What It Takes to Be a Successful Scrum Master

Anh Chi Nguyen

anguyen@akvagroup.com

Abstract

This paper focuses on the lessons learned along the way as I transitioned from Developer to Scrum Master. I'll share my own personal story in why I decided to take on this transition. My main desire was to help people. I wanted to help my fellow developers and product owners to solve daily Scrum communication problems and adopt an Agile mindset. However, with my lofty aspirations, it was a tough mission. As a developer at my organization, you mostly work alone at your desk. But I quickly realized that I could not do things on my own terms anymore. Rather, I needed to collaborate with people and look for solutions which could work for all. Rather than "ME", it is now a "WE". As a Scrum Master, my tasks have become more abstract and personnel related.

In sharing my story, I'll discuss what I have experienced working in a Scrum Master role for more than 4 years. This includes my successes, challenges, and yes… failures. I hope that this can provide valuable insight (including tips and lessons learned) to other developers & testers who may be considering transitioning to a Scrum Master role. Some particular points I'll cover include:

1. Understand human psychology to decode team behaviors. Why is it that some team individuals behave in 'odd ways' that you can't understand on the surface? What's going on here?!

2. Select communication methods that could help your team to solve collaboration problems. Keep in mind, what works for you does not mean it will work for your team. Test it and adjust!

3. How to measure and interpret work progress in your team and look for team improvements.

4. Do Agile vs. Be Agile learned lessons

Biography

Anh Chi Nguyen holds a MSc degree in Information Systems at Norwegian University of Science and Technology (NTNU). She has worked in Nordic IT industry for 12 years where of 4 years as a Scrum Master. Her current job is Agile Coach for AKVA group where she coaches multiple development teams to adopt an Agile mindset through a practical set of Scrum practices.

1 Introduction

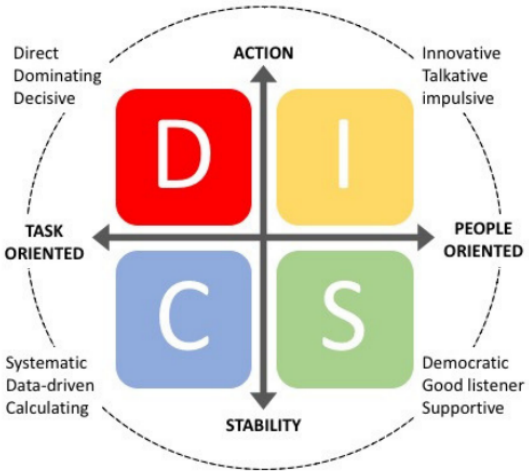
Scrum is no longer a new topic in IT industry. Yet, challenges of doing Scrum effectively persist. Back in 2007, I was a software developer when my past Norwegian company started to adopt Scrum in my team. Norwegian software companies are known for their flat culture, which is an advantage to adopt Scrum. However, there are certain real-life challenges when working with Scrum that you hardly find explanation in Scrum training and books. My paper will cover some of these challenges reflected from my own experience.

2 Psychology approach to understand human behaviors

Most communication conflicts are caused by misreading others' behaviors. In Scrum, communication conflicts may happen frequently. This is because teams are empowered to make important decisions together without hierarchical authority. To solve communication conflicts effectively, I chose to look at human psychology.

2.1 Background of DISC model

DISC stands for Dominance, Influence, Stability and Compliance. This model is invented by an American psychologist named William Moulton Marston. The author proposes a technique to categorize people's behavioral traits into these types of DISC in order to understand their emotional reactions. For each type, there are certain communication challenges. Below is an illustration of main traits in each DISC type.



DISC model [1]

In my experience, everyone is a combination of these four types. However, their dominant type can highly influence their overall reactions. By knowing about their dominant type, I could adjust my communication approach to obtain a more effective result and improve collaboration.

2.2 My stories

I was Scrum Master before I learned about the DISC model. There were certain situations that I could have handled better if I had been equipped with a proper understanding of how different others were from me. DISC helped me to gain a fundamental understanding of human behaviors, and I made better decisions in this field with my tasks.

2.2.1 The agreeable person

Agreeable people can initially be perceived as polite and peaceful, yet they can be a frustrating factor in teamwork where we require individual opinions and consensus. I had such an experience with Dave, a developer who is the agreeable type. One time, Dave came to me and complained that his Sprint work got disturbed by our elite colleagues Alf and product owner Bob. In the team, we all knew there was some tension going on between Alf and Bob on product decisions and we had not found an effective resolution. Bob felt developers need to comply with the design he wanted for his product, while Alf argued how the product interacted should be a developer's decision. They both went for their own ways, and Dave felt stuck, uncertain and disturbed as a team member.

To prevent unnecessary tension, I brought Alf to talk with Dave first and discuss how to help him remove work disturbance. Alf suggested a few collaborative things and Dave agreed with them all. Surprisingly, the same outcome happened when I brought Bob to help Dave remove work disturbance. I knew suggestions from Alf and Bob were different and impossible to implement both, but Dave just accepted them. I must emphasize that Dave accepted both Alf and Bob's suggestions when he was in the room *with each of them*. When he left the room, he came to me and complained that Alf and Bob's suggestions made him stress out. That made me feel frustrated. What did not work here? I had brought them to communicate and agree on a solution. I could not understand Dave. If he thought Alf and Bob's suggestions were difficult for him to try, he could have told them when he was in the room with them. Why had he accepted it and now complained? At last, I had to gather them all again to clarify product goals and role scopes; we formed a new mutual agreement.

Had I known about the DISC model, I would have helped all of them more effectively sooner. Later, when learning about the DISC types, I reflected on this story and realized that Dave's behavior traits were considered as those of the Stability type, so he always tended to avoid conflict. On the other hand, Alf presented a Compliance type which tended to judge every detail in a logical aspect and fought for what he felt to be correct. Bob was a typical Dominance type who chose to see his plan implemented with a strong will and might overlook important details. When a Dominance type communicates his goal with a Compliance type, he could get into conflict due to their different focus. Unfortunately, Dave was brought in between them, and with his Stability nature of good collaboration and conflict avoidance, it made sense he felt totally helpless.

2.2.2 Anxiety spreading due to differences in perception

One time I worked to kick start Agile processes for an R&D team in an accounting software company. They had strong domain knowledge, but their old-fashioned software development slowed them down in a competitive market. Management planned to hire external consultants to speed up development for new short-term projects which they hoped to roll out for an early win. The team had a history of "anti-outsider" which hindered management to hire external experts.

Management gathered us to confirm about external hiring. I sensed there was an underneath tension from developers towards management's decision. Henry, my boss who was a Dominance type, briefly announced about the hiring and requested the team to prepare an onboarding plan for newcomers. Ted, a backend developer who was a Stability type, expressed concern about why Henry needed to hire outsiders this time. He felt that the team with several years of experience should have been considered to take charge of new projects. He also reasoned that bringing new people to the team now would affect their productivity due to time spending on training the new ones. Henry explained to them that it would have taken too much time if the team had handled new projects, due to their lack of modern technical competence. Henry emphasized a wish to roll out new products as soon as possible. Therefore, he suggested the team to transfer domain knowledge to the new ones and let them handle new projects alone. As a Dominance type, he was very direct and to the point in conversation. Arvid and Luke, Compliance type developers, said no word but they wrote something in their notebooks. Three other Stability type developers Larry, Jane and Laura exchanged comments but preferred not to say it out loud. As a Scrum Master, I had to observe my team well to detect hidden reactions. My job was to facilitate discussion but not actively attend the discussion. I asked if the team had more concerns to bring up with

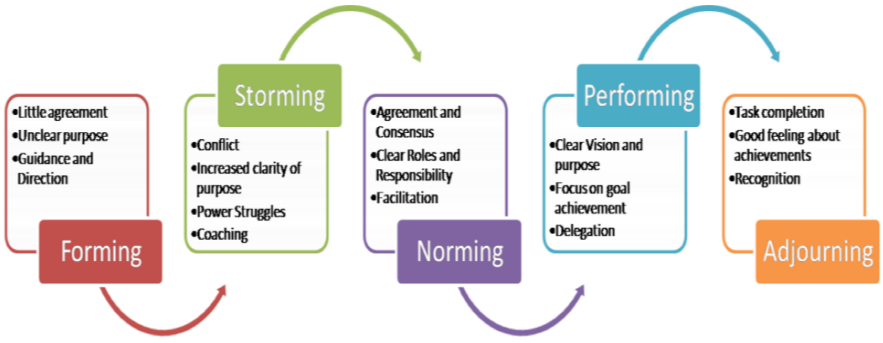
Henry. Luke raised questions about how long those people would stay with the team. Henry showed the gesture of being unsure but confirmed that he would soon find out when the new guys started their first project. Arvid wrote something more while Ted looked puzzled at Henry but said nothing. The meeting was dismissed in agreement that the team would prepare an onboarding plan as soon as possible.

The next day, Larry and Jane came to talk to me. They were worried about management's hiring decision. I asked them why they didn't say a word in the meeting yesterday. They explained that they needed to discuss with their team first. I realized there was a border in communication which separated "my side" and "your side". They worried about their future when management focused on hiring outsiders. They could be replaced one day if the management kept on bringing in outsiders. I was struck with their thought. Although I knew Stability type was not comfortable with change and I was aware of the team historic issue, it seemed Larry and Jane perceived the situation too pessimistic. Worse, their anxiety spread to the rest of the team. Ted, then Luke came to me the day after and expressed the same anxiety. The team could not work productively, so I had to step in. I called in everyone and confirmed that I would help them find out more details about the hiring plan from management. I showed them my booked meeting with management in the upcoming week. I also assured them that the outsiders would not involve any potential replacement to the team. I had confidence to say this because I was in a discussion with management where this issue was brought up as a team mental challenge, and we all agreed a hiring plan would not involve replacement to current employees. Had Henry and I have remembered to mention about this agreement with the team, they would have not built up anxiety and spread it to others. Henry and I simply thought the team should not have been concerned as management had already checked for them. This story taught me a lesson about being aware of differences in perception. What I perceive and make sense of can be different from others. I need to put myself in their shoes to feel how my message is perceived. In addition, I ask assertive questions to ensure that my audience understand correctly about what I intend to communicate.

3 Communication approach based on development stages

3.1 Background of Tuckman's Group Development stage

Bruce Tuckman has provided a clear path of how a team develops its maturity from Forming stage to Adjourning stage. At each stage, the team exposes certain traits. Although the path seems to be linear, real-life teams have their own tempo with a few back and forth shifts. If we are aware of the team's current stage based on Tuckman's predefined traits, we can find effective ways to improve the team's communication and productivity.



Tuckman's stage of team development [2]

3.2 My stories

For some reason, I have been mostly working with teams belonged to Storming, Norming and Performing stage. My stories present some memorable challenges I have collected through my Scrum Master journey.

3.2.1 Team struggles to move forward

I had been assigned to a team which had several skilled developers but struggled with moving forward. Team productivity was low, and everyone felt unhappy about their progress. They missed delivery deadlines and no one from outside the team managed to step in and take charge. Management told me that they had talked to the team several times, but they could not reach a solution. The team said their productivity was struggling due to a work distraction of helping newcomers onboard and offboard quite often, while management believed newcomers would never affect team productivity since they were skilled people. I suggested the team to hold a Retrospective session. We connected the events which happened along recent Sprints with respective velocities. We found out that the team got productivity issues at each new onboarding and offboarding period due to unpredicted time spent on helping newcomers. In addition, team development stage seemed to shift back and stay long in Storming since there was not enough time for the team to stabilize team spirit before a new onboarding and offboarding event. We shared our findings with management. Management now understood the problem and agreed with the team to decide a new scope of work and resources, so that the team could get stabilized soon and improve productivity.

3.2.2 Team struggles to implement a change

One time I had been assigned to a software team which had a history of change avoidance. The team contained people who worked together for many years and was at Norming stage. Everyone understood their peers' working style and kept a decent collaboration. But, new change suggestion is an unfavorable topic in the team. I was aware of that and tried to figure out how to help them overcome this hindering.

At one meeting, I introduced a new routine on technical documentation to the team. Being aware of the team change avoidance tendency, I was careful to present my idea which contained only a few simple steps to try. Three developers showed positive signs towards the idea and three others looked reserved. I asked how the team thought about this idea. Jane, one of those positive developers, said she would like to try it and she thought the idea should be also applied for release notes to customers. She is an Influence type person and full of new ideas and positive spirit. I complimented Jane on her opinion with a smile and waited for others to give inputs to the idea so that we could agree to start applying it tomorrow. I looked for inputs from the reserved ones. They looked at each other. Fred, one of them, finally said they would not hinder the team if the team chose to apply this change. I was not experienced enough to understand his intention correctly. At that moment, I just thought he was agreed about applying the change in his team. Two days later, I came to the team to check out how the change was going. Fred and one of the reserved developers told me that they didn't do anything yet about it. I was surprised. I asked for a reason. They told me the team didn't start applying the new routine and that was not their fault. More than that, they emphasized that they didn't hinder the team about applying this change. To me, they looked like frustrated victims in this case. I came to Jane, the developer who showed most positive engagement towards the change discussion. I wanted to hear what she thought about this case. Opposite to my expectation, she just confirmed that the team should decide what to do with my idea. A big BUT appeared in my head, "… BUT you guys already agreed to try it with my suggested steps. What more needs to be decided?" I didn't know what to say other than thank you and went back to my desk. For the first time, I realized the real problem with this team. Team accountability was lacking. Empowerment was lacking. It seemed that individuals waited for someone else to make decision. That's exactly what hindered them to grow to a Performing team. They needed an action plan and someone to hold them accountable for their delegated tasks in the plan. The next day, I gathered the team to set out a minimum workable plan which could be delegated to all team members and we agreed on a weekly follow up session. One week later, the change seemed to start showing signs in their documentation routine. The team also showed better engagement and enthusiasm towards this change as they could finally see the progress. I came to learn that sometimes a team struggled to implement a change because they had no one to give guidance and follow up. Scrum Master should guide them to create a workable plan and make them accountable for implementation.

4 Team progress measurement

Scrum Master does not only help team to achieve excellent communication, but also knows how to measure and interpret team progress.

There are several courses and training materials giving you guidance on what and how to track work progress in a Sprint as well as what you should do when you detect a problem. To prevent repeating these topics, I will only provide you some personal insights that goes beyond pure theory.

4.1 Velocity is not everything

Velocity can be tracked by number of tasks or story points in a Sprint. This indicator is essential to help team forecast Sprint capacity, but Scrum Master needs to understand it properly in order to benefit from it. Factors such as team atmosphere and product focus may affect velocity numbers so it's important to be aware of them and evaluate team performance in the big picture.

4.1.1 High velocity but delivery delay

I had been assigned to work with a Scrum team that looked very productive from outside. I was surprised to hear management complain about their delivery delays, because I saw each Sprint velocity based on the data looked very strong and stable on the chart. However, I soon found out the hidden cause to delivery delays. Something was going on with the product focus. The product owner was too busy with two teams, so he could not always manage the product backlog priority on time. The team also had a problem with having him available to clarify the tasks. He at times could not attend some of the Sprint planning meetings. In order to protect their Sprint velocity, the team had to pick up whatever tasks in the backlog that they could proceed with. In addition, there was no available burndown chart to reflect where the Sprint scope started to creep i.e. new tasks were added into the Sprint after Sprint planning. I brought the team together with management to present my findings and asked for suggestions. Management took charge of getting the product owner a solution for his more availability in the team. I helped the team to refine our measurement approach which emphasized on product focus, scope and time. We still used velocity as an indicator but we would not depend solely on it like before. After all, we had learned that numbers could hide us from seeing real issues. When there is a lack of communication like in this case, product development will get in trouble regardless of how good the velocity numbers look.

4.1.2 Struggle to get a stable velocity

One of my past teams was upset because after six Sprints, they still could not get a stable velocity. The team was coming into the Performing stage and they all wished to stabilize the velocity for a better capacity prediction. Yet, they seemed to struggle to achieve it. I also shared that frustration as I didn't know the root cause of our struggle. Looking back at earlier Sprints, the team observed stable results in some Sprints, and unstable results at some other Sprints. Velocity seemed stable for a while, and then unstable again. There might be some pattern there, I thought at last. I then gathered the team and suggested we put all the previous Sprints' velocities together with a timeline, then we saw a pattern. We found out the Sprints before a release as well as the Sprints during a release, there appeared to be an unstable velocity. We explored deeper in our reflection and found that those Sprints usually connected with a lot of testing and fixing. The team had to deal with emergency work like customer support, stakeholders meeting and bug fixing. These efforts were not counted as user story points for velocity. Worse than that, these efforts normally happened unexpectedly and therefore came in the Sprint after the team finalized a forecasted Sprint work. We all now understood why we had that pattern. Learning about this insightful analysis, we came to modify our measurement by setting a velocity-free reservation for each Sprint. This means we reserved a time buffer up to 20% of a Sprint for handling emergency incidents in that Sprint. With the new measurement, my team felt more confident in velocity prediction as well as sufficient time handling should emergency tasks have happened.

A lesson here was do not plan a 2-week Sprint with an expected 2-week velocity work. There is always an unknown factor somewhere which may show up right after we settle the plan, for example, sick leave,

critical bug found, unplanned downtime or meetings etc. These emergency cases certainly affect the Sprint velocity. Therefore, reserving a velocity-free time buffer in a Sprint will help secure planned velocity better.

4.2 Sprint work means work forecast, not work commitment

Many factors contribute to planned Sprint tasks unachievable at the end of a Sprint. It is therefore insufficient to judge team commitment using Sprint work as the only indicator. Scrum Guide 2011 [(3)] also made the change from "commit" to "forecast" to reflect this reality. Here is a list of the most troublesome factors to a Sprint in my experience.

- Communication problems

  Communication problems cause team individuals to have frustration, confusion and lack of motivation towards Sprint work. Sometimes it happens because of different perceptions on the message communicated between sender and receiver. Sometimes it happens because of human behavior misunderstandings between sender and receiver. Looking back at the DISC model and my shared stories in section 2, you can obtain insights on how to better understand your peer colleagues in a working context. My advice to you as a Scrum Master is, always pay attention at daily team communication and Sprint workflow to detect early communication issues.

- Unplanned meetings, external emergency requests and similar issues

  One of the biggest disturbances which developers call as "a mental drain", comes from outside the team. Ironically speaking, external disturbances share team's Sprint time but not its Sprint work. Team members can always be exposed to external disturbances. Scrum Master should help team to protect themselves by setting "unavailability" rules which inform external side about when the team becomes unavailable. External sides can still reach the team by a written message like email or message via chat program, but they must expect there will be delay in receiving an answer.

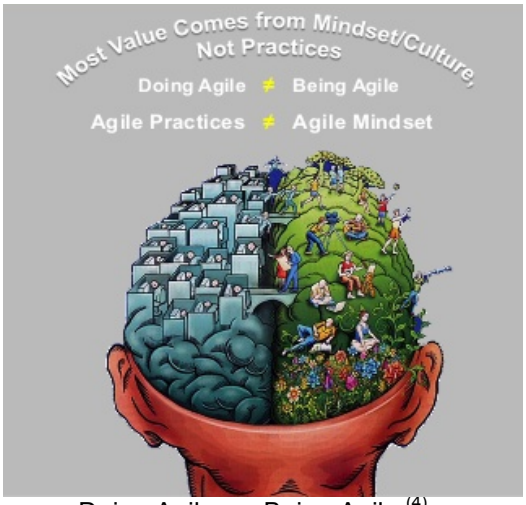
  I have set a 10 AM rule for all my teams which means they are not available for external requests before 10 AM every morning. With this rule, my teams can optimize a quiet morning to concentrate on their work. In addition, I set the no-meeting Wednesdays in their calendar so they can expect all Wednesdays are only reserved for internal team discussions or no gatherings at all except daily stand up. My teams think the rules help them focus more on Sprint work and able to plan out the day for both internal and external matters.

- Technical problems affect development progress such as server downtime, expired licenses, and unforeseen forbidden access

  These problems are popular and it's difficult to predict them all at Sprint planning. Therefore, it would be wise to always reserve a time buffer in a Sprint. This time buffer can be then used for critical bug fixing, extra meetings, technical problems etc, which helps the team protect planned capacity for the Sprint work (mentioned in section 4.1.2).

5 Do Agile vs. Be Agile

Scrum methodology is a double-edged sword. The approach that Scrum Master chooses to lead Scrum process in team can either yield benefits or cause burdens for them. Following Scrum practices by the book is good with caution. By that, I mean Scrum Master needs to understand team maturity level and their specific needs for Scrum in order to help them build a workable process.



Doing Agile vs. Being Agile [4]

5.1 Adding resource does not increase team capacity (velocity) immediately

One common problem of management, especially from sales and finance department, is that they believe adding resource to a process yields an instant benefit. In real-life development scenarios, this belief can face challenges. I have shared a story illustrated a similar case in section 3.2.1.

While there is an expectation that velocity will increase soon after adding new resource, reality may show a different result. Indeed, velocity can drop since team members must share their Sprint time with newcomers in an onboarding process. Internal disturbance is expected to the fullest when onboarding newcomers. Instead of setting a high velocity goal in early Sprints, management and team should allow velocity adjustment to accommodate the change and look for insights into new potential capacity of the team.

5.2 Agile does not welcome all kinds of change in product development

I've worked with management who wanted to adopt Agile because they believed it would welcome all changes. In theory, this belief can be valid. However, adopting Agile in theory and reality still present a big gap. My experience shows that unnecessary changes of scope, political influences, individual motivation conflicts, etc should not be considered as productive changes which Agile supports. Indeed, these kinds of change only cause negative effect to team process. People, who allow these changes to happen and expect that Agile will fix them, are misusing Agile's "embrace change" principle.

Once I came to work with a corporate team which was directed by two managers. Jake was product manager and Luke was technical manager; both were assigned to collaborate on a new product development. Trouble happened when they had different interests for the product and manipulated product roadmap in their favor. Jake was passionate to build more product features, while Luke aimed for delivering a technical masterpiece. They influenced team members to gain their support. Somehow, the team was split in three groups: Jake's group, Luke's group and a neutral group. Communication in team was frustrating when priority kept changing in product roadmap. The team lost both motivation for the product and trust for the managers. At that time, I was very new to the team and Scrum Master role. I

didn't expect to face this complex situation. The Scrum books I had read didn't mention any similar case. I felt stuck to see two passed Sprints which were unproductive. Worse than that, Jake and Luke expected Agile process would embrace these changes in product development; they requested me to find a solution. I decided to ask for help in my company's internal Scrum Master group. Luckily, one of senior Scrum Masters, Mark, agreed to step in to help my struggling team. He had much Scrum Master experience in corporate teams and earned high respect from management.

Mark called in a meeting with Jake, Luke and me. He explained to all of us that what was going on now in my team was a result of a political conflict between the two managers. I had eye-opening moment when Mark went on explaining why this conflict likely introduced unnecessary change in scope and affected team collaboration. He referred to the unproductive Sprints which the team had just experienced. Jake and Luke reasoned that their interests were aligned with a purpose of making the product better. Mark listened to them carefully and then replied in a neutral way "Yes, political conflicts may start from a good will of making a better product. However, when two parties choose not to be open minded to consider each other's opinions in a mutual concern for the product, collaboration goes broken. The conflict then only creates frustration and bad team spirit." The room was silent as Mark let his message sink in for the managers. Then Mark offered to help them solve this conflict by building an agreement on product goal and responsible scope. He set the agreement's code of conduct: trust and accountability. Mark said to Jake and Luke that they needed to build collaboration on trust and accountability. He advised the managers to optimize product development by using each other's strength and keep each other accountable. Mark suggested that the managers should collaborate on the product backlog priority before coming to Sprint planning with the team. Keeping one mutual priority for Sprint work will help team focus on the right thing and move forward. The meeting gave me deep insight about how Agile process could help solve anti-Agile issues. I realized that true power of Agile stayed in building a mindset around it, and this mindset will be used to navigate direction through real-life challenges in Scrum teams.

5.3 Exceptions in Sprint can be positive

5.3.1 Sprint length

Scrum process encourages teams to keep a stable Sprint length to benefit velocity measurement and capacity prediction. However, it's not always simple to keep desired length because of team availability. How productive is it to have two-week Sprints in summer time in Norway when developers take overlapped four-week vacation? Or in Christmas holiday when people combine holiday and family vacation? Or the team simply attends a one-week conference? Keeping two-week length in these cases is frustrated and ineffective. My teams usually extend Sprint length to four weeks or even six weeks to cover frequent lack of resource in these occasions. We have a good long Sprint planning before anyone starts vacation, then the team organizes internal sync (update round) every time a member gets back to work from vacation.

5.3.2 Sprint review

Scrum books make an impression that team must demonstrate something that they have achieved at Sprint review session. At least, that's the reason to invite stakeholders and all who are interested in the product to come at the Sprint review. But what if that Sprint contains most of back-end work and impossible to have a visualized demonstration? It's not good to cancel the Sprint review as the team did achieve something, but how to show it then? My experience showed that a presentation from the team can be a good solution in this case. With visualized illustrations and user-oriented explanation, the team can inform stakeholders about their achievement in an effective way.

Working with Agile / Scrum, you must be prepared to face unknown challenges. Don't be caught up in rigid process from limited teamwork scenarios in Scrum books. You should be open-minded and question purposes behind a suggested process. Scrum framework is meant for adopting and adjusting to suit your team's specific needs. Scrum practices should be there to serve your team, not vice versa.

6 Conclusion

I hope my shared stories will provide insight about challenges you may encounter in your Scrum Master journey. Although the stories come from my own experiences with a limitation on specific teams, working environment and culture, they reflect common challenges in real-life Scrum teams. The key to success is to keep an open-minded attitude and continuous learn and improve your soft skills. Good luck!

References

(1) Right people group. The four different communication types. “Cheatsheet to customer communication.” https://rpg-cify0074508w.netdna-ssl.com/wp-content/uploads/2018/09/DISC-profile-cheat-sheet-to-customer-communication.pdf (accessed June 10, 2019).

(2) Wageningen University & Research. Tuckman (forming, norming, storming, performing). http://www.mspguide.org/tool/tuckman-forming-norming-storming-performing (accessed June 10, 2019)

(3) Scrum Guides. Scrum Guide Revisions. https://www.scrumguides.org/revisions.html (accessed August 3, 2019)

(4) Agile Exchange. Doing Agile ≠ Being Agile. https://agilityexchange.com/doing-agile-is-not-being-agile/ (accessed June 15, 2019)