Software Education

Expected and Unexpected Lessons Learned from Leading Workshops About Geographically Distributed Agile Teams

First Published in the IEEE Software.
Johanna Rothman, Shane Hastie, "Lessons Learned from Leading Workshops about Geographically Distributed Agile Teams," *IEEE Software*, vol. 30, no. 2, pp. 7-10, March-April 2013, doi:10.1109/MS.2013.33

Expected and Unexpected Lessons Learned from Leading Workshops About Geographically Distributed Agile Teams

We have been teaching workshops about geographically distributed agile teams together and separately since March 2011. In that time, we have led the workshops singly and jointly in New Zealand, Australia, Israel, Germany, multiple times in the United States, Saudi Arabia, and Canada. It doesn't matter where we lead the workshops. It doesn't matter where the project leaders or the company senior managers are. We hear many of the same stories.

There are some common factors which we've encountered. The players vary, the context is often different but underneath there are a set of common mistakes and some success factors that we have seen. In this paper we bring these common aspects together into a set of lessons learned, with some concrete advice on how to avoid the mistakes and leverage and amplify the successes.

We separated these lessons into project lessons, management lessons and culture lessons.

First, the project lessons.

Project Lesson 1: "We thought Scrum would work."

Almost every project manager or lead who participated in our workshops started off by saying, "We thought Scrum would work." Scrum is a wonderful project management framework. And, it's designed for a 5-7 person cross-functional co-located team. Because our participants have geographically distributed projects, they fail the co-location test.

But more importantly, almost all of these teams are part of a larger effort: a program of work. A program is the coordination of several related projects to meet one larger business objective (1). (Note that a program of projects is different from a computer program!) Because the program requires coordination and risk management across the teams, and within the organization, and Scrum by itself only provides Scrum of Scrums, the program doesn't have sufficient coordination and risk management. Does that mean Scrum can't work? No. It means that extending Scrum to a new-to-agile geographically distributed team is non-trivial.

There are a number of scaled agile models which have been identified, and they offer some good advice in terms of going beyond the small, co-located team that has been described as the agile “sweet spot” Kruchten (2), Rothman (3). Finding the right scaling structure for your project and environment will take some experimentation, use agile approaches to experiment and adapt to find what fits best for you.

Project Lesson #2: “We’ll make a cross-functional team out of these people.”

One of the reasons Scrum didn’t work was because senior management said, “Go do Scrum with these people from these silo’d teams. We’ll give you testers in India, developers in Germany, developers in Sweden, developers in Israel, product owners in Brazil, business analysts in North Carolina, Scrum Masters in California. We empower you to make yourselves a team.”

This is an example of just one of the geographically distributed teams we met. In this case, the developers were pretty close in time zones. But they were far away from their product owner and their business analyst. At least, they were close to their tester, their one lonely and under-educated tester.

Even when collocated it takes time and effort for a group of individuals to become a team, this is harder and takes longer if they start out distributed. Put in the effort and time to allow people to get to know each other so they can truly form a team.

We thought people knew Tuckman’s model of Form/Storm/Norm/Perform (4). The model says that people need time to get to know each other to work together as a team, and that teams have stages that they proceed through. If you don’t take the time to form, you spend a lot of time in storming, and never get to norming, never mind performing.

The team members we met do know the model. But the managers who create these far apart distributed teams appear to be unfamiliar with Tuckman’s model or have forgotten that teams need the time to get to know each other.

Why did the managers forget? We suspect it’s because the managers work with each other all the time. That is, the managers continue to work with their peers on an ongoing basis. The managers’ team doesn’t change, however the project teams do change, often out of sight of the managers who inflict the change.

The more time zones apart the team members are, the longer it takes to form a team. Teams need interaction to be able to storm – to figure out their collaboration process, their ways of

communicating, their shared norms. The harder it is to storm—the effect of many time zones—the longer it takes to get to norming. If you want to get to performing, it's much easier if people travel to meet each other.

Project Lesson #3: No matter how much it costs to bring people together at least once, it will always be cheaper than the cost of not doing so

The teams that traveled to meet each other succeeded better than the teams that didn't travel. In every single case.

We developed our workshop together as a geographically distributed pair. But we developed it after meeting each other, after establishing trust, and after seeing how we each worked. We met each other first, before deciding to work together remotely (Boston and New Zealand).

What astonished us was how many senior management teams flung people together, called them a team, and gave them no chance to establish trust before entrusting multi-million dollar efforts to a team. When our workshop participants asked for travel budgets, they were told there was no money for travel.

It seems crazy to us that there is no money to bring seven to ten people together to break bread for a week to learn how to work together so you can ensure the work of a project team. Or, if those people are a program team, so they can bring the program charter and vision back to their respective locations and spread the trust.

In every case, the teams who spent the money and traveled at the beginning of the project to establish trust enjoyed a project with fewer misunderstandings and a product with fewer defects. The return on investment of the relatively small amount of money spent bringing the teams together could be measured in better collaboration, shortened decision making cycles, less misunderstandings and rework and higher quality products with shorter time to market.

Management Lesson #1: Don't underpower the infrastructure in "remote" locations

We heard astonishing stories of how the "remote" teams did not have the same access to tools, sources, knowledge and talent as the teams closer to headquarters. The closer the teams were to the senior management teams, to the headquarters, the more likely the teams were to be properly funded. This automatically creates an "us and them" distance between the groups, and disadvantages the whole team.

With distributed teams the mere ability to collaborate is determined by the tools available to the members with the lowest capability – it doesn't help having multi-megabit having fibre-to-the-office in one location if another is restricted to a single dialup line shared between 8 people. A daily standup over a Skype call that drops out every 40 seconds just frustrates everyone involved, and people quickly stop trying to communicate.

Management Lesson #2: Don't select the testers to be 12 hours away from the developers to save money

Managers too often selected testers to be 12 hours away from the developers because they do not understand that building software is a creative design activity. Unfortunately, some managers think that software is a manufacturing activity. As long as managers think of software as construction, they will believe that they can make cost-based decisions. The managers thought they were saving money on testing. In reality, they were costing the project time and money, confusing project cost and labor cost (5).

Distributing work to reduce cost is one of the worst reasons to form distributed teams. Examine why the team are being distributed and ensure the goals are actually achievable.

Management Lesson #3: Managers can't keep their hands off the people and their hands on the project portfolio

As soon as you have a geographically distributed team, it's crucial to manage the project portfolio and have work flow through the team. The team has to learn to work together. Moving people on and off the team disturbs the team and makes it difficult for the team to succeed at all. But many of the people we met had managers who were accustomed to picking people up and moving them around the organization. These managers were not accustomed to managing the project portfolio.

The managers were now hit with a double whammy. They had to leave the team alone to complete project work either in flow or in an iteration, and they had to manage the project portfolio (6). These two challenges often drive the managers into such chaos that they found it difficult to continue to support the transition to agile. So, just about the time the team discovered its stride, the managers are ready to stop the transition to agile, because the managers can't take the transparency. It's enough to make everyone berserk.

Culture Lesson #1: Different is not wrong, but it takes conscious effort to overcome the built in bias

People are all different, and in distributed teams we often compound those differences because we take people who would otherwise most likely never come across one another and tell them "you're now a team – work together productively right now!"

We all bring our background, biases, experiences, history and our culture with us – these things make us who we are. In your home culture there are hundreds of unwritten rules of behaviour that "everyone knows." We understand the social norms, the unspoken hierarchies, even the simple things like which cutlery to use (or not use) depending on the meal being served.

When we create distributed teams, the members don't have the same understanding of each other's culture. Our natural tendency is to assume that everyone has the same set of unwritten norms as we do. Of course this is wrong, but that doesn't matter—unless team members take the time to stop and think about how they communicate and what messages they are conveying through their behaviour. If team members don't take the time to stop and think through their messages, especially their emails if they have not met first, then misunderstanding and miscommunication are rife.

There is a lot of research into different cultures, see Hoefstede cultural dimensions (7). It is worth spending time in a team talking about each other's cultures. Respectful curiosity goes a long way to creating understanding. Bringing people together and giving them the space to have these conversations is a great way to start a team building their own shared norms.

Culture Lesson #2: Sharing food is an important social bonding activity

One proven and powerful technique for building understanding in the team is to share food together, taking the time to understand the message of the meal.

When you bring people together, as in Project Lesson #3, do it the way Johanna did many years ago with her geographically distributed project team. She had the team work together during the day on "real" project work. One evening, she invited them to her home where they would all prepare dinner.

Everyone prepared their special food. The self-professed non-cooks brought flowers, seltzer, bread, beer and wine. The rest of us cooked up a storm one night. The kitchen was a disaster and we all ate too much. But we all learned a lot about working together in a too-small kitchen, and how we could work together on the project. Johanna says, "Don't do this the first day of the project. Do this after you have been working together for a few days, and you understand some people's sense of humor!"

If we had not been together, we could never have had this experience, and it was high point of the week we had together. We referred to it again and again during that project. We learned that some people *must* wash knives by hand, and that issue never occurred to others. You might not think knife washing is a cultural difference, but we assure you, it can be.

For us, preparing food and sharing it was part of building the team. You don't have to go as far as we did. But take advantage of any opportunity when you visit any of your geographically distributed teammates to share a meal. Sharing food will bond you in a way that a call or email will not.

Culture Lesson #3: Create team norms around risk and done

When we taught the workshops, the participants often asked us, "How do we help our project teams get to "done" at the end of the iteration?" Another frequent question was "How do we get to the end of the project?"

Colocated teams have these problems also. But they can easily conduct retrospectives and examine their behaviors and artifacts. It's not so easy for a geographically distributed team to do so.

One of the most important ongoing discussions an agile geographically distributed team needs is to keep asking "What does done mean for us?" and "How do we manage our risks?". Define "done" for the story, for the iteration, for the release, for the project and for the program, and then make sure the definition of done is understood and agreed by all members of the team, and management who guide them.

We, Johanna and Shane, believe in small stories that you finish in one- or two-week iterations that you then demo to your product owner or customer. That's how you manage done and project risk. You have release criteria for the project (1). You never let your work in progress grow, and you know where the project as a whole is headed. You keep heading on a course, checking in with the customer. But, not every team believes in this approach.

So, keep having the conversation, "What does done mean for us?" and "How much risk can we take?" If you keep discussing done and risk, you will make the issues obvious. And, you will discover the cultural differences that often hide project risk until it explodes.

Geographically Distributed Agile is Not an Oxymoron

You can make geographically distributed agile teams work. But you need to think about how to do it. Our lessons are:

1. Consider which agile approach you want to use. Do consider Scrum. But if you want to use Scrum with a new-to-agile team, do get training and use a coach. What we have seen to be an even better idea for many teams is to use an unbranded form of agile that fits their specific context. (8)
2. Make sure you have feature teams, not silo'd teams. There are smart people all over the world, and you should take advantage of them. But taking one person from here and one person from there and somehow calling those people a team as if they were items from a menu does not make them a team. Creating feature teams does work.
3. Outfit each team with all the infrastructure each team requires. Every team we teach this workshop to complains about this problem. We still shake our heads, and are still disappointed that we need to say this, but we still do. Infrastructure can even mean training. Don't expect that the person who can spell "agile" can do "agile."
4. Beg, plead, whine, do whatever you need to do to get the project team or the core program team together at the beginning of the project/program to learn who each other is and how to work together. You will recoup that money and time very quickly.
5. Most teams new to agile still have trouble with too-large stories. This occurs whether the team is geographically distributed or in the same room. The problem is exacerbated when the team is distributed. We recommend that the team moves to a one-week iteration and/or uses a kanban board so they can see and feel the work in progress and then eliminate work in progress.

Our Next Steps

We are considering adding a workshop for managers, because many of the problems our participants encounter are management-generated. However, the managers don't feel the pain, so we have to find another way to reach the managers. And, many potential participants said, "We want this workshop as webinars, because we have no travel budget to attend a workshop that would help us fix our troubled projects!"
We are working on learning how to do experiential training online. So far, we don't know how. But we are open to suggestions.

References

1. Rothman, Johanna. *Manage It! Your Guide to Modern, Pragmatic Project Management.* Pragmatic Bookshelf. Dallas and Raleigh. 2007.
2. Kruchten, Philippe, *The frog and the octopus: a conceptual model of software development:* http://arxiv.org/abs/1209.1327
3. Rothman, Johanna. *Agile Program Management: Collaborating Across the Organization.* Leanpub. Anticipated 2013
4. Tuckman, Bruce W.and Mary Ann C. Jensen. *Stages of small group development revisited*. Group and Organizational Studies, 2:419– 427, 1977.
5. http://www.jrothman.com/blog/mpd/2010/03/wage-cost-and-project-labor-cost.html].
6. Rothman, Johanna. *Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects*. Pragmatic Bookshelf. Dallas and Raleigh. 2009.
7. Hofstede, Geert, Gert Jan Hofstede, Michael Minkov, *Cultures and Organization: Software of the Mind*. McGraw Hill. New York. 2010.
8. http://www.jrothman.com/blog/mpd/2012/02/geographically-distributed-agile-teams-have-choices-for-their-lifecycles.html]

In addition, we recommend that people read Carmel, Erran and J. Alberto Espinosa. *I'm Working While They're Sleeping: Time Zone Separation Challenges and Solutions*. Nedder Stream Press. 2012.