What happens when you take an agile coach and force them to "eat their own dog food"? This is my personal story about transitioning from an enterprise agile coach to a programmer on a team, and what I learned about agile, coaching, and myself in the process.

1. INTRODUCTION

Agile coaching as a discipline has experienced incredible growth in recent years. The market for coaches has exploded as companies scramble onto the agile bandwagon. The concept of “agile coaching”, and even “coaching” in a broader corporate sense, is a relatively new phenomenon. Lyssa Adkins' seminal book, “Coaching Agile Teams” first published in 2010, and Google Trends has shown an exponentially increasing relevance of the search term “agile coach” (see Figure 1). Anecdotal evidence suggests more are companies are establishing internal agile coaching structures and roles.

I am personally fortunate to know many wonderful, capable agile coaches whom are passionate about bringing both productivity and joy back to our workplaces. However, given the meteoric rise in a discipline that did not exist a decade ago, it may be worthwhile to pause and consider whether “agile coaching” structures should be institutionalized, permanently woven into the fabric of our organizations? I recently had the opportunity to consider these questions from two different first-hand perspectives: that of an enterprise coach, and a team member.

There are plenty of stories of developers, quality specialists, project managers, and managers making the leap to coaching – how often do we hear of the transition in the other direction: an agile coach becoming a developer? After two years of coaching teams and executives around the world, I became disillusioned by “coaching” and volunteered to instead be a developer on a team, my first time in a programming role in 15 years. The experience of returning as a programmer was enlightening; it substantially changed my perspective on coaching and agile. Having worn the shoes of a team member, I acquired a new empathy for their world and perspective. This is my personal story of entering the world of agile coaching, moving to enterprise coaching, eventual disillusionment, then to being a team developer. Ultimately, after experiencing “the other side of the coin”, I personally found myself more skeptical, if not outright cynical, about the long-term effectiveness of coaching in an organization.

My agile origins began as a product owner. The technical leaders with whom I worked were self-taught agilists - students of lean and agile principles, voracious readers, theorists, and experimenters. In the era before agile certifications became popular (no one at the company held an agile certification) we delivered production-ready software every two weeks, and eventually every week. The development teams were not even necessarily aware when a release went into production. We practiced test-driven development. We did not flinch when priorities changed. The experience was refreshing and liberating compared to the ponderous and paperwork-heavy pace I had experienced with other companies. I was sold on agility.

In 2013, I fell into the world of agile coaching by accident when I was hired by a small, privately owned oil & gas software company of around forty developers to assist them with their agile transformation. I jumped at the opportunity for what I saw as the ultimate leadership challenge: to earn the trust and respect of my new colleagues and influence the organization without explicit authority. I wanted to take all I had learned at my previous company – the leadership, technical and product management practices that enabled us to release production-ready software every two weeks – and help bring that to a new organization. At the time, neither the company nor I had ever heard the term “agile coaching”; we did not know what to call my position, but it did not matter because we had a common vision of the outcome.

The six teams I began working with were ripe for change: they were enthusiastic and had a thirst for learning and progress. Their culture already valued adaptability. I quickly went to work by taking a theoretical approach to agile training; I was deliberately non-prescriptive about any specific methodology like Scrum or Kanban. I began illustrating to teams how work-in-progress, decision gates, and batching all inadvertently hurt productivity. I believe this equipped future generations of leaders to understand the “why” behind agility and pass on the knowledge forward.

In hindsight, one of the criteria I believe made this “transformation” successful was that the person who had hired me, “CJ”, was the head of the software engineering organization, and one of the partners of the privately-held company, treated me as an entrusted personal advisor. I felt I had an influence on CJ, as I could see him shift his thinking and approach after the many long conversations in his office.

Within a couple months of my hiring, my new company was acquired by a much larger, international public company with thousands of developers spread across the world. They, too, were undergoing an agile transformation, so it was an exciting time to be a coach in this new dynamic world. In the meantime, though, my focus remained with the local, co-located teams. After approximately a year and a half of coaching these teams, I began to appreciate my value to them was declining, as they had picked up agile and were running with it. I was particularly encouraged with how product management was on-board. I declared success and took the opportunity to transition to coaching roles in the wider company.

I entered the world of Enterprise Agile Coaching; I now had the opportunity to coach and advise up to Vice Presidential level. I was buoyed from learning that, in some corners of the company at least, I had acquired a positive reputation for my non-prescriptive approach to agile, and what I felt, was my emphasis on explaining agility in business and economic terms that resonated with executives. I was proud of my ability to nudge and influence the organization, slowly but surely, I felt in the right direction.
2. DISILLUSIONMENT AND A NEW DIRECTION

My feeling of accomplishment as an Enterprise Coach slowly wore off, and it was not long before I began to feel entirely disillusioned with coaching. I felt I was no longer experiencing the success as I had with the teams I first coached. Various managers in the company routinely sought my coaching for their; despite this, I was under the growing impression that my coaching was having much long-term effect. Initially, I had a difficult time understanding or articulating the source of my frustration. That would come later, but in the meantime, I knew that I needed a change.

After observing my friend, "CE", take on a managerial role with a different team, I developed a hypothesis: perhaps the best way to coach a team was to influence from within? Perhaps being a team-member, wholly invested in the work and the outcomes, with a knowledge on agile values and principles, provided the best platform to inject these ideas and considerations into the day-to-day interactions of the team?

In my time coaching, I developed some empirical evidence to support this hypothesis; teams that had a strong agile change agent within, someone who understood the "why" behind agile, the principles and values, had a much greater chance of avoiding dogmatism. My theory was that the "dark matter" of agility came from 85% of that which was difficult to observe: the innumerable number of micro-decisions and interactions made by team members on a daily basis. Like the mysterious hypothetical "dark matter" that makes up 85% of the universe, I posited that the agile "dark matter", contributes to the majority of the team's true agility, yet is unobservable in the standard Scrum ceremonies. This harkens back to my first experiences of Agile, where strong team leads, extremely knowledgeable in lean-agile theory, had the largest impact on myself.

Frustrated in my role as Enterprise Agile Coach, I took a bold and risky move: to transition to a developer on a team, the team on which CE was now managing. The reason I became a developer over another organizational role (say manager or product owner) was partly because that role was most accessible in the organization at the time.

It would be the first time in fifteen years that I would hold an explicitly development job. The fear was compounded by a new product, with new team members whom I did not know, and did not have an established rapport.

3. LIFE ON A TEAM

Fortunately, many of my fears of joining a new team were unfounded. My new team members were supportive, welcoming and very good at their jobs. Being on a team, I was quickly forced to "eat my own dog food" – all those practices I preached as important as a coach – stories, stand-ups, retrospectives, story points, showcases – I would have to experience first hand now as a developer. Initially, I thought that my previous experiences as a Product Owner and later Agile Coach would be sufficient agile experience; I did not think I could possibly learn anything "new" about agile from being a developer, but I quickly discovered how wrong I was.

It was not long before I sensed a tinge of agile cynicism from my team members. The history of agile on the team was long and storied. Agile came to the team around six or seven years before my arrival as a top-down mandate without much explanation of what or why. My impression of the team's approach to agility was that they went through the motions of Scrum because there was the expectation they had to, but it was not something they would continue to do of their own accord given the choice. However, I soon began to share my team's agile cynicism.

4. I BEGAN TO LOATHE THE STANDUPS

Very quickly, I began to loathe the standup time. I would usually find myself deep "flow", the state of deep concentration and incredible sense of productivity and accomplishment, described by Mihály Csíkszentmihályi in the book of the same name. Inevitably, I would find someone tapping on my shoulder to remind me it was time for the standup. I would usually curse under my breath, as I would flush my mental stack on the problem I was tackling, and impatiently stare into the distance while waiting for my turn to mutter off a few words about what I was working on.

The agile coach in me could quickly diagnose the issues: each team member was working on largely independent work, and therefore there was less benefit in intra-team status updates to coordinate. Secondly, we should choose a standup time that works for us as a team (easier said than done.) Over time, we as a team began to converge towards collectively working on only one user story at a time. The standups became marginally more interactive and valuable, as now we needed to coordinate our collective efforts. However, this was not always the case all the time for all stories.
My experience with the standups was my first and most painful reminder of how easily going through the Scrum motion can miss achieving their intended purpose. I came to believe that the standups were not being done for the benefit of the team members, but they were done because there was some unstated expectation from above that they should be done.

5. WASTEFUL RETROSPECTIVES

As a coach, I used to tell teams that they can change anything about their process, but keep the Retrospective, as it was the sacred ceremony for process improvement, therefore I was surprised when I began to detest the retro almost as much as the standup. Many of the reasons behind my dislike for the retro were similar to those of the standup: I wanted to be productive. I wanted to code. I wanted to get in the “flow” and stay there. Retrospectives were an impediment to all of these.

I was certainly an advocate for continuous improvement; I saw no shortage of opportunities to make things better. However, rarely would we delve deep into any of these topics in the retro. There always seemed to be elephants in the room, or tacit assumptions that some things were too big to change. Many of these were related to the codebase in which we were working – we would be constantly tripped up by the antiquity of the codebase, yet doing something about it seemed out of reach. Other “elephants” were related to product management, an unshakable “all or nothing” mentality that got in the way of prioritization.

As a result, the retrospectives tended towards debating insignificant minutia, (such as agonizing over specific wording of acceptance criteria) when compared against the overwhelming organizational impediments that we would conveniently ignore. To me, we were investing in micro-optimization, while simultaneously ignoring the glaring organizational impediments around us. This frustration was compounded by the tacit organizational expectation that we hold retrospectives (less we don’t “look” agile). It seemed as if the retrospectives were going through the motions for the sake of agile appearances.

In an effort to combat this, as a team, we ran an experiment: we tried having our retrospectives late on a Friday afternoon. The knowledge that the rest of the day would be a write-off anyways was a convenient antidote to the desire to get back to the desk. Further, we had the retrospectives offsite in a casual setting. Personally, I felt this very much opened up the conversation. We spoke more candidly, and we also began envisioning grand futures and possibilities. I found these conversations to be much more energizing and useful to build common clarity of purpose for our team and where we wanted to go technically. The shift to an off-site and unstructured retrospective, in my opinion at least, led to much more creative thinking.

6. SOMETIMES STORIES CANNOT BE SPLIT

As a coach, I was a firm believer that stories could be easily split. I would react with immediate suspicion whenever a team would suggest that a story could not be split into smaller stories. Surely, there must be a way? And yes, there often was, but for every time I helped a team find a way to split a story, I encountered many other teams who were splitting stories that no longer honored the value of the story – they split along technical or task lines to force it into the iteration. Consistent with Bill Wake's INVEST criteria for good user stories, a user story must be deliver business value when complete. Where I was non-prescriptive on specific ceremonies and practices, as a coach, I shared no such flexibility about honoring the definition of valuable user stories. To me, playing fast and loose with the definition of a story’s “value” fundamentally broke the purpose of agile: to obtain feedback through early and often delivery.

Now, as a team member, we began working together on an initiative to simplify and add new functionality to certain back-end parts of an application. We encountered the problem of defining “small” stories; as we were wading through 20+ year old code across five different languages, we realized that even the most trivial “valuable” change required significant efforts. Every time we thought we found a legitimate approach to keep the stories small by containing the extent of change, we were continuously proven wrong once we began coding. Repeatedly, we would come to realize that our presumably small change would force numerous unforeseen changes elsewhere. The high degree of coupling placed a fundamental boundary on story splitting.

After trying five or more different approaches for splitting while honouring “value”, we eventually exhausted any belief that we could find a way to contain the changes to small, incremental stories. Each approach came with “legs” that would spider out significantly increasing the size of the story into multi-week endeavours. Finally, we had to accept, even having all team members “swarm” on stories, that there would be a
minimum multi-week investment before we could realize the first increment of value. It was a slow, long, tough slog and we finally had to accept that.

Legacy, tightly coupled code was largely responsible in this instance, however the experience forced me to abandon my earlier naïve coaching perspective that there must always be a way to split a story into something small.

7. WHY I BECAME DISILLUSIONED IN THE FIRST PLACE

With several months of experience on the team, I had an opportunity to look hindsight and evaluate both how I approached agile coaching, and what led me to become disillusioned with it in the first place. What on the surface looked like success -- mid-level managers contacting directly to seek my coaching services -- I now realize was a losing proposition in the long-term. Every time I would engage with a new team under the request of a mid-level manager, I would consistently discover similar situations: competent teams of developers behaving completely rationally in the context that had been created for them. On more than one occasion did I hear that the developers needed to be “whipped into shape” – and the agile coach was expected to be the one to do it. Frequently, “coaching” was sought by management who believed something was “wrong” with the team. Leadership expert Dan Rockwell once said, “If you want something different, do something different.” Unfortunately, to many leaders, “doing something different,” meant simply “hire agile coach and inflect them on your teams.”

My experience as a developer unexpectedly reinforced this. While I tried to escape the macro environment of enterprise coaching, I came to appreciate that what I had been trying to avoid in coaching, I was now experiencing from a different perspective as a team member. I began to appreciate ever more acutely that my team was behaving perfectly rationally in the context that had been created for them. A risk averse, “keep your head down” mentality was a culture that had been cultivated around them, by management who had not yet internalized how agile values and principles impacted the way in which they themselves needed to change.

If agile represents values and principles, local economics within the organization (incentives, pressures, rewards, punishments) had to be aligned. Managers held control over these economics, and therefore it was imperative not only for them to change these economics through changing their own behaviours, but also if they were to perpetuate them in a sustainable organization, they had to understand them well enough to teach and develop them in others.

In my first coaching engagement, I did not experience these same frustrations. I posit this critical difference was the nature of my relationship with the executive level; CJ sought my advice of his own accord and allowed his own thinking to be influenced. Unfortunately, this type of relationship did not repeat in my subsequent engagements, at least not anywhere to the same degree.

Managers at all levels, as those responsible for holding the power to control local economics, had to internalize agile principles and values, to such a degree that over the long term, the need for a separate coaching role would be obviated. As those responsible for evaluating, developing and promoting future leaders in the organization, managers not only required a deep understanding of agile principles, but also the skills to teach and pass them on. I came to the belief that to effect this type of persistent transformation, managers needed to seek coaching, not for their subordinates, but for themselves.

8. CONCLUSIONS

After spending several months with the team, I moved onto a technical, more R&D focused role, in the company. I am grateful for the experience and the support of my teammates for the time I had with them.

Looking back, despite my prior experiences as a product owner, team coach and enterprise Coach, I was shocked at the extent I did not fully appreciate life on an Agile team until I had experienced it first hand as a developer. As a developer, I longed to be productive, to get into the flow and write software. It was very natural to feel disdain for anything that got in the way of this, including anything under the “Agile” banner. I quickly went from being a coach, to being “that” team member, the jaded, cynical one that every coach dreads. I empathize with the teams that have agile coaching “inflicted” on them.

The experience drove home what I believe is a foundational concept behind agile that I do not believe is necessarily well-communicated by coaches: developers want to be productive and are in the best position to determine how to maximize productivity, but are too frequently impeded by the organizational context around them. My hypothesis of “influence from within” was somewhat disproved. The “dark matter” I expected to find, the multitude of small factors that influenced a team’s agility, did exist, but its source tended not to lie within
the team itself, but the many environmental influences on the team: product management strategy, incentives and disincentives towards risk-taking, and management performance expectations. No amount of team coaching could have a long-term impact without changing the context; coaching influence was not needed for the team, but above the team. This discovery was consistent with the frustration I had experienced as an enterprise coach: an organizational tendency for managers to seek out coaching not for themselves, but a force to be brought in to “fix” what they perceive as unproductive teams.

While my own experience is too narrow to draw a general conclusion about agile coaching as a whole, it has certainly caused me to be much more skeptical about the long-term efficacy of coaching as an organizational strategy. I now hypothesize that the accessibility of coaching services can be a threat to sustainable agility for an organization: coaching services (whether through an internal coaching structure, or outside consultants) may subtly encourage managers and executives to “outsource” agile knowledge and provide a convenient excuse to not learn it themselves. How can agile values be organizational values if the leadership chain does take responsibility for internalizing them?

If I were to go back to coaching, I would be much more insistent that several conditions exist for the coaching to have sustained impacts on the organization:

- Leaders (managers and executives) seek out coaching only for themselves, not for their teams, subordinates or others. Leaders must “lead by example” and demonstrate the vulnerability associated with seeking coaching;

- Leaders appreciate that agile values are organizational values that influence decisions and behaviours at all levels across the organization; and

- Coaching should almost universally begin and focus on the product management organization; influencing the mindset of how products were managed and work prioritized was critical.

Looking at the agile coaching industry gold rush, I am personally worried these lessons might get lost.

9. ACKNOWLEDGEMENTS

I would like to thank my shepherd, Mary Ann Lapham, for her guidance; her support is what made this paper possible. Thanks also to my mentors and friends, Todd Little, Chris Edwards and Meg Ward, whom are perpetually supportive and whose candid feedback was very much appreciated.

REFERENCES
Adkins, Lyssa “Coaching Agile Teams” Addison-Wesley Professional, 2010