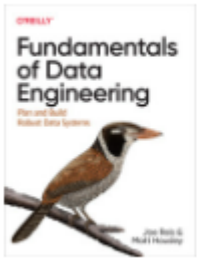

Fundamentals of Data Engineering

Joe Reis, Matt Housley

Published by O'Reilly Media, Inc.

Chapter 1. Data Engineering Described

If you work in data or software, you may have noticed data engineering emerging from the shadows and now sharing the stage with data science. Data engineering is one of the hottest fields in data and technology, and for a good reason. It builds the foundation for data science and analytics in production. This chapter explores what data engineering is, how the field was born and its evolution, the skills of data engineers, and with whom they work.

What Is Data Engineering?

Despite the current popularity of data engineering, there's a lot of confusion about what data engineering means and what data engineers do. Data engineering has existed in some form since companies started doing things with data—such as predictive analysis, descriptive analytics, and reports—and came into sharp focus alongside the rise of data science in the 2010s. For the purpose of this book, it's critical to define what *data engineering* and *data engineer* mean.

First, let's look at the landscape of how data engineering is described and develop some terminology we can use throughout this book. Endless definitions of *data engineering* exist. In early 2022, a Google exact-match search for "what is data engineering?" returns over 91,000 unique results. Before we give our definition, here are a few examples of how some experts in the field define data engineering:

Data engineering is a set of operations aimed at creating interfaces and mechanisms for the flow and access of information. It takes dedicated specialists—data engineers—to maintain data so that it remains available and usable by others. In short, data engineers set up and operate the organization's data infrastructure, preparing it for further analysis by data analysts and scientists.

From "Data Engineering and Its Main Concepts" by AlexSoft[1]

The first type of data engineering is SQL-focused. The work and primary storage of the data is in relational databases. All of the data processing is done with SQL or a SQL-based language. Sometimes, this data processing is done with an ETL tool.[2] The second type of data engineering is Big Data–focused. The work and primary storage of the data is in Big Data technologies like Hadoop, Cassandra, and HBase. All of the data processing is done in Big Data frameworks like MapReduce, Spark, and Flink. While SQL is used, the primary processing is done with programming languages like Java, Scala, and Python.

Jesse Anderson[3]

In relation to previously existing roles, the data engineering field could be thought of as a superset of business intelligence and data warehousing that brings more elements from software engineering. This discipline also integrates specialization around the operation of so-called "big data" distributed systems, along with concepts around the extended Hadoop ecosystem, stream processing, and in computation at scale.

Maxime Beauchemin[4]

Data engineering is all about the movement, manipulation, and management of data.

Lewis Gavin[5]

Wow! It's entirely understandable if you've been confused about data engineering. That's only a handful of definitions, and they contain an enormous range of opinions about the meaning of *data engineering*.

Data Engineering Defined

When we unpack the common threads of how various people define data engineering, an obvious pattern emerges: a data engineer gets data, stores it, and prepares it for consumption by data scientists, analysts, and others. We define *data engineering* and *data engineer* as follows:

Data engineering is the development, implementation, and maintenance of systems and processes that take in raw data and produce high-quality, consistent information that supports downstream use cases, such as analysis and machine learning. Data engineering is the intersection of security, data management, DataOps, data architecture, orchestration, and software engineering. A data engineer manages the data engineering lifecycle, beginning with getting data from source systems and ending with serving data for use cases, such as analysis or machine learning.

The Data Engineering Lifecycle

It is all too easy to fixate on technology and miss the bigger picture myopically. This book centers around a big idea called the *data engineering lifecycle* (Figure 1-1), which we believe gives data engineers the holistic context to view their role.

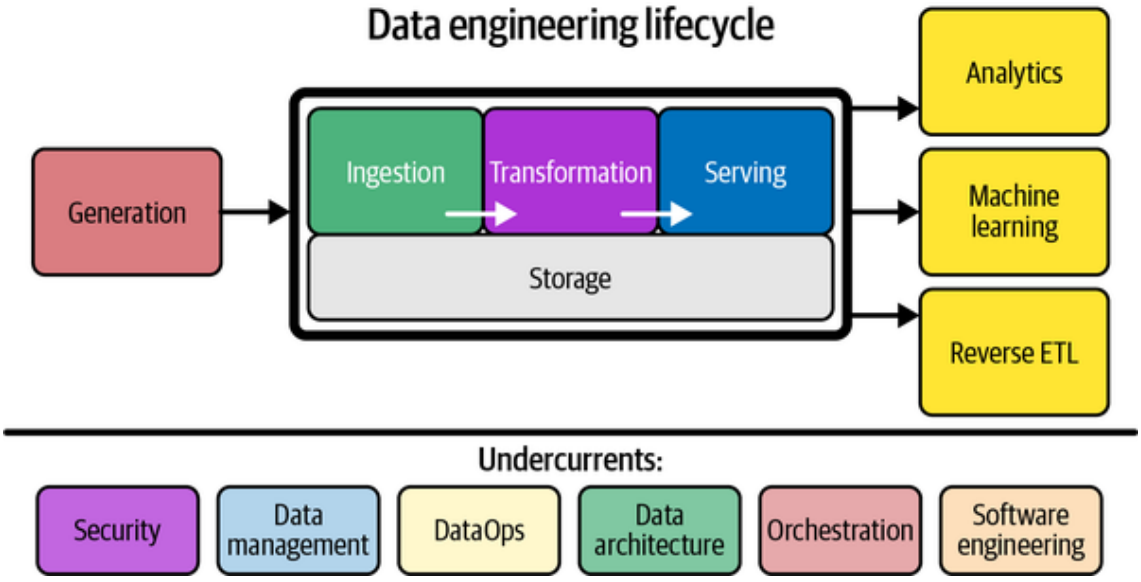

Figure 1-1. The data engineering lifecycle

The data engineering lifecycle shifts the conversation away from technology and toward the data itself and the end goals that it must serve. The stages of the data engineering lifecycle are as follows:

- Generation
- Storage
- Ingestion
- Transformation
- Serving

The data engineering lifecycle also has a notion of *undercurrents*—critical ideas across the entire lifecycle. These include security, data management, DataOps, data architecture, orchestration, and software engineering. We cover the data engineering lifecycle and its undercurrents more extensively in Chapter 2. Still, we introduce it here because it is essential to our definition of data engineering and the discussion that follows in this chapter.

Now that you have a working definition of data engineering and an introduction to its lifecycle, let's take a step back and look at a bit of history.

Evolution of the Data Engineer

History doesn't repeat itself, but it rhymes.

A famous adage often attributed to Mark Twain

Understanding data engineering today and tomorrow requires a context of how the field evolved. This section is not a history lesson, but looking at the past is invaluable in understanding where we are today and where things are going. A common theme constantly reappears: what's old is new again.

The early days: 1980 to 2000, from data warehousing to the web

The birth of the data engineer arguably has its roots in data warehousing, dating as far back as the 1970s, with the *business data warehouse* taking shape in the 1980s and Bill Inmon officially coining the term *data warehouse* in 1989. After engineers at IBM developed the relational database and Structured Query Language (SQL), Oracle popularized the technology. As nascent data systems grew, businesses needed dedicated tools and data pipelines for reporting and business intelligence (BI). To help people correctly model their business logic in the data warehouse, Ralph Kimball and Inmon developed their respective eponymous data-modeling techniques and approaches, which are still widely used today.

Data warehousing ushered in the first age of scalable analytics, with new massively parallel processing (MPP) databases that use multiple processors to crunch large amounts of data coming on the market and supporting unprecedented volumes of data. Roles such as BI engineer, ETL developer, and data warehouse engineer addressed the various needs of the data warehouse. Data warehouse and BI engineering were a precursor to today's data engineering and still play a central role in the discipline.

The internet went mainstream around the mid-1990s, creating a whole new generation of web-first companies such as AOL, Yahoo, and Amazon. The dot-com boom spawned a ton of activity in web applications and the backend systems to support them—servers, databases, and storage. Much of the infrastructure was expensive, monolithic, and heavily licensed. The vendors selling these backend systems likely didn't foresee the sheer scale of the data that web applications would produce.

The early 2000s: The birth of contemporary data engineering

Fast-forward to the early 2000s, when the dot-com boom of the late '90s went bust, leaving behind a tiny cluster of survivors. Some of these companies, such as Yahoo, Google, and Amazon, would grow into powerhouse tech companies. Initially, these companies continued to rely on the traditional monolithic, relational databases and data warehouses of the 1990s, pushing these systems to the limit. As these systems buckled, updated approaches were needed to handle data growth. The new generation of the systems must be cost-effective, scalable, available, and reliable.

Coinciding with the explosion of data, commodity hardware—such as servers, RAM, disks, and flash drives—also became cheap and ubiquitous. Several innovations allowed distributed computation and storage on massive computing clusters at a vast scale. These innovations started decentralizing and breaking apart traditionally monolithic services. The "big data" era had begun.

The *Oxford English Dictionary* defines big data as "extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions." Another famous and succinct description of big data is the three *V*s of data: velocity, variety, and volume.

In 2003, Google published a paper on the Google File System, and shortly after that, in 2004, a paper on MapReduce, an ultra-scalable data-processing paradigm. In truth, big data has earlier antecedents in MPP data warehouses and data management for experimental physics projects, but Google's publications constituted a "big bang" for data technologies and the cultural roots of data engineering as we know it today. You'll learn more about MPP systems and MapReduce in Chapters 3 and 8, respectively.

The Google papers inspired engineers at Yahoo to develop and later open source Apache Hadoop in 2006.[6] It's hard to overstate the impact of Hadoop. Software engineers interested in large-scale data problems were drawn to the possibilities of this new open source technology ecosystem. As companies of all sizes and types saw their data grow into many terabytes and even petabytes, the era of the big data engineer was born.

Around the same time, Amazon had to keep up with its own exploding data needs and created elastic computing environments (Amazon Elastic Compute Cloud, or EC2), infinitely scalable storage systems (Amazon Simple Storage Service, or S3), highly scalable NoSQL databases (Amazon DynamoDB), and many other core data building blocks.[7] Amazon elected to offer these services for internal and external consumption through *Amazon Web Services* (AWS), becoming the first popular public cloud. AWS created an ultra-flexible pay-as-you-go resource marketplace by virtualizing and reselling vast pools of commodity hardware. Instead of purchasing hardware for a data center, developers could simply rent compute and storage from AWS.

As AWS became a highly profitable growth engine for Amazon, other public clouds would soon follow, such as Google Cloud, Microsoft Azure, and DigitalOcean. The public cloud is arguably one of the most significant innovations of the 21st century and spawned a revolution in the way software and data applications are developed and deployed.

The early big data tools and public cloud laid the foundation for today's data ecosystem. The modern data landscape—and data engineering as we know it now—would not exist without these innovations.

The 2000s and 2010s: Big data engineering

Open source big data tools in the Hadoop ecosystem rapidly matured and spread from Silicon Valley to tech-savvy companies worldwide. For the first time, any business had access to the same bleeding-edge data tools used by the top tech companies. Another revolution occurred with the transition from batch computing to event streaming, ushering in a new era of big "real-time" data. You'll learn about batch and event streaming throughout this book.

Engineers could choose the latest and greatest—Hadoop, Apache Pig, Apache Hive, Dremel, Apache HBase, Apache Storm, Apache Cassandra, Apache Spark, Presto, and numerous other new technologies that came on the scene. Traditional enterprise-oriented and GUI-based data tools suddenly felt outmoded, and code-first engineering was in vogue with the ascendance of MapReduce. We (the authors) were around during this time, and it felt like old dogmas died a sudden death upon the altar of big data.

The explosion of data tools in the late 2000s and 2010s ushered in the *big data engineer*. To effectively use these tools and techniques—namely, the Hadoop ecosystem including Hadoop, YARN, Hadoop Distributed File System (HDFS), and MapReduce—big data engineers had to be proficient in software development and low-level infrastructure hacking, but with a shifted emphasis. Big data engineers typically maintained massive clusters of commodity hardware to deliver data at scale. While they might occasionally submit pull requests to Hadoop core code, they shifted their focus from core technology development to data delivery.

Big data quickly became a victim of its own success. As a buzzword, *big data* gained popularity during the early 2000s through the mid-2010s. Big data captured the imagination of companies trying to make sense of the ever-growing volumes of data and the endless barrage of shameless marketing from companies selling big data tools and services. Because of the immense hype, it was common to see companies using big data tools for small data problems, sometimes standing up a Hadoop cluster to process just a few gigabytes. It seemed like everyone wanted in on the big data action. Dan Ariely tweeted, "Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."

Figure 1-2 shows a snapshot of Google Trends for the search term "big data" to get an idea of the rise and fall of big data.

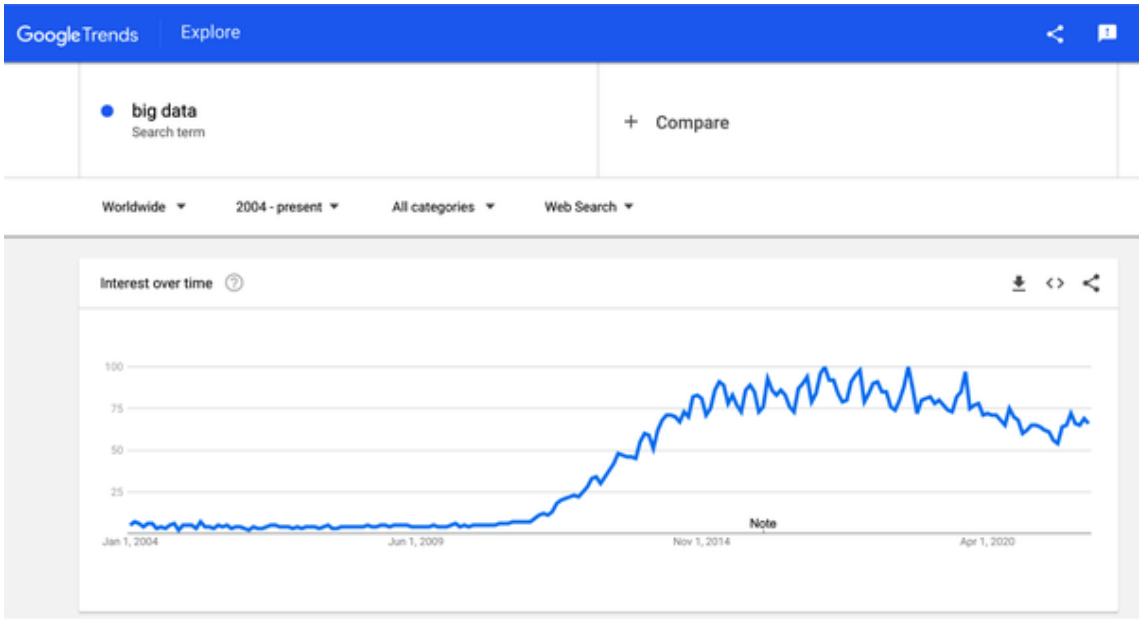

Figure 1-2. Google Trends for "big data" (March 2022)

Despite the term's popularity, big data has lost steam. What happened? One word: simplification. Despite the power and sophistication of open source big data tools, managing them was a lot of work and required constant attention. Often, companies employed entire teams of big data engineers, costing millions of dollars a year, to babysit these platforms. Big data engineers often spent excessive time maintaining complicated tooling and arguably not as much time delivering the business's insights and value.

Open source developers, clouds, and third parties started looking for ways to abstract, simplify, and make big data available without the high administrative overhead and cost of managing their clusters, and installing, configuring, and upgrading their open source code. The term *big data* is essentially a relic to describe a particular time and approach to handling large amounts of data.

Today, data is moving faster than ever and growing ever larger, but big data processing has become so accessible that it no longer merits a separate term; every company aims to solve its data problems, regardless of actual data size. Big data engineers are now simply *data engineers*.

The 2020s: Engineering for the data lifecycle

At the time of this writing, the data engineering role is evolving rapidly. We expect this evolution to continue at a rapid clip for the foreseeable future. Whereas data engineers historically tended to the low-level details of monolithic frameworks such as Hadoop, Spark, or Informatica, the trend is moving toward decentralized, modularized, managed, and highly abstracted tools.

Indeed, data tools have proliferated at an astonishing rate (see Figure 1-3). Popular trends in the early 2020s include the *modern data stack*, representing a collection of off-the-shelf open source and third-party products assembled to make analysts' lives easier. At the same time, data sources and data formats are growing both in variety and size. Data engineering is increasingly a discipline of interoperation, and connecting various technologies like LEGO bricks, to serve ultimate business goals.

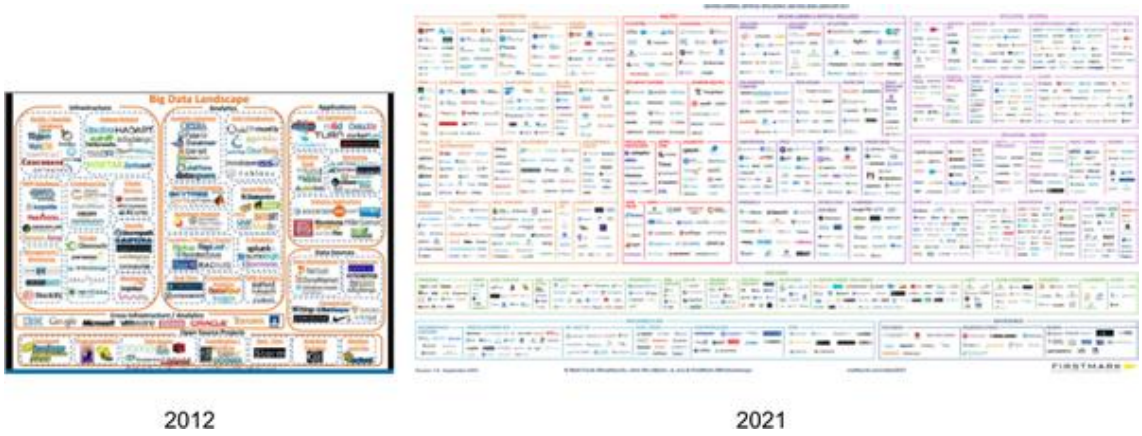

Figure 1-3. Matt Turck's Data Landscape in 2012 versus 2021

The data engineer we discuss in this book can be described more precisely as a *data lifecycle engineer*. With greater abstraction and simplification, a data lifecycle engineer is no longer encumbered by the gory details of yesterday's big data frameworks. While data engineers maintain skills in low-level data programming and use these as required, they increasingly find their role focused on things higher in the value chain: security, data management, DataOps, data architecture, orchestration, and general data lifecycle management.[8]

As tools and workflows simplify, we've seen a noticeable shift in the attitudes of data engineers. Instead of focusing on who has the "biggest data," open source projects and services are increasingly concerned with managing and governing data, making it easier to use and discover, and improving its quality. Data engineers are now conversant in acronyms such as *CCPA* and *GDPR*;[9] as they engineer pipelines, they concern themselves with privacy, anonymization, data garbage collection, and compliance with regulations.

What's old is new again. While "enterprisey" stuff like data management (including data quality and governance) was common for large enterprises in the pre-big-data era, it wasn't widely adopted in smaller companies. Now that many of the challenging problems of yesterday's data systems are solved, neatly productized, and packaged, technologists and entrepreneurs have shifted focus back to the "enterprisey" stuff, but with an emphasis on decentralization and agility, which contrasts with the traditional enterprise command-and-control approach.

We view the present as a golden age of data lifecycle management. Data engineers managing the data engineering lifecycle have better tools and techniques than ever before. We discuss the data engineering lifecycle and its undercurrents in greater detail in the next chapter.

Data Engineering and Data Science

Where does data engineering fit in with data science? There's some debate, with some arguing data engineering is a subdiscipline of data science. We believe data engineering is *separate* from data science and analytics. They complement each other, but they are distinctly different. Data engineering sits upstream from data science (Figure 1-4), meaning data engineers provide the inputs used by data scientists (downstream from data engineering), who convert these inputs into something useful.

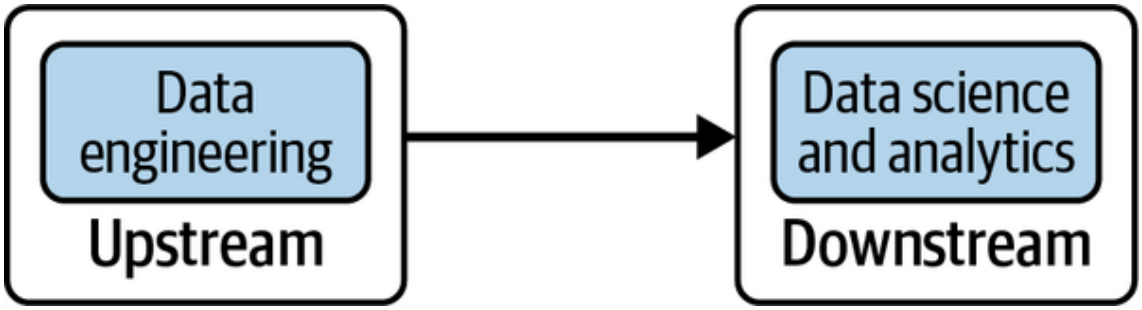

Figure 1-4. Data engineering sits upstream from data science

Consider the Data Science Hierarchy of Needs (Figure 1-5). In 2017, Monica Rogati published this hierarchy in an article that showed where AI and machine learning (ML) sat in proximity to more "mundane" areas such as data movement/storage, collection, and infrastructure.

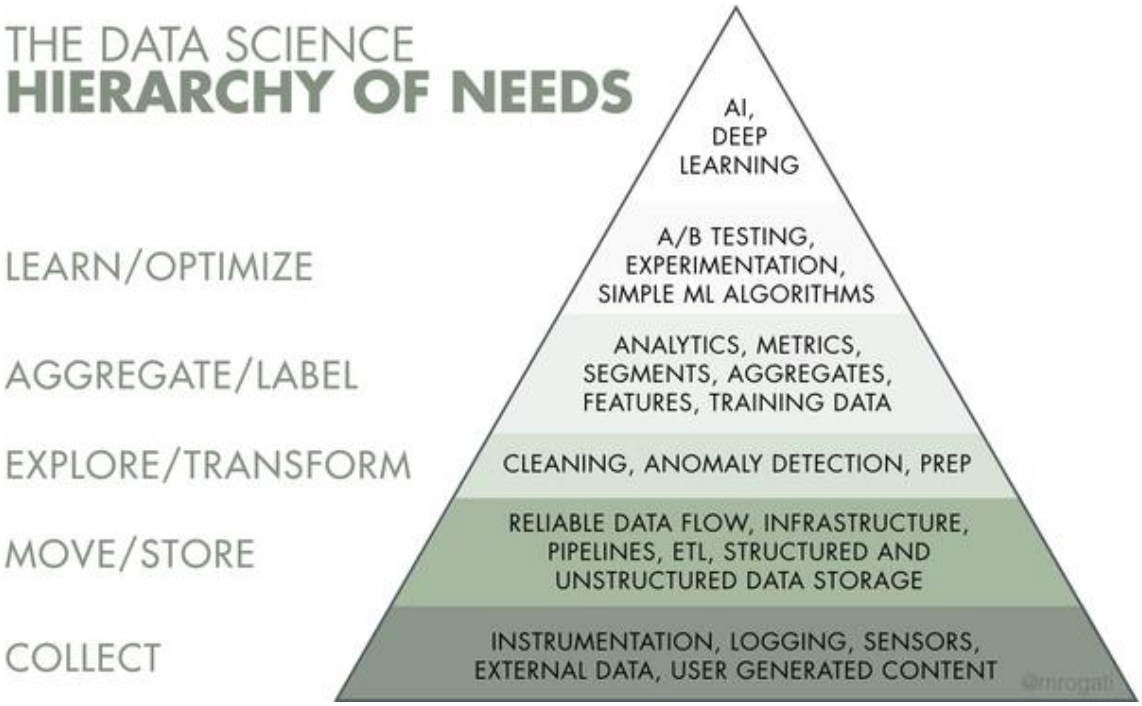

Figure 1-5. The Data Science Hierarchy of Needs

Although many data scientists are eager to build and tune ML models, the reality is an estimated 70% to 80% of their time is spent toiling in the bottom three parts of the hierarchy—gathering data, cleaning data, processing data—and only a tiny slice of their time on analysis and ML. Rogati argues that companies need to build a solid data foundation (the bottom three levels of the hierarchy) before tackling areas such as AI and ML.

Data scientists aren't typically trained to engineer production-grade data systems, and they end up doing this work haphazardly because they lack the support and resources of a data engineer. In an ideal world, data scientists should spend more than 90% of their time focused on the top layers of the pyramid: analytics, experimentation, and ML. When data engineers focus on these bottom parts of the hierarchy, they build a solid foundation for data scientists to succeed.

With data science driving advanced analytics and ML, data engineering straddles the divide between getting data and getting value from data (see Figure 1-6). We believe data engineering is of equal importance and visibility to data science, with data engineers playing a vital role in making data science successful in production.

Figure 1-6. A data engineer gets data and provides value from the data

Data Engineering Skills and Activities

The skill set of a data engineer encompasses the "undercurrents" of data engineering: security, data management, DataOps, data architecture, and software engineering. This skill set requires an understanding of how to evaluate data tools and how they fit together across the data engineering lifecycle. It's also critical to know how data is produced in source systems and how analysts and data scientists will consume and create value after processing and curating data. Finally, a data engineer juggles a lot of complex moving parts and must constantly optimize along the axes of cost, agility, scalability, simplicity, reuse, and interoperability (Figure 1-7). We cover these topics in more detail in upcoming chapters.

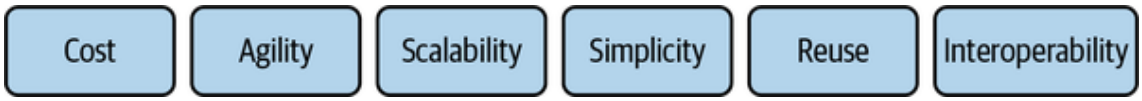

Figure 1-7. The balancing act of data engineering

As we discussed, in the recent past, a data engineer was expected to know and understand how to use a small handful of powerful and monolithic technologies (Hadoop, Spark, Teradata, Hive, and many others) to create a data solution. Utilizing these technologies often requires a sophisticated understanding of software engineering, networking, distributed computing, storage, or other low-level details. Their work would be devoted to cluster administration and maintenance, managing overhead, and writing pipeline and transformation jobs, among other tasks.

Nowadays, the data-tooling landscape is dramatically less complicated to manage and deploy. Modern data tools considerably abstract and simplify workflows. As a result, data engineers are now focused on balancing the simplest and most cost-effective, best-of-breed services that deliver value to the business. The data engineer is also expected to create agile data architectures that evolve as new trends emerge.

What are some things a data engineer does *not* do? A data engineer typically does not directly build ML models, create reports or dashboards, perform data analysis, build key performance indicators (KPIs), or develop software applications. A data engineer should have a good functioning understanding of these areas to serve stakeholders best.

Data Maturity and the Data Engineer

The level of data engineering complexity within a company depends a great deal on the company's data maturity. This significantly impacts a data engineer's day-to-day job responsibilities and career progression. What is data maturity, exactly?

Data maturity is the progression toward higher data utilization, capabilities, and integration across the organization, but data maturity does not simply depend on the age or revenue of a company. An early-stage startup can have greater data maturity than a 100-year-old company with annual revenues in the billions. What matters is the way data is leveraged as a competitive advantage.

Data maturity models have many versions, such as Data Management Maturity (DMM) and others, and it's hard to pick one that is both simple and useful for data engineering. So, we'll create our own simplified data maturity model. Our data maturity model (Figure 1-8) has three stages: starting with data, scaling with data, and leading with data. Let's look at each of these stages and at what a data engineer typically does at each stage.

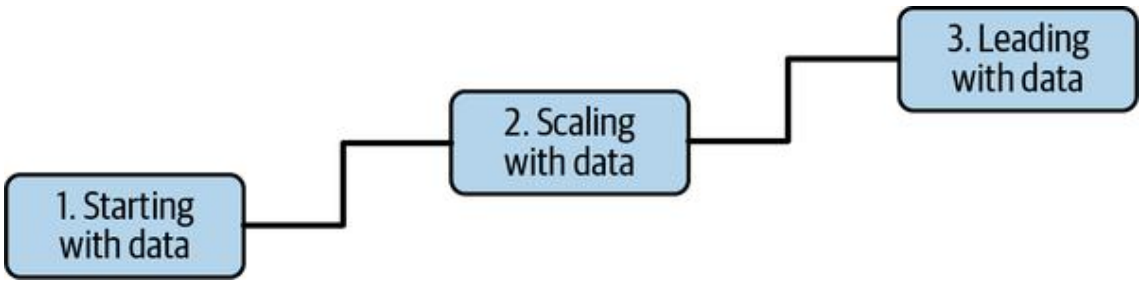

Figure 1-8. Our simplified data maturity model for a company

Stage 1: Starting with data

A company getting started with data is, by definition, in the very early stages of its data maturity. The company may have fuzzy, loosely defined goals or no goals. Data architecture and infrastructure are in the very early stages of planning and development. Adoption and utilization are likely low or nonexistent. The data team is small, often with a headcount in the single digits. At this stage, a data engineer is usually a generalist and will typically play several other roles, such as data scientist or software engineer. A data engineer's goal is to move fast, get traction, and add value.

The practicalities of getting value from data are typically poorly understood, but the desire exists. Reports or analyses lack formal structure, and most requests for data are ad hoc. While it's tempting to jump headfirst into ML at this stage, we don't recommend it. We've seen countless data teams get stuck and fall short when they try to jump to ML without building a solid data foundation.

That's not to say you can't get wins from ML at this stage—it is rare but possible. Without a solid data foundation, you likely won't have the data to train reliable ML models nor the means to deploy these models to production in a scalable and repeatable way. We half-jokingly call ourselves "recovering data scientists", mainly from personal experience with being involved in premature data science projects without adequate data maturity or data engineering support.

A data engineer should focus on the following in organizations getting started with data:

- Get buy-in from key stakeholders, including executive management. Ideally, the data engineer should have a sponsor for critical initiatives to design and build a data architecture to support the company's goals.

- Define the right data architecture (usually solo, since a data architect likely isn't available). This means determining business goals and the competitive advantage you're aiming to achieve with your data initiative. Work toward a data architecture that supports these goals. See Chapter 3 for our advice on "good" data architecture.
- Identify and audit data that will support key initiatives and operate within the data architecture you designed.
- Build a solid data foundation for future data analysts and data scientists to generate reports and models that provide competitive value. In the meantime, you may also have to generate these reports and models until this team is hired.

This is a delicate stage with lots of pitfalls. Here are some tips for this stage:

- Organizational willpower may wane if a lot of visible successes don't occur with data. Getting quick wins will establish the importance of data within the organization. Just keep in mind that quick wins will likely create technical debt. Have a plan to reduce this debt, as it will otherwise add friction for future delivery.
- Get out and talk to people, and avoid working in silos. We often see the data team working in a bubble, not communicating with people outside their departments and getting perspectives and feedback from business stakeholders. The danger is you'll spend a lot of time working on things of little use to people.
- Avoid undifferentiated heavy lifting. Don't box yourself in with unnecessary technical complexity. Use off-the-shelf, turnkey solutions wherever possible.
- Build custom solutions and code only where this creates a competitive advantage.

Stage 2: Scaling with data

At this point, a company has moved away from ad hoc data requests and has formal data practices. Now the challenge is creating scalable data architectures and planning for a future where the company is genuinely data-driven. Data engineering roles move from generalists to specialists, with people focusing on particular aspects of the data engineering lifecycle.

In organizations that are in stage 2 of data maturity, a data engineer's goals are to do the following:

- Establish formal data practices
- Create scalable and robust data architectures
- Adopt DevOps and DataOps practices
- Build systems that support ML
- Continue to avoid undifferentiated heavy lifting and customize only when a competitive advantage results

We return to each of these goals later in the book.

Issues to watch out for include the following:

- As we grow more sophisticated with data, there's a temptation to adopt bleeding-edge technologies based on social proof from Silicon Valley companies. This is rarely a good use of your time and energy. Any technology decisions should be driven by the value they'll deliver to your customers.
- The main bottleneck for scaling is not cluster nodes, storage, or technology but the data engineering team. Focus on solutions that are simple to deploy and manage to expand your team's throughput.
- You'll be tempted to frame yourself as a technologist, a data genius who can deliver magical products. Shift your focus instead to pragmatic leadership and begin transitioning to the next maturity stage; communicate with other teams about the practical utility of data. Teach the organization how to consume and leverage data.

Stage 3: Leading with data

At this stage, the company is data-driven. The automated pipelines and systems created by data engineers allow people within the company to do self-service analytics and ML. Introducing new data sources is seamless, and tangible value is derived. Data engineers implement proper controls and practices to ensure that data is always available to the people and systems. Data engineering roles continue to specialize more deeply than in stage 2.

In organizations in stage 3 of data maturity, a data engineer will continue building on prior stages, plus they will do the following:

- Create automation for the seamless introduction and usage of new data
- Focus on building custom tools and systems that leverage data as a competitive advantage
- Focus on the "enterprisey" aspects of data, such as data management (including data governance and quality) and DataOps
- Deploy tools that expose and disseminate data throughout the organization, including data catalogs, data lineage tools, and metadata management systems
- Collaborate efficiently with software engineers, ML engineers, analysts, and others
- Create a community and environment where people can collaborate and speak openly, no matter their role or position

Issues to watch out for include the following:

- At this stage, complacency is a significant danger. Once organizations reach stage 3, they must constantly focus on maintenance and improvement or risk falling back to a lower stage.
- Technology distractions are a more significant danger here than in the other stages. There's a temptation to pursue expensive hobby projects that don't deliver value to the business. Utilize custom-built technology only where it provides a competitive advantage.

The Background and Skills of a Data Engineer

Data engineering is a fast-growing field, and a lot of questions remain about how to become a data engineer. Because data engineering is a relatively new discipline, little formal training is available to enter the field. Universities don't have a standard data engineering path. Although a handful of data engineering boot camps and online tutorials cover random topics, a common curriculum for the subject doesn't yet exist.

People entering data engineering arrive with varying backgrounds in education, career, and skill set. Everyone entering the field should expect to invest a significant amount of time in self-study. Reading this book is a good starting point; one of the primary goals of this book is to give you a foundation for the knowledge and skills we think are necessary to succeed as a data engineer.

If you're pivoting your career into data engineering, we've found that the transition is easiest when moving from an adjacent field, such as software engineering, ETL development, database administration, data science, or data analysis. These disciplines tend to be "data aware" and provide good context for data roles in an organization. They also equip folks with the relevant technical skills and context to solve data engineering problems.

Despite the lack of a formalized path, a requisite body of knowledge exists that we believe a data engineer should know to be successful. By definition, a data engineer must understand both data and technology. With respect to data, this entails knowing about various best practices around data management. On the technology end, a data engineer must be aware of various options for tools, their interplay, and their trade-offs. This requires a good understanding of software engineering, DataOps, and data architecture.

Zooming out, a data engineer must also understand the requirements of data consumers (data analysts and data scientists) and the broader implications of data across the organization. Data engineering is a holistic practice; the best data engineers view their responsibilities through business and technical lenses.

Business Responsibilities

The macro responsibilities we list in this section aren't exclusive to data engineers but are crucial for anyone working in a data or technology field. Because a simple Google search will yield tons of resources to learn about these areas, we will simply list them for brevity:

Know how to communicate with nontechnical and technical people.

Communication is key, and you need to be able to establish rapport and trust with people across the organization. We suggest paying close attention to organizational hierarchies, who reports to whom, how people interact, and which silos exist. These observations will be invaluable to your success.

Understand how to scope and gather business and product requirements.

You need to know what to build and ensure that your stakeholders agree with your assessment. In addition, develop a sense of how data and technology decisions impact the business.

Understand the cultural foundations of Agile, DevOps, and DataOps.

Many technologists mistakenly believe these practices are solved through technology. We feel this is dangerously wrong. Agile, DevOps, and DataOps are fundamentally cultural, requiring buy-in across the organization.

Control costs.

You'll be successful when you can keep costs low while providing outsized value. Know how to optimize for time to value, the total cost of ownership, and opportunity cost. Learn to monitor costs to avoid surprises.

Learn continuously.

The data field feels like it's changing at light speed. People who succeed in it are great at picking up new things while sharpening their fundamental knowledge. They're also good at filtering, determining which new developments are most relevant to their work, which are still immature, and which are just fads. Stay abreast of the field and learn how to learn.

A successful data engineer always zooms out to understand the big picture and how to achieve outsized value for the business. Communication is vital, both for technical and nontechnical people. We often see data teams succeed based on their communication with other stakeholders; success or failure is rarely a technology issue. Knowing how to navigate an organization, scope and gather requirements, control costs, and continuously learn will set you apart from the data engineers who rely solely on their technical abilities to carry their career.

Technical Responsibilities

You must understand how to build architectures that optimize performance and cost at a high level, using prepackaged or homegrown components. Ultimately, architectures and constituent technologies are building blocks to serve the data engineering lifecycle. Recall the stages of the data engineering lifecycle:

- Generation
- Storage
- Ingestion
- Transformation
- Serving

The undercurrents of the data engineering lifecycle are the following:

- Security
- Data management
- DataOps
- Data architecture
- Orchestration
- Software engineering

Zooming in a bit, we discuss some of the tactical data and technology skills you'll need as a data engineer in this section; we discuss these in more detail in subsequent chapters.

People often ask, should a data engineer know how to code? Short answer: yes. A data engineer should have production-grade software engineering chops. We note that the nature of software development projects undertaken by data engineers has changed fundamentally in the last few years. Fully managed services now replace a great deal of low-level programming effort previously expected of engineers, who now use managed open source, and simple plug-and-play software-as-a-service (SaaS) offerings. For example, data engineers now focus on high-level abstractions or writing pipelines as code within an orchestration framework.

Even in a more abstract world, software engineering best practices provide a competitive advantage, and data engineers who can dive into the deep architectural details of a codebase give their companies an edge when specific technical needs arise. In short, a data engineer who can't write production-grade code will be severely hindered, and we don't see this changing anytime soon. Data engineers remain software engineers, in addition to their many other roles.

What languages should a data engineer know? We divide data engineering programming languages into primary and secondary categories. At the time of this writing, the primary languages of data engineering are SQL, Python, a Java Virtual Machine (JVM) language (usually Java or Scala), and bash:

SQL

The most common interface for databases and data lakes. After briefly being sidelined by the need to write custom MapReduce code for big data processing, SQL (in various forms) has reemerged as the lingua franca of data.

Python

The bridge language between data engineering and data science. A growing number of data engineering tools are written in Python or have Python APIs. It's known as "the second-best language at everything." Python underlies popular data tools such as pandas, NumPy, Airflow, sci-kit learn, TensorFlow, PyTorch, and PySpark. Python is the glue between underlying components and is frequently a first-class API language for interfacing with a framework.

JVM languages such as Java and Scala

Prevalent for Apache open source projects such as Spark, Hive, and Druid. The JVM is generally more performant than Python and may provide access to lower-level features than a Python API (for example, this is the case for Apache Spark and Beam). Understanding Java or Scala will be beneficial if you're using a popular open source data framework.

bash

The command-line interface for Linux operating systems. Knowing bash commands and being comfortable using CLIs will significantly improve your productivity and workflow when you need to script or perform OS operations. Even today, data engineers frequently use command-line tools like awk or sed to process files in a data pipeline or call bash

commands from orchestration frameworks. If you're using Windows, feel free to substitute PowerShell for bash.

The Unreasonable Effectiveness of SQL

The advent of MapReduce and the big data era relegated SQL to passé status. Since then, various developments have dramatically enhanced the utility of SQL in the data engineering lifecycle. Spark SQL, Google BigQuery, Snowflake, Hive, and many other data tools can process massive amounts of data by using declarative, set-theoretic SQL semantics. SQL is also supported by many streaming frameworks, such as Apache Flink, Beam, and Kafka. We believe that competent data engineers should be highly proficient in SQL.

Are we saying that SQL is a be-all and end-all language? Not at all. SQL is a powerful tool that can quickly solve complex analytics and data transformation problems. Given that time is a primary constraint for data engineering team throughput, engineers should embrace tools that combine simplicity and high productivity. Data engineers also do well to develop expertise in composing SQL with other operations, either within frameworks such as Spark and Flink or by using orchestration to combine multiple tools. Data engineers should also learn modern SQL semantics for dealing with JavaScript Object Notation (JSON) parsing and nested data and consider leveraging a SQL management framework such as dbt (Data Build Tool).

A proficient data engineer also recognizes when SQL is not the right tool for the job and can choose and code in a suitable alternative. A SQL expert could likely write a query to stem and tokenize raw text in a natural language processing (NLP) pipeline but would also recognize that coding in native Spark is a far superior alternative to this masochistic exercise.

Data engineers may also need to develop proficiency in secondary programming languages, including R, JavaScript, Go, Rust, C/C++, C#, and Julia. Developing in these languages is often necessary when popular across the company or used with domain-specific data tools. For instance, JavaScript has proven popular as a language for user-defined functions in cloud data warehouses. At the same time, C# and PowerShell are essential in companies that leverage Azure and the Microsoft ecosystem.

Keeping Pace in a Fast-Moving Field

Once a new technology rolls over you, if you're not part of the steamroller, you're part of the road.

Stewart Brand

How do you keep your skills sharp in a rapidly changing field like data engineering? Should you focus on the latest tools or deep dive into fundamentals? Here's our advice: focus on the fundamentals to understand what's not going to change; pay attention to ongoing developments to know where the field is going. New paradigms and practices are introduced all the time, and it's incumbent on you to stay current. Strive to understand how new technologies will be helpful in the lifecycle.

The Continuum of Data Engineering Roles, from A to B

Although job descriptions paint a data engineer as a "unicorn" who must possess every data skill imaginable, data engineers don't all do the same type of work or have the same skill

set. Data maturity is a helpful guide to understanding the types of data challenges a company will face as it grows its data capability. It's beneficial to look at some critical distinctions in the kinds of work data engineers do. Though these distinctions are simplistic, they clarify what data scientists and data engineers do and avoid lumping either role into the unicorn bucket.

In data science, there's the notion of type A and type B data scientists.[10] *Type A data scientists*—where *A* stands for *analysis*—focus on understanding and deriving insight from data. *Type B data scientists*—where *B* stands for *building*—share similar backgrounds as type A data scientists and possess strong programming skills. The type B data scientist builds systems that make data science work in production. Borrowing from this data scientist continuum, we'll create a similar distinction for two types of data engineers:

Type A data engineers

A stands for *abstraction*. In this case, the data engineer avoids undifferentiated heavy lifting, keeping data architecture as abstract and straightforward as possible and not reinventing the wheel. Type A data engineers manage the data engineering lifecycle mainly by using entirely off-the-shelf products, managed services, and tools. Type A data engineers work at companies across industries and at all levels of data maturity.

Type B data engineers

B stands for *build*. Type B data engineers build data tools and systems that scale and leverage a company's core competency and competitive advantage. In the data maturity range, a type B data engineer is more commonly found at companies in stage 2 and 3 (scaling and leading with data), or when an initial data use case is so unique and mission-critical that custom data tools are required to get started.

Type A and type B data engineers may work in the same company and may even be the same person! More commonly, a type A data engineer is first hired to set the foundation, with type B data engineer skill sets either learned or hired as the need arises within a company.

Data Engineers Inside an Organization

Data engineers don't work in a vacuum. Depending on what they're working on, they will interact with technical and nontechnical people and face different directions (internal and external). Let's explore what data engineers do inside an organization and with whom they interact.

Internal-Facing Versus External-Facing Data Engineers

A data engineer serves several end users and faces many internal and external directions (Figure 1-9). Since not all data engineering workloads and responsibilities are the same, it's essential to understand whom the data engineer serves. Depending on the end-use cases, a data engineer's primary responsibilities are external facing, internal facing, or a blend of the two.

Figure 1-9. The directions a data engineer faces

An *external-facing* data engineer typically aligns with the users of external-facing applications, such as social media apps, Internet of Things (IoT) devices, and ecommerce platforms. This data engineer architects, builds, and manages the systems that collect, store, and process transactional and event data from these applications. The systems built by these data engineers have a feedback loop from the application to the data pipeline, and then back to the application (Figure 1-10).

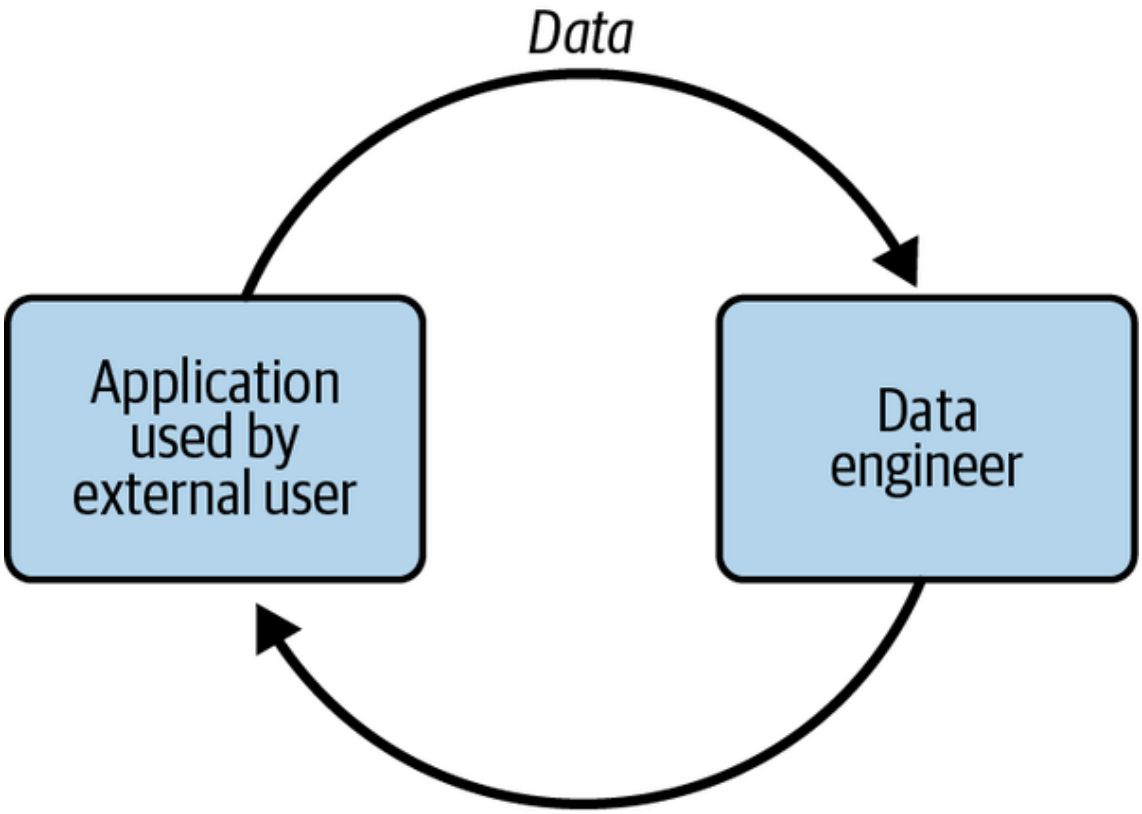

Figure 1-10. External-facing data engineer systems

External-facing data engineering comes with a unique set of problems. External-facing query engines often handle much larger concurrency loads than internal-facing systems. Engineers also need to consider putting tight limits on queries that users can run to limit the infrastructure impact of any single user. In addition, security is a much more complex and sensitive problem for external queries, especially if the data being queried is multitenant (data from many customers and housed in a single table).

An *internal-facing data engineer* typically focuses on activities crucial to the needs of the business and internal stakeholders (Figure 1-11). Examples include creating and maintaining data pipelines and data warehouses for BI dashboards, reports, business processes, data science, and ML models.

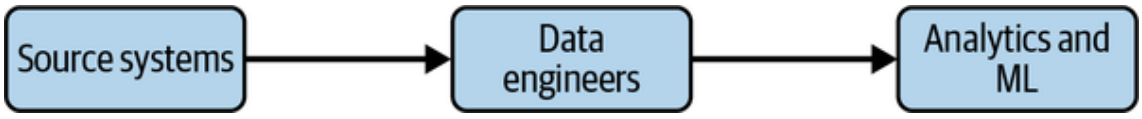

Figure 1-11. Internal-facing data engineer

External-facing and internal-facing responsibilities are often blended. In practice, internal-facing data is usually a prerequisite to external-facing data. The data engineer has two sets of users with very different requirements for query concurrency, security, and more.

Data Engineers and Other Technical Roles

In practice, the data engineering lifecycle cuts across many domains of responsibility. Data engineers sit at the nexus of various roles, directly or through managers, interacting with many organizational units.

Let's look at whom a data engineer may impact. In this section, we'll discuss technical roles connected to data engineering (Figure 1-12).

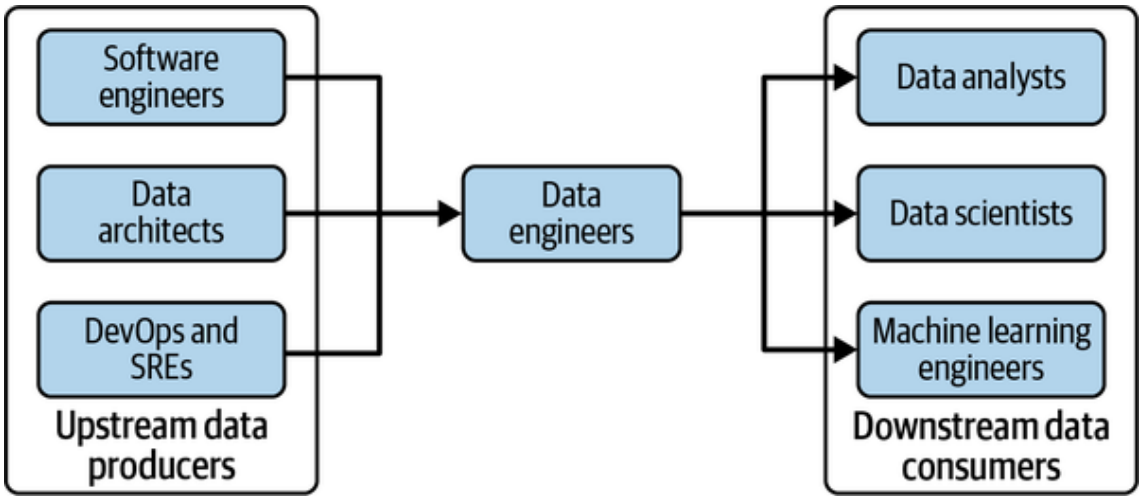

Figure 1-12. Key technical stakeholders of data engineering

The data engineer is a hub between *data producers*, such as software engineers, data architects, and DevOps or site-reliability engineers (SREs), and *data consumers*, such as data analysts, data scientists, and ML engineers. In addition, data engineers will interact with those in operational roles, such as DevOps engineers.

Given the pace at which new data roles come into vogue (analytics and ML engineers come to mind), this is by no means an exhaustive list.

Upstream stakeholders

To be successful as a data engineer, you need to understand the data architecture you're using or designing and the source systems producing the data you'll need. Next, we discuss a few familiar upstream stakeholders: data architects, software engineers, and DevOps engineers.

Data architects

Data architects function at a level of abstraction one step removed from data engineers. Data architects design the blueprint for organizational data management, mapping out processes and overall data architecture and systems.[11] They also serve as a bridge between an organization's technical and nontechnical sides. Successful data architects generally have "battle scars" from extensive engineering experience, allowing them to guide and assist engineers while successfully communicating engineering challenges to nontechnical business stakeholders.

Data architects implement policies for managing data across silos and business units, steer global strategies such as data management and data governance, and guide significant initiatives. Data architects often play a central role in cloud migrations and greenfield cloud design.

The advent of the cloud has shifted the boundary between data architecture and data engineering. Cloud data architectures are much more fluid than on-premises systems, so architecture decisions that traditionally involved extensive study, long lead times, purchase contracts, and hardware installation are now often made during the implementation process, just one step in a larger strategy. Nevertheless, data architects will remain influential visionaries in enterprises, working hand in hand with data engineers to determine the big picture of architecture practices and data strategies.

Depending on the company's data maturity and size, a data engineer may overlap with or assume the responsibilities of a data architect. Therefore, a data engineer should have a good understanding of architecture best practices and approaches.

Note that we have placed data architects in the *upstream stakeholders* section. Data architects often help design application data layers that are source systems for data engineers. Architects may also interact with data engineers at various other stages of the data engineering lifecycle. We cover "good" data architecture in Chapter 3.

Software engineers

Software engineers build the software and systems that run a business; they are largely responsible for generating the *internal data* that data engineers will consume and process. The systems built by software engineers typically generate application event data and logs, which are significant assets in their own right. This internal data contrasts with *external data* pulled from SaaS platforms or partner businesses. In well-run technical organizations, software engineers and data engineers coordinate from the inception of a new project to design application data for consumption by analytics and ML applications.

A data engineer should work together with software engineers to understand the applications that generate data, the volume, frequency, and format of the generated data, and anything else that will impact the data engineering lifecycle, such as data security and regulatory compliance. For example, this might mean setting upstream expectations on what the data software engineers need to do their jobs. Data engineers must work closely with the software engineers.

DevOps engineers and site-reliability engineers

DevOps and SREs often produce data through operational monitoring. We classify them as upstream of data engineers, but they may also be downstream, consuming data through dashboards or interacting with data engineers directly in coordinating operations of data systems.

Downstream stakeholders

Data engineering exists to serve downstream data consumers and use cases. This section discusses how data engineers interact with various downstream roles. We'll also introduce a few service models, including centralized data engineering teams and cross-functional teams.

Data scientists

Data scientists build forward-looking models to make predictions and recommendations. These models are then evaluated on live data to provide value in various ways. For example, model scoring might determine automated actions in response to real-time conditions, recommend products to customers based on the browsing history in their current session, or make live economic predictions used by traders.

According to common industry folklore, data scientists spend 70% to 80% of their time collecting, cleaning, and preparing data.[12] In our experience, these numbers often reflect immature data science and data engineering practices. In particular, many popular data science frameworks can become bottlenecks if they are not scaled up appropriately. Data scientists who work exclusively on a single workstation force themselves to downsample data, making data preparation significantly more complicated and potentially compromising the quality of the models they produce. Furthermore, locally developed code and environments are often difficult to deploy in production, and a lack of automation significantly hampers data science workflows. If data engineers do their job and collaborate successfully, data scientists shouldn't spend their time collecting, cleaning, and preparing data after initial exploratory work. Data engineers should automate this work as much as possible.

The need for production-ready data science is a significant driver behind the emergence of the data engineering profession. Data engineers should help data scientists to enable a path to production. In fact, we (the authors) moved from data science to data engineering after recognizing this fundamental need. Data engineers work to provide the data automation and scale that make data science more efficient.

Data analysts

Data analysts (or business analysts) seek to understand business performance and trends. Whereas data scientists are forward-looking, a data analyst typically focuses on the past or present. Data analysts usually run SQL queries in a data warehouse or a data lake. They may also utilize spreadsheets for computation and analysis and various BI tools such as Microsoft Power BI, Looker, or Tableau. Data analysts are domain experts in the data they work with frequently and become intimately familiar with data definitions, characteristics, and quality problems. A data analyst's typical downstream customers are business users, management, and executives.

Data engineers work with data analysts to build pipelines for new data sources required by the business. Data analysts' subject-matter expertise is invaluable in improving data quality, and they frequently collaborate with data engineers in this capacity.

Machine learning engineers and AI researchers

Machine learning engineers (ML engineers) overlap with data engineers and data scientists. ML engineers develop advanced ML techniques, train models, and design and maintain the infrastructure running ML processes in a scaled production environment. ML engineers often have advanced working knowledge of ML and deep learning techniques and frameworks such as PyTorch or TensorFlow.

ML engineers also understand the hardware, services, and systems required to run these frameworks, both for model training and model deployment at a production scale. It's

common for ML flows to run in a cloud environment where ML engineers can spin up and scale infrastructure resources on demand or rely on managed services.

As we've mentioned, the boundaries between ML engineering, data engineering, and data science are blurry. Data engineers may have some operational responsibilities over ML systems, and data scientists may work closely with ML engineering in designing advanced ML processes.

The world of ML engineering is snowballing and parallels a lot of the same developments occurring in data engineering. Whereas several years ago, the attention of ML was focused on how to build models, ML engineering now increasingly emphasizes incorporating best practices of machine learning operations (MLOps) and other mature practices previously adopted in software engineering and DevOps.

AI researchers work on new, advanced ML techniques. AI researchers may work inside large technology companies, specialized intellectual property startups (OpenAI, DeepMind), or academic institutions. Some practitioners are dedicated to part-time research in conjunction with ML engineering responsibilities inside a company. Those working inside specialized ML labs are often 100% dedicated to research. Research problems may target immediate practical applications or more abstract demonstrations of AI. DALL-E, Gato AI, AlphaGo, and GPT-3/GPT-4 are great examples of ML research projects. Given the pace of advancements in ML, these examples will very likely be quaint in a few years' time. We've provided some references in "Additional Resources".

AI researchers in well-funded organizations are highly specialized and operate with supporting teams of engineers to facilitate their work. ML engineers in academia usually have fewer resources but rely on teams of graduate students, postdocs, and university staff to provide engineering support. ML engineers who are partially dedicated to research often rely on the same support teams for research and production.

Data Engineers and Business Leadership

We've discussed technical roles with which a data engineer interacts. But data engineers also operate more broadly as organizational connectors, often in a nontechnical capacity. Businesses have come to rely increasingly on data as a core part of many products or a product in itself. Data engineers now participate in strategic planning and lead key initiatives that extend beyond the boundaries of IT. Data engineers often support data architects by acting as the glue between the business and data science/analytics.

Data in the C-suite

C-level executives are increasingly involved in data and analytics, as these are recognized as significant assets for modern businesses. For example, CEOs now concern themselves with initiatives that were once the exclusive province of IT, such as cloud migrations or deployment of a new customer data platform.

Chief executive officer

Chief executive officers (CEOs) at nontech companies generally don't concern themselves with the nitty-gritty of data frameworks and software. Instead, they define a vision in collaboration with technical C-suite roles and company data leadership. Data engineers provide a window into what's possible with data. Data engineers and their managers

maintain a map of what data is available to the organization—both internally and from third parties—in what time frame. They are also tasked to study primary data architectural changes in collaboration with other engineering roles. For example, data engineers are often heavily involved in cloud migrations, migrations to new data systems, or deployment of streaming technologies.

Chief information officer

A chief information officer (CIO) is the senior C-suite executive responsible for information technology within an organization; it is an internal-facing role. A CIO must possess deep knowledge of information technology and business processes—either alone is insufficient. CIOs direct the information technology organization, setting ongoing policies while also defining and executing significant initiatives under the direction of the CEO.

CIOs often collaborate with data engineering leadership in organizations with a well-developed data culture. If an organization is not very high in its data maturity, a CIO will typically help shape its data culture. CIOs will work with engineers and architects to map out major initiatives and make strategic decisions on adopting major architectural elements, such as enterprise resource planning (ERP) and customer relationship management (CRM) systems, cloud migrations, data systems, and internal-facing IT.

Chief technology officer

A chief technology officer (CTO) is similar to a CIO but faces outward. A CTO owns the key technological strategy and architectures for external-facing applications, such as mobile, web apps, and IoT—all critical data sources for data engineers. The CTO is likely a skilled technologist and has a good sense of software engineering fundamentals and system architecture. In some organizations without a CIO, the CTO or sometimes the chief operating officer (COO) plays the role of CIO. Data engineers often report directly or indirectly through a CTO.

Chief data officer

The chief data officer (CDO) was created in 2002 at Capital One to recognize the growing importance of data as a business asset. The CDO is responsible for a company's data assets and strategy. CDOs are focused on data's business utility but should have a strong technical grounding. CDOs oversee data products, strategy, initiatives, and core functions such as master data management and privacy. Occasionally, CDOs manage business analytics and data engineering.

Chief analytics officer

The chief analytics officer (CAO) is a variant of the CDO role. Where both roles exist, the CDO focuses on the technology and organization required to deliver data. The CAO is responsible for analytics, strategy, and decision making for the business. A CAO may oversee data science and ML, though this largely depends on whether the company has a CDO or CTO role.

Chief algorithms officer

A chief algorithms officer (CAO-2) is a recent innovation in the C-suite, a highly technical role focused specifically on data science and ML. CAO-2s typically have experience as

individual contributors and team leads in data science or ML projects. Frequently, they have a background in ML research and a related advanced degree.

CAO-2s are expected to be conversant in current ML research and have deep technical knowledge of their company's ML initiatives. In addition to creating business initiatives, they provide technical leadership, set research and development agendas, and build research teams.

Data engineers and project managers

Data engineers often work on significant initiatives, potentially spanning many years. As we write this book, many data engineers are working on cloud migrations, migrating pipelines and warehouses to the next generation of data tools. Other data engineers are starting greenfield projects, assembling new data architectures from scratch by selecting from an astonishing number of best-of-breed architecture and tooling options.

These large initiatives often benefit from *project management* (in contrast to product management, discussed next). Whereas data engineers function in an infrastructure and service delivery capacity, project managers direct traffic and serve as gatekeepers. Most project managers operate according to some variation of Agile and Scrum, with Waterfall still appearing occasionally. Business never sleeps, and business stakeholders often have a significant backlog of things they want to address and new initiatives they want to launch. Project managers must filter a long list of requests and prioritize critical deliverables to keep projects on track and better serve the company.

Data engineers interact with project managers, often planning sprints for projects and ensuing standups related to the sprint. Feedback goes both ways, with data engineers informing project managers and other stakeholders about progress and blockers, and project managers balancing the cadence of technology teams against the ever-changing needs of the business.

Data engineers and product managers

Product managers oversee product development, often owning product lines. In the context of data engineers, these products are called *data products*. Data products are either built from the ground up or are incremental improvements upon existing products. Data engineers interact more frequently with *product managers* as the corporate world has adopted a data-centric focus. Like project managers, product managers balance the activity of technology teams against the needs of the customer and business.

Data engineers and other management roles

Data engineers interact with various managers beyond project and product managers. However, these interactions usually follow either the services or cross-functional models. Data engineers either serve a variety of incoming requests as a centralized team or work as a resource assigned to a particular manager, project, or product.

For more information on data teams and how to structure them, we recommend John Thompson's *Building Analytics Teams* (Packt) and Jesse Anderson's *Data Teams* (Apress). Both books provide strong frameworks and perspectives on the roles of executives with data, who to hire, and how to construct the most effective data team for your company.

Note

Companies don't hire engineers simply to hack on code in isolation. To be worthy of their title, engineers should develop a deep understanding of the problems they're tasked with solving, the technology tools at their disposal, and the people they work with and serve.

Conclusion

This chapter provided you with a brief overview of the data engineering landscape, including the following:

- Defining data engineering and describing what data engineers do
- Describing the types of data maturity in a company
- Type A and type B data engineers
- Whom data engineers work with

We hope that this first chapter has whetted your appetite, whether you are a software development practitioner, data scientist, ML engineer, business stakeholder, entrepreneur, or venture capitalist. Of course, a great deal still remains to elucidate in subsequent chapters. Chapter 2 covers the data engineering lifecycle, followed by architecture in Chapter 3. The following chapters get into the nitty-gritty of technology decisions for each part of the lifecycle. The entire data field is in flux, and as much as possible, each chapter focuses on the *immutables*—perspectives that will be valid for many years amid relentless change.

Additional Resources

- "The AI Hierarchy of Needs" by Monica Rogati
- The AlphaGo research web page
- "Big Data Will Be Dead in Five Years" by Lewis Gavin
- *Building Analytics Teams* by John K. Thompson (Packt)
- Chapter 1 of *What Is Data Engineering?* by Lewis Gavin (O'Reilly)
- "Data as a Product vs. Data as a Service" by Justin Gage
- "Data Engineering: A Quick and Simple Definition" by James Furbush (O'Reilly)
- *Data Teams* by Jesse Anderson (Apress)
- "Doing Data Science at Twitter" by Robert Chang
- "The Downfall of the Data Engineer" by Maxime Beauchemin
- "The Future of Data Engineering Is the Convergence of Disciplines" by Liam Hausmann
- "How CEOs Can Lead a Data-Driven Culture" by Thomas H. Davenport and Nitin Mittal
- "How Creating a Data-Driven Culture Can Drive Success" by Frederik Bussler
- The Information Management Body of Knowledge website

- "Information Management Body of Knowledge" Wikipedia page
- "Information Management" Wikipedia page
- "On Complexity in Big Data" by Jesse Anderson (O'Reilly)
- "OpenAI's New Language Generator GPT-3 Is Shockingly Good—and Completely Mindless" by Will Douglas Heaven
- "The Rise of the Data Engineer" by Maxime Beauchemin
- "A Short History of Big Data" by Mark van Rijmenam
- "Skills of the Data Architect" by Bob Lambert
- "The Three Levels of Data Analysis: A Framework for Assessing Data Organization Maturity" by Emilie Schario
- "What Is a Data Architect? IT's Data Framework Visionary" by Thor Olavsrud
- "Which Profession Is More Complex to Become, a Data Engineer or a Data Scientist?" thread on Quora
- "Why CEOs Must Lead Big Data Initiatives" by John Weathington

1 "Data Engineering and Its Main Concepts," AlexSoft, last updated August 26, 2021, *https://oreil.ly/e94py*.

2 *ETL* stands for *extract, transform, load*, a common pattern we cover in the book.

3 Jesse Anderson, "The Two Types of Data Engineering," June 27, 2018, *https://oreil.ly/dxDt6*.

4 Maxime Beauchemin, "The Rise of the Data Engineer," January 20, 2017, *https://oreil.ly/kNDmd*.

5 Lewis Gavin, *What Is Data Engineering?* (Sebastapol, CA: O'Reilly, 2020), *https://oreil.ly/ELxLi*.

6 Cade Metz, "How Yahoo Spawned Hadoop, the Future of Big Data," *Wired*, October 18, 2011, *https://oreil.ly/iaD9G*.

7 Ron Miller, "How AWS Came to Be," *TechCrunch*, July 2, 2016, *https://oreil.ly/VJehv*.

8 *DataOps* is an abbreviation for *data operations*. We cover this topic in Chapter 2. For more information, read the DataOps Manifesto.

9 These acronyms stand for *California Consumer Privacy Act* and *General Data Protection Regulation*, respectively.

10 Robert Chang, "Doing Data Science at Twitter," *Medium*, June 20, 2015, *https://oreil.ly/xqjAx*.

11 Paramita (Guha) Ghosh, "Data Architect vs. Data Engineer," Dataversity, November 12, 2021, *https://oreil.ly/TlyZY*.

12 A variety of references exist for this notion. Although this cliche is widely known, a healthy debate has arisen around its validity in different practical settings. For more details,

see Leigh Dodds, “Do Data Scientists Spend 80% of Their Time Cleaning Data? Turns Out, No?” Lost Boy blog, January 31, 2020, *https://oreil.ly/szFww*; and Alex Woodie, “Data Prep Still Dominates Data Scientists’ Time, Survey Finds,” *Datanami*, July 6, 2020, *https://oreil.ly/jDVWF*.