From “Team” to “Wow team”: an Agile team’s journey

Gourav Tiwari
Tata Consultancy Services Ltd
Jersey City, United States
e-mail: gouravtiwari21@gmail.com

Zainab Alikhan
New York, United States
e-mail: zainab_alikhan@mckinsey.com

This paper describes different approaches to successfully conquering complacency and ruts in agile practice. It relates how a team on a slide, that was drifting away from the “spirit” of agile, changed itself to a “wow” team by seeing every problem as an opportunity, converting small thoughts into great ideas and finding innovative solutions for problems.

Keywords-Agile development; Scrum; agile practices; team rooms; retrospectives; innovation, start small, culture change.

I. INTRODUCTION

A. Company Context

We work for the internal application development group of a multinational company that began its agile adoption four years ago, in 2007. Pre-agile we were a development group that had primarily home grown systems (mainly Java) along with some off-the-shelf systems that were highly customized. Our department strength was approximately 350 (including onsite and offshore vendor count).

After our year long transition to agile, which began in mid 2007, we felt the benefits of agile across all application development. We had smaller teams that had a sense of ownership. Our time to production went from 6 - 18 months down to 1 - 3 months; we had a new focus on business value, better quality, and felt a slew of other efficiencies across the board.

This paper begins in early 2010 when all teams in the department had been running completely agile for about two years – they were fairly mature. We were quickly converting our majority technology footprint from Java to Ruby on Rails, and starting to deliver more off-the-shelf applications (for the most part non-customized). We were more efficient and our department strength was down to approximately 150 (including onsite and offshore vendor count). Agile was the biggest catalyst in this change - teams had learned how to self-organize, have short iterations, greater transparency with clients, and overall get more done with less.

B. Team Context

Our team consisted of one developer pair (Gourav Tiwari and Ritcha Malhotra) and one Product Owner / Scrum master (Zainab Alikhan). We were staffed as an “intact” Ruby team for a business area that had multiple Ruby on Rails applications and several Ruby projects (both enhancements and build-from-scratch) in the pipeline. “Intact” meant that while the project / product might change, the team members would remain constant.

The developers had one to two years experience with Ruby. Our first assignment was to build a fairly complex application from scratch in two to three months. Quite a challenging task for a small team that was relatively new to the development platform.

II. OUR STORY

A. Where We Started

Being on an agile team is itself a wonderful experience. But what if you’ve been working on agile for two or three years and the “wonderfulness” and energy is waning?

Over the course of prior agile projects, we had become used to repeating the same practices over and over again. The feeling at the beginning of our new project was similar to the feeling at the beginning of most of our previous projects: “…there’s going to be a backlog, we’ll have a kickoff and a release planning. Then we’re going to do stories. It’s going to be the same old thing.”

B. Impetus for Change

Like many other teams, we had hit a plateau in terms of growth.

We had learnt practices advocated by agile and scrum, both process (sprint planning, sprint reviews, stand-ups, done list, working software) and technical (test driven
development, building up an automated test suite, pair programming) which had formed part of our toolkit. But little had changed during our last few projects. We weren’t staying in touch with the external world and keeping ourselves updated on new ideas to try out. We weren’t growing or innovating. And like Stephen Covey says, “Sharpening the Saw keeps you fresh... You increase your capacity to produce and handle the challenges around you. Without this renewal, the body becomes weak, the mind mechanical.” [2]

Worse, on our prior projects there had been an increasing tendency to cut corners, skip the basics, such as:

- When retrospectives were tending to become slow and not generate many ideas, we just let them fall off the calendars rather than recognizing their value and trying to identify and address the cause behind their ineffectiveness.
- We were treating stand-ups as a status update-justifying how our time was being spent rather than leveraging them to ensure that a sprint was successful.

In addition, our first three-month project had some unexpected twists. What had seemed to be a small simple Ruby application when we did our release planning turned out to be a tool that had several touch points outside of the Ruby component, thus increasing the complexity beyond what we had predicted. As a result we failed to identify stories relating to the integration and monitoring of those components. Given our limited toolkit and inability to respond quickly to things going wrong, it took us a few false starts to get the project back on track – for example as new stories came up, rather having an open conversation with our internal client and resetting scope we kept trying to squeeze in the missing stories into the existing timeline. The low point of the project was the time when the first production deployment that was going to be shared with the senior client leadership; a week prior was the first time it dawned on us how much we had underestimated the project and just how much we left. We didn’t share the misgivings that we individually felt and kept pushing to deliver on an unrealistic timeline. In spite of our best efforts at making the deadline, one simple thing went wrong in production. And we had no time to fix it because we had shaved the deployment timeline so fine. Eventually the clients had to pull back their demos hours before the meeting. This rocky project made it clear to us that we had to do something different in order to make the next few projects go smoother.

In addition to the need to overcome these challenges, we wanted to try to experiment to see if we could get our energy up, inspire ourselves, grow and do more than one might expect out of a team with one developer pair.

III. THE MEAT OF IT – HOW WE DID IT

A. Rejuvenating the Basics

Basics are important in any methodology or technology or anything you learn/practice. Our first change was to breathe new life into our agile fundamentals. These are some of the things we did to get the basics back on track.

- Core meetings back to order - Even though we only had the usual limited scrum meetings (planning, review, retrospective) every single one would run over to the point that to our internal clients it was beginning to feel like “scrum = meetings”. During one retrospective we realized that we had a tendency to break into problem-solving that really had no place in the planning or the review meetings. Gourav volunteered to take on this retrospective item and become our “meeting watchdog”. It was his responsibility to do what it took to eliminate tangents and ensure that meetings ended on time. He started to mark those side conversations as “parking lot” items. As soon as he’d spot a tangent, he simply wrote down the topic on a post-it and put it on a Parking Lot wall (along with an “owner” to make sure it was followed up later) and then signaled a timeout to the offenders. The Parking Lot wall got revisited at the beginning of every sprint review - only to take completed items off the wall. In two or three weeks spotting and avoiding tangents became a habit with every member, and we didn’t need a watchdog to avoid tangents.
- Take standups outside - We used to do our standups at 10 a.m. in the team room, but they were increasingly getting to be tedious and long. One day one of us suggested in a retrospective that we take the standups outside the team room for a change. The change of scene not only instantly perked us up, but unexpectedly we saw that our standups were now finishing on time if not early (sometimes they’d end in five minutes instead of fifteen). The reason was simple. When we were doing standups next to our workstations, someone might respond to a ping or email (bad etiquette!). But when we were away from our laptops we didn’t have this option. The product owner of a neighboring team commented to us “You are the only team which literally does the stand-up where everyone actually stands up and are focused. Your motivation is great…”
- Scrum factoid – At every retrospective we started to share a “Did you know...?” fact about scrum (or agile). Team members had the rotating responsibility to bring a factoid to the retrospective. It could be something new that they found, something they felt the team needed a reminder of, a fresh idea that they’d read about, or just a favorite fact about scrum or agile. We not only began to read more, but it made us begin to think again about “why” we were doing what we were doing rather than merely the “what”.

B. Remaining fresh and innovative

We learned how important it was to nip staleness in the bud. Staleness created a lack of energy, bad moods, and invariably had an impact on our stories. We came up with several ways to free ourselves from these kinds of ruts.

We found it helpful to always do something outside the project. At first it might seem like a waste of time. Why
spend precious development time on side projects that aren’t focused on creating working software? But we found the reverse to be true. We always had a side project going on, and these energized us, and gave us a lift that made stories go quicker. In fact, the more side projects we added, the faster we got. Some of our side projects:

- **Most Teamish Room** - Once someone passing by our team room stuck a post-it on outside with the words “this is the most teamish looking room” scrawled on it. At the time, there was little innovation going on in the room – but it was more than there was in other team rooms. This lit a spark – let’s make our team room truly the Most Teamish Room. Our acceptance criterion was, “Anyone that passes by feels like coming in.” The owner of this retrospective item pulled the three of us together to brainstorm and paint a picture of what “Teamish” was. It turned out that our idea of an inviting teamish room included elements of being engaging, tidy/pleasant, colorful, not stagnant, and with walls that spoke to people.

  Based on this we made a Teamish Room backlog, prioritized by voting on it, and addressed one story at a time (see Fig. 1). At the end of week three, we had had three passers-by come in and say “your room looks so inviting”...that week we stopped working on our Teamish Room and moved onto a different side project.

- **Game zone** - Keeping the energy level of a team high can be very challenging, especially when you’ve been working together for several months. You need a fresh breeze to reduce the lethargy that creeps in. We found that games, like Connect 4, Su-Do-Ku, Solitaire and Go that could give us a quick diversion from work easily filled the team room with energy (see Fig. 2).

- **Thought for the week** – Each week we stuck a colorful printout of a thought for that week. We rotated responsibility to find a thought provoking idea and put it up. Then we decided that instead of discarding the previous weeks’ thoughts, we would line them up along the top of the room. They always made for interesting conversation when someone wandered into the room (see Fig. 3).

- **Snack penalty** - We kept a log of latecomers to any meeting. Three minutes late would earn you a cross against your name. Whenever a person had three crosses they had to bring in something yummy and healthy for the snack corner. This made us self-aware of how often we really were late. We found we didn’t mind bringing in something and that the team room became even more inviting, especially on long boring afternoons or late nighters for “day before production code freeze.”

C. **Raise your hand**

If the discussion was not relevant, we decided that everyone had the responsibility to raise their hand, which
was a way of saying “is this conversation really needed?? Let's FOCUS!” People would feel bad verbally interrupting conversations, but with this unspoken sign made it much easier to interrupt and we all became meeting police. It was a simple but effective way to keep focus. Slowly, the irrelevant discussions naturally reduced.

D. No idea too silly

We developed a philosophy of never shutting down ideas, hearing through any idea that any of us produced, no matter how outlandish. This encouraged people to be creative and turned out to be the source of many of our ideas. Examples of ideas that may not have made it through the chopping block without this philosophy – most teamish room, a department-wide debate on retrospectives, game zone, and thought for the week.

E. Start small – don’t try to perfect everything at once

We didn't take on lofty goals, didn't shoot for perfection, and didn't have high expectations. Whatever small idea anybody volunteered was good enough. Our approach to everything was simply to do enough to ensure that the ball didn’t slow down to a halt – “what’s the minimal that can be done to keep the ball rolling.” It was better to at least walk if we were not able to run.

For example, when we started as a team we had trouble working with our internal client who had a perception that agile was magic. His interpretation of “changing requirements are good” was “requirements can change anytime”. And they did...even minor edits the day of production deployment! This was not his fault at all – he just had not had the coaching behind the “why” of agile as we had had. How could we even begin on this journey of educating folks that we worked with so that we could work with them as a strong team. We began with a small step: we involved our clients in the retrospective so that they could hear our own self-improvements, what was working and what was not. They began contributing retrospective items. After a few weeks we added a scrum factoid to each retrospective (see section 3A). These gave us forums to let out sound bytes about why scrum did things the way it did. It really helped win their confidence. We were more productive and our sponsors were feeling like they had a voice and were a part of the team. Together we were able to deliver better quality products.

IV. NO NEED TO EVANGELIZE - PRIDE, ENERGY AND INNOVATION ARE INFECTIOUS

How could we spread this rejuvenation of energy and its benefits beyond our team? “Lunch and learns” and other such means are considered “preachy” and we knew they wouldn’t go very far. Instead we tried out some creative ways of spreading our ideas such as taking on debate.

A. Provoke ideas rather than preach

We were proud of our retrospectives. They had done us wonders and in a department where less than twenty percent of teams even attempted to have regular retrospectives, the three of us had become strong champions of this meeting. Rather than try to evangelize them, we took on a public debate with another team that did not schedule retrospective meetings. This debate was competitive and there was a build up of anticipation (spread via posters) and good-natured rivalry. Both teams prepared like this was a do or die test. The debate was held to a full house (we would never have had this size of audience this if we'd held a meeting to evangelize retrospectives!) The debate was passionate and extremely entertaining. It got the reputation as “the most entertaining AppDev meeting ever.” But more importantly, we got to put forward our reasons for supporting retrospectives and planted seeds in other team members’ minds about their complete write-off of retrospectives. From previously being a non-topic, the retrospective was elevated to at least being considered by teams, if not revered. We provoked thought by way of a healthy debate.

B. Information Radiators to Energy Radiators!

People walked past our room and found the energy from it infectious. They stopped and looked in, and wanted to model it. When we took standups outside the room into the corridor, we posted our sprint backlog on the wall as well (see Fig. 4). People passing by began to expect us to be there at 10 a.m., and commented about how they wanted to make their standups more like ours - purposeful and fun instead of drab status updates.

Figure 4. Taking the standup to the corridor

C. Started sitting in other rooms to learn and share

We sent a meeting invite to the team to block off time on their calendars to start interacting with other teams. During this time, none of us were permitted to be in our own team room. We HAD to find some other room to be in. As a result, we found that cross-learning shot up instantly - sitting with others brought up conversations and idea-sharing that would never have happened otherwise.

V. THE IMPACT OF OUR LEARNING AND GROWING

It’s all very well being more agile, more energetic and having more fun. Did all of this actually help our project and product?
The change in the way the team executed our projects as we grew with this learning and innovation was clear. The most telling example is in the way we handled pressure and how we responded to things going wrong.

In section II we spoke about the production deployment that went badly, resulting in consequences for our client. This incident had occurred three months into the time our team had been working together. When we discovered the deployment issue, we hadn’t been able to pull together as a team and make a plan to address it. We felt helpless, kept trying to save the day with stop gap measures and when we finally ran into issues in spite of our best efforts our response was “there’s nothing that we can do about it”, and called it a night at 7 p.m. The internal client felt quite helpless, and had to do what he could to save the situation with his clients.

In contrast, we had another deployment issue that happened three months later (six months into the time our team had been working together). The issue was more serious - it involved production data loss, and the production teams had told us there was no hope to get it back as recent backups had failed. However within our team there was a determination to find a way, to take one step at a time, not blame, but simply focus on the task on hand. We spent a slightly panicked night but came out of it with an innovative way to have the data restored. All credit to this calm and competent handling of the incident was the capability that the team had gained with its continuous learning and growth.

And as this paper is written, seven months after the second production deployment incident we have had no further incidents in spite of deploying every two to three weeks on average.

Our reputation also grew. James Torio, a UX expert on a different project who is a frequent visitor to our team room said when asked about what he thought of our team, “your team handles a suite of applications and is very agile to switch between projects. You have the most rigorous agile process and you are very much process driven. It’s great to see the team members embrace the team’s face and team is priority not the individuals…”

Our sponsor, Rohit Sareen’s comments on the team that he’s seen evolve: “I consider the team one of the more high performing ones I have worked with. The team is able to change direction and work on multiple areas without stress. Team members feel empowered to take appropriate decisions and reach for counsel as needed and self-improvement is encouraged and taken seriously.”

Our retrospective wall however (see Fig 5) is the best indicator of the performance of the team with contributions from team members and the client such as “Team velocity is increasing as planned”, “Pair programming and collaboration are helping”, “We are getting a handle on <product name> finally!” and “It’s truly a high performing team!”

In conclusion, our takeaways are that after agile adoption, teams tend to go into cruise mode. In order to switch a team from cruise mode back to growth, experimentation and innovation is key.

For organizations that adopt agile, it is vital not only to learn the basics, but to have a way of keeping agile fresh. Without this there is a risk of a compromise on agile fundamentals leading to decline in performance. Learning how to do basics right is just the first step. Taking it to the next level requires experimenting and breaking the rules is just as important as in the Japanese Shu Ha Ri (a Japanese martial art concept that describes the stages of learning to mastery involve learning the traditional wisdom, then breaking with tradition to experiment, and finally reaching mastery).

However it doesn’t take anything big to get this freshness and experimentation. Renewal can start at the grass roots, with one person who sees this need. It can start with tiny steps. And these steps one at a time add up and can be transformative to an organizational culture.

ACKNOWLEDGMENT
We would like to thank Ritcha Malhotra our third team member and partner in crime, Rohit Sareen our sponsor for his willingness to fly with our crazy ideas and our colleagues Belkis Vasquez-McCall, James Torio, Niranjan Sarade, Surbhi Bhati and Yanwing Wong for their encouragement and for the invigorating, thought provoking conversations in their visits to our team room that often recharge us.
REFERENCES

[1] J. Collins, Good to Great

[2] S. Covey, “The 7 Habits of Highly Effective People”, Habit 7: Sharpen the Saw