CHAPTER 1

Data Teams

Oh I get by with a little help from my friends

—"With a Little Help from My Friends" by The Beatles

Making use of big data is a team sport. It takes several different kinds of people to get things done, and in all but the smallest organizations, they should be organized into multiple teams. When the help from these friends comes together, you can do some awesome things. When you're missing your friends, you fail and underperform.

Just who are these friends, what should they be doing, and how do they do it? This book answers these questions. The book covers many facets of forming data teams: what kinds of skills to look for in staff, how to hire or promote staff, how the teams should interact with each other as well as with the larger organization, and how to recognize and head off problems.

Big Data and Data Products

To make sure this book is right for you—that my topics correspond to what your organization is working on—I'll take some time to explain the kinds of projects covered in these pages.

How could we start a book about big data management without a definition of big data? What I want to do here is go beyond buzzwords and get to a definition that really helps management.

J. Anderson, *Data Teams*, https://doi.org/10.1007/978-1-4842-6228-3_1

The Terrible 3s, 4s, 5s…

Everybody accepts that big data is a rather abstract concept: you can't just say you have big data because the sizes of your datasets hit certain metrics. You have to find qualitative differences between small and big data. That gets hard.

One of Gartner's original attempts to define big data led to the creation of the 3 Vs. Originally the Vs were variety, velocity, and volume. It's difficult for management to understand this definition. It was too broad. As a result, every company said their product was big data, and management still didn't understand the definition.

This led to people choosing their own definition. Pick a number between 3 and 20. That's the number of Vs that were defined.

Instead of providing clarity, these definitions really confused the issue. People were just looking through the dictionary for Vs that sound like they should fit. Managers were learning nothing that helped them manage modern data projects.

The Can't Definition

For management, I prefer the *can't* definition. When asked to do a task with data, the person or team says they can't do it, usually due to a technical limitation. For example, if you ask your analytics team for a report, and they say they can't do it, you probably have a big data problem.

It's imperative that the *can't* be due to a technical reason instead of the staff's skill. Listen to the reasons that the team says they can't do the task. These are some examples of technical reasons for a *can't*:

- The task is going to take too long to run.
- The task will bring down or slow down our production database.
- The task requires too many steps to complete.
- The data is scattered in too many places to run the task.

Obviously, your more technically trained people will offer a more precise and technical definition. I highly suggest you verify that your data teams really understand what is and isn't big data. If they don't, you could be relying on people who don't really understand the requirements.

Some organizations are smaller or are startups. What should they do, since they aren't saying "can't"—yet. The question should then be: will the organization have big data in the future? This future big data is really where many companies are focused. It's difficult to go back and reengineer lots of pipelines and code to use a different technology stack. Some organizations prefer to solve these problems from the very beginning instead of waiting.

Why Management Needs to Know the Definition of Big Data

It's crucial for management to understand what constitutes big data, and they should be guided by technically qualified people. This is because the mismatch of big and small data problems can really crush productivity and value creation.

Using small data technologies for big data problems leads to can'ts. Using big data technologies for small data problems is also a problem and not just because it leads to overengineering—it's creating major costs and problems.[1]

Just because many big data technologies are open source doesn't mean that they're cheap. Your costs will go up for infrastructure and salaries. Big data technologies tend to be pointy and filled with thorns, whereas small data technologies have fewer nuances that you'll have to fight. While small data lets you run all of your processes on a few computers, the number of computers explodes with big data. All of a sudden, your infrastructure costs are way higher. With big data, your response times, processing time, and end-to-end times could go up. Big data isn't necessarily faster; it's just faster and more efficient than using small data technologies for data that's too big.

Management needs to know when to push back on the hype for big data. Management shouldn't let engineers convince them to use something that isn't the right tool for the job. This could be resume polishing from the engineering side. But in the case of can'ts, big data could be the right approach.

Finally, if you're barely making it with small data technologies, big data technologies will be even more difficult. I've found that when an organization can barely exploit or productionize small data technologies, the significant jump in complexity leads to failure or underperforming projects.

[1]If you have small data, don't take this as belittling your use case, company, or project. Rather, you should be relieved that you've dodged the bullet known as big data (see www.jesse-anderson.com/2018/07/saying-you-have-small-data-isnt-belittling-your-use-case/).

Why Is Big Data So Complicated?

Big data is 10–15 times more complicated to use than small data.[2] This complexity extends from technical issues to management ones. Misunderstanding, underestimating, or ignoring this significant increase in complexity causes organizations to fail.

Technically, this complexity stems from the need for distributed systems. Instead of doing everything on a single computer, you have to write distributed code. The distributed systems themselves are often difficult to use and must be chosen carefully because each has specific trade-offs.

A *distributed system* is a task broken up and run on several computers at once. This could also mean data broken up and stored on multiple computers. Big data frameworks and technologies are examples of distributed systems. I'm using this term instead of talking about a specific big data framework or program. Honestly and unfortunately, these big data frameworks come and go.

Management becomes more complex, too, because the staff has to reach across the organization at a level and consistency you never had to before: different departments, groups, and business units. For example, analytics and business intelligence teams never had to have the sheer levels of interaction with IT or engineering. The IT organization never had to explain the data format to the operations team.

From both the technical and the management perspectives, teams didn't have to work together before with as high of a bandwidth connection. There may have been some level of coordination before, but not this high.

Other organizations face the complexity of data as a product instead of software or APIs as the product. They've never had to promote or evangelize the data available in the organization. With data pipelines, the data teams may not even know or control who has access to the data products.

Some teams are very siloed. With small data, they've been able to get by. There wasn't ever the need to reach out, coordinate, or cooperate. Trying to work with these maverick teams can be a challenge unto itself. This is really where management is more complicated.

[2]For a full explanation of this increase in complexity, I suggest you read www.oreilly.com/learning/on-complexity-in-big-data/.

Data Pipelines and Data Products

The teams we're talking about in this book deal with *data pipelines* and *data products*. Briefly put, a data pipeline is a way of making data available: bringing it into an organization, transferring it to another team, and so on—but usually transforming the data along the way to make it more useful. A data product takes in a dataset, organizes the data in a way that is consumable by others, and exposes in a form that's usable by others.

More specifically, a data pipeline is a process to take raw data and transform it in a way that is usable by the next recipient in the organization. To be successful, this data must be served up by technologies that are the right tools for the job and that are correct for the use cases. The data itself is available in formats that reflect the changing nature of data and of the enterprise demand for it.[3]

The output of these data pipelines are data products. These data products should become the lifeblood of the organization. If this doesn't happen, the organization isn't a data-driven organization and is not reaping the benefits of its investment in data.

To be robust, data products must be scalable and fault-tolerant, so that end users can reliably use them in production and critical scenarios. The data products should be well organized and cataloged to make them easy to find and work with. They should adhere to an agreed-upon structure that can evolve without massive rewrites of code, either in the pipelines creating the data products or among downstream consumers.

Data products usually are not one-off or ad hoc creations. They are automated and constantly being updated and used. Only in certain cases can you cut corners to create short-lived or ad hoc data products.

The veracity and quality of data products are vital. Otherwise, the teams using the data products will spend their time cleaning data instead of using the data products. Consistently low-quality data products will eventually erode all confidence in the data team's abilities.

[3]It's also worth noting other people's definition of data pipelines and related questions (see www.jesse-anderson.com/2018/08/what-is-a-data-pipeline/).

Common Misconceptions

Before we focus on how to make big data work for you, we need to dispel some myths that prevent managers from recognizing the special requirements of big data, hiring the right people to deal with big data, and managing the people effectively.

"It's All Just Data"

Sometimes people say it's all just data. They're trying to say there isn't a difference between small and big data. This sort of thinking is especially bad for management. It sends the message that what data teams do is easy and replaceable. The reality is that there is a big difference between small and big data. You need completely different engineering methods, algorithms, and technologies. A lack of appreciation for these differences is a cultural contributor to project failures.

There may be reasons that people think this. They may be in an organization with good data engineering. They're simply taking for granted the hard work of the data teams. From this person's point of view, it's all easy. This is one of the marks of a successful data engineering team.

Another reason may be that the organization doesn't have big data. They've been able to get by without having to deal with the complexity jump caused by big data.

"Isn't This Just Something Slightly Different from…?"

A common misconception with data teams is that they are only slightly different from a more traditional existing team in the organization. Managers think we're just going overboard with job titles and teams to make things more complicated. This sort of thinking contributes to failure because the wrong or unqualified team is taking the lead.

Business Intelligence

Some people believe that business intelligence is the same thing as data science. Yes, both teams make extensive usage of math and statistics. However, most business intelligence teams are not coding. If they are writing some code, they're not using a complex language. Put simply, a data scientist needs to know more than SQL to be really productive. They'll need to know a high-level and complex language to accomplish their goals.

Data Warehousing

Others think data engineering is the same thing as data warehousing. Yes, both teams make extensive usage of data. They'll both use SQL as a means of working with data. However, data engineering also requires intermediate-level to expert-level knowledge of programming and distributed systems. This extra knowledge really separates the two teams' underlying skills.

Although this book talks a bit about the continuing roles of database administrators (DBAs) and data warehouse teams, I'm not including their work as part of the data teams discussed in the book. The data products that they can create are just too limited for the types of data science and analysis covered in the book. Yes, some teams are able to create some products of value, but they aren't able to create a wide variety of data products that today's organizations need.

Operations

Another source of confusion is on the operational side. This also comes back to distributed systems. Keeping distributed systems running and functioning correctly is difficult. Instead of data and processing being located on a single computer, it is spread out over multiple computers. As a direct result, these frameworks and your own code will fail in complex ways.

The operational problems don't end with just software. Operations have to deal with data itself. Sometimes, the operational issues come from problems in the data: malformed data, data that isn't within a specific range, data that doesn't adhere to the types we're expecting—the list goes on. Your operations team will need to understand when a problem stems from data, the framework, or both.

Software Engineering

Finally, software engineering and data engineering look really similar from the outside. You might see a pattern here, but data and distributed systems make all the difference. Software engineering is no exception, and software engineers will need to specialize in order to become data engineers.

I've worked with software engineers extensively and throughout the world. Software engineering skills are close to data engineering skills but not close enough. For most of a software engineer's career, the database is their data structure and storage mechanism. Others may never have had to write code that is multiprocess, multithreaded, or

distributed in any way. The majority of software engineers haven't worked with the business in as deep or in as concerted way that is needed for data engineering.

The distributed systems that data engineers need to create are complex. They also change quite often. For most data pipelines, data engineers will have to use 10–30 technologies to create a solution, in contrast to the three or so technologies needed for small data. This really underscores the need for specialization.

Why Are Data Teams Needed for Big Data?

You might have checked out this book because you're part of a brand-new project, or one that's already underway, and have seen something amiss but can't put your finger on what's happening. I find this is often the case when someone is having problems with their data projects.

In this book, data teams are the creators and maintainers of data products. I have mentored, taught, and consulted with different data teams all over the world. After being brought on board, I have repeatedly encountered teams in various stages of failure and underperformance. The failure of data projects would prevent the business from benefiting or gaining any return on their big data investments. At some point, organizations would just stop investing in their big data projects, because they became a black hole that sucked up money while emitting little to nothing of value.

At risk of oversimplifying, I'll use the term *small data* for the small-scale work that organizations everywhere are doing with SQL, data warehouses, conventional business intelligence, and other traditional data projects. That term contrasts with *big data*, a buzzword we'll look at in the next chapter. The new big data projects seem familiar and yet also different and strange. There is an ethereal and hard to quantify the difference that you can't quite express or put your finger on.

This difference you're seeing is the missing piece that determines whether data projects are successful or not. Without this understanding that informs your creation of data teams and interaction with them, you can't be successful with big data. The key is getting the right balance of people and skills—and then getting them to collaborate effectively.

Why Some Teams Fail and Some Succeed

Years ago, I could only spot when a team was about to fail. I couldn't tell them what do to fix things, improve the team, or prevent failure. I felt quite powerless. I was seeing a train speeding toward a concrete wall, and I could only tell the passengers to get off before it crashed.

I knew that wasn't enough.

Figuring out the why, what, and who of these failures became somewhere between a challenge, obsession, and quixotic adventure for me. It became my research project. I absolutely had to figure out why very few teams succeeded and why so many teams that showed promise at the start ended up failing.

I became a success junkie, crazy-focused on people and organizations who were or said they were successful. When I found these people, they faced a barrage of specific and demanding questions: Why were you successful? How did you become successful? What did you do differently to become successful? Why do you think you were successful? Even as I'm writing this, I'm remembering people's faces as I asked these weird questions. The only way I could decipher this puzzle was by asking questions and assimilating other people's experiences.

I learned a lot from the projects that failed too. Encountering a project teetering on the edge of failure, I looked at the usual suspects but couldn't find anything obvious. Did people work hard? Yes. Were they smart? Yes. Was the technology to blame? No—at least not usually. I had to look deeper.

It was easy to blame the technologies, and that's what most organizations did after a failure. But I knew that was a total cop-out. Yes, there are problems and limitations with the technologies. Other organizations were able to productionize the same technologies, deal with the issues that the technologies had, and be successful. We're mostly talking about projects that failed well before the project ever made it into production or the first release.

No, these projects were failing so early in the cycle that there was a different culprit. Most of the time, the failures were the same over and over again. And yet, you couldn't really blame the staff or the management. There wasn't any body of work out to say why or what was happening.

Personally, I've been on this mission to create the body of work for management. I wanted to dispel the ignorance that led to all that waste. This book is a compendium of that effort. You'll see some of my related writings on the topic pop up as footnotes throughout the book.

The Three Teams

To do big data right, you need three different teams. Each team does something very specific in creating value from data. From a management's 30,000-foot view—and this is where management creates the problem—they all look like the same thing. They all transform data, they all program, and so on. But what can look from the outside like a 90 percent overlap is really only about 10 percent. This misunderstanding is what really kills teams and projects.

Each team has specialized knowledge that complements the other teams. Each team has both strengths and weaknesses inherent to the staff's experiences and skills. Without the other teams, things just go south.

We're going to go deeply into each one of these teams in Part 2, but I want to briefly introduce each one here. Like a dating show, let's bring on our three bachelors!

Data Science

Bachelor #1 likes counting things, math, and data. He's learned a little about programming. Meet the data science team!

When most managers think or hear of big data, it's in the context of data science. The reality is that data science is just one piece of the puzzle.

The data science team consumes data pipelines in order to create derivative data. In other words, they take data pipelines that were previously created and augment them in various ways.

Sometimes the augmentation consists of advanced analytics, notably the machine learning (ML) that is so hot nowadays. The members of a data science team usually have extensive backgrounds in mathematical disciplines like statistics. With enough of a background in statistics, math, and data, you can do some pretty interesting things. The result is usually a *model* that has been trained on your specific data and analyze it for your business use case. From models, you can get fantastically valuable information such as predictions or anomaly detection.

My one-sentence definition of a data scientist is:

> A data scientist is someone who has augmented their math and statistics background with programming to analyze data and create applied mathematical models.

At this initial juncture, there are few main things to know about data scientists:

- They have a math background, but not necessarily a math degree.
- They have an understanding of the importance and usage of data.
- They usually have a beginner-level understanding of big data tools.
- They usually have beginner-level programming skills.

This beginner-level skill is important to understand. We'll get deeper into this later—just know that this is a big reason we need the other teams.

A common mistake made by organizations just starting out with big data is to hire just data scientists. This is because they are the most visible element of big data and the face of the analytics created. This mistake is like trying to get a band together with just a lead singer. That might work if you're going to sing *a cappella*. If you're going to want any musical accompaniment, you'll need the rest of the band. A big focus of this book is to help you see and understand why each team is essential and how each team complements another.

Data Engineering

Bachelor #2 likes building model airplanes, programming, data, and distributed systems. Meet the data engineering team!

Going from data science in the lab to running data science at scale in the business isn't a trivial task. You need people who can create maintainable and sustainable data systems that can be used by nonspecialists. It takes a person with an engineering background to do this right.

The data engineering team creates the data pipeline that feeds data to the rest of the organization, including the data scientists. The data engineers need the skills to create data products that are

- Clean
- Valid
- Maintainable
- Usable at scale
- Sustainable for additions and improvements

Sometimes, the data engineering team rewrites the data scientists' code. This is because the data scientists are focused on their research and lack the time or expertise to use the programming language most effectively.

The data engineers are also responsible for choosing the data infrastructure to run on. This infrastructure can vary from project to project and usually consists of open source projects with weird names, or related tools provided by cloud vendors. This is a place where data teams get stuck, choose the wrong thing, or implement things wrong and get waylaid.

My one-sentence definition of a data engineer is:

> A data engineer is someone who has specialized their skills in creating software solutions around big data.

This team is crucial to your success. Make sure you have the right people with the right resources to guide you effectively. At this initial juncture, there are a few main things to know about data engineers:

- They come from a software engineering background.
- They have specialized in big data.
- Their programming skills are intermediate at a bare minimum, and ideally expert.
- They may be called upon to enforce some engineering discipline on the data scientists.

These data engineers are—at their heart—software engineers, with all the good and bad that comes along with it. They, too, need others to complement their shortcomings. So, in addition to interacting heavily with other teams, the data engineering team itself is multidisciplinary. Although it will be made up primarily of data engineers, there may be other job titles as well. These extra staff will fill out a role or skill that a data engineer doesn't have, or a task of lower difficulty that can be done in conjunction with a data engineer.

Operations

Bachelor #3 likes trains that run on time, hardware, software, and operating systems. Meet the operations team!

Running distributed system frameworks in production ranges from rock-solid to temperamental. Your code will likely have the same range of behavior. Who is responsible for keeping these technologies running and working? You need an operations team to keep the ship moving and everything chugging along.

Organizations accomplish their operational goals in two different ways.

The first is a more traditional operations route. There is a team that is responsible for keeping everything running. That team does not really touch the data engineer's code. They may have a hand in automation, but not in writing the code for pipelines.

The second is more of a practice than a team. This practice mixes data engineering and operational functions. The same team is responsible for both the data pipeline code and keeping it running. This method is chosen to prevent the quintessential "throw it over the fence" problems that have long existed between developers and operations staff where developers create code of questionable quality that the operations team is forced to deal with the problems. When the developers are responsible for maintaining their own code in production, they will need to make it rock-solid instead of leaving quality as someone else's problem.

Whether a separate team or a function of engineering, operations are responsible for keeping things running. This list of "things" is pretty and long, underscoring the necessity of operations. These things include

- Being responsible for the operation in production of the custom software written by your data engineers and data scientists (and maybe other people too)
- Keeping the network optimized, because you're dealing with large amounts of data and the vast majority of it is passed through the network
- Fixing any hardware issues, because hard drives and other physical hardware will break (less common in the cloud, but still occasionally requiring troubleshooting knowledge)
- Installing and fixing the peripheral software that may be needed by your custom code
- Installing and configuring the operating systems to optimize their performance

That list might sound like any operations, but let me add the things that really kick the big data operational team into overdrive. They must be:

- Responsible for the smooth running of the cluster software and other big data technologies you have operationalized
- More familiar with the code being run than usual, and understand its output logs
- Familiar with the expected amount, type, and format of the incoming data

My one-sentence definition of an operations engineer is:

> An operations engineer is someone with an operational or systems engineering background who has specialized their skills in big data operations, understands data, and has learned some programming.

At this initial juncture, there are few main things to know about operations engineers:

- They come from a systems engineering or operational background.
- They have specialized in big data.
- They have to understand the data that is being sent around or accessed by the various systems.

It's important to know there's really a different mindset between data engineers and operations engineers. I've seen it over and over. It really takes a different person to want to maintain and keep something running rather than creating or checking out the latest big data framework.

Why Are Three Teams Needed?

We've just seen the gist of all three teams. But you may still have questions about what each team does or how it differs. We're going to get deeper into the differences and how the teams support each other.

For some people or organizations, it's difficult to quantify the differences because there is some overlap. It's important to know that this overlap is complementary and not the source of turf wars. Each one of these teams plays a vital role in big data.

Sometimes managers think it's easier to find all three teams' skills shoved into one person. This is really difficult, and not just because there are so many specialized skills. I've found that each team represents a different personality and mindset. The mindset of science is different from engineering, which is different from operations. It's easier and less time-consuming to find several people and get them to work together.

Three Teams for Small Organizations

Small teams and small organizations represent a unique challenge. They don't have the money for a 20-person team. Instead, they have one to five people. What should these really small teams and organizations do?

What happens, of course, is that the organization asks certain staff to take on multiple functions. It's a difficult path. You'll have to find the people who are both competent to take on the functions and interested in doing so. These people are few and far between. Also, know that these people may not fulfill that role 100 percent. They can fill in for a function in a pinch, but they're not the long-term solution. You'll need to fill in those holes as the size of your organization grows.

What Happens If an Organization Doesn't Manage Correctly?

To round out this first chapter's overview, I want to share the effects of skipping some teams or placing the wrong people on them. Put simply, it condemns organizations to fail or get stuck at their big data projects. The specifics depend on complexity, use case, and what's missing. In any case, it leads the organization down a sad path.

I often see the second- or third-order effects of a missing team. I can see what is missing in the team by the explanation of the problem. I recognize what's missing because I've seen other organizations with the same problem. The root cause of the issue almost always falls into one of just a few categories. Sometimes, there are several layers to the problem, and you'll need to peel them back until you get to the roots of the problem. Only then can you really achieve the maximum potential from your data teams.

This book will help you get your organization and teams back on track or fix what's missing.