

Instilling Agile Values for Creativity, Self-Improvement and Organizational Change - A Manager's Perspective

Ken H Judy

<http://judykat.com/ken-judy>

Overview

The scale and speed of an agile adoption are external measures that don't speak to the founding values of the practice. Collective ownership, continuous improvement and trust are hard won but lead to craftsmanship and joy. They are enabling conditions for innovation and beneficial change.

I will retrospect on my contributions both positive and negative towards cultivating these values in two organizations. The first was a practice that matured over four years, led to a new mission for the team and direct collaboration with the founder and CEO. The second is a team establishing its own agile practice after winding down an engagement with a much larger agile offshore team.

What will I do more of? What will I do less of? What impediments got in the way?

Who am I

I am a software executive manager, developer, product owner and agile coach.

I am a practitioner. Not a consultant. I'm not exceptional. I'm just another dude on the bus trying my best to get somewhere just like you.

I started learning agile practices eight years ago. My experiences writing code and leading projects got me there.

When I started, I worked alone crashing for a deadline. Classic code and fix. I generally delivered on time. Often through sheer force of will.

As I moved onto larger projects, I suffered under well-intentioned but corrosive attempts at waterfall. Some of these projects were successful but the process prized simple agreement over trust. At it's worst it created false hierarchies which hid incompetence and fetishized heroics.

Over time I learned techniques for iterative planning, risk management, and estimation. Through Earned Value Planning, regular inspection points and risk lists we built transparency into our practices. With realistic schedules we were able to maintain a reasonable work-life balance.

The weakness I saw in my team and in my leadership style was relying way to heavily on the abilities and day to day motivation of individuals. I had to manage the project. My best developer had to work on it. If one of us had a bad day the project might grind to a halt. Our whole wasn't adding up to the talents of the individuals.

It took me a while to really grok that excellence isn't about adopting a shared set of practices. **It's rallying around a shared set of values.** I shifted from mentoring my team on how I did things to a conversation about why we do what we do.

Agile Values

...the meteoric rise of interest in and sometimes tremendous criticism of Agile Methodologies is about the mushy stuff of values and culture.

– Jim Highsmith (<http://agilemanifesto.org/history.html>)

Pecha Kucha (The existential joys of agile development)

Earlier this year, I participated in a Pecha Kutchu at agile nyc. 20 slides/20 seconds per slide. I used the opportunity to describe why pursuing agile values brings me joy...

In agile practice I follow a family tradition of care and craft

My mother is college educated but made her living through physical labor. She made valances on fancy drapery and upholstered fine furniture.

She took pride in her work matching the pattern at the seams no matter how complex. And she worked long hours.

She has arthritis from years handling heavy fabric.

My father is a retired engineer.

He's always pursued hobbies with an engineer's precision. Book binding, restoring tube amplifiers, annealing, reshaping and tempering fishing hooks into an authentic 19th century fly fishing hook shape.

If people offered to pay him for his hobby, he'd move on to something else. He did these things for pleasure.

My ten-year old daughter

...Aspires to be an engineer or scientist. She been a member of a Lego FIRST Robotics team since she was seven.

When she first tried out, her teachers wrote:

“You were chosen based on your ability to work well with your team and how well you cooperated with others.

We also looked at your ability to problem solve, on your own and within the group, your endurance, enthusiasm, and your handle and care for the pieces.”

My girl is a born agilist...

As agile becomes popular it becomes a buzzword.

It gets promoted as a tool that solves problems when at its heart it is a set of values that encourage you to confront problems.

We should all recognize these organizing principles...

- Collaboration over negotiation
- Working software over specification
- People over process
- Responding to change over following a plan
- In addition, Bob Martin's "Quality over crap"

Let's talk about following a plan... worshipping a plan

I think of this every time I think of worshipping a plan...

A driver put her faith in her satellite navigation system. It told her to turn onto a bridge. Problem was the bridge had been washed away. She drove her \$160,000 Mercedes into the flood where it was swept away. She had to be rescued as it sank.

Where the customer doesn't entirely know what will succeed...

Where they aren't entirely steeped in the technology...

Specifications become a black hole so dense with detail that even light cannot escape.

Project schedules become the most boring fairy tails ever told.

Mocks mock us.

And process gates ("handoffs") kill collaboration.

We put a lot of energy into delivering the wrong thing on time and on budget.

And we don't even recognize or care about that thing by the time it goes live – if it ever does.

I want to live in our imperfect reality.

...Focus on what I did, what I'm doing and what I want to do next.

I want to know what we are trying to achieve and converse with people I'm achieving it with.

I accept failure

...if we call it out as we recognize it, applaud the attempt and make changes so that we don't repeat that exact failure again.

In short, I love an iterative, reflective way of working because I dearly want to spend each day doing a little less crap and a little more not crap than the day before.

And I want to do it without simply handing off my crap onto others.

Cathie Black was Chancellor of New York City Schools for three months.

She was hired despite having no education experience and no affinity for public schools, parents, teachers and students because she was, “an excellent manager”.

I love that agile doesn't celebrate management. It relies on individual contributors. It relies on community.

The oozy failure wrapped in the chocolatey success of agile

is when we focus on process mechanics and lose sight of people.

If we do, our practice becomes arbitrary and abstract.

There's a study that claims the best and worst performers have more in common with each other than those in the broad middle.

While the best are energized by their caring and use that passion to drive to the best outcomes,

the worst are demoralized and ruined by it.

The indifferent middle, they just plug away.

When we impose a process upon a workplace to avoid failure. We rob the best performers of opportunities to engage and care.

We preclude the best in an attempt to avoid the worst and ensure mediocrity.

I acknowledge that successful products can emerge from awful workplaces. And that that good teams often create failed products.

But working in a way that tears down talented people's desire for work is tragic. To repeatedly do this this is to sap the world of its limited supply inspiration, creativity and joy.

Agile values call for honesty and trust.

A shared ambition to do better and be better while causing each other less unnecessary pain.

I try to remember this in one on ones, retrospectives, coaching and in reflecting on my own decisions and actions.

The great thing about these values is that even as you strive towards them your co-workers will give you permission to demand more of them.

Just as they will demand more of you.

This demand gives you an angel on your shoulder.

Watching you as you work. It inspires even as it shames you into substantial actions that go against your nature. And you do this because your team needs you to.

You invest in the hard daily work of adjusting your own bad habits one behavior at a time in the interests of the people you work with and the work you do together.

This isn't easy. It's mortifying. It's scary.

But the reward is that you get to be the same person

with your boss that you are with your peers that you are with your staff.

The reward is that you get to work at your best with other people working at their best.

And you carry that potential with you as you move on to other projects and other teams.

Ultimately, I want more than success on a project or in a particular job.

I want a career.

I want to be proud of my accomplishments and I want to be proud of who I was as I attained them.

I want to spend my life loving what I do.

And I want to build things that are useful and delightful to people.

This petcha kutchra was inspired by Samuel Florman's book, [The Existential Pleasures of Engineering](#)

Instilling (a retro*)

Disclaimer

* Clearly I will respect my past and current employers, coworkers and colleagues.

Definition of the word

in·still - To introduce by gradual, persistent efforts; implant: "Morality . . . may be instilled into their minds" (Thomas Jefferson).

People

The desired outcome:

Create a workplace that attracts and retains passionate, courageous and smart people

Passionate

- care about quality

- care about improving their own skills
- need to accomplish something

Courageous

- admit their own limitations
- take criticism from peers
- give constructive criticism to peers

Smart

- Sensitive to other people
- Know their stuff.

What I should keep doing/do more of

For my relationship to the team

Don't lie to them

Be honest about the challenges they face with the larger business but also challenge them to see progress when it happens.

Example: Acknowledge that a product owner does not have enough grounding to really engage with the team in a meaningful conversation of priorities and trade offs but let them know what you are trying to do with senior management to improve the situation.

Defend them. Support team's initiative around craft and quality. Let the team take risks and hold them accountable

Team building is not a linear progression to happiness. There will be wrong turns. The team will also be held accountable to things they have no control over. You have to advocate for your team sometimes in the face of consequences to your job security.

Example: Major refactor of a less than one year old codebase visibly adding cost, risk and time to a major project. I defended that decision because I believed they had the hands on knowledge I did not, because I knew they could pull it off, and because I need a team that rallies to a cause and to each other, champions quality and owns the code.

Identify an "other"

As you introduce change and especially as you build a team it helps if you can contrast what your trying to achieve with another group inside or outside the company. At best, this can be a form of healthy competition (bounded cohabitation) with another group. But it can also serve morale to just be better and happier than another group. As the team begins to perform they'll let go of this and may even reach out to improve the state of that other group of people. Of course, there can be a backlash but attempting to change things isn't without risks.

Make changes if they are in the best interests of the team even if they are difficult for individuals

Example: removing reporting relationship to managers

Example: turning private offices into group space

For peer to peer relationships in the team

Encourage the team to make major decisions that affect their working conditions

Example: Invest the time to make hiring of new team members a group activity. Pre-screen then do group interviews. Have the team construct a structured set of questions that test both skills and team fit.

Example: In one team, we spent the better part of three months teasing out the definition of roles for UX, product owner and developer. It took some fraught conversations teasing out what issues were tied to business value and which to subtleties of user experience. The product owner had to learn to not answer questions and let the UX director step into the vacuum.

Allow myself to be overruled

Example: On that same team, we were planning to divide the team in two to take on different streams of work. The team refused and opted to work on both backlogs within one sprint. We worked on a 60%-40% allocation. Because they owned the decision they took upon themselves the extra work to make it happen.

Use the expectations of fellow co-workers as a corrective force for team members

Example: Rather than become a repository of individuals impressions of each other, turn them into three way conversations with the team member in question

Encourage the team to review and critique their managers

Example: Only group in company to perform 360 reviews rather than the standard annual one way performance review

Bring in a “closer” but not a hero

At least one person in the team or scrum master or delivery manager role who closes out tasks, who can be trusted to trail blaze a new role and who gives me unvarnished feedback.

For individuals in the team

Have patience and faith that people can adapt their behavior over time

Example: Aggressively motivated developer (“hero”) who would drive himself so hard he made bad decisions and wrote unmaintainable code. Experience and responsibility made him over time into an excellent leader and craftsman.

Get to know what intrinsically motivates people and use that to help them grow.

Example: Skilled developer held back by pairing with others who aren’t as skilled. Reward and compensate that person for mentoring.

Encourage team to participate/contribute to the community

Example: Co-author with one of my product owners and then send the whole team to a conference as developers put together their own presentation topics

What I should do less of/not do at all

Let bad patterns fester

Example: Three person team pairing with one junior developer. Was unproductive for that junior dev who ended up opting out of the team.

Example: Team stuck in forming. Contrived collegiality.

Wait too long to manage people off the team

Example: The developer who knows the legacy systems but shows no aptitude or initiative in learning new skills or in integrating into the team. Presence of developers in the room who are not integrated into the team has always been a fail for me.

Lose touch with the team

Example: Spending too much time on external projects , vendors and business meetings to spend predictable windows of time in the team room. Don't lose touch with the human dynamic in the team even if you've delegated management to others.

Lose touch with the work

Example: find time to mix in and code if possible but don't become a bottleneck. Focus on maintenance work and troubleshooting. The little overhead you create will be made up for your improved ability to feel your team's pain and to root out their impediments.

Lose touch with individuals

Example: When I made a commitment to meet with each team member one on one for an hour, I ended up re-organizing the team reporting and set clear development plans for specific individuals.

Get sucked into the worm hole of recruiters

Going through recruiters is one of the worst ways to find talent. All the resumes are so touched up they often don't reflect reality. Once the recruiters find out your looking, they flood your phone. Find ways to build a reputation for your team and relationships with the larger development community so I don't have to resort to recruiters.

Promote the guy who eats a sandwich during meetings

Almost every good leader I've worked for has at one time or another had a trusted lieutenant who didn't carry their weight. He's the guy who sits in the meeting reading his blackberry or eating his lunch. Who is always very busy but who doesn't DO anything. What they're good at is taking credit for other people's

work and selling it to their leader. I neither want to be that person nor do I want to shield one of those people.

Ecosystem

The desired outcome:

An organization within which your team will thrive and contribute. A diverse organization that entrusts teams of individuals to own their work.

What I should keep doing/do more of

For the team

Articulate a cause

Put the work in context of what senior executives are trying to achieve whether those goals are tangible and measurable or not: revenue, customer acquisition, consumer attention, industry reputation... Place all work in the portfolio within that context (even if it's not strictly my job to do so)

Celebrate victories

Acknowledge those small victories along the way. Incremental changes that improve the team or the relationship of the team to the larger company. Definitely cheer the big victories, even if you find yourself the only one giving a standing ovation in the auditorium.

Make enemies

When doing so creates necessary change. Sometimes you need to fire the client even if the client is another department. Better have earned some trust with your boss before you do this.

Not make enemies

When doing so does not effect change. This just enflames the politics. You all lose sight of why you're there.

Get the team into conversations to brainstorming the thing

In the media companies I've worked for, the decision making tends to be hierarchical with the effect that many of the most significant decisions about a product/project are made before the actual individual contributors are given a chance to participate in the process.

At Oxygen our CEO encouraged the team to brainstorm product ideas and present them un-filtered to her. Direct access between her and the team led to an ambitious effort to expand the company's mission into consumer software. We made more progress in 8 months towards achieving this strategic goal for her than we had the prior 7 years of the company's existence.

Bring in outside expertise to challenge and validate

If your leadership really respects authorities, a thought leader can add lift to your changes. But even better, if it will work, is to collaborate with successful peers in your own software development community. Practitioners helping practitioners. But beware of “certified agile” see below...

For myself

Choose leaders that are worth working for

You're entrusting them with your labor and you're recruiting and building a team off of you're own reputation. Make sure you have something to learn from them and something to accomplish with them. Make sure they will protect you if you need and deserve protection.

Demonstrate loyalty - earn trust

People seem to be surprised that they have to earn trust. My experience is that you always have to earn trust particularly if you want permission to build the team you want and manage them the way you want. The best way I've found to earn trust is to deliver on an ambitious goal but it also involves small actions that let your boss and co-workers know you have their back. Genuine collegiality helps. False collegiality hides dysfunction and is a form of politics.

Walk in other people's shoes

A cliché, but don't judge a co-worker until you've tried to do what they do. Get their permission to work with them. For example, helping an organization move from over reliance on hi-fidelity mockups by volunteering to help the UI team with html mocks.

What I should do less of

For the organization

Allow the business to duck the why this? question (business value)

We go into so many projects with subjective and volatile measures of success. How can a team rally to that. Fight to isolate a set of measurable benefits but realize that if they don't capture the real reasons why a project is being done delivering on them wont matter. Be willing to acknowledge intangible benefits: corporate reputation, demonstration of ability, learning for the company...

Allow the business to hand over a static mockup as viable description of a feature

Visual people tend to approach software stories by thinking of how they want screens/pages to look. They spend enormous effort documenting these details and then hand this off to development to build. What's missing in that approach is a conversation on the value of the feature, on the interactions that are at the heart of the feature, and the relative tradeoffs in execution that can drastically affect the cost of implementing the feature while still deriving it's essential value.

Functional roles as process boundaries (“handoff's suck”)

Allowing the organization to isolate meaningful product conversations and suck up calendar time on fuzzy front end thinking on the business side before engaging development to build the thing.

For the team

“certified agile”

Time to stop pushing agile as a thing or the practices as an end of themselves but get back to getting buy in on what we're trying to achieve and why any particular practice or decision supports that goal.

I spent one year un-packing the definition of “agile” created by the outsource provider an employer contracted with to build out our websites. They won agile adoption for the company - something I might not have been able to do myself. They delivered the project. But they left a legacy of magic consistency built on floating resources and unsustainable pace, points as dollars, development held accountable for product owner responsibilities, and heavy weight reporting.

Not one team - offshoring

I allowed a relationship with an offshore contract team to suffer because I didn't challenge the assumption they would be rolling off in the near term. So, I didn't invest in travel exchange as much as I should have or take the hit of blending the backlogs across the teams. Then more money would be found and the relationship would be extended. I came on mid-stream and never wrangled control over the funding as I should have. I better not make that mistake again.

Allow a team to be banished to second class

Challenge yourself to not divide work into convenient silo's backend/front end, admin tools versus consumer facing app, maintenance versus new development. I've had to dig out of the first two divisions.

For myself

Not knowing when I'm winning

Best advice I ever got was to know when things are turning in your direction because there's a delay between when you affect change and when you can feel that change and during that time the people who have lost influence will raise a stink. If you engage to loudly in these parting shots, you can actually give them credibility and slow your own progress.

not-interesting work, rewarded accomplishments for thankless ones, and create inherent bottlenecks.

Giving too much information

In my effort to be transparent to my teams, I can tell them too much about the impediments I was working on for them. As one of my team once told me, “I don't need to know this because I can't do anything about it. All it does it depress me.”

Becoming a black hole of information

It's very difficult to balance protecting the team - screening out noise and becoming an impediment yourself by having too many conversations and influencing too many decisions that affect your team's ability to self-organize around the work. It's painful when the team has to unpack and reverse decisions made without their input. It's even worse when you're the culprit.

Things I can't change (impediments)

time - lifecycle of the business/lifecycle of the team (local optima)

There's a huge risk it will take longer to build a real team than a business can maintain the specific environment required to sustain that team. Sounds negative but it sucks to spend years building something to have the business change management and for that team to hollow out in weeks.

top secret

Any business has secrets. Sometimes those secrets are material to decisions you would make about a product roadmap. Sometimes, the people shaping that roadmap don't know. Sometimes knowing that information would change the product enough to avoid failure.

the budget cycle

If we don't know what we want until we see it, how can we know how much it will cost months before we start work. Annual budget cycles divert so much energy and create so many artificial constraints on projects. Because the problem is, sometimes you can't solve a time and budget constraint by trading scope. Sometimes you're just baking in failure.

hierarchy (over centralization)

command and control wastes so much initiative and hands on knowledge.

fiefdoms (over compartmentalization)

individuals at all levels carve out little domains over which they exercise control at the expense of the strategic interests of the company which makes the organization inflexible and unfocused. Examples can be IT organizations, project management, development, product, marketing... why stop...