

Against all oddities

*or how a developer became an agile project manager*¹

Abstract

A retrospective of the transition I had to make as I moved from being a developer to playing the role of a project manager on a large software development project.

1 In the beginning

During my several years as a developer (and for those of you snickering, I still am one!), I've always wondered what exactly the big deal about project management was. It certainly seemed like a sinecure - apparently, all you needed in the way of qualifications was a hard-boiled exterior that could withstand the boredom of long and arduous meetings. I also felt compelled to make light of the project manager role, thinking that any one could do it, and it took real talent to mess up a project. Oh, and I also thought I could easily do the job, maybe even better. I once even wished I'd get a chance to play the manager, just to prove to myself that I could put my money where my mind was.

Like they say, be careful what you wish for. The past year has been a time of transition - as I felt my way through the journey of leaving familiar territory of developer-land to the somewhat more eccentric world of software project management. The following is some of the strangeness that I witnessed and the lessons I learnt as I made this fantastic voyage.

2 First impressions

Here are the things that I saw and observed first. Of course, today they seem a bit mild, but during those first few days, everything seemed strange and unfamiliar. Why were people not doing anything? What did these folks bring to the table, really? What were those people thinking? All these seemed questions with impolite answers.

2.1 No stuff, just fluff?

Within the first couple of days, something very disturbing struck me. I didn't think I was doing a lot in the way of tangible work. I mean, I was attending

¹DRAFT - 7th December 2006. Written by Amit Rathore, ThoughtWorks Inc., Chicago, Illinois. www.thoughtworks.com, www.epistemologic.com

meetings, talking to people, making *plans*, and all that sort of stuff, but I didn't check anything in when I was done. Nor did a green bar ever tell me I was on the right track. The end of the day was no longer marked with a victory declared against an insidious bug but with a question mark - what on earth did I really accomplish today?

Luckily, this was a very large project - there were about eight teams running in parallel with about 10 - 15 people on each. This also meant that each sub-team had a project manager who operated under the umbrella of a program management office - a program manager, a product manager and an overall project manager. Whatever else I thought all these people did everyday, I hoped they could help me with one thing - explain and justify my existence.

Thankfully for my idea of self-worth, they explained that these are necessary functions. A large effort like the one I was part of couldn't exist without some overall planning, some overall vision - and a bunch of people whose sole job it was to "oversee" the whole operation. I figure that it all boils down to accountability. It is all very well to think that a self-motivated, talented set of software developers can deliver working software. This is true in many cases, yes - but the more I thought about it, and looked around me at the apparent chaos - I realized that there is a certain aspect of responsibilities that really don't belong to these fine folks. It is a cross-cutting concern which is critical for a software team to be successful but does not truly belong to it. Enter, the project manager (among his/her other responsibilities.)

2.2 Feeling the elephant

As a developer, I thought I was good at looking at things from different levels of abstraction. I could look at the entire project, and think about architecture and decoupling and integration and so on. I could also look at a method on a class and make changes based on the recommendations of my olfactory senses. And I could switch easily between these different levels of abstraction, and talk about the system as a whole and also about the impact of a particular choice of pattern used in a module.

Within the first few days of being a project manager, or perhaps because of hanging about in the company of my distinguished counter-parts from other subteams, these technical details started to seem only mildly interesting. I was sure that they were, still in fact, very important - just on a different plane of existence. This was strange - and I found myself saying things to my team-members that I had heard my project managers say to me in the past. My *deja-vu* began to have *deja-vu*...

The fact of the matter, it turns out (and I didn't really accept this without

a good amount of kicking and screaming), is that from the vantage point of a project manager - while these other things matter a lot, they're only one of many issues that are on the watch-list. The project-management team, if you will, grapples with issues like - what date to commit to with the senior management and marketing folks, how to present this latest setback so that we can get extra budget approved and not have the project cancelled, how to ensure that all external teams play well with each other so that there are no unnecessary delays, how to get more communication going so it obviates having to use a heavy weight CMM Six Sigma Seventy Seven Delta process, and so on and on. There are literally dozens of such issues that need to be thought about, and for which solutions need to be devised, and whose progress, in turn, needs to be monitored so that nothing unexpected ends up biting the project in the backside.

3 Responsibilities

When I used to be a developer, I always thought this whole idea of *people* management was a bit far-fetched - after all, it seemed to me that all my team-mates were intelligent, motivated people and we could self-organize into a highly productive, well-oiled machine that produced excellent code. What was there to manage, really?

3.1 Risks, schmisks

As described previously, most projects have several potential problems lurking about that could cause the project to flounder. In fact, a list of these potential issues is maintained by all good project managers in Microsoft Excel (TM) (the tool of choice for any self-respecting project manager) and it is dubbed the Risk Management Spreadsheet.

The idea is that all the stake-holders (project managers and product managers) get together in a room, and go over this list together, adding to it any new ones that have been discovered, and updating existing ones with the status of their mitigation. This ensures there is a constant, proactive overview of potential things that could have disastrous consequences for the project if they were to happen. Some people say that risk discovery, mitigation and monitoring is at the core of a project manager's job - and I tend to agree with this point of view.

3.2 The people stuff

This one seemed to me, in my past life, the most icky of all things that “management” does. They talk about *morale* and *feelings* and *empowerment* and things like that. All very management like. I always figured that it was just my particular project team that had issues which needed to be sorted out, and that most other teams in the world were just fine. What I found out surprised me.

It certainly appears as though everyone on the team, and I mean everyone, has some or the other problem bothering them. I even had one person tell me that he was deeply concerned that he had not been asked about his vacation schedule yet - and that clearly, I didn't think he was important to me or to the team. When I first heard that, I couldn't help but look at him in total astonishment. I was bewildered, really - I certainly thought he was a fine member of the team and I already knew he wasn't going to take any vacation during the time period under consideration. And yet, I had managed to, well - hurt his feelings. While it is possible to attribute this one incident to an overly sensitive ego (which I did for the most part), I realized that it is dangerous to take things for granted. Anything and everything that happened on my team had someone disagreeing with some aspect of it, and it fell (at least partly) to me to ensure that no one was feeling unvalued. There - I used the F word.

We all say, in fact it is one of ThoughtWorks' values - people over process. When I was a developer, I certainly appreciated the latter half of that - in fact, I would take *anything* over a rigid process. In the past year however, I've truly begun to appreciate what is meant when it is said that people are the most important asset. A well-gelled team can only happen if the people comprising it are comfortable with each other, if the environment created for them in the team room is collegial enough to be conducive to working well, and if part of the job of the project manager is nanny. Just kidding.

Like I said earlier, I used to be of the opinion that a lean, mean, code-producing machine can be formed by well-meaning and motivated technologists without any other input from any other type of team member. It turns out - that although it may be possible in some cases - and even then it certainly needs a lot of luck, most often these teams are grown with a bunch of support from the project manager. It behooves the project manager to be on a constant lookout for potential trouble-spots within the team, to shield the team from outside trouble and to generally ensure that the team has what it needs to be at its operating best. Regular and frequent one-on-one meetings with the team members is a good way to keep up with the pulse of the team. To summarize, while “growing a team” sounds like a rather

wishy-washy goal, it is one of the important aspects of a project managers job.

3.3 The lean, mean, value-machine

Coming from the mindset of process being a bad word in general (translates to lots of unnecessary documentation, meetings, bureaucracy and a lumbering slowness that slows progress to a crawl), I realized that not all process is bad or unnecessary. Some is needed. In a way, I guess I knew it all along - after all, Agile is not chaotic or messy. It is a process that needs lots of discipline and structure. The important thing is to defend this simplicity and constantly find ways to make the process leaner.

In the past year of playing the role of project manager on my team, I've had to, on several occasions, fight back ideas to add "structure" to our process. In my view (and in that of my counter-parts in other teams) these additions to the process were meant only to satisfy someone's idea about improving overall team productivity through process improvement. They didn't actually fulfill the purpose they were meant for, but they *seemed* like they might. It is important to constantly ask why. Why, why, why? And only if all those questions can be answered satisfactorily, should the project manager allow the team to do something that does not directly add value to the system being developed.

The other thing I realized as I played this role was that size matters. The size of the project team(s), I mean. One can get away with little or no formal process when team sizes are tiny to small. However, as both the number of teams and the size of each team begins to grow, it becomes important to throw in the towel. Uh, I mean... to understand that a structured process is needed to manage the interactions of all these people. The process, just like a technical solution, should be simple, elegant and lightweight - but no simpler than the inherent complexity of the situation. So, the point of this tirade is this - a developer trying to learn the art of project management needs to accept process as a necessary evil, strive to keep it within limits so as to ensure it doesn't get in the way of real work, and constantly defend his team from having to do more of it - all within reason.

I might fall asleep during the 7th meeting of the day, but I consider my job done if I could keep my team members (who do the actual work of producing working code) out of these meetings. And in those meetings, if someone were to propose something that just adds non-work to my teams already large backlog, then I consider it my duty to rouse myself from my (light) slumber and ferociously question the value of the proposal before it succeeds in detracting from the leanness of our process.

3.4 Sharpening the saw

Speaking of ferocity, it is also the job of the project manager to pounce like a panther on every little opportunity for improvement that comes up. Being lean and staying lean also involves cultivating a certain mindset - of reducing the *mudda* or waste from every process. Anything that doesn't directly produce value is a candidate to become roadkill.

So, if the processes or practices can be altered to add more value, it should be done. In the same breath, the project manager should also keep in mind, and this is especially true in larger organizations, that pace of change is slow - in fact, there may be a strong resistance to change. You might hear that change is the only constant, that it's the adaptive species that survives, and all that good stuff - but let's face it - change is plain annoying - it's heavy, rattles about in your pocket and if you drop it, the coins roll all over the place.

On a more serious note, however, despite the appeal of radical innovation and dramatic changes that fix everything at once (*kaikaku*, as the Japanese say), what seems to work better is slower, smaller changes as things come up that allow these to happen (*kaizen*, in that same Oriental tongue). As a developer I get impatient when things seem broken and I can't fix them right away - and I had to learn that in this world, fixes are organic and take time. It's a hard thing to internalize, but required (in order to avoid those ulcers).

3.4.1 Picking your battles

The corollary to the above is that a good project manager should carefully pick the fights to umm, pick. There are probably going to be many things that as a developer seem just wasteful or absurd, but not all are worth the effort to change. It's like optimization - without measurement (or careful thought, in this case) it is wasteful to try and squeeze out more performance from the system. It may be that the place you're optimizing is just not the bottleneck.

Of course, shaking things up is fun, so this advice may be ignored in favor of a good time.

3.5 Grub

I shall not make light of this serious topic. Food is just too important.

Snacks are a great way to get the team talking to each other, playing around, and in general, gelling together. On a couple of projects, we've used a chart on a wall that tracked the teams current favorite snacks and the

date of the next snack-drop. It was mounted right next to velocity charts, scope burn-downs and so on. There is always place on a team room wall for important information.

The other important classes of social eating events are the team lunches and dinners. These are probably one of the most important factors that count towards closely knit teams. It provides a forum for team-members to unwind in a non-office environment and let their hair down (or their feet up, whichever one you prefer). After all, a team that eats together, stays together.

3.6 Looking back

A retrospective is basically a look-back on what the team has accomplished and what has occurred over the past iteration or a given duration. It affords an objective look at things the team did or did not do, and allows suggestions to be made around what to change. It also provides a forum to recognize good things that happened during that time-frame, done by either the team as a whole or by individual team members.

The most important goal of a retrospective, of course, is to find ways to improve the overall productivity of the team and do so through ways that the team buys into. Retrospectives are facilitated sessions (either the project manager does the facilitation or more ideally, gets someone from outside the team to do it) - and should be conducted on a regular basis throughout the project duration. I will not go into more details about this here, but ensuring successful retrospectives are carried out and that the findings are incorporated back into the team process and culture is a responsibility that falls squarely on the project manager.

Finally, retrospectives are a great venue for practical jokes. I retract my previous point, this is the most important aspect of a retrospective.

3.7 Time passes, will you?

In many organizations, the one thing that is the most not looked forward to is annual reviews. Not so at ThoughtWorks, at least in general, but they can be unpleasant at times. Luckily (for my reviewees) this was not the first time I did people's reviews, but even so, I'd not played a real project manager before. So, to ensure that I did at least a half-decent job at this, I asked around for advice on the matter.

I will not wax eloquent on this topic too much, but the only things worth saying were - never surprise anyone during their review (as in, let them find out they were being measured on something and they didn't know it or that

they were doing a bad job of something but it was never mentioned to them before), give constructive feedback with specific examples (for things both good and bad) and always, always travel with a towel.

3.7.1 Project Managers Anonymous

I think the most fun I've had this past year is hanging out with my colleagues, learning and laughing together about everything that happens at work. Well, it was more than just learning and laughing - there was discussing concerns, risks, planning things, worrying about staffing, about the right balances between functional and non-functional work, etc. etc. But the wonderful thing was knowing that if there was ever an issue that I faced and didn't know just quite how to handle, I could rely upon these comrades of mine for help. And the same was true of me - I'm always glad to be able to help should anyone need me.

4 Culture

After having moved over to playing this role, I've had the opportunity to be privy to the inner workings of the exalted *middle-to-less-than-middle-management*. In this group, I've tried to understand attitudes, misconceptions, and thinking towards developers. For the record, I've always been a double-agent. Unfortunately I often let my developer side show through my carefully cultivated mature-and-knowing project manager facade. This may have lost me some points with this group, and I may have been excluded from their innermost machinations, but here's what I did learn about the culture of this rather odd species.

4.1 Them. Those *other* people...

Ask a typical project manager who is more important - a good developer or a good project manager. The answer is always interesting (unfair questions with no real answers are always fun; try asking someone what a question with no answer is called) - it says something about the person (of course, I'm not entirely sure what...)

Whatever other people believe, it is my firm belief that the team - comprising of developers, testers, business analysts, these are the people that do the real work of building working software that satisfies user needs. Project management is an over-head function, and the real job of project managers is to ensure that the team can operate at maximum productivity. Whatever

is needed to be done for this to be possible is fair game, and indeed, is the duty of the project manager to perform.

4.2 Trust issues

To this end, it is also the job of the project manager to ensure that team members are working on the right things and that nothing is blocking them from completing their work. Now, on more than one occasion, I've seen peoples' shoulder-blades misting up because their managers were breathing down their necks for every task they had to do. The manager was "ensuring" that they really were working on their tasks, and would ask for status about once every forty-two minutes.

In my experience, both as a developer and as a project manager, what has worked really well is trust. By this I mean that the manager trusting his team-members to do their best as responsible professionals and team-members trusting each other so that they can work together with an implicit understanding that each will do his or her job well. And, also, the team trusting their project manager to do whatever it is he does, so that they can complete their work as efficiently as possible.

This does take a certain amount of self-confidence; it ultimately leads the project manager to ask himself (or herself) if he really understands what is going on around him, if he understands software development, team-dynamics, leadership, general theory of non-annoying-behavior and a myriad other things. In general, I've also found that project managers who have never actually written any code in their past are more susceptible to micro-managing their teams. It is my belief that if someone wants to be a good manager, they owe it to themselves and to the people who work on their teams to learn a little bit about software - it would be nice if they were able to refer to what developers do as... "whatever it is they do... over there"... This lament however, probably ought to be a topic for another essay.

In any case, it would be nice if a developer told his manager that a story is going to take more time because the architecture team decided that the call ought to be asynchronous... and the manager actually understood! If not actually *nice*, it would certainly be a lot better for the manager than having to constantly ask the developer if they were there yet. And finally, it would be great if the project manager were to monitor progress at reasonable intervals while trusting the team and staying out of their way so they can actually get some work done.

4.3 Meetings. Whee.

One does not have to be a manager of a software development team to love meetings. Managers all over the world love the social ritual affectionately known as the *meeting*. They love it too much for anyone to be able to do anything about it. Enough said.

4.4 Meetings - no, really!

Seriously though, this topic is a rewind of an idea mentioned earlier - staying lean. Meetings are the bane of productivity - so much so that it is pretty much the only answer given to the question - "what is the difference between a real day and an ideal day?". The conversation usually goes something like this - Mr. New-to-agile asks the above-mentioned question, and the answer is "on real days you have other things interrupting you from work - you know, meetings and other things." I really don't know what these other things are...

It is the project manager's responsibility to limit the number of meetings his people have to attend. I found that my team was most grateful if I could get them out of having to attend another three hour session on how service oriented architecture was going to change their lives. In the past, my manager tried to go to meetings that he could attend if that meant we didn't have to go. I've tried hard to follow his method. It is also important to try to change the meeting culture to the extent possible, and increase the communication that occurs at the team level. A lot of meetings are convened because the issue at hand crosses team boundaries - if the team-members who're ultimately going to have to do the work involved talk instead, communication decay and overall turn-around time would be reduced. It's things like this that can achieve a leaner process. It does fall to the project manager to make it possible within the constraints of the organization.

Overall though, the reality is, a developer looking to become a project manager has to attend about nineteen meetings a day. And often, these meetings begin before a normal day starts - *yes, before 9 AM!* - and becoming a project manager sometimes entails going to bed earlier.

4.4.1 I'm listening, *really* listening

OK. This one took a little effort, because clearly, what did I, as an obviously superior developer, need to listen to all these other people for? They just talked too much and did too little. In the world of project management, however, these other people became the all important *customers*. Really.

One of the biggest lessons I learnt as started to deal with people I had mostly worked with only indirectly before, was, to shut up and listen for a bit. Many times, I even learned something valuable.

A lot of this has to do with being consultative. Clients have issues, which is why we as consultants have jobs. There have been times in the past when I was so blinded by myself and my past experiences and (what I considered to be) successes - that I felt that I knew how to solve the customers' problems without even listening. A cursory glance at their card wall (or horrors - Rose diagrams) was enough. The truth is - it's just not. In all cases, listening - without mental screens (alright, at least until absolutely essential) was absolutely required, and turned out to be quite helpful.

And while we're on the topic of listening, there have been *plenty* of instances when after feeling good about myself about having listened and really understood what the client was saying, I plain forgot parts of the discussion. And let things drop because of it. Bad project manager! No gantt chart for you! In this case though, the solution was easy. I began to *write things down!* This technologically advanced solution rivaled the bread-slicer. I recommend it.

The final point I wanted to make about being consultative is a realization that came to me fairly recently. When I say came to me, I really mean, drilled into me by my program manager. Repeatedly, until I began to believe. The point I'm referring to is this - *obvious things simply aren't*. There were absolutely *millions* of things that I thought ought to be painfully, stupidly obvious to everyone - that were, in fact not so obvious, and that stating them seemed to make clients think we were worth the big bucks they paid us. I mean the company I work for. Anyway, in the end, things began to work better for me (and I was able to restrain from chewing my elbow off in frustration) once I stopped making assumptions about what people understood and what they were thinking. Thinking aloud turned out to be a good strategy and helped in creating a common understanding and helped in getting things moving quickly. Well - thinking aloud turned out to be a good idea in *most* cases, anyway.

4.5 It will take *how long*?

This is probably another topic for another essay, but I did want to mention it. It is somewhat related to the topic about micromanagement and trust. It is critically important to continue to let developers do the estimates, and to never forget that they are, in the end, just estimates. Finally - try to convince your team that estimation ought to be done in relative complexity points for they are the only objective measure of stories, and the only ones

that are least likely to decay.

The other important point here is that for those project managers that aren't technical - it is critical to trust the team - and to also perhaps have one or two developers that they are comfortable working with, to be sort of their technical lieutenants. This way, whenever in doubt about anything technical, they can be sure they can rely on someone to get answers in enough detail to truly understand the issues.

5 Tools - sort of

I hate to be called a microsoftie, but unfortunately, I've had to use Excel, Powerpoint and Word on an almost daily basis in this role. There's nothing to be said about it, there it is. The advice I've received, and have yet to act upon, is this - get good with Excel.

6 Reading up

There are plenty of good books that I read after researching the aisles of amazon.com, and on the advice of other more experienced project managers. Here are some of the best - *Waltzing with Bears* by Tom Demarco for an excellent overview of risks and risks management, *Radical Project Management* by Rob Thomsett for general project management reading, *Project Retrospectives* by Norman Kerth for authoritative commentary on all things related to retrospectives, *Peopeware* by Tom Demarco for general project management, *Critical Chain* by Eli Goldratt on system throughput, bottlenecks and buffers, *The Fifth Discipline* by Peter Sengue on systems thinking, *Lean Thinking* by James Womack and *Lean Software Development* by Mary Poppendieck on lean processes and applying lean ideas to software. There are many, many more books I can list here, but there are plenty of resources on the internet that contain more exhaustive lists.

These are certainly all books I enjoyed reading.

7 Mentors

Good project management is not rocket-science. Good developers who have an inclination towards playing this role can easily become good project managers. Yes, this is probably is a somewhat biased statement - but hey, I'm writing the article. Jokes apart, however strong my management skills may or may not be, I benefitted a lot from my mentor. Books help but nothing

can replace being able to corner your mentor in the hallway and asking those questions, as often as you like.

When I started on this project, it was an entirely new role for me. The very first week, I spoke to the program manager (I'd heard he was very, very good at his job) and asked him if he would guide and help me through this transition to becoming a successful manager. He agreed, and we've had weekly sit-down sessions, where we discuss project management issues - both related to the project at hand and those that were not. These sessions were very useful to me, as I it gave me a place to vent, to question, to discuss, to bounce ideas off someone who's done a lot of this stuff before, and to just generally learn about managing software projects and about leadership in general.

8 Finally

I wanted to play the role of a project manager to see if I could do it, and to learn what I could from being on the "other side". I also had some ideas in my mind that I wanted to try - since I had thought that they were better than how I saw things being done.

The past year (and the present) has shown me things from a very different point of view. It made me think along lines I hadn't really thought along before and made me think about certain issues I hadn't thought about before. Some of my ideas turned out to be useful, some were plain wrong. It certainly made me appreciate where my project managers had been coming from in the past, and the kinds of issues they were dealing with and were trying to balance. All the non-work makes sense now, at least a little.

Overall though, without a doubt, I believe that I'm now in a place where I can be a much better team-member than where I was a year ago. This would be true even if I went back to being a developer (I probably will switch between these two roles as often as I can). I heartily recommend all developers give this role a shot, if they're at all so inclined. After all, there's only one way to find out...

8.1 P.S.

For those who want to make fun of me by calling me post-technical (I'd like to get my hands on the clown who coined that particular phrase), I have this to say - because this was an article remotely about project management (aka, a non-technical topic) - I decided to learn latex, wrote a Ruby script that

would convert my simplified mark-up to latex and an emacs minor-mode to help with my simple syntax. So there.

8.2 One last thing

All characters appearing in this work are fictitious. Any resemblance to real persons, living or dead, is purely coincidental. Except for Christine.