Five Lessons I’m Thankful I Learned in my Agile Career

by Mike Cohn •

27 Comments

Image not readable or empty
/uploads/blogs/2017-11-21-fivelessonsthankfulilearnedinmyagilecareer.jpg

Five Lessons I’m Thankful I Learned in my Agile Career

Since it’s Thanksgiving week here in the United States, I took some time out of my schedule to reflect on some lessons I’m very thankful to have learned through my career. While these lessons are not unique to Scrum or even agile, each has been a big part of my success with agile.

For each lesson, I’ll share what I learned and tell a brief story of how I learned it. In doing so, I'm hoping to help you avoid the mistakes I made before these lessons became second nature to me.

When There Are Two Ways to Do Something, Do It the Right Way

One of my first professional programming jobs was working in the litigation support division of one of the big consulting companies. Much of our work was driven by sudden demands from the opposition’s attorneys. We developed software that helped our attorneys comply with those demands.

This often meant writing programs that would be used once, but that were needed within 24-48 hours and needed to be perfect. The boss who had hired me was a Unix shell scripting wiz and he’d managed much of the development with no concern for reuse. Every program was 100 percent new. He didn’t even develop new programs by starting with old ones and extending or generalizing them.

This wasn't a bad decision necessarily. It was often the fastest way to comply with the attorney's needs. But over the longer term, I knew we could respond even more quickly if we started to assemble a library of common code. But would it be worth it? Did we even have the time to slow down 10 percent today to be 50 percent faster later?

Soon after joining the project, my boss was promoted and began working elsewhere on the same overall project. And I had a decision to make: continue as I had for the first two weeks on the project or focus on developing a reusable library.

I remember very clearly being in the office one Saturday with the two other programmers on the project, Sean and David. We discussed whether we should start building a reusable library or whether the urgency of deadlines was such that we had to stay focused on just getting things done. It turned out to be an easy decision. We agreed to start assembling the library every time we worked on something.

It was one of the best decisions I've ever been involved in making. The payback from that change came much faster than I ever expected because just a few months later we were faced with challenges we could not have met if we hadn't chosen reusability.

And so, the first thing I'm thankful for is that I learned the lesson:

When there are two ways to do something, do it the right way

The right way may seem more time-consuming or more difficult but in my experience, doing it the right way is always worth it.

Life Is Too Short to Work With People You Don't Like and Respect

One of the last jobs I had before Mountain Goat Software involved working in a highly dysfunctional culture. That culture was in place long before I got there, and, unfortunately, I didn't detect it during the interviews. I think perhaps I was so excited by the cool software I'd be involved in that I pushed the culture out of my mind and took the job.

It was horrible. I was one of five VPs reporting to our CEO. She decided one day that people

needed to work more hours. Oh not because of some urgent deadline, just because. She assigned each of us VPs a different night of the week and we were expected to stay in the office until 7:00 P.M. so that employees would see us staying late. And that would motivate them to do so as well.

If you know me, you'll know I'm a complete workaholic because I love what I do. But I also have a mental screw loose. No matter how big the project is, I have a feeling that if we all stay late tonight--pull an all-nighter--we can finish the project by tomorrow. What? Linux rewritten? Let's work through the night and finish it! The rational part of me knows it can't happen, but it doesn't stop me sometimes from trying.

My point is, I work long hours. But I work them on my terms. I like finishing my day at a reasonable time--perhaps 5:00 P.M. and then exercising for a bit before having dinner with my family. And then I'll work more--often much more--later in the night.

But tell me I *must* stay until 7:00 P.M. and the anti-establishment part of me kicks in, and I want to rebel. Even though I was often in the office until 7:00, or later, the boss telling me I had to do it pissed me off. And I didn't like the message it was sending others. I would literally wait until the CEO left (normally no later than 5:15) and then I'd go tell anyone still in the office to go home.

The CEO wasn't the only dysfunctional person in that company. There were many. There was the architect who wiretapped the boardroom so he could listen in on meetings. There was the developer who claimed to be allergic to our building and went out on medical leave two days after starting. I could go on.

One day our CEO announced some new ludicrous policy--I don't even remember the specifics. I was in another VP's office when he read her email to me. We both looked at one another and came to the same conclusion:

Life is too short to work with people you don't like and respect.

We both decided we would never again work with people we didn't like and respect. It just isn't worth it. Within a month, we'd both left that company, and we haven't looked back. Today we each run successful agile coaching and training businesses. We've never been happier.

If you find yourself working with people you don't like and respect, work to get yourself out of that situation. You'll be much happier.

Removing Someone from the Team Never Hurts as Much as You Think It Will

Firing someone is never easy, even when it's for just cause. I had to fire one system engineer who was stealing hardware from our company and selling it online. The police had caught him fencing expensive hardware that he'd purchased for the company, was missing from our data center, and matched serial numbers from the vendors.

I had no doubt about his guilt. His trial was pending. Yet my boss was reluctant to support me in the decision that he needed to go. My boss's reasoning was, "Do you know what they do to pretty boys like him in prison?" Yes, he really said that to me and hesitated over doing the right thing because he'd seen too many movies. Still this guy had to go, yet it was hard to fire even someone like this. It's always hard to fire someone.

Sometimes firing someone is hard because the team has become highly dependent on that person as the only one who can do a certain job. And you're worried that if you fire that person, the team will be slowed dramatically.

My experience is that those fears are way overblown. In a couple of cases I had to fire someone who was the only person who knew how to do some vital thing or was the only person familiar with some very crucial code. But letting those individuals go and watching their teams quickly learn whatever needed to be learned taught me that.

Removing someone from the team never hurts as much as you think it will.

Teams are amazingly resilient. If there is something specific to be learned, they will dig in and learn it.

Also, by the time a manager realizes someone needs to be fired, gathers the energy to do it, and gets the human resources group on board with the idea, the team has thought the person should be fired for months. Teams typically realize these things much faster than managers.

Letting someone go should never be done without significant deliberation. But it's never as painful as you fear it might be.

The Smartest Person in the Room Is Not Smarter than the Whole Room

A lot of leaders work their way up their careers by being very good at what they do, having that be noticed by someone senior, and then getting promoted. And so many leaders and managers have for at least part of their careers been the smartest person in the room.

And when a decision needed to be made, they would state their opinion, defend it, win the argument over how to do things, and very often lead the team to the right decision. But no matter how smart the smartest person in the room is, it is highly unlikely that the smartest person is smarter than the collective wisdom of the entire team.

I learned this lesson early on in my career. I was a software team leader, which meant I had perhaps 3–5 more years of experience than the average person on my team. And, I probably was the smartest person the room when we were debating design decisions and such.

I remember one debate in which one of the more junior team members changed his opinion without much argument at all. I asked him why and he said that because I was the team lead, I must know best. And that scared me. Even when it might have been true in some cases, I never wanted my teammates backing off their positions and agreeing with mine just because of some job title I had.

Since then I've hated job titles. They are how we present ourselves to the outside world. Within a company, job titles should be meaningless.

I am positive that this junior developer did sometimes have better ideas than I did. And what a shame it would have been if he'd been reluctant to share because I had a fancier job title than he did.

This incident and other similar ones led me to learn that

The smartest person in the room is not smarter than the whole room.

No matter how smart the one person may be or how much experience that person might have, the collective wisdom of the team is greater. The best ideas and decisions will be born from the discussion rather than from the mind of just one person.

If You Don’t Manage Expectations, Expect to Fail

I learned this next lesson from perhaps the least technically savvy boss I ever had. But he understood the importance of managing expectations.

We were building a large call center system that was to be used by nurses who worked in our company. The project ultimately was very successful and was instrumental in helping the company grow from 100 employees to over 1,600 employees within a couple of years.

But if I hadn’t shifted my efforts a few months before the end of the project, that same project would have been viewed as a complete disaster.

The problem was that the nurse’s expectations for what the software would do were through the roof. They had somehow started to believe that the software was going to do things that still aren’t possible 20 years later. Some of the things they had just made up in their heads--and then shared among themselves--were amazing.

I was aware of this disconnect between expectations and reality. But I was too busy with the overall technical aspects of the project to worry about it. Until one day my boss gave me some advice that made me reconsider. He told me that to be successful, the project needed to do two things:

- provide the necessary functionality
- meet or exceed user expectations

He educated me that if we failed at either one, the project would be viewed as a failure. I knew right away that we could never meet or exceed the nurses’ current, inflated expectations. Since I

couldn’t change our technical capabilities, I began working to change our users’ minds.

I immediately shifted the focus of my time, spending about half of each week talking to the nurse users about what the system would and wouldn’t do. I traveled to each of the company’s four call centers around the United States every few weeks and presented the equivalent of a sprint review to nurses in each location.

I’d learned the lesson that

If you don’t manage expectations, expect to fail.

If I had stayed focused purely on the technical delivery of that product, our users would have looked at it and said, “Is that all there is?” Their unrealistic (impossible!) expectations would have led them to be disappointed in what was actually a very good product--one which ultimately made billions of dollars for that company.

Giving Thanks for These Lessons

There are many more lessons I’m thankful to have learned. I often have to live through an experience a couple of times before the truth hits me. Each of the truths I’ve shared with you so far is something that I learned relatively early and that had a significant impact on my career.

But there’s one last lesson that I need to share:

I couldn’t do what I do if it weren’t for you.

I wrote above that I love what I do, and that’s why I’m a workaholic. Except most days it doesn’t really feel like work to me. I couldn’t do what I do if it weren’t for the people who visit this blog or who subscribe to my weekly tips.

And for that I am very thankful to each of you.

What Lessons Are You Thankful to Have Learned?

What are you thankful to have learned in your career? Please share a lesson or story in the

comments section below.

Posted: November 21, 2017

Tagged: management, expectation management, leadership

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.
