Designing Agile Feedbacks for Agile Learning

Tathagat Varma,
VP Strategic Process Innovations,
[24]7 Innovation Labs

Abstract

Feedback is perhaps the most important aspect of the overall agile lifecycle - without a proper, honest and timely feedback, there is no ‘adapt’ step in the inspect-adapt cycle. The absence of such feedback only ensures there is no early opportunity to ‘respond to changes’ and teams will have no option but to simply keep ‘following the plan’ thereby violating a key agile value. Starting with the TDD loop to the CI systems, we are constantly seeking feedback on our outputs - in ever shortening feedback cycles as technologically possible. However designing a proper feedback instrument for a human-human interaction, like a training program, is a totally different thing because it entails imprecise measurements that are often influenced by people’s mental models, skills and experiences, and not to mention - their calendars! Needless to say, these feedbacks could mean anything to different people on different days.

If the feedback required is too ‘wide and shallow’, it can be obtained very quickly but it won't give enough actionable feedback. On the other hand, a ‘narrow and deep’ feedback could be more actionable but might take relatively more time, and it might also fail to register feedback outside its focus area (what if you were focusing on the wrong problem?). So, how does one go about designing feedbacks that enable agile learning? We call them agile feedbacks.

In this brief session, we will share an experience from designing agile feedbacks for agile trainings and workshops. The objective was to get **most critical feedback** in **shortest amount of time** to **enable quick action planning**. We created feedback that took a maximum of just **5 minutes** per respondent, and enabled the most important learning in both, focused as well as open-ended manner that allowed us to focus on the most critical items. We employed elements of Design Thinking and Rapid Iterative Testing and Evaluation (RITE) to improve the process and quality of feedback themselves.

Background

[24]7 Innovation Labs is chartered with developing products in the areas of Big Data, Machine Learning, Predictive Analytics and Product Development in the domain of customer service, and has 400+ product developers, data scientists, service delivery, user experience in geographically distributed teams across India and US.

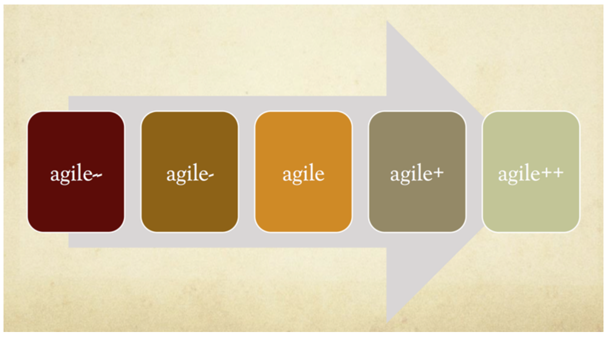
We operate in a B2B segment, and serve some of the biggest global customers by delivering hosted solutions powered by our global contact centers. Our customers expect a 100% mature system delivered under a specific contract thereby making the process more akin to a bespoke product developed and delivered under a traditional SDLC. Needless to say, this is a vastly different context than, say, consumer internet, where consumer-facing practices like continuous deployment are table stakes today.

As one of the long-term goals of the company to become a true SaaS product-enabled service delivery business, a need was felt to create agility across the entire product development and service delivery cycle by creating an end-to-end execution framework spanning the entire pre-sales, product and solution development and delivery cycle. Recognizing that traditional agile methods don't necessarily lend to such end-to-end notion of agility at business scale (*e.g., it is 'easy' to build a system using agile methods but what does it mean to sell in an agile way when the customer is a Fortune 100 company seeking an 80-page contract on highly specific deliverables that must be specified at pixel-levels and byte-levels, and delivered by a given date or face penalties, and yes, did you say continuous deployment?...well, our change management doesn't allow push to production that frequently due to multiple compliance requirements..we prefer the system going live 100% in one shot!*), we adopted agile and lean philosophy, as opposed to agile methods, as the bedrock of our transformation strategy. Clearly, it takes a whole lot of stuff to scale up agile thinking from an individual team to the entire organization!

Starting Jun 2013, we embarked upon agile transformation journey. Training was identified as an activity to not only build an increased awareness and knowledge about agile methods, but also as means to identify the key missing areas, as described before. This data was then used to create execution framework and detailed process definitions and guidelines, as needed. We chose to be source-agnostic and borrowed heavily from classical Agile, XP, Scrum and Lean, as well as the more contemprary and evolving Scaled Agility Framework (SAFe), Kanban, Lean Startup, Customer Development, and Business Model Canvas, etc. Apart from process frameworks, we also recognized that unless we change the underlying organization structure and processes appropriately, no process change could sustain. To that end, we have been inspired by elements of Complex Adaptive Systems, Holacracy and such.

Approach

When we started agile transformation program, some teams had already been practicing a few agile practices like timeboxed sprints. However, it was not uniformly agreed upon as to how well agile could apply to our context. As a first step towards creating a wider awareness of agile mindset and methods, we identified a very immersive training to get people started. We identified a one-day session on introduction to agile methods, followed by another one-day session for product owners or a half-day session for scrummasters. This was meant to be a two-way street – collect real-life issues from the teams engaged in product development and 'guiding' them to 'discover' what are the best ways to improve their team process before really exposing them in the agile theory and methods. This was a critical and conscious decision because often people don't want to sit in a daylong session to be told what they already (think they) know from various sources on internet, but they want to know how it can really solve their problems. Even importantly, do they even know their problem? So, the approach adopted was roughly like this -



At the start of the session, all participants were asked to give their rating on an 'agile scale' without worrying what agile meant. The idea was to get a very unbiased pre-training feedback on where they thought they were. After that, they worked in teams of 5-6 each and identified specific practices behind their rating. It was not only important to get people to voice their concerns first thing in the session to create an environment conducive for learning, it was also extremely helpful in making people communicate among each other, and have them focus on the big picture.

Unsurprisingly, the majority of people came up with 'agile -' rating in most cases, which meant they kind of knew what was missing in their way of working that would make them more agile. This was a great starting point on what to focus on, but also from how well the attendees might be open to learning. Each specific practice (or the lack of it) was voted upon by the group on its perceived criticality/importance across the board using simply fist to five method. This would take roughly 2 hours out of the training, and even before a single word on agile was spoken, we already had the 'process backlog' of things that we needed to fix inside the organizaiton, duly prioritized by the actual practitioners, which consisted of development, testing, product management, user experience design, IS, finance, production operations, network operations, and so on. Needless to say, the attendees had already identified and socialized the problems, kind of agreed upon the solution and the trainer was now only needed to create a shared mental model in terms of a common vocabulary, frameworks and methods using the training material. Rest of the session would introduce agile, scrum, XP, lean and kanban practices to the attendees but the focus was to integrate the specific problems that had been raised by the working groups.

In some sense, it can be called "Test-Driven Training" - we first found out what was missing and then proceeded to take up topics to fill up those specific gaps, while also taking on agile concepts and methods.

After the training was completed, a very lightweight questionnaire was created on google docs (for reasons of simplicity in administering the questionnaire, and collecting and collating the feedback). The questionnaire had just five questions - so it was quick to design, easy to administer and equally quick for attendees to answer.

We looked at some of the oldest and most established training evaluation models, like the Kirkpatrick's Model, and made changes to meet our criteria of an agile feedback - easy to administer, fast to fill up, focus on vital few and keep the trainee's interest in the center. The result was a questionnaire with just 5 questions as below:

The training helped me understand/improve concepts behind Agile Software Development and Scrum framework (Strongly Agree / Agree / Disagree / Strongly Disagree)
What are the Top 5 impediments to improving my current team performance? (Choose from list of options, or add your own)
I will be able to implement some of agile and scrum practices in my project in the coming months (0%, ~20%, ~50%, ~80~, 100%)
Adoption of agile and scrum practices can help improve my team's performance (time to market, delivery predictability, quality, amount of rework or wastage, etc.) by (No improvement, 0-20%, 20-40%, 40-60%, 60-80%, Beyond 100%)
How can future trainings on agile and scrum be made more effective?

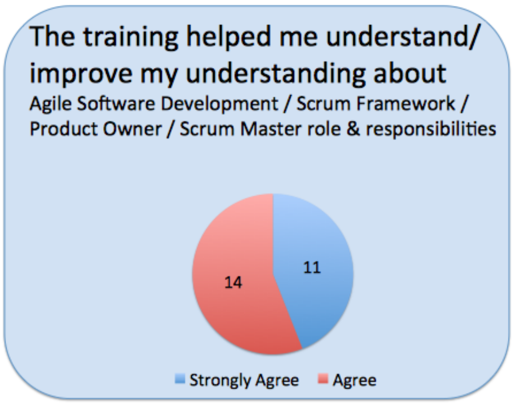
Each question was asked keeping the attendee's interest, knowledge and abilities in mind (rather the usual ego-gratifying questions like 'was the trainer knowledgeable?' or 'did the trainer answer the queries well?'. This helped attendees to respond timely, while also helping us crowdsource the improvement suggestions, and corroborate them them against the pre-training feedback. Fourth question was used to gauge willingness and readiness of teams and individuals for the change (results discussed below) and the last question was used to improve next training. This is an important activity for any change program, and unless people have high belief they can improve, chances are low that a change program will succeed, let alone scale up and sustain in the long run.

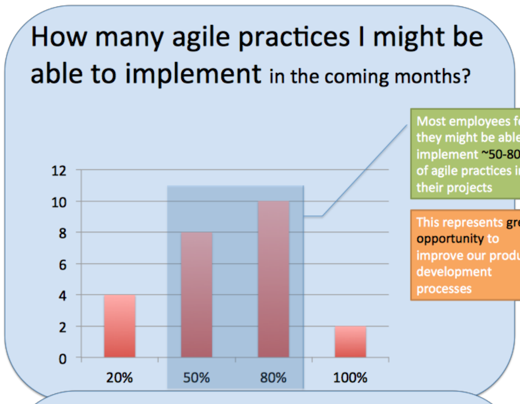
Based on all the trainings delivered in India and US offices, the in-training and post-training data was collated and process improvement framework was designed in accordance. Full-scale rollout started from

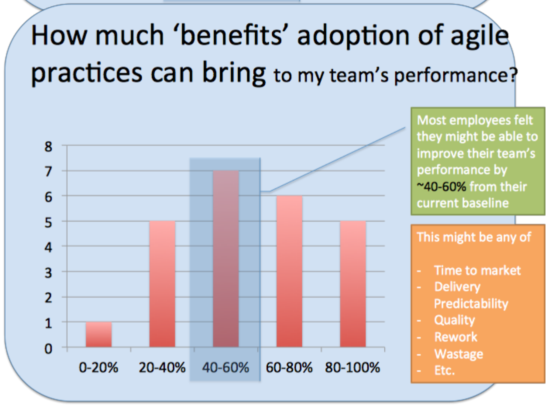
Oct and the data is collected and analysed during retrospectives. In addition, a post-training survey is planned in Jan to gauge if those attendees actually got what they thought they would get from agile practices, and determine the next steps.

Feedback

The actual in-training and qualitative feedback can't be shared as it pertains to specific action items internal to the company. However, some of the data pertaining to how people responded to the questionnaire is shared here, along with the key insights learnt.







The feedback was designed keeping the learning needs of trainees in mind rather than conventional methods of gauging the trainer's ability or knowledge. Mostly, those methods are self-serving the trainers and don't really reflect what the participants really got out of training.

The feedback clearly showed that attendees found the training helpful. I believe a key reason is that it was very solution-focused and the goal of the session was not to teach agile or scrum but to find solutions to the problems people were facing.

(Only the 'strongly agree' and 'agree' options were chosen. So clearly there was a high positive feedback.)

Second question was an interesting question because it probed people in how much they felt they will be able to implement vaious agile practices. It was an indirect question to know what people felt were the gaps in their current process. As we saw from the data, most people felt they could implement 50-80% of the practices that we discussed in the training session. Asking this feedback also made people think about those practices, and it is more likely that when those practices are implemented, the buy-in will be higher.

As opposed to someone insisting that a given set of practices are the best way (really?) of accomplishing something, asking people what might be able to implement opens them up and reduces resistance when the change is eventually brought about.

For the fourth question, it was an interesting experiment. Instead of making grand promises in thin air about how much productivity improvement these practices would bring to them, we asked them how attendees felt how much they could improve, on whatever is their yardstick of measurement was. Again, sceptics might disagree that such a question is too vague, but I think some of the most interesting patterns emerge when the questions are deliberately vague and subject to individual perception.

The feedback was an interesting distribution with most people centered around 40-60% improvement to their current performance. Again, it is important for people to believe they could improve rather than having someone else insist that they can improve their performance by 5x or some such grand number. It is the self-belief that brings about more meaningful change than an external perspective.

The last question was meant for open-ended feedback and was used to change subsequent training content, or the style of delivery. Typically these feedbacks were simple to design and administer, they could be rolled out within a day of the training, and most of the feedback would be available in couple of days - just in time for the next training session. When I look at how we conducted the first session to the time we came to the tenth session, the entire approach, contents and style of delivery had undergone massive changes. Utilizing such agile feedbacks helped improve the agile training sessions tremendously.

Conclusions

The key takeaway is from this experience is that agile feedbacks are all about focusing on 'vital few' vs. 'trivial many', and must be rolled-out quickly, and key changes must be incorporated before the next iteration. While an agile feedback might not solve all problems under the sun, the idea is to take up one or two key issues at a time and quickly iterate on it, and build a virtuous cycle of continuous improvement.

Some other insights from this experiment were -

- If the feedback instrument is simple to design and administer, it will get rolled out quickly.
- If the feedback instrument is quick to fill-up, more people are likely to give feedback.
- If the trainer takes prompt action on feedback, subsequent trainings will benefit from the improvement actions undertaken in the previous cycle, thereby leading to a virtuous cycle.
- A pre-training feedback is a great way to condition participants for the upcoming changes.
- If the practitioners themselves agree on what plagues their way of working, and can come up with how to fix it, there can't be anything better for a trainer or the coach. Trainers need to change their style and contents based on what the trainees already know!
- The self-belief to change and improvement in practitioner's minds is much more important compared to an external prescription about how much one can improve based on some data in faraway lands! What matters is 'here and now'.
- Though agile doesn't really talk about introspections, but I believe introspection has an even bigger role to play compared to retrospection. I would call the pre-training feedback as one of the introspectives.

We continue in our journey. Like any happy ending in the making, we too continue to hypothesize, experiment, fail, take feedback, learn from failures, pivot ourselves, and try yet again!

-end-