Engineers Shouldn’t Write ETL: A Guide to Building a High Functioning Data Science Department

JEFF MAGNUSSON

March 16, 2016 - San Francisco, CA

“What is the relationship like between your team and the data scientists?” This is, without a doubt, the question I’m most frequently asked when conducting interviews for data platform engineers. It’s a fine question – one that, given the state of engineering jobs in the data space, is essential to ask as part of doing due diligence in evaluating new opportunities. I’m always happy to answer. But I wish I didn’t have to, because this a question that is motivated by skepticism and fear.

Why is that? If you read the recruiting propaganda of data science and algorithm development departments in the valley, you might be convinced that the relationship between data scientists and engineers is highly collaborative, organic, and creative. Just like peas and carrots. However, it’s not a well kept secret that this is seldom the case. Most shops foster a relationship between engineers and scientists that lies somewhere in the spectrum between non-existent[1] and highly dysfunctional.

A Typical Data Science Department

Most companies structure their data science departments into 3 groups:

- **Data scientists:** the folks who are “better engineers than statisticians and better statisticians than engineers”. Aka, “the thinkers”.
- **Data engineers:** these are the folks who build pipelines that feed data scientists with data and take the ideas from the data scientists and implement them. Aka, “the doers”.
- **Infrastructure engineers:** these are the folks who maintain the Hadoop cluster / big data infrastructure. Aka, “the plumbers”.

Data scientists are often frustrated that engineers are slow to put their ideas into production and that work cycles, road maps, and motivations are not aligned. By the time version 1 of their ideas are put into an A/B test, they already have versions 2 and 3 queued up. Their frustration is completely justified.

Data engineers are often frustrated that data scientists produce inefficient and poorly written code, have little consideration for the maintenance cost of productionizing ideas, demand unrealistic features that skew implementation effort for little gain… The list goes on, but you get the point.

Infrastructure engineers get frustrated with everyone for overloading the clusters and filling up disk space. They are kept at arm's length from the scientists and engineers, which means they never gain a solid context into how the infrastructure is being used, or the business and technical problems that it needs to be used to solve. This makes them feel powerless to improve the situation. Instead, they react by making the infrastructure more restrictive. In turn, everyone becomes frustrated with them.
It's a vicious cycle.

What Went Wrong?

We all know that the standard is substandard, and that the recruiting hype is just that. So, why don't we fix it? Why does every data science and algorithms development team seem to slide into the same dysfunctional model?
I blame two things, offered here in the form of a couple observations:

You Probably Don't Have Big Data

Data processing tools and technologies have evolved massively over the last five years. Unless you need to process over many petabytes of data, or you're ingesting hundreds of billions of events a day, most technologies have evolved to a point where they can trivially scale to your needs.

Unless you need to push the boundaries of what these technologies are capable of, you probably don't need a highly specialized team of dedicated engineers to build solutions on top of them. If you manage to hire them, they will be bored. If they are bored, they will leave you for Google, Facebook, LinkedIn, Twitter, … – places where their expertise is actually needed. If they are not bored, chances are they are pretty mediocre. Mediocre engineers really excel at building enormously over complicated, awful-to-work-with messes they call "solutions". Messes tend to necessitate specialization.

Everybody Wants to be the "Thinker"

Because it sounds like such a cool role! You get to sit around all day, think up better ways to do things, and then hand off your ideas to people who eagerly rush to put them into production. Go ahead, pitch it to somebody on the street. I bet they jump at the opportunity! Data scientists, especially those who are newer to the industry and don't know any better, are especially vocal about desiring such a role.

That’s because we have trained them to desire it. We have bigger, more established companies to thank for that. Companies that had business intelligence departments before the Big Data craze…

A traditional business intelligence department consists of three roles: ETL engineers, report developers, and DBAs. ETL engineers move the data into the data warehouse. They are obsessed with Kimball and his guide to dimensional modeling. Report Developers, on the other hand, are folks who have made a career around designing reports in a specific tool (e.g. Microstrategy, et al). They are specialists. DBAs (and a team of other tool administrators) do their best to just keep things running.

Here’s the thing. ETL engineers, Report Developers, and DBAs are all “Doers”. So, 10 years ago or so, when Big Data and data science started to become buzzwords, there were well-established BI departments who had plenty of Doers and not enough Thinkers. So, they made “Thinker” a role. We integrated data scientists with established BI departments by promising them the ability to fiddle around with data and change the course of the business. In reality, this didn’t happen. These data scientists occasionally manage to create some pretty cool and effective solutions, but by and large they focus on performing slightly higher level Report Developer-ing back to the business (which largely ignores their advice).

But the role sounds really nice, and it’s easy to recruit for. Thus was born the traditional, modern day data science department: data scientists (Report developers aka “thinkers”), data engineers (ETL engineers aka “doers”), and infrastructure engineers (DBAs aka “plumbers”).
Whoops. It would seem that the business intelligence department never really changed, we just added a Hadoop cluster and started calling it by a new name.

Is it Really that Bad?

In truth, it depends what you’re looking to achieve. If you buy in to the argument above, then you have to accept that it has allowed companies to get by for many, many years… since the advent of BI. But I believe it’s a terribly inefficient model if you want your data science team to truly do more than build PowerPoint decks and dashboards.

The fundamental flaw that prevents the Thinker and Doer model from living up to its recruiting hype is the assumption that there exists an army of soulless non-mediocre Doer engineers who eagerly implement the ideas and vision of data scientists. Does that sound like the profile of any talented engineers that you know?

In this model the Doers are solely accountable for implementation, failure, and support of other people’s ideas, while the Thinker is rewarded for their success. This is at the heart of the contention and misalignment between the teams. It creates an IT group rather than an engineering team.

In order to attract talented engineers into a role like that, you need some really big scaling problems to serve as a distraction to the soulless, subservient role you have hired them into. You need the type of problems created by the existence of Big Data. And, I'm sorry, but you don't have Big Data.

Instead, you will hire mediocre engineers. They will create tremendously over complicated messes. This will exacerbate the contention. Welcome to the Vicious Cycle. The end result is a team of data scientists who are empowered to be little more than report developers because they lack the support of a solid, innovative data platform. And if your recruiting hype had pitched them on the Report Developer role, they would have run the other way. After all, they're Thinkers, not Doers!

A Different Kind of Data Science Department

Rather than try to emulate the structure of well-known companies (who made the transition from BI to DS), we need to innovate and evolve the model! No more trying to design faster horses…
A couple years ago, I moved to Stitch Fix for just that very reason. At Stitch Fix, we strive to be Best in the World at the algorithms and analytics we produce. We strive to lead the business with our output rather than to inform it. Unless you're willing to boldly challenge and rethink the things you know to be substandard, that is a damn hard proposition to fulfill.
After seeing the department grow and develop over the last two years, I am confident to share what we are up to.

Given that the goal is to lead rather than to inform, I would like to propose to you what I believe is A Better Way to structure a data science department. A way that allows for autonomy in roles, true ownership all the way into production, and accountability for output. A way that is well suited for a company with a quickly evolving business (and data) model.

What follows is a blueprint for building a data science team that can pivot and react quickly, so as to lead and innovate through the production of thought-leadership, APIs, and code, rather than react to changes and throw together some PowerPoint presentations in a desperate attempt to redirect gut feelings and intuitions.

Enable Everyone to be Best in the World

Let's forget the traditional roles, and instead think about the intrinsic motivations that get folks excited to come to work in the morning.

Regardless of role, a fundamental differentiator between adequate and great people lies in their desire and talent for being creative. Great people are able to identify and creatively solve problems that would absolutely baffle the mediocre. They excel in and crave for an environment of autonomy, ownership, and focus.

The assembly line handoff from scientist to engineer creates the polar opposite environment. (Truth is, even the Thinker resents having to rely on the Doer). The trick is to create an environment that allows for autonomy, ownership, and focus for everyone involved.

However, it is important to recognize that engineers and data scientists are impassioned by very different tasks:

Data Scientists: Data scientists love working on problems that are vertically aligned with the business and make a big impact on the success of projects/organization through their efforts. They set out to optimize a certain thing or process or create something from scratch. These are point-oriented problems and their solutions tend to be as well. They usually involve a heavy mix of business logic, reimagining of how things are done, and a healthy dose of creativity. Thus, they require a deep understanding of how specific portions of the business operate and a high degree of partnership with business verticals.

Engineers: Engineers excel in a world of abstraction, generalization, and finding efficient solutions in the places where they are needed. These problems are usually horizontally oriented in nature. They can be most impactful when applied broadly. They require a good overall understanding of how the business operates, but the abstracted nature of solutions mean they are light on business logic and do not require a heavy partnership with or deep understanding of verticals within the business.

Hybrid Thinker-Doers

A common fear of engineers in the data space is that, regardless of the job description or recruiting hype you produce, you are secretly searching for an ETL engineer.
In case you did not realize it, Nobody enjoys writing and maintaining data pipelines or ETL. It’s the industry’s ultimate hot potato. It really shouldn’t come as a surprise then that ETL engineering roles are the archetypal breeding ground of mediocrity.

Engineers should not write ETL. For the love of everything sacred and holy in the profession, this should not be a dedicated or specialized role. There is nothing more soul sucking than writing, maintaining, modifying, and supporting ETL to produce data that you yourself never get to use or consume.

Instead, give people end-to-end ownership of the work they produce (autonomy). In the case of data scientists, that means ownership of the ETL. It also means ownership of the analysis of the data and the outcome of the data science. The best-case outcome of many efforts of data scientists is an artifact meant for a machine consumer, not a human one. Rather than a report, dashboard, or PowerPoint presentation, it is some sort of algorithm or API that is integrated into the engineering stack – something that fundamentally changes the operation of the business. Autonomy means the data scientists own that code as well. All the way into production. They

should be able to develop and deploy it without asking the permission of engineers, be accountable for support, be held to performance, latency, and SLA requirements, etc.
This puts vertical responsibility and focus squarely into the hands of data scientists. But, data scientists are not typically classically trained or highly skilled software engineers. You could say they are adequate, at best. So you would expect that they would create a Big Mess.
This is one reason why ETL and API / production algorithm development is typically handed off to an engineer in assembly line style. But, all of those tasks are inherently vertically (point) focused. Talented engineers in the data space are almost always best focused on horizontal applications.

So, then, what is the role of an engineer in this new, horizontal world? To sum it up, engineers must deploy platforms, services, abstractions, and frameworks that allow the data scientists to conceive of, develop, and deploy their ideas with autonomy (such as a tool, framework, or service used to build, schedule, and execute ETL). I like to think of it in terms of Lego blocks. Engineers design new Lego blocks that data scientists assemble in creative ways to create new data science. This is definitely easier said than done, but:

- Engineer’s work can be completely horizontal in nature. This allows them to focus on building technology that is broadly applicable across multiple data science problems. This maximizes leverage of engineering output. Which is great, since you probably have far more data scientists than you do engineers in your data science department.
- Engineers get to focus on what they do best: abstracting, generalizing, and creating efficient, scalable solutions where they are needed.
- Engineers get to operate with autonomy. The production from an engineering team deployed in this manner should look like “magic”. Things should just “fall into place” for the data scientists because their needs are anticipated and the scaling and resiliency is taken care of the platform, services, and frameworks they are using.
- In order for this to work well, most of the time the engineers need to anticipate the needs of the data scientists. They should be developing multiple steps ahead.
- For highly talented and creative engineers and data scientists, it’s a hell of a lot more fun.

So, the Scientists Do All The Work?

No, not at all. If anything, engineers have a much more challenging and demanding role than they do in the standard model. The data scientists probably do as well. We are not optimizing the organization for efficiency, we are optimizing for autonomy. What is offered is clear ownership of ideas and accountability for their delivery.
These are roles that are very attractive to folks who embrace an entrepreneurial mindset. It allows for quick movement, eliminates the need for building unnecessary consensus, and opens the door to disruptive innovation. But it does come at the cost of specialization, and thus efficiency.

The expectation, however, is not that data scientists are going to suddenly become talented engineers. Nor is it that the engineers will be ignorant of all business logic and vertical initiatives. In fact, partnership is inherent to the success of this model. Engineers should see themselves as being “Tony Stark’s tailor”, building the armor that prevents data scientists from falling into pitfalls that yield unscalable or unreliable solutions.

In the absence of abstractions and frameworks for rolling out solutions, engineers partner with scientists to create solutions. But, not in the form of a hand off. Rather, the engineering challenge becomes one of building self-service components such that the data scientists can iterate autonomously on the business logic and algorithms that deliver their ideas to the business. After the initial roll out of a solution, it is clear who owns what. The engineers own the infrastructure that they build, and the data scientists own the business logic and algorithm implementations that they provide. No form of tight coupling is required to iterate.

A Challenging Road

At this point, you may be skeptical that it’s possible to pull something like this off. However, I think the payoff is well worth the risk. Here are a couple of things to watch out for that can hamper or revert progress:

People are averse to change. People tend to want to recreate environments that they are used to working in. This creates pressure to revert to the Thinker-Doer model. New hires need to get on board with the new structure quickly. Vigilance is especially warranted when a project encounters problems – e.g. an API breaks or an algorithm serves bad results.
People behave in a very reactive way in those circumstances. They will insist that engineers should take over. But, they are addressing a symptom rather than the problem. Engineers should instead build in better platform support, visibility, abstractions, and resilience. And, they should realize that engineers break things too… no one is immune from making a mistake and breaking production.

It is absolutely essential for platform engineers to stay ahead of the data science teams. You need very sharp platform engineers who can make intuitive decisions about what services, frameworks, and capabilities need to be in place before they are desperately needed. The lack of hand off from scientist to engineer means that the engineers do not get the luxury of reacting to requirements delivered by the scientists.

Remember, the engineers are creating lego blocks, and the data science teams are assembling them. If the data science teams don’t have to right blocks to assemble, they will forge ahead nonetheless to create a solution. They’ll solve their problem either by assembling the wrong blocks (square peg in a round hole), or by creating their own. Usually, they will create a Big Mess. One that is hard to undo once it has been created.

Don’t Fear Inefficiency

A consequence of empowering data scientists to take on such a breadth of the stack is that they will be unlikely to produce code and solutions that are as technically efficient as an engineer’s. We are sacrificing technical efficiency for velocity and autonomy. It is important to recognize this as a deliberate trade off.

There is, however, a set of less obvious efficiencies that are gained with end-to-end ownership. The data scientists are experts in the domain of the implementations they are producing. Thus, they are well equipped to make trade offs between technical and support costs vs. requirements. For example, they can decide to sample data in certain places, use approximate methods where they make sense, and make decisions to nix or punt features that may produce only very marginal business impacts but come with extremely high development or support costs. These things seldom happen (and when they do, usually require numerous negotiations) in the assembly line model of hand off between scientists and engineers.
In aggregate, it is hoped that the benefits of autonomy and the innovation that can be produced as a result will outweigh the technical inefficiencies of the lack of technical specialization in allowing data scientists to own their full stack.

The Future

I’ll make no claim that we have discovered the best way to structure a data science department, or that this is the best structure for your organization. But, it is definitely not an attempt to build a faster horse and I feel strongly it is a better solution for Stitch Fix.
It’s my sincere hope that in sharing what we have done that it will encourage others with a non-traditionally structured department to do the same, inspire leaders of data science departments that are in a formative stage to think outside the box and find the courage to challenge tradition, and inform engineers and data scientists who are frustrated by traditional roles that there are different types of environments available to operate in.