effects of team member changes are reduced.

Thank You

I would like to thank the following people for their contribution in the reviewing process of this paper: Cesario Ramos, Eric Naiburg, Lindsay Velecina, Illia Pavlichenko, Irina Skrypnyk, Jowen Mei, Pascal Dufour, and Alexey Pikulev.

About the Author

Roland Flemm is an independent product development consultant. He works as a consultant on large- and small-scale Scrum adoptions and as a Professional Scrum Trainer. Roland has a background in application development (.Net and Java) and in infrastructure engineering for over 20 years. Since 2013 he is dedicated to working as a Scrum Master and LeSS Coach. Besides coaching, he regularly publishes articles, he developed the Scrumcards (www.scrumcards.org) and is the creator of humorist agile movies. Roland enjoys speaking at international conferences and strongly believes that fun and diversity at the workplace is essential for success.

You can contact Roland at roland@agilix.nl

References

Adkins, Lyssa. *Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Addison-Wesley, 2010.

Aghina, Wouter, et al. "How to select and develop individuals for successful agile teams: A practical guide." Survey. McKinsey & Company; Scrum.org, 2019. < https://www.scrum.org/resources/how-select-and-develop-individuals-successful-agile-teams-practical-guide>

AgiliX. *How you can accelerate knowledge-transfer with cross-location MOB-programming*. 23 May 2019. <https://agilix.nl/blog/how-you-can-accelerate-knowledge-transfer-with-mob-programming/?lang=en>.

Akgün, Ali E and Gary S Lynn. "Antecedents and consequences of team stability on new product development performance." *Journal of Engineering and Technology Management* 19.3-4 (2002): 263-286. < https://www.sciencedirect.com/science/article/pii/S0923474802000218>

Bonner, Bryan L. and Alexander R. Bolinger. "Bring Out the Best in Your Team." *Harvard Business Review* (2014). < https://hbr.org/2014/09/bring-out-the-best-in-your-team>

Coutu, Diane. "Why Teams Don't Work." *Harvard Business Review* (2009). < https://hbr.org/2009/05/why-teams-dont-work>

Duhigg, Charles. "What Google Learned From Its Quest to Build the Perfect Team." *The New York Times Magazine* (2016). < https://www.nytimes.com/2016/02/28/magazine/what-google-learned-from-its-quest-to-build-the-perfect-team.html?smid=pl-share>

Hackman, Richard J. *Leading Teams: Setting the Stage for Great Performances*. Harvard Business Press, 2002.

Helfand, Heidi. *Dynamic Reteaming*. n.d. <http://www.heidihelfand.com/dynamic-reteaming/>.

Katz, Ralph. "The Effects of Group Longevity on Project Communication and Performance." *Administrative Science Quarterly* 27.1 (1982): 81-104. < https://www.jstor.org/stable/pdf/2392547.pdf?seq=1#page_scan_tab_contents>

Knight, Pamela J. "Acquisition Community Team Dynamics: The Tuckman Model vs. the DAU Model." Monterey: Naval Postgraduate School, 2007. < https://apps.dtic.mil/dtic/tr/fulltext/u2/a493549.pdf>

Morgan, B. B., E. Salas and A. S. Glickman. "An analysis of team evolution and maturation." *The Journal of General Psychology* 120.3 (1994): 277-291. <https://en.wikipedia.org/wiki/Group_development#Morgan,_Salas_&_Glickman's_TEAM_model>

Norton, Doc. *Tuckman Was Wrong!* 5 May 2017. <https://onbelay.co/articles/2017/5/5/tuckman-was-wrong, https://www.infoq.com/news/2019/04/tuckman-team-model-wrong/>.

Scrum PLoP! [1]. *Stable Teams***. n.d. <https://sites.google.com/a/scrumplop.org/published-patterns/product-organization-pattern-language/development-team/stable-teams>.

Scrum PLoP! [2]. *Norms of Conduct*. n.d. <https://sites.google.com/a/scrumplop.org/published-patterns/product-organization-pattern-language/norms-of-conduct>.

Sutherland, Jeff and Ken Schwaber. *The Scrum Guide*. 2017. <https://scrumguides.org/scrum-guide.html>.

Tuckman, Bruce W. "Developmental sequence in small groups." *Psychological Bulletin* 63.6 (1965): 384-399. < https://psycnet.apa.org/doi/10.1037/h0022100>

Vodde, Bas and Craig Larman. 2019. <https://less.works/>.

Wang, Yanqing, et al. "On measuring team stability in cooperative learning: An example of consecutive course projects on software engineering." 24 January 2014. *Cornell University.* <https://arxiv.org/abs/1401.6244>.

Wikipedia. *Paul Watzlawick*. n.d. 2019. <https://nl.wikipedia.org/wiki/Paul_Watzlawick>.