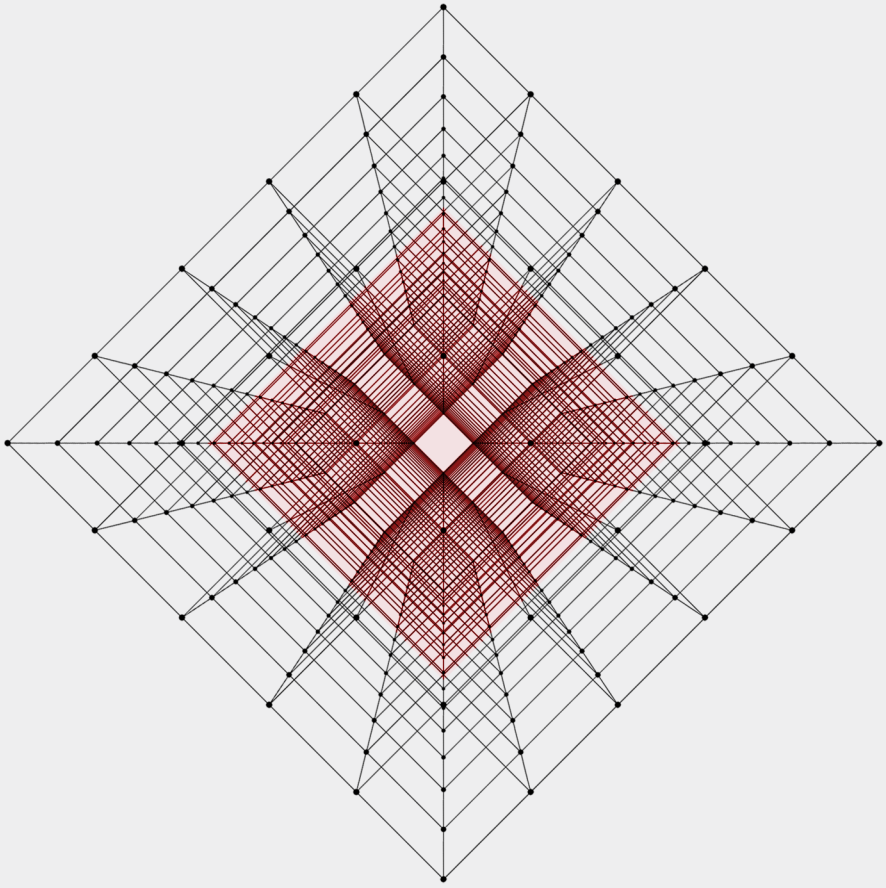
ELEMENTS OF
DATA STRATEGY

A Framework for the Analytics Manager



BOYAN ANGELOV

Elements of Data Strategy

A Framework for the Analytics Manager

Boyan Angelov

This book is for sale at http://leanpub.com/elementsofdatastrategy

This version was published on 2022-09-26

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Boyan Angelov by spreading the word about this book on Twitter!

The suggested hashtag for this book is #elementsofdatastrategy.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#elementsofdatastrategy

Contents

Preface

> "Software eats the world, but AI will eat software."
> –Jensen Huang

Those are the words of the CEO of NVIDIA. While they are full of commitment, most of us agree we are still far from this lofty goal. This becomes clearer if we look at the current state of enterprise data efforts. Despite the tremendous potential new data technologies offer, few organizations currently reap the benefits of data-driven digital transformation. Adoption and deployment of data and AI technologies remain rare, contrasting with big words from executives and their significant financial commitments. But why? Indeed, at this point, progress in the field should not be hampered by the lack of technical talent anymore. Or lack of tools and frameworks available? Why is it so hard to follow in the footsteps of the Googles and Amazons of the world and take full advantage of data?

This book aims to answer this question by providing a framework for action - a *data strategy*. Data strategy is not just a supplement to a business strategy, but it's most vital and full of potential element. These pages will show you how to design and deliver a successful data strategy.

Boyan Angelov,
Berlin, Fall 2022

Introduction

Motivating factors—Book structure—Learning how to fish—Systems 101—The skills of a data strategist—Defining data strategy—StratOps

Why did I write this book?

To be a good data strategist (more on this title in a bit), you must first gain a diverse set of experiences. I was fortunate enough for this to be the case with me. It took me a while to appreciate that being a jack of all trades can provide tremendous advantages down the line in a new job title, moving beyond the data scientist or engineer. DJ Patil and Tom Davenport, in their 2022 article "Is Data Scientist Still the Sexiest Job of the 21st Century?"* support this view: while data science has had meteoric growth (and will continue to do so in the foreseeable future), it spawned even more fast-growing disciplines, such as data strategy. Here are some of the diverse experiences I obtained so far in my career: worked on metagenomic data at the Max Planck Institute, built machine learning models and architectures for startups and large organizations, led my teams to do that at scale as a CTO, and have been guiding industry leaders in helping their organizations do the same. One thing stood out when looking back at all those different areas: the increased complexity of modern-day work. Each of those experiences required its own set of skills, tools, associated frameworks and ways of working, and best practices. What kept me going was the vast array of tutorials, courses, books,

*There's a follow-up to this article, asking the same question as of 2022.

and articles at my disposal - I rarely felt starved for help. This is why I was quite surprised to find a massive lack of resources on data strategy - when I started this position, I was mostly left to my own devices.

I remember those first days as a data strategist. They were full of questions and confusion. When do we do a CSA[*]? Before the Gap Analysis or after? What other elements are necessary for a comprehensive data strategy? What do we do with the deliverables of such work? How do they relate to each other? The issue wasn't limited to forming a brand new vocabulary; I was used to working with many new terms as a data scientist, after all. The more I thought about it, the more I realized that I couldn't even answer the most fundamental of questions: **what does it mean to be a data strategist?** What are the skills and experience necessary to get the job done? I searched, read, and asked - yet I could still not find confident and conclusive answers. While the *why* was clear to me from the beginning, I expected a manual on the *what* and the *how* of data strategy. I resorted to learning the hard way by listening to experienced leaders in the field and combining their knowledge into tangible and concrete concepts. Slowly the different concepts and definitions clicked together, and the answers to my questions became visible. Soon other people started approaching me with the same questions I had, and that's when I decided it was time to share what I learned, shaping it into a book you're reading now. "Elements of Data Strategy" [†] is the book I wish I had when starting in data strategy.

> **About the cover**: the book's cover represents two fundamental ideas. First: data strategy should be at the core of the business strategy. Second:

[*] Current State Analysis. More on this in DUE DILIGENCE.

[†] The name was inspired by one of my favorite books: "Elements of Statistical Learning" by Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie.

> this work showcases a *framework*: concepts, building blocks, and ideas that guide and inform the design of your strategy - it is still up to you to fill the missing pieces that fit your organization.

A good first question that the attentive reader will have is whether reading this book is sufficient to turn them into data strategists. I'll be honest - it won't. I came to realize this role requires a vast amount of experience across many fields; it's inherently transdisciplinary. Still, this book would teach you two things that I believe are necessary to keep at your side as you gain more practical experience: the concepts of data strategy and a holistic way of thinking about it. Those two combine into a framework you can apply to any strategic work in data. A second point I want to make here is on technical terms. There are more than enough resources on data lakes, algorithms, and self-service analytics - often written by their creators. Thus, instead of spending valuable time defining all of them, I'll point to the source. I will be referring to such concepts wherever necessary. Still, the focus will always be on providing the framework for data strategy and its main elements and connections.

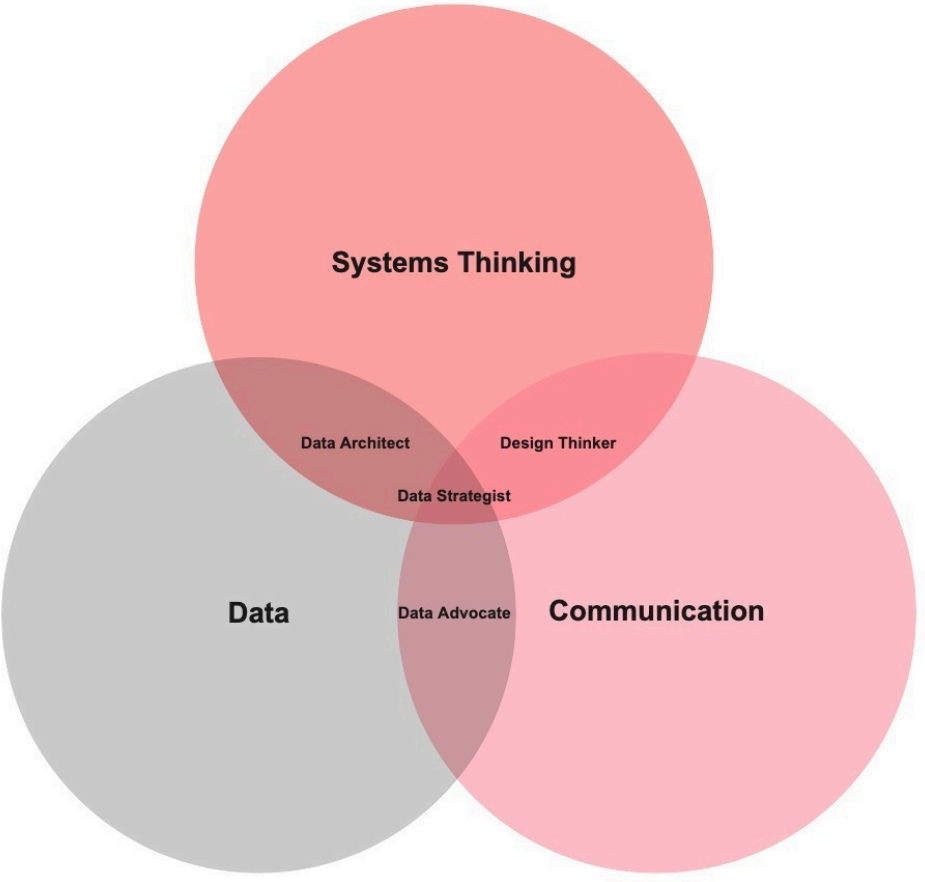
Another potential goal that I'll forgo is convincing the reader that you need a data strategy or even need to care about data. Many such books start with endless factual explanations of the low adoption of AI and the need for planning data projects. I assume you would not have this book in front of you if you thought otherwise, and if you still require convincing - there are more than enough materials.

A third housekeeping item acknowledges that this work represents a set of "working definitions". I realize that the field of data and data strategy, in general, is constantly evolving. By the time this book turns its first birthday,

many things will have changed. This is one of the reasons why I stayed away from using the term "modern" anywhere in the book (it would be arrogant to assume this will hold for years into the future). Treat this book as what it is - a framework, and I encourage its adaptation and reworking in the wild! The core concepts and way of thinking should remain constant.

As the first step in understanding data strategy, we should define the skills of a data strategist. In the early days of data science, there was a quite popular article called "The Data Scientist Venn diagram" by Drew Conway. Data science suffered from the same fate as data strategy today - it started as a new field (often dismissed as a fad), and it took a while for skills to be established, especially between different flavors of data science. I would argue that data strategy will follow a similar trajectory towards becoming well-established[*]. Now - let's have a look at a Venn diagram of the skills of a data strategist:

[*]More on this in my wide-ranging discussion with June Dershewitz, Data Strategist at Amazon, in the Interviews chapter.



The Data Strategist Venn Diagram

The grey circle, *Data*, contains all the (technical) domain knowledge required. Of course, this term itself is quite broad, but we can at least specify its higher-level internal components. Data comprises of Data Science, Data Engineering, and Business Intelligence (BI). In other words, in this circle, we have everything hands-on about the job. One of the most common questions I've received here is, to what extent a data strategist needs to be

familiar with those topics? There are different answers to this, but a data strategist with a solid grasp of the other circles of the Venn diagram will be able to compensate for any lack in this area. But as a rule of thumb, a data strategist should have a sufficient technical understanding of the three data areas I mentioned to the extent that allows them to manage a team of technical people. I'll cover this topic in more detail later in this chapter.

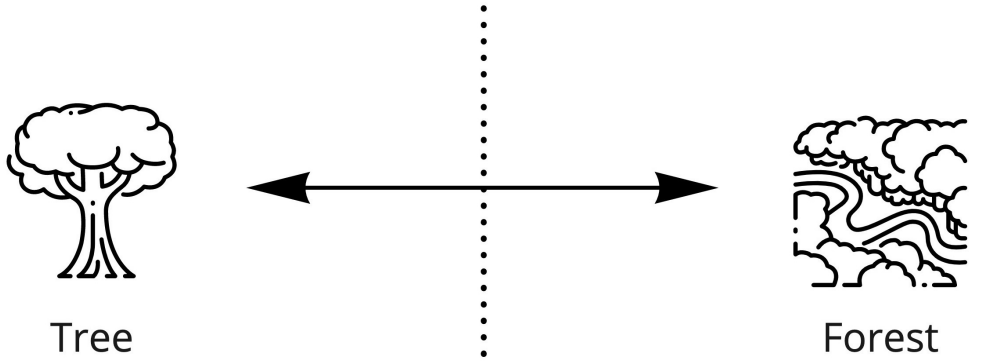
Now, topic number two: *Communication*. This collection of skills is frequently mentioned in technical circles as desirable but still - at least in my opinion - not emphasized enough. A common misconception about communication (at least in the technical realm) is that it's a skill that some people are just born with. I wholeheartedly disagree - while it's easier for some, it can be mostly learned - with deliberate practice. Why should we learn to communicate better, though? An effective data strategist should be able to discuss topics on many different levels, from the day-to-day data work (the items from the "data" circle), to high-level conversations, presentations, and strategy formulations with the C-level people. This concept is often described in data strategist job description as *"the ability to translate requirements and concepts between technical and executive stakeholders"*. I like the word "translation" - it even enhances the job title itself in some organizations, where they refer to us as "data translators" [*].

Finally, those two circles form the support for the integrative skill, which binds them together into a complete package - *System thinking* (ST). A thorough definition of ST is available in a paper by Arnold and Wade[1], I'll go into more detail on ST in `SYSTEMS 101`:

[*] DBT calls such roles "purple people", which is admittedly catchy. Have a read here.

> "Systems thinking is a set of synergistic analytic skills used to improve the capability of identifying and understanding systems, predicting their behaviors, and devising modifications to them in order to produce desired effects." - R. Arnold

There are overlaps between those different sections, resulting in the jobs of Data Architect, Data Advocate, and Design Thinker. All of those are important on their own, but our focus is on the middle of the diagram - the data strategist. What is important to consider when thinking about the skills of a data strategy is that all of those lie in a balance. Every individual data strategist possesses a unique combination of those three areas, and while with a sufficient dedication, this balance can be acquired - and you can be proficient in all three - it's impossible to be perfect. This concept is commonly referred to as the T-shaped skill proficiency. For example, if you dedicate much effort to technical skills, you also pay an opportunity cost for not learning business skills. For all of us, there are just twenty-four hours in a day, and we also have limited energy available. Thus a tradeoff between the three elements of the Venn diagram occurs, and since such tradeoffs are typical in any ST book - this becomes a pattern. I call it a "tension diagram" (we'll see many such patterns later in the book). Have a look at a classic example below. As data strategists, we often need to focus on both the forest and the tree, and this balance is often unclear:



The Data Strategist's Attention

Tension diagrams. The data strategist represents a balance between different systems of equal importance. This can also be described as a "tension diagram". For example, a data strategist often needs to pay attention to things at different levels. At the same time, they need to be looking at the details (the tree), and designing and delivering a strategy. There is always a balance between the two, and the tradeoffs are clear.

I'll be using a few ST terms frequently throughout this book. This is why I decided to dedicate a separate section later in this chapter to provide several key definitions - `SYSTEMS 101`.

Who is this book for?

There are two defining attributes of this book. First: the goal of this book is not to describe all concepts and processes in data. As I mentioned, many books go deeper into details, such as the difference between inferential

and descriptive statistics, what AI is, what data mesh is, and how to implement it. There are already excellent books (for example *Driving Digital Transformation Through Data and AI: A Practical Guide to Delivering Data Science and Machine Learning Products* by A. Borek and N. Prill[2]), tutorials, and courses available on all those topics - often by their creators as well. I will be linking and citing all those resources in the areas where necessary. Still, my goal is to provide a conceptual framework, allowing you to *think* about data strategy holistically. To use the common biblical analogy - I don't want to give you fish; I want to teach you *how to fish*. You'll still need to gain practical experience by applying the framework. If this is how you think, this is the book for you.

Second, many of the data strategies I've designed have been from a consultant's point of view, certainly the larger ones. I thought - this means that the book's target audience should perhaps be mostly other consultants. In some way, many of the concepts presented here would be easier to grasp for readers with consulting experience. Still, I believe, on some level, everyone is an in-house consultant, depending on the context. This is why I selected the book's subtitle as *"A Framework for the Analytics Manager"*. Everyone in a position to make decisions on data projects should derive benefits from this work, whether working with their own internal teams or as an external consultant. So when I use the word "client", I mean this as the recipient of the data strategy, whether they are external to your organization, or internal.

How is this book written?

In the Data Strategist's Venn diagram, I put a special emphasis on the systems skills. Those are the foundation of data strategy, so I designed this

as a *systems book*, first and foremost. I provide a holistic[*] model of data strategy. Building such mental models is fundamental in systems science and model thinking. Naming the elements of data strategy (giving them logical boundaries), and identifying the relationships between them lays the foundation for any hands-on work. Such a structure allows the reader to explore a data strategy inductively, communicate their ideas about it, and enable information gathering. Remember that Zen saying, where the Zen master said to their student: don't mistake my finger, pointing to the moon, for the moon itself. I believe the only way for a successful data strategy model to be designed is by letting go of our desire to perfectly model every situation and the arrogance of believing we actually can. Instead, we should take this for what it is - a model; the rest of the work we need to do ourselves. Remember - all models are wrong, but some are useful[3]. I certainly hope this one is!

The book comprises three large phases, capped by a final chapter with interviews with thought leaders in the field of data strategy worldwide. Each phase contains a suite of elements. Since those are commonly referred to throughout the work, they are indicated by a `CAPITALIZED AND MONOSPACED` font to draw attention. The main deliverables of each element are presented in the beginning of each subsection (with the exception of the `DELIVERY` phase, which is more about process than concrete deliverables). Here are the main phases of data strategy:

Phase I (`DUE DILIGENCE`): Here you'll understand the organization's overall business objectives and how data strategy must align to those. Then I'll explain what a Current State and Gap analysis are and how they form the critical core of information gathering. After completing this section, you'll

[*] A good definition is available from the Oxford English Dictionary: "[…] characterized by the belief that the parts of something are intimately interconnected and explicable only by reference to the whole".

have the foundation to create a customized data strategy.

Phase II: (DESIGN): I'll show you how to build a complete data strategy. It needs to contain plans on how to decide on optimal architecture and technology stacks; methods for selecting the best use cases, the ones positioned to deliver the greatest success for the business; design an effective and efficient operating model for them to be delivered later; take care of governance, ethics and security elements.

Phase III: (DELIVERY): Even the best strategy is useless unless delivered successfully. In this part, you'll learn how to ensure the data strategy is used by applying data-specific flavors of two methodologies: *soft agile* and *lean data*.

`Interviews`: This work has been heavily influenced by my conversations with data leaders worldwide. I'm privileged to share some of those in this section. Read them to understand how the principles we have covered are common in successful data strategies across many industries.

How to read this book?

Quite a few books on data are written in a reference format. With such books, you can pick any topic you find exciting and dive straight into it - without paying attention to any assigned order. I structured this book differently. Creating a data strategy is, by necessity, a sequential process. The phases and their elements are building on top of each other; the outputs of a phase become the inputs of the following one (you'll understand this in SYSTEMS 101). Those elements might not make sense if consumed in haste and out of order. I suggest reading the book following the designated order first and only afterward using it as a reference manual in case you want to refresh

your memory and knowledge on a specific element.

Another point to remember is that no two data strategies are identical. They should all follow a similar structure, but you as a strategist will always need to shift focus and order were necessary for your specific case. That Zen teaching rings true once again. It would be arrogant to assume that one strategy template is all you need to make any large or small organization data-driven.

The book also has a companion website, containing additional materials. You can see it by visiting this link or scanning the QR code:

Additionally, there are several types of blurbs that you'll encounter throughout the book

Information: Since data strategy is a transdisciplnary and expansive field, I'll be adding any further reading here.

Warning: I have learned the hard way how a journey through data strategy can be sidetracked. I'll emphasize those situations here.

Tip: Here I'll provide practical words that can give you the edge.

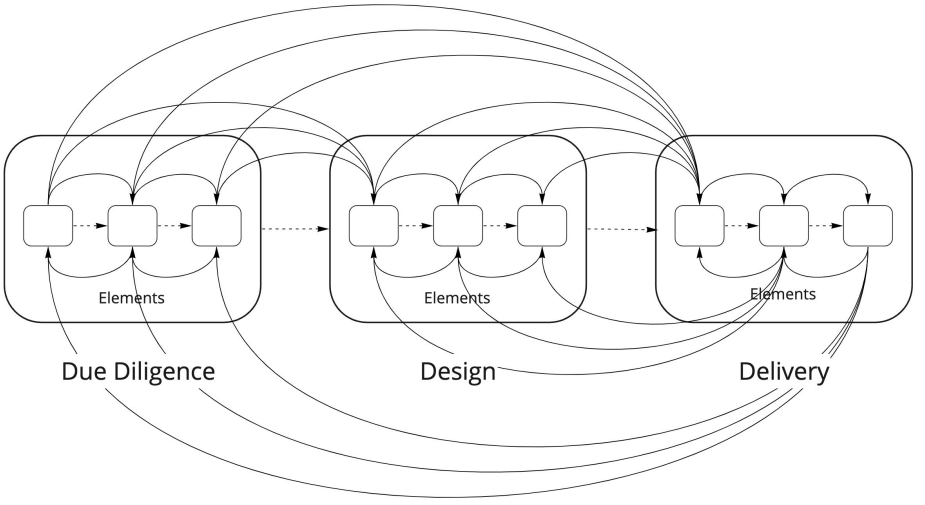
Discussion: The topics covered in those blurbs are meant to be discussed as a group, so this is a good opportunity to engage with your fellow data strategists, or online.

> **Asides and sidebars**: Some topics don't fit perfectly well in the scope of the book, but can be useful. Those will be mentioned here.

Systems 101

There's a lot of literature on ST - and that's both a blessing and a curse. A blessing since we have an abundance of resources to gain an understanding from. A curse since this amount of information can be, at times, overwhelming. To complicate the matter further, the field is full of conflicting views and overlaps heavily with other areas, such as operations research and complexity management. Later I'll go into deeper detail as to why definition setting is essential, but suffice to say that working with vague terms during already complex work, such as data strategy, can only add unnecessary confusion.

The Elements of Data Strategy is a **systems book**. Have a look at what I mean by that in the figure below:



Systems approach to data strategy.

The three phases of data strategy don't exist in isolation. The output of an element within `DUE DILIGENCE` can become an input to an element in `DESIGN`, or `DELIVERY`. For example, the `DATA DICTIONARY` is relied upon heavily during `ARCHITECTURE AND TECHNOLOGY`. The loops are not only moving forward but also backward - providing essential feedback. For example, if during `DELIVERY`, the strategist can realize that an adjustment is needed in it's design - the project's successful implementation needs to be supported by a different operating model*. This diagram demonstrates how the concepts of boundaries, feedback loops, abstraction levels, and other related ones are essential.

In the following subsections, I'll define all crucial ST concepts specifically.

*See StratOps below.

Boundaries for systems discovery

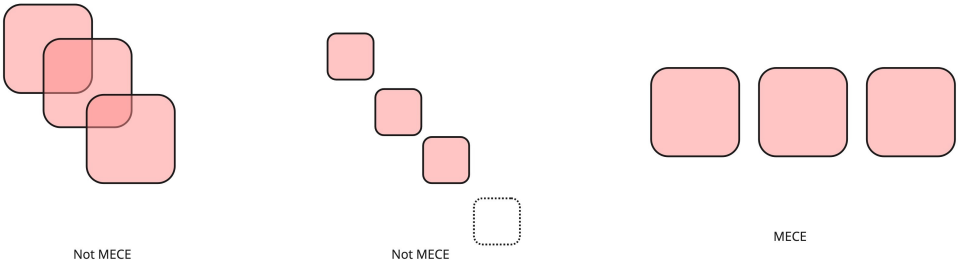
The first essential concept is probably the most abstract: *boundaries*. As you'll see later in the book (in CHANGE MANAGEMENT), the primary challenge for a data strategist is communication. So many abstract terms need to be explained, and any provided definitions can be vague. For a data strategy to be successfully implemented down the line, the communication between the strategist and the client or in the strategy document itself needs to be spot on. Having concise and clear words for the concepts we discuss, and a shared understanding ultimately enables us to be productive and focus on the actual work.

This is also one of the main reasons I wrote this book - I lacked the common understanding of terms to discuss how to do data strategy. Now: every time we communicate complex topics, we set *boundaries*. Knowing where those are in different systems is also an essential task in CSA.

Definition setting. For many data strategy elements, I recommend setting a common vocabulary of definitions at the beginning of each engagement. This is essential to avoid confusion down the line since we are discussing challenging topics to understand by default. We must be precise, even if it frequently feels indulgent and wasteful of time. Definition setting is also essential when more technical members start to use the data strategy; without it, the work can lead to miscommunication and subsequent frustration. To provide you with a specific example, one time with a client, I talked about whether we should be doing data anonymization or pseudonymization. Much later, both parties realized what was meant by those terms. To avoid issues moving forward, we must sit together and define the standard terms.

The reason why I cover the concept of boundaries first is that this is the defining feature of a system. A boundary is where one system ends, and another one begins. Philosophically, everything is a system - one within another*. Almost all the data strategy activities start with explicitly defining the systems we are working with and understanding their boundaries. With my colleagues, I used to joke that the number one skill of a successful data strategist is to draw boxes! For example, we might draw the organization's different departments as different systems. We can then proceed to draw the boundaries of the teams. Within those boundaries, we can fit other elements, enabling us to talk concretely about otherwise abstract terms.

My method of choice for boundary setting is MECE:, standing for *Mutually Exclusive, Collectively Exhaustive*. I learned about it from the excellent book *Technology Strategy Patterns* by E. Hewitt[4]. Have a look at the diagram below:



Defining MECE. It's defined much better by what it's not, rather than what it is.

Two common sources of error appear when we break down a concept into parts. The first one is that two (or more) elements have sub-elements in common, blurring the separation between them. The second source of error is that we are not presenting the whole picture; some elements are

*Much like the Russian dolls where you can put one into another many times over.

missing. If we want to avoid both, we should always attempt to break down a larger concept into a complete set of parts without anything missing. Those parts should have clear borders between them delineated so there's no ambiguity. This is an essential tool in reducing complexity and the fundamental principle in designing the elements of data strategy. All of the elements of data strategy in this book are designed to be MECE.

Complexity

One of the most overused and dangerous words in data strategy (and management in general) is *complexity*. To paint a picture of how frustrating it is to define this word, I'll share a quote (unfortunately, I can't recall by who): "Complexity is so hard to define that even the eponymous book, *Complexity* doesn't define it in any of its 600 pages". Complexity is the biggest enemy of a data strategist and any knowledge worker. And for such an important term, it's mind-boggling to realize that a definition and a universally agreed upon measurement method are not available.

A good working definition of what a complex system is available from the Santa Fe Institute. It can be summarised in several properties: nonlinearity, randomness, collective dynamics, hierarchy, and emergence.

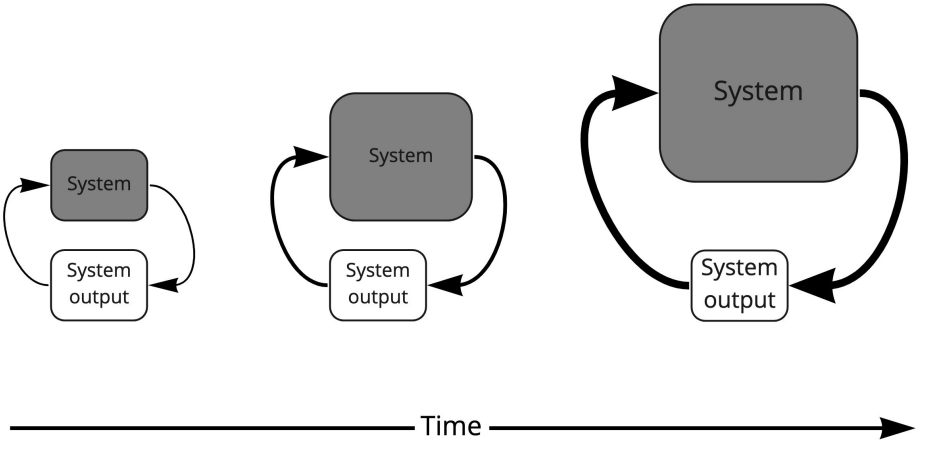
Complex systems: human generated and natural.

What are some examples of complex systems? The nervous system, cities, society in general, the jungle, the ocean, a large corporation.

Feedback loops

The easiest way to understand what feedback loops are (you can also read Donella Meadow's great systems book[5]) is with an example from our daily life. We all know what we talk about when we say, "I got off the wrong side of the bed today". Suppose you have a negative experience first thing in the morning. In that case, things can cascade further down - your already bad mood will likely attract more negative experiences and, consequently - an even worse mood. This is an example of a positive feedback loop. Positive,

not because it's a nice thing to happen, but because the system continues to grow with time. Have a look at the following diagram:



Positive feedback loop.

An example of feedback loops in data strategy is the *inertia* of larger organizations in changing the ways they operate. This is an issue in almost any data strategy engagement and can be summarised in the sentence "we've always done things this way, and it worked out". Unfortunately, inertia is hard to fight. The good news is - once the data strategy is in place and the feedback loop starts to go into the opposite, more positive, and organized direction - it would be equally hard to reverse.

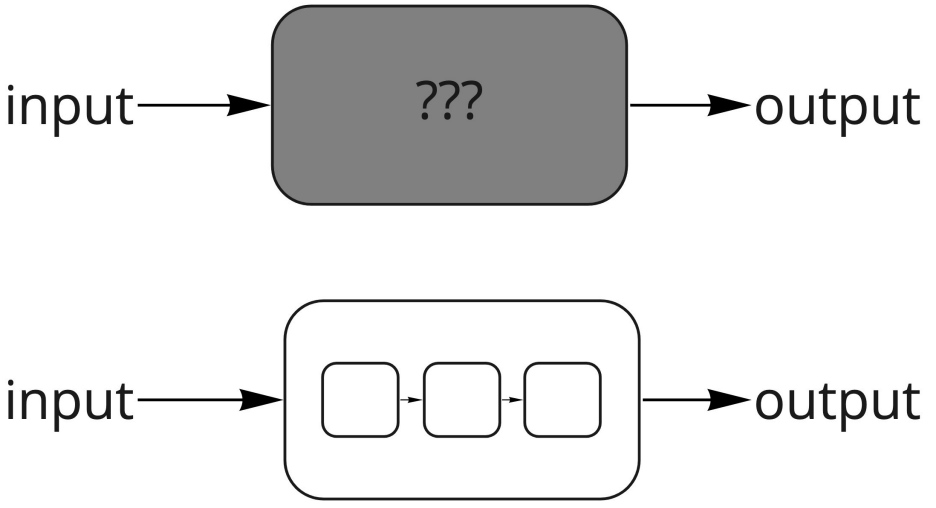
Black boxes

Let me practice what I preach and start with a definition:

> A black box is any system whose internal operation and subsys-

tems we don't understand, yet need to work with.

This is visualized below:



Black box vs. transparent box scenarios.

Making technology decisions on well-documented and stable systems can already be challenging, but how about opaque ones - when we deal with a black box? As technology leaders, we almost always operate within constraints - but a lack of general understanding of a system is perhaps the most challenging (especially under time pressure and if the system in question is mission-critical, such as payments). Here I'll go through common black box scenarios and provide advice on how to deal with them:

- Legacy software: When we inherit software written by others, we rely solely on adequate documentation, which is unfortunately rare.

- Senior engineering talent leaving: Even if they wrote good documentation, understanding complex systems takes time.
- Poorly written systems.

Of course, we can understand any black box if unlimited time is available, but that's seldom the case. Yet, there are several things that we can do:

- Measure outputs and inputs: While the systems can be opaque, what goes into and out of them is not. Analyzing those points can yield valuable insights into the system's inner workings.
- Adopt a scientific approach: Conduct experiments, test hypotheses, and document the results. Observe the behavior of the system patiently over time.
- Set up feedback loops: While running the system, prod it and observe any changes in behavior and performance.
- Isolate subsystems: Break down the black box into separate components and measure their inputs and outputs instead of the system as a whole.
- Replicate: Replace different components piece by piece, culminating in a complete copy.

These solutions will help engineering leaders guide their teams around the frustrations and dangers of working with black box systems.

System types

This is a great time to discuss another system thinking concept - that of viable systems. In the systems thinking literature, those are also known as Viable Systems, or Complex Adaptive Systems (CAS). We can look at our

data strategy as a system - and our goal would be to ensure that the data strategy we create is adaptable instead of rigid. “Fragile” systems should be self-explanatory - they can break anytime. Imagine a house of cards, where the whole system falls down with the pull of only one element. An example of this in data strategy is a recommendation of the wrong vendor. For example, a company’s CTO decides on a specific technology only because they have prior experience with it or for other political reasons - instead of whether it’s fit for purpose for the particular use case. Such a strategic recommendation can create a very fragile system. Let’s focus on a more common type of data strategy systems (and planning systems in general) - the rigid ones. While nobody wants to build a fragile system, a rigid one can be tempting - especially one designed by high-paid external experts. The most common cognitive bias is the illusion that we can predict the future with any degree of accuracy. This, together with our domain expertise, can lead to having supreme confidence in creating plans. This scenario is even more dangerous because, in the short term, such a strategy can work and inspire confidence. Everybody wants to follow a leader with a clear plan, and nobody wants to hear or present thoughts of uncertainty. Still, such systems are bound to fail in the mid-to-long term since they collide with the complexity of the real world.

Abstraction levels

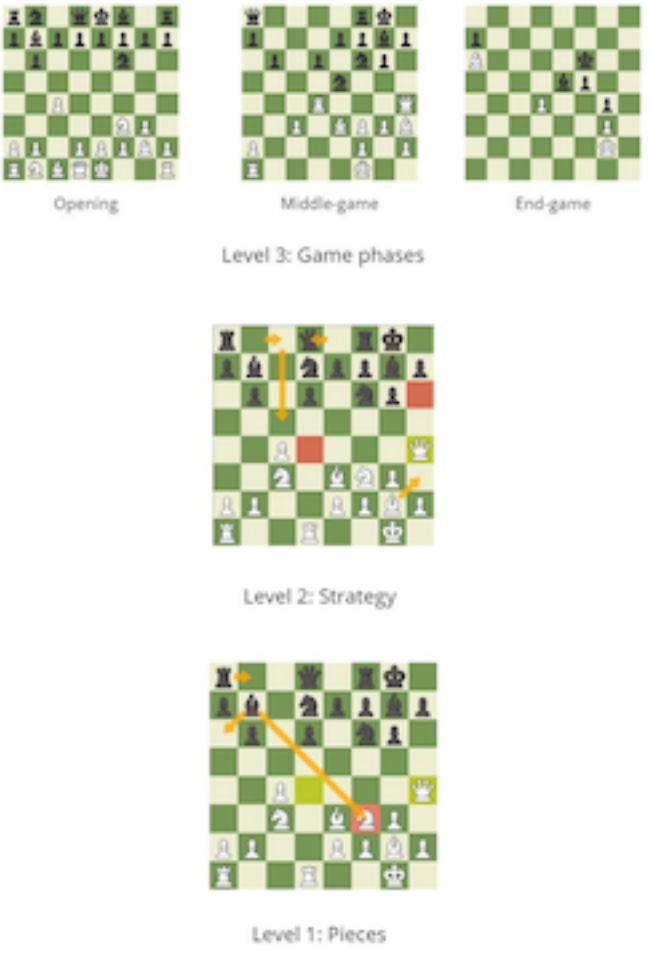
One of the best ways to deal with complex systems is by using *abstractions* (in systems and complexity science this is almost equivalent to the concept of scale[6]). The human mind is uniquely capable of reasoning through the same problem in various ways. Since, by definition, we can’t understand a black box (complex system), we need to apply abstractions. You can imagine them as mental maps we sketch to think differently and be productive. This

also allows us to look at problems with less detail necessary. After all, we don't need to understand a system's inner workings, but mostly its outputs and inputs. Let's define this concept:

> Abstraction levels are a stack of abstractions, or mental maps (models), from different viewpoints and angles[*].

To drive the point home, I'll illustrate with an example from arguably the most strategic game invented: chess. Look at the boards below (you'll need just a basic understanding of chess, don't worry):

[*] This method will come up in handy in the CSA section.



Abstraction levels in a chess game

Remember our attention tension diagram? It is relevant at every move in a chess game. At each point, the player needs to think on three levels: individual pieces, strategy, and game phase. Looking at the game in this way allows the player to be as creative as possible. Every abstraction layer provides a set of rules: the rules on how individual pieces move are self-explanatory, but the other two levels are more complex. For example, on a strategic level, the player needs to pay attention to the overall balance of

the board, perhaps planning their attacks in one area only. At the same time, any decisions they make on this second strategic abstraction layer need to consider the overall phase of the game - the long-term picture.

One of the most famous visualizations of abstraction levels is by Picasso. His famous 1946 series of eleven lithographs “Bull” shows several bulls, sketched at different levels of abstraction, from most complexity to least.

Once you get used to thinking in abstraction levels, you’ll see them everywhere. Using this ST method will allow the strategist to operate in black box scenarios under constraints effectively and efficiently.

Technical skills for data strategists

One of the most common questions is how “technical” a data strategist needs to be. My experience is that of someone from a very technical background (science and software engineering). Still, I can draw from it and my observations of successful colleagues in the field.

As a natural first step, we must define what “technical” means. This term is represented in two circles in the data strategist Venn diagram I presented earlier in the chapter - Data and Systems Thinking. “Technical” relates to those two terms - this is the ability to both guide the implementation of a data (software) system and its connection to other systems (which can be nontechnical). There’s one thing we should already get out of the way - it is certain that the more hands-on experience you have with software and data technologies, the better. The real question we’re after here is that for an inherently cross-disciplinary position such as that of a data strategist,

all the three circles of the Venn diagram need to be balanced. If one has spent all possible time and effort in their career in the technical area only (or any of the others, for that matter), they are bound to have missed out on developing their other skill sets. In such a case, with all the technical talent and experience in the world, their effectiveness will be limited if the data strategist is not strong in systems and communication.

The balance I just talked about is always different from person to person. There will always be data strategists of slightly different focuses - and that's ok. The key lies in having an honest view of your specific skillset, both competencies and some shortcomings. Based on this information, you can try to surround yourself with other data strategists (or even technical business people) so that you can offer the complete package together. Most of the knowledge work in the 21st century can only be accomplished by a team, and data strategy is no exception.

Now that we know that balance needs to be kept, what is a good level of technical and systems expertise? This is such a complex question that we need to answer it appropriately. Instead of providing a giant list of programming languages, frameworks, databases, and cloud services to master, I'll illustrate cases from the real world with examples.

Let's say that as a part of the data strategy, all the data in the organization needs to be organized and stored correctly. On top of this, good access policies must be implemented (data governance, more on this in a later subsection in this part) - striking a good balance between restrictive and open. For the first task, a data strategist should understand the use cases, the data formats, quality and size, and different data storage terms (data lake, warehouse, relational and non-relational databases etc.), enough to

provide guidance to the implementers. This should be enough technical expertise.

What can be more challenging here is the systems expertise. The data strategist should be excellent in how different systems (in this case, a combination of cloud services, such as EC2, scripts, and databases) operate with each other and the pros and cons of different configurations (architectures). This is a much harder skill set to acquire since it requires a lot of experience across the whole data stack (from business intelligence to science and engineering).

If a data strategist is capable of selecting the optimal technologies, combining them to form solutions to common use cases, and making informed decisions on resources, budgets, and constraints on such systems - they are good enough for the job.

Hopefully, you can appreciate the sheer breadth of skills necessary to become a data strategist and understand why it's hard; let's focus on what you *can* get out of this work.

Defining data strategy

> Strategy: a plan of action designed to achieve a long-term or overall aim.
> –Oxford English Dictionary

There's an additional reason why this is a systems book. For a strategy to be effective, it needs to be viewed from a specific angle: as a roadmap - but not a "static", unchangeable one - but a living, malleable and adaptable to the continuously changing data landscape. Currently, most organizations

look at strategy as a collection of slides, delivered at the end of a strategic engagement, often by expensive external consultants. Yes, those slides should serve as a blueprint, but most would agree they are useless if not applied in practice[*]. They also fall short of a real strategy that can evolve and be updated long after the engagement has concluded. The fast-moving and changing world of data are far too complex for static plans - they are prone to become obsolete at the very moment they are presented.

Making sure that a data strategy is future-proof is achieved by adding an analytics playbook to the more static stages. This playbook will serve both leaders and implementers of the data strategy initiatives. The company Taival provides an excellent comparison between a traditional strategy process and what they call *StratOps*, inspired by Tom Paterson:

STRATEGY	**STRATOPS**
Inwards directed	Customer driven and ecosystem centric
C-level driven	Across the organization (inclusive)
Lengthy & Standardized	Agile (dynamic strategy)
Backward-looking	Future-back
Annual or bi-annual	Continuous

We'll follow the StratOps approach. Now that we understand why we need a data strategy, what it is, and how systems thinking is essential - let's get to work.

[*]More on this in my conversation with Nicolas Averseng in the Interviews chapter.

Part I: Due Diligence

Why due dilligence is hard—Alignment with business strategy—Steering group—Current State Analysis—Via negativa—Maturity assesment—Pilotitis—Systems audit—Competitor analysis—Futures thinking—Ambition level setting—Gap analysis

Overview

> "Give me six hours to chop down a tree and I will spend the first four sharpening the axe."
> –Abraham Lincoln

How can we know where to go if we don't know where we are? This philosophical question applies not only to our personal lives but to data strategy. Assessing the current state might not feel like making inroads toward digital transformation (and in some cases might even open some unhealed wounds). However, it still is a foundational element of any successful data strategy project. I group all such activities focused on information gathering in the first phase under the umbrella term DUE DILIGENCE (DD). You might have previously encountered this term in another context, perhaps from the venture capital sector. There it roughly translates to "collection of information, in preparation for action", in other words: *before we can strategize, we need information*. It's not only arrogant but also irresponsible to make decisions or provide a strategy in the dark. Moreover, how would members of our organization or clients trust us if we immediately jump to conclusions without first getting the lay of the land?

Before diving into the nitty-gritty of DD, let's have a few words about mindset and honesty - two essential qualities of a data strategist. Keep in mind that from all elements of data strategy, DD is the most commonly assigned to an external party (such as a management consulting firm). There are several reasons for this. External consultants are primarily hired to alleviate pain

points; any organization would prefer to solve its issues (especially those of a more "strategic" nature) internally. What's behind this lack of trust in internals can be summarized in three reasons:

Political: Management would like to hear an external perspective on the current state without bias and politics. Internal team members often have agendas that might not necessarily align with the organization.

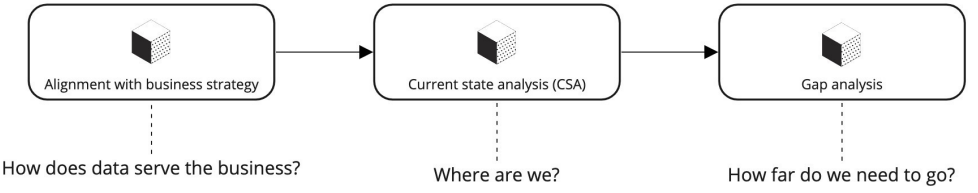
Practical: It's expected that a fresh look, combined with specific expertise, can break through barriers that were too complicated to overcome for the internal team. In Zen philosophy, there's a term called "beginner's mind": a viewpoint free of prejudice and baggage can provide a new, previously unseen perspective.

Economic: External consultants are indeed expensive (at least more so than regular, in-house employees), but the per hour cost might make up for itself mid-to-long term, especially if those consultants are deployed at critical junctions and are involved in strategic decisions.

> **Extreme Red Teaming.** An interesting idea I had ever since becoming a management consultant was to perform an extreme version of dealing with the political component of an external DD process. As external consultants, we try to be as honest with the client as possible. Still, in the real world, additional political considerations limit this particular aspect of communication (negotiation) with the client. Often, the measure of a good consultant is to secure repeat business opportunities. Sometimes, we cannot be as direct with the client as we would like since it might jeopardize the engagement. This can contribute to tension between our desire for honesty on one side and the internal political goal for project continuation on the other one. Being direct can sometimes damage any relationship. But what if we send several consultants, with the explicit

> intent to be as direct as possible, just for the DD part of the data strategy and then let them leave the project? In this way, we would avoid the potential fallout of radical honesty. In military circles (where a lot of strategy takes place), this is known as a "red team": a group of experts hired specifically to poke holes in your theories and work to improve them afterward.

Now that we are familiarised with the rationale behind DD, I'll show you how to do it. Its elements are shown below:



Elements of Due Diligence

As I wrote in INTRODUCTION, many activities in the framework are designed for sequential execution in mind - as shown in the diagram above. We begin by ensuring we understand the strategic business objectives of the whole organization are, and how those are aligned (or not) to those of the data strategy. Then we assess the current situation with the aptly named `Current State Analysis` (`CSA`). Building on top of the information gained there, we continue by determining the *gap* between the current state and where the company should aim for (its ambition level), with a `Gap Analysis` (`GA`). With all three elements complete, we have obtained all information needed to design a data strategy in the next phase.

Alignment with Business Strategy

Deliverables

- Documentation on current business strategy
- List of concrete steps for data strategy team to align further
- Filled RACI diagram for next steps

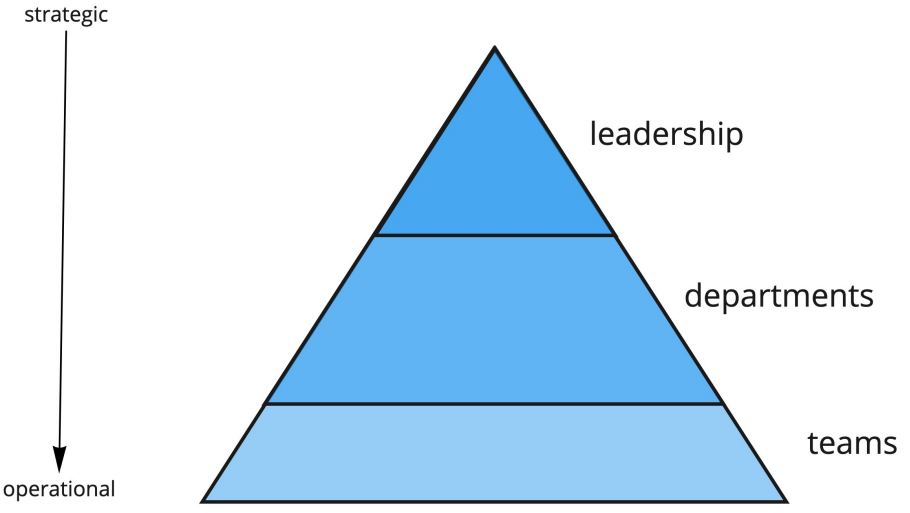
Data strategy initiatives are multi-department and often multi-year projects requiring significant amounts of resources and commitment. This makes buy-in from senior leadership essential. The most common concern from the leadership team is how the data strategy pushes in the same direction as the business strategy and how we can ensure it stays that way. Two elements of data strategy support this activity. First, the strategist needs to uncover the business strategy. After all, how are we to align to it if it's not clear to us? Second, a steering group is formed. The purpose of this group is to be involved in all phases of data strategy. This steering group is responsible for maintaining alignment between the stakeholders and operational team at all times by having a permanent seat at the table.

Uncovering the Business Strategy and Goals

This first element might sound obvious, but for large organizations with multi-layered internal structures (visions, departments, and teams), identi-

fying true business plans and objectives can prove more challenging than one may think. This activity requires a thorough look at both company's ongoing and planned strategic initiatives on many levels of the organization. We should not limit this investigation to the business units interfacing directly with data.

The best way to approach this challenge is to use a top-down approach, as illustrated in the figure below (this is also a tension diagram, imagine the individual trees at the bottom and the forest on top):



Strategic vs. operational gradient across the organizational pyramid.

We want to start this way because the complexity of business goals tends to increase from top to bottom organization layers as they become less strategic and more operational. We can approach this work from a consultant's perspective: in many cases, you are an outsider to the goals of other organizational units. For the learning curve for grasping the whole business strategy to be lower, we first start gathering information on the

top by talking to the C-level suite. Most organizations run several strategic projects simultaneously. Depending on their digital maturity, this number can be between one and more than ten [*]. Those initiatives tend to be long-term (longer than one year) focused. An example of a strategic business goal driving such an initiative can be "increase the market share of our products in the EMEA region from 7% to 10%" (Borek and Prill provide many specific examples[7]). If we were to stop the alignment between data and business strategy on this level, we would probably miss the mark - this is too vague and not actionable enough. How are we supposed to connect such a goal directly to our data strategy?

> Most of the data strategy work in DD is accomplished by conducting either **interview sessions** or **workshops**. Essential skills of a data strategist in this phase are asking good questions, taking notes, and facilitating more interactive sessions.

Next, we climb down a level and uncover the business goals of the individual departments. Let's take marketing as an example. Their goals should be informed by the C-level ones - but are probably more specific and focused on marketing projects and related operations. We need to talk to the *functional leadership*. These are the leaders of non-data departments and teams. In this case, the marketing leaders think about how they can support the overall strategic goal to increase the market share of products sold within a specific region. While doing this, they come up with their version of the overall business objective: *"Deploy a marketing campaign in EMEA which increases the conversion rate for the region by 2%"*. Successfully achieving this goal contributes directly to the overall one. Working directly with functional

[*] I'll show you how to establish this for any organization during the CSA section.

leadership dramatically reduces the complexity of the task since they are at the right level of familiarity with the work - at the same time, not too strategic, but also not too operational. These people are valuable allies for the data strategist.

And finally, we can also go to the most fundamental level of the organizational pyramid and look at what individual teams have selected as objectives. This would be more relevant if we design a data strategy for a smaller organization where the employee headcount numbers are in the low hundreds. The information obtained by aligning with specific teams is probably redundant for a larger one. That could change in the use cases part of the data strategy design, where goals and targets become more specific. We can repeat the exercise we did for the middle section and discover what the goals of individual teams are.

Now that we went through the different levels of the organization to discover their goals, I would advise looking at the business strategy pyramid at the right level, where the goals are not too operational but at the same time also not too strategic. With this information, we can proceed with the other parts of the data strategy. We need to refer to the organization's goals or the separate functional parts in several elements of data strategy, most notably in `USE CASES`.

Steering Group Formation

The second alignment step specifies the internal team leading the data strategy efforts and ensures that the necessary functions are actively and continuously involved in this process. As I previously mentioned, having a dedicated working group participating in the data strategy at all levels

is essential for progress. This can be difficult due to political reasons and the number of touchpoints (stakeholders) that data and technology affect within even a middle-sized organization. I recommend approaching this by filling a Responsibility and Assignment (RACI) matrix. Have a look at the template below:

Empty RACI matrix

The RACI matrix should involve a mix of the people designing the data strategy and the steering group members. In the first column, we add all the planned activities (elements) to create the data strategy, such as a `DIGITAL MATURITY ASSESSMENT` or `AMBITION LEVEL SETTING`. At this stage, it's crucial not to zoom into details; it's enough to specify the major components only. This way, we reserve space for adjustments (which will always be necessary) in advance. They can then be used during the data strategy design work. On the horizontal axis, we can then add the participants. We can label them in

one of the five categories below:

Responsible: Executing the hands-on work on the activities (interviews, workshops, design documents, and other data strategy deliverables).

Accountable: The go-to point of contact; this is the person to whom external stakeholders can address the questions and ad-hoc requests.

Consulted: Participating in the topic, but without daily engagement - someone kept in the loop, participating with advice.

Informed: Passive participant, but being updated on the progress through the steering group at semi-regular intervals.

Below you can find a RACI in its filled form:

RACI matrix

Owner / Action item	Anne	Thomas	Peter	Elena
Kickoff organisation		R	C	
Digital maturity assesment	R	I		R
Ambition level	A	R	Q	
...	...	...	...	...

Filled RACI matrix

An essential property of RACI is that it's constrained to one person per action item. This ensures accountability for the process (see the aside on

ownership below). The final step after this is to setup up the meeting and reporting structure for this steering group. Still, beyond saying that this should be on a semi-regular basis (for example, bimonthly), we leave it to the discretion of the data strategist.

> **Ownership.** An essential quality for teams tackling complex projects. We must ensure that ownership boundaries are clear to all stakeholders - everyone should know where their responsibilities and focus areas lie. This is challenging for projects led by multiple stakeholders and can result in conflicting priorities, scope creep, and politics in general. All items that we would rather avoid! This is why there should always be a single person held accountable for any data strategy element.

Current State Analysis

Deliverables:

- Systems audit documentation
- Digital maturity score card

Let's start with a thought exercise. We can look at every organization as a living organism: it constantly changes and evolves. It can get sick and then healthy again. Now, imagine your first task in learning about the organization is to take a picture of it so that you can study it in the lab better. Even for an experienced data strategist (the "photographer" in this analogy) this can pose a challenge. Any snapshot we take is bound to become outdated quickly, and we're in danger of taking the photo from the wrong distance or at the wrong time. This analogy represents the two main challenges in conducting a successful CSA: *timing* and *resolution*.

> One of my favorite tools for decision-making is *via negativa*, popularized in Nassim Taleb's work[8]. It stands for "the negative path" in Latin. Let's illustrate this concept with an example. Remember the advice we would often hear from our parents, especially when we are just at the start of our life: get a good job, be a good person and work hard? All of us can relate to the feeling of frustration when exposed to such advice. The most normal immediate reaction is rolling our eyes. We think: "Sure, thanks for

> the advice, but isn't this obvious? How is this actionable?"
> Interestingly, we *can* derive actionable advice here. We need to flip the sentence. Instead, let's define a "bad" job or bad "person". Those concepts are much easier to define than the positive ones - everyone knows what *not* to do. Maybe don't get a job that does not develop your skills, which doesn't have a good mentor available, or a tedious one. If we just approach our parent's advice this way, by just avoiding obvious mistakes, we can set ourselves up for success in the future. Via negativa is a method of flipping a positive statement into a negative one to get actionable insights.

So what would happen if we apply via negativa to the definition of CSA: what is a "bad" CSA? Let's say we take our snapshot at the wrong *time*—a typical case when the strategist jumps straight into details after the start of the engagement. As Robert Galford thoroughly illustrates in his book, *The Trusted Advisor*[9], the fundamental basis of trust needs to be established before you become technical and concrete. If you rush too fast into the current state exploration (remember - this can be a painful topic), you might succeed in "taking a photo", but question marks will surround how much it represents the truth. Jumping to immediate conclusions can be error-prone, since much of the information you gather early on is probably obfuscated by layers of politics and complexity.

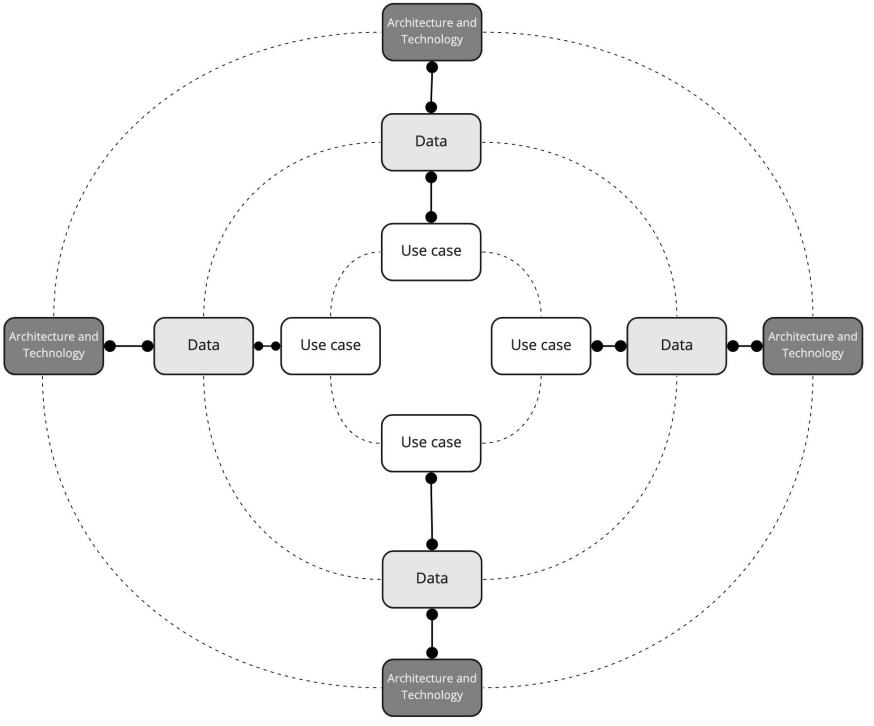
Now, how about the second dimension of CSA? What if we take the snapshot at the wrong *resolution*? This means we're missing the forest for the trees (again, remember the tension diagram from the INTRODUCTION). This can be an easier mistake since a careful balance is required. On one extreme, the strategist might spend too much time on the operational end of the work - such as investigating daily practices and technical implementation details. On the other extreme lies conducting a CSA with the executive suite only -

focussing on the more strategic and high-level view, and ignoring the rest. A good CSA requires a balance between both views, hence the constant need of a data strategist to adjust what they pay attention to.

This is what a CSA is all about - taking a picture of the organization at the right time and the right resolution. Now let's dive into the components of a thorough CSA.

Systems Audit

To deliver a comprehensive assessment of the organization's data capabilities, we need to audit several vital areas - the systems of interest:



A 360 degree view of CSA target systems

You can look at the current state of an organization as an onion, whose layers we need to peel off one at a time to obtain the complete picture. As with many other concepts in this book, those layers are closely intertwined, and the borders between them can be blurry.

Activities format. Most of the activities in CSA, and many in DESIGN are implemented with two methods: interviews and workshops. The former when you want to gather quick information or when the element is of low complexity, and the latter when you need to dig much deeper to uncover information or make a decision. There are many resources to help the strategist improve in those areas, for example, this great Harvard Business Review article on asking questions. Further insight is presented in USE CASES on the topic of workshops.

We can split the CSA target systems into three discrete types: use cases, data and architecture, and technology. A dataset and architecture, and technology support each use case. Perhaps differently from how we would peel an onion, we first need to start with the center. This is because the use case is the fundamental value-generating unit of any data project, and if we put it at the center of our work at all times, we can keep the focus on delivering value. A second reason is that any change in the use case can have cascading (and sometimes unpredictable) effects on the other two system types (a topic covered in more detail in INFLUENCE CASCADE in DELIVERY).

Every system type needs to be audited. I'll provide a list of motivating factors, a brief description, and potential questions that need to be asked or clarified during the interviews and workshops.

Use Cases Audit

As a first step, the data strategist must uncover what data (and related) use cases are currently pursued across the organization. Those, together with new ones, will represent the organization's data portfolio (I'll go deeper into how to manage this in PORTFOLIO MANAGEMENT in DELIVERY). It's not

guaranteed that those same use cases will be worked on as part of the data strategy - the scope is often changed during the DESIGN. Still, knowing what is currently being pursued is crucial since it might need to be stopped (to conserve resources), improved (if it's a valid use case), or provided necessary information and code for new use cases.

Question	Example
What use cases are pursued?	Topic modeling for customer support data
What technologies are used in each?	Python, LDA
What architecture components are used in each?	S3, Glue
What datasets are used?	CRM data from SalesForce
Which people are involved?	J. from marketing, A. as a customer support manager
What are the budgets?	EUR 35000, 2 FTE[^fte] over two quarters

Deliverables format. While for activities, you are mainly limited to two: interviews and workshops; in terms of deliverables, the options are many more. The most traditional result would be a slide deck, and this is a deliverable commonly expected by most business stakeholders. Additional and often more suitable formats for your work are wiki pages (such as Confluence or SharePoint), boards (such as Miro), Jira roadmaps, and many others.

Data Audit

Now we can dive into the oil that powers the use cases, the datasets themselves. A word of caution is required here. If you remember what we

discussed about the data strategist's attention earlier, this is one element that can drain the attention of the data strategist if left unchecked. There is so much information here that if the data strategist wants to compile an exhaustive report on it, they can waste much time - even in the case of relatively small and homogenous datasets. The challenge then lies in deciding the appropriate assessment for your specific case.

The data strategist needs to ensure that necessary data is not only present but is also of sufficient quality for use in the use cases. Fortunately for us, this is not an entirely new problem - there are quite a few established frameworks focused on auditing data. The two popular frameworks for data audits are FAIR and the 4Vs. I'll refer you to read on them separately and instead provide an example of a data audit below:

Question	**Example**
Who are the data custodians?	Marketing department
Who are the data consumers?	Operations
Is there a data dictionary present?	yes, but partial, covering just 40% of the fields
What is the data quality?	30% of the rows in the aggregated dataset are missing

The answers in the example are too simplistic in this case; this is for illustration purposes only. Typically, especially for the data quality part, you will have much more to fill in, and the deliverable probably won't fit this table but be a whole document instead.

Architecture and Technology audit

The final audit we need to conduct to get a complete CSA is that of the technology and architecture. Those are the elements that power the use cases, and often a lot of the make-or-break issues occur here.

Question	Description
What are the data source systems	Salesforce
What are the data storage systems	flat files on S3
How much is on cloud versus on-premise	40% on cloud
What languages and frameworks are used	Python, Scala, Java, Keras
What are potential issues with security and compliance?	RPO and RTO are not set, only weekly backups

Same as the other audits - here, you usually would go much deeper. For each of those items, it's also important to note any potential concerns or upcoming plans associated. When working through the use cases in DESIGN, you'll need to have those at hand.

Finally, all those audits can help us do the DIGITAL MATURITY ASSESSMENT.

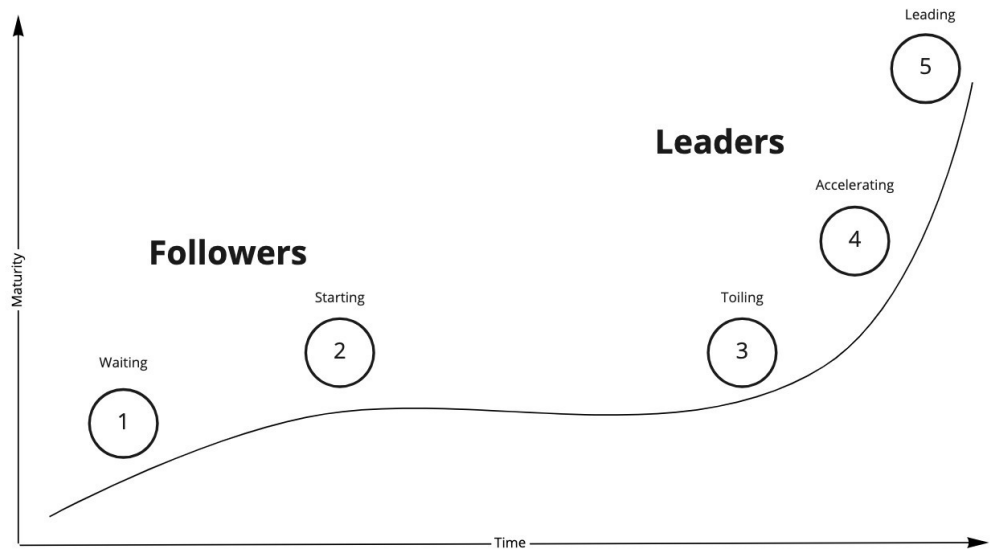
Digital Maturity Assessment

In many strategic engagements, an organization's digital transformation is often presented as a journey (to the top peak of a mountain[*]). Such a viewpoint certainly makes sense if you think deeper - this process is rarely

[*] This and other useful analogies for data strategy are covered in DESIGN.

instantaneous and much more likely to be incremental, with long periods of steady climb interspersed with sparse jumps of progress. It also makes sense to view a company's different stages and maturity development throughout this journey. To provide good data strategy advice, we need to use the knowledge we have obtained in the last two elements (ALIGNMENT and SYSTEMS AUDIT) to place the organization along this journey. This element is called DIGITAL MATURITY ASSESSMENT (DMA).

Before we decide where our organization currently stands, look at the diagram below to see the different categories of organizations.



The different types of organizations are sorted according to their maturity level over time.

It's not surprising that the journey is not a straight line, but why this shape in particular? I'll show you by describing what the five stages stand for. We'll go through the lowest maturity level to the highest:

Waiting (1)

- Data are seen as only a by-product of technological processes;
- No organizational functions responsible for data;
- Little awareness of the possibilities of taking advantage of data assets;
- No forward-thinking investment in technology and people regarding

data initiatives;

Starting (2)

- Understanding the necessity for change;
- Appreciation of data as an asset;
- First initiatives looking to exploit data potentials;
- First roles and ownership established for data projects;

Toiling (3)

- More significant investment in data and people;
- Initial pilot projects deployed, but in isolation;
- Dedicated teams established for data;

Accelerating (4)

- Broad up-skilling efforts established for data literacy company-wide;
- Movement from *pilotitis* (I'll explain this term in a bit) to not reinventing the wheel in the delivery of data projects;
- Data are seen as a product;
- Company-wide understanding of data assets and AI;

Leading (5)

- Known in their industry to be at the front of data technologies;
- Development of new methods for data;
- Development of new frameworks for data;
- Large and exponential increase in revenues from data, reinvested into itself;

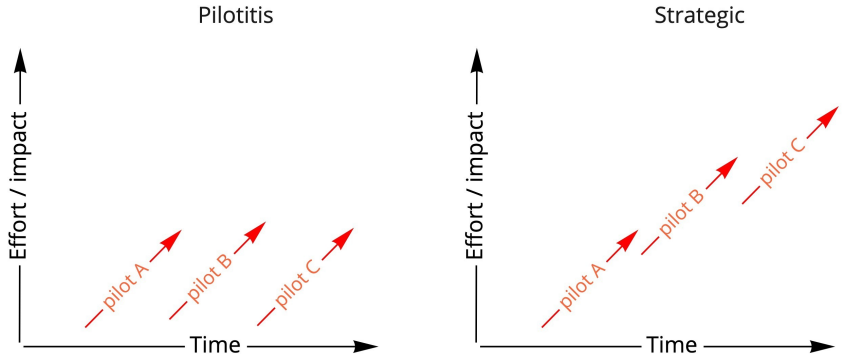
> DMA's have been around for quite some time, and many consultancies offer comprehensive solutions. My recommendation would go to appliedAI's tool. Another great source is the DMBOK book[10], the Data Management Maturity Assessment chapter. As a general rule of thumb for this work, you will always need to devise a scoring mechanism.

Based on those criteria, you can score and classify the organization. In this work, you'll need copious amounts of trust once again.. In most cases, the organization won't be a leading one - if it was, it would already have a designed data strategy at its disposal. Thus you'll find yourself in a situation where you need to communicate that the organization falls short of the ideal. Nobody wants to hear this, least of all senior executives - and those are often your target audience. To cushion this potential blow, you'll need to demonstrate how being honest with the situation and looking at the facts is the first step to making progress. With the right strategy, even giant corporations can achieve dramatic turnarounds - so it's not all doom and gloom. Those goals are your North Star[^north_star].

> **Pilotitis**: A term that I'll often refer to in the book, *pilotitis* describes the propensity of organizations to commit to individual, isolated pilot projects only without contemplating the need to build on a solid foundation and

scale. Data products are worked on in isolation. So what are the causes of this condition? First, it's the *easy* way (more on this in the LEAN DATA and SOFT AGILE in DELIVERY). No need to establish a complex organizational structure; make a small team of three to five people, and off you go. Also no need for complex IT or data architecture changes. Second - it's cheap. Doing things at scale requires an investment, both in people and technology. And third (perhaps most insidiously): it's *fun*. It plays to the human susceptibility to the Dunning-Kruger effect[11]. We feel accomplishment and progress early on, but this will not last.

A comparison between an organization suffering from *pilotitis*, versus one which has a strategic approach to data projects is visualized in the figure below:



Pilotitis vs. Strategic approaches

Pilot projects are often low in effort (also in impact). Perhaps more crucially, as time goes on, such work does not build on top of each other. Granted, it requires a more significant upfront investment to structure work properly. Still, the impact will also be much higher - to the long-

> term satisfaction of the implementation teams and the executives. Let's illustrate this point with a practical example. Imagine that in the first pilot project, you create a data pipeline. For example, perhaps you decided to use AWS S3 to store the raw data, then AWS Athena to visualize it. Even if this pilot project ultimately proves unsuccessful, in the future, you might still reuse much of the same architecture (or at least just with minor *ad hoc * adjustments) for the next pilot project. This is much more productive than every time inventing something from scratch. Such project management needs to be strategic - looking at a wider angle at the use cases, further into the future, and better understanding the available resources.

After scoring the organization, we would know where it stands in the overall journey. Still, to obtain the complete picture, we will need to expand our context and look outside the organization. We need to look at what the enterprise's competitors are doing, and what the state of the art looks like.

Gap Analysis

Deliverables:

- Documentation of competitor state
- Documentation of ambition level

Let's set some signposts for the road ahead from different angles. This element will require us to move beyond the technical and work with our stakeholders on a strategic level again. We must be mindful of our balance here and strike a good balance between optimism and realism. Remember that data and AI work starts with the burden of high expectations, fear of the unknown, and escalating costs. We have the job cut out for us.

The gap analysis aims to set the ambition level for the organization and determine how far off that goal is. Two methods are essential for this work; let's tackle them one at a time.

Competitor Analysis

Analyzing the state of competitors is essential for any business strategy. Thus it's not surprising to see its importance for data strategy. As an initial analysis, the strategist can use the DMA we covered previously to see which

group the competitor organizations fall into. After this initial binning of the competing into its maturity state, one has to go deeper and try to answer the following questions:

Competitor analysis questions
What organizational structures does the competitor have regarding data?
How siloed are their data initiatives?
What data sources do they use?
What positions are they hiring for? What are the technology requirements for those? What are the salaries?
What are the profiles of people employed by this organization (for data roles)? Do they come from specific industries or university majors? What is their seniority?
Does the company have any open source technologies available?
Are they seen as thought leaders in the field, and if so - how?

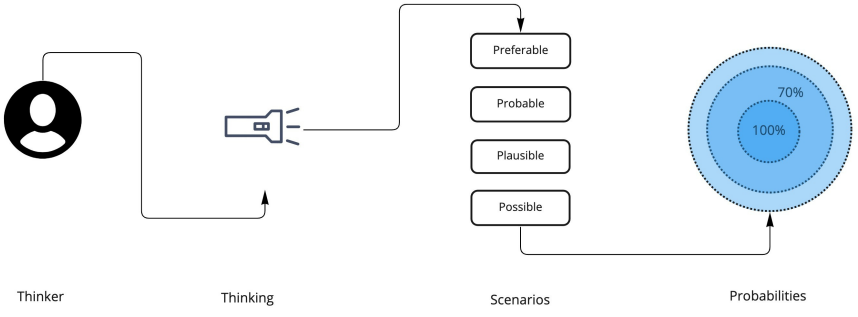
With the answers to those questions, we should be closer to having a better context. But we need to answer one final important one: what is state of the art in the domain? Almost in all cases, another organization is a leader (remember our diagram about the digital maturity journey). If there's no clear one, here we are allowed to take one from an adjacent field. We need to see where the signposts for the future are to benchmark our work accordingly.

Ambition Level Setting

Knowing the state of the art, we can go back to our organization and talk to the senior stakeholders to determine the right ambition level.

This is another vague and politically charged topic. A common thread in this book is that we are dealing with complex, evolving systems. This distinction is crucial because it helps us avoid making static, oversimplified assessments, leading to quickly outdated decisions. This pattern continues - circumstances are bound to change. Humans are notoriously bad at predicting the future, especially mid-to-long term, but valuable tools can help with that. Artificial Intelligence (AI) is one of those technologies whose impact is overestimated in the short term and underestimated in the long one.

A vital part of a data strategy is anticipating future trends and scenarios internally and externally. For this purpose, we can use a tool called the *futures thinking toolkit* (there is a multitude of other great tools, including McKinsey's Horizon Innovation Framework). This is a whole field of its own. Still, we'll take its highest-level tool and fit it to our purpose. Have a look at the diagram below:



Futures thinking visualized

This model can help us think about future scenarios. They all have different probabilities of occurring, which are not set in stone. There is an element of uncertainty in this work. Let's go through them one by one and provide an example. Remember, we can apply this futures thinking to several elements.

We can use it for the current initiatives at the company or the wider environment. My recommendation is to do both. From the CSA perspective, having the internal scenarios discovered is more useful in the short term.

Preferable: This is the scenario that we want to happen. It can be the easiest to estimate since most organizations have at least some idea about what they would like to happen. This one is probably the smallest in scope, including the least amount of information.

Probable: Discovering what is the most likely outcome can be very hard. It requires careful consideration of many factors and a deep understanding of the context.

Plausible: This is where the scope widens. Those scenarios can be deduced by extrapolating from the preferable and probable scenarios, albeit with minor adjustments.

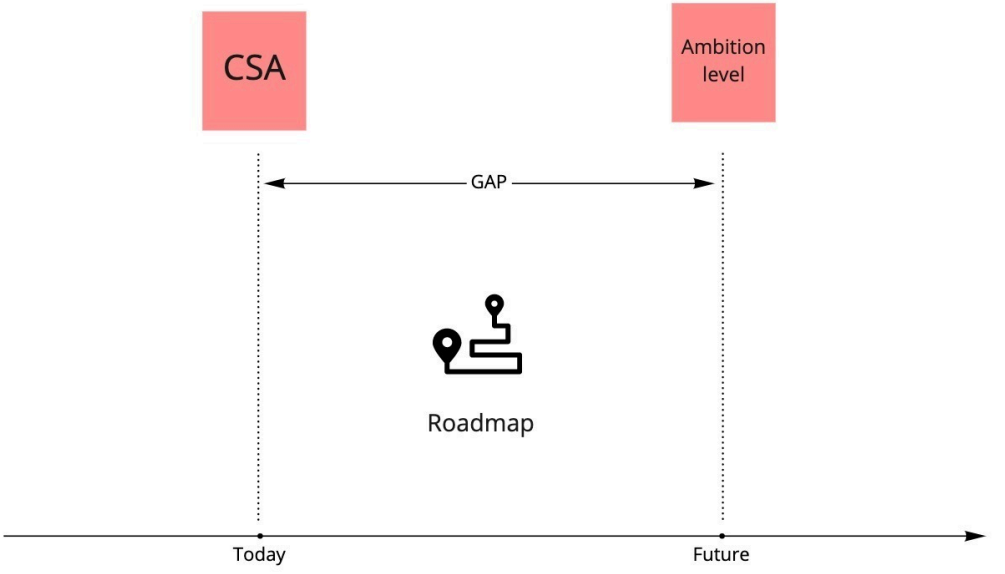
Possible: Here, we enter a more creative territory, and this scenario has the most extensive scope. The purpose here is to discover potential missed opportunities (similar to Data Thinking exercises we'll cover in the USE CASES section), and black swan events[12].

We're getting close. Now, armed with a solid understanding of where the future can take us, we can have another internal discussion with the senior stakeholders about where the organization's ambition lies regarding data.

Now, there can be several ambition levels organized on a timeline. Such increased resolution can help the strategy become more specific and move beyond the "we want to be the Google of restaurant chains" (this is a typical example of goals masquerading as strategies, explained very convincingly in *Good Strategy, Bad Strategy* by Richard P. Rumelt (also recommended in my interview with Martin Szugat)[13]. Much input for this step is required from the other elements from the CSA - the competitor analysis, maturity

assessment, the data audit and dictionary, and the futures thinking work. It can be helpful to position the company around the competitors based on the different assessments.

Now we can close the circle of the DUE DILLIGENCE phase and reveal the whole picture:



How data strategy fits between the signposts of CSA and the Ambition Level

Here we can see how the CSA and the ambition level setting elements enable the GAP ANALYSIS. And guess what's inside of it? The actual data strategy that we'll start designing in the next phase.

Summary

At this point, we know where we are and can proceed further in designing a confident data strategy. Let's recap what we learned so far in DUE DILIGENCE.

In the beginning, we established the key motivating factors behind conducting a thorough DD process, even before we started the first actual data strategy design tasks. This requires a lot of confidence (and trust), and working on the right level of abstraction while gathering information also requires experience. We can go a long way to set up for success if we establish the right level of trust within the organization. Any data strategy not aligned with the overall business objectives is doomed to fail at the start. We dive deep into the business strategy by conducting interviews with various organizational stakeholders. We establish a steering group using tools such as RACI to ensure we have an excellent team for the subsequent work. After establishing alignment with other business units, it's time to take a confident step toward gathering information in the CSA. We start by conducting a SYSTEMS AUDIT, followed by the DMA. Knowing where the organization is, their mid- to long-term ambitions (based on a COMPETITOR ANALYSIS) are then used in a GAP ANALYSIS. This can be our North Star, allowing both to set the expectations straight and establish motivating targets for the data strategy delivery.

This will form a great start and a solid foundation for the subsequent work! Next, we'll start designing the actual data strategy, building upon a solid foundation of information about the organization.

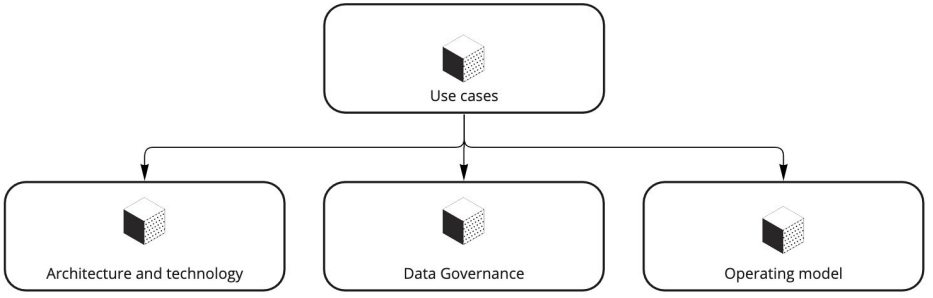
Part II: Design

Useful analogies—The Influence Cascade—Use case ideation, feasibility study and prioritization—Data architecture and technology—Data governance—Operating model

Overview

> "A system is never the sum of its parts. It's the product of their interaction."
> –Russell Ackoff

In the DD part of the book, we invested a significant amount of time and effort into gathering information on the organisation's ability (or lack thereof) of delivering value from data. This groundwork can already prove useful to the organzation as is: it can be used to take initial tentative steps in the right direction, even without further recommendations from the data strategist. Still, at this point there's little in ways of actionable advice, our deliverables have more of a diagnostic than prescriptive character, if you excuse the medical analogy. While we know the state of the organization, we haven't taken a single step in providing a strategy. This situation can be frustrating for the data strategist. Fortunately, the tangible fruits of labor become apparent in the design phase. Moreover, with good materials from the first phase, progress here will be swift and you'll be able to deliver value. Remember, while it's tempting to jump directly into designing the strategy, without DD as a basis, we might just as well copy a data strategy prepared for another organization. Without essential elements such as CSA, DMA and GA such recommendations would only based on gut feeling rather than facts and the organisation's particular circumstances. Shooting in the dark is never a viable strategy! Let's see what are the different components of a data strategy - as always we'll strive to keep them MECE:



Use-case driven design elements of data strategy.

The execution of elements is presented differently here, as compared to the DD part. The only constraint is starting with the USE CASES. With that element out of the way the data strategist can on the others in parallel. The reasoning for this particular sequence is described in INFLUENCE CASCADE in depth, but the main benefit is a constant focus on the practical value of data strategy - represented by the use cases that need to be eventually delivered. If we lose sight of that goal we could easily fall into a trap of spending too much time focusing on other elements. They are essential, but fullfill only a supporting role for successful use case delivery.

We begin by selecting the cases we want to build (or extend ones already in operation) in USE CASES. After this we commit to technologies and architecture to support their delivery in ARCHITECTURE AND TECHNOLOGY. The use cases also require data access, and that can be a challenging topic in larger organizations (due to compliance, security and infrastructural constraints) hence the need to design a DATA GOVERNANCE (in this element we'll rely heavily on the SYSTEMS AUDIT deliverables). And, finally, to expedite successful delivery, an effective and efficient OPERATING MODEL is designed: composed of blueprints on organizational structure in terms of roles, teams, projects and processes.

Keep in mind that no two data strategies are the same. The complexity of the task at hand ensures that there are few copy-paste solutions for delivering this work: every time you design a data strategy, you'll face challenges you've not experienced before. You'll need to mix and remix the elements to make them suit the company, projects, and associated to them people.

A note on assumptions. In my conversation with Martin Szugat (in INTERVIEWS) we covered why assumptions are frequent in data strategy work, and how their validity can only be tested during delivery. You'll need to be constantly aware of this as you design the elements of data strategy: you'll need to show a lot of flexibility in adjusting your recommendations during the third phase, DELIVERY.

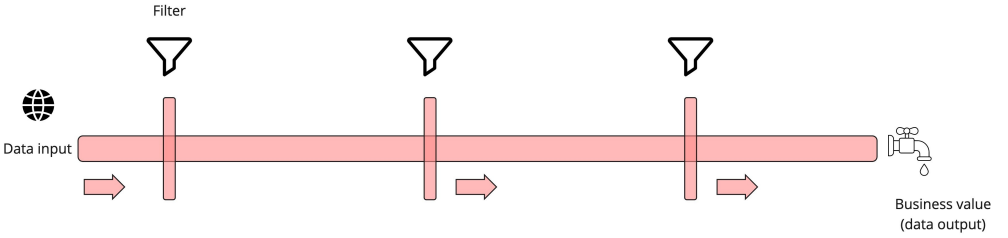
Before diving into the DESIGN elements I want to provide you with a good tool that served me well. To design a good data strategy, and communicate it successfully to the stakeholders, you can utilize an essential item of every good data strategist's toolkit: *analogies*. They can make the often opaque world of data more transparent for people with no or limited technical background. In the following section I'll cover some common helpful analogies in data strategy work.

Data Analogies

Talking about data projects can be confusing, even for experts. This is primarily due to the extensive technical terminology used in the field. When walking into a technical data meeting you'll often hear opaque concepts such as data assets, ingestion layers, data marts, anomaly detection models, and everything in between. It might seem challenging to find good analo-

gies for such a sweeping range of terms, but there are some good ones, tried and tested throughout many consulting engagements.

The Oil Analogy



The oil analogy

You might have heard this one: "data is the new oil" (here's also the opposite view, where that data is the new water: while it's true that organizations are sitting on piles of data, making them actionable is as big of a challenge as it's ever been[*]). There are several good reasons why this analogy has proven to be popular and pervasive in the community. First, it relates to the idea that data is firmly on track of becoming more valuable than oil. It powers not only our digital lives, but our workspaces, government and vital infrastructure. This influence can be seen on par as oil throughout the industrial revolution. Second, this analogy successfully covers the idea of data processing and enrichment (this is a topic into which we go deeper in `DATA ASSETS`, and the `BSG` concept). Similar to how oil needs refinement before becoming useful, data requires similar upfront investment of work before we can reap its benefits. And finally, data also travels through pipelines, gradually improving in quality and suitability for business use (becoming

[*] See the vanity projects from Joe Reis and Matt Housley's "Fundamentals of Data Engineering"

fit-for-purpose). The oil analogy maps well to data engineering pipelines, which are always at the core of any data architecture.

> **Mapping user journeys with data**. Business stakeholders are increasingly comfortable with another software concept, "user journeys", which is also helpful when discussing pipelines. In one consulting project I worked on mapping a target data architecture design to a user journey. This was very helpful for the customer to understand how the architecture supports the product and have a direct impact on the customer and how they use the data-powered digital products.

The Kitchen Analogy

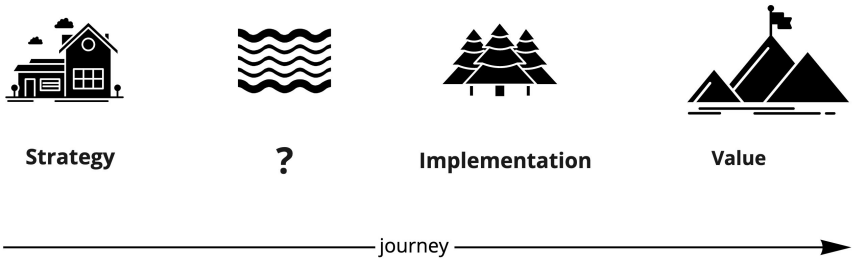
This second analogy became popular more recently. It was been advocated for by Google's Chief Decision Scientist, Cassie Kozyrkov[*]. This one can be more playful (and less environmentally disturbing) than the oil one. I found it particularly helpful in describing more AI-centric data terminology and processes. Here, for example, we can say that the raw ingredients represent data. The kitchen appliances describe algorithms. A recipe stands for a model, and finally, the product is the dish (this gives a new meaning to the term "serving predictions"!).

The Journey Analogy

The journey analogy is often used to describe digital transformation efforts (remember the DMA?). It depicts the journey of an organisation to derive

[*] Her original article introducing this analogy is called "Why Businesses fail at Machine Learning" and you can read it on here.

value from data as ascending a long path up a mountain, eventually reaching success at the peak:



The road to value

The arrow represents both the passage of time and associated effort to reach the final goal, the peak of value. In the beginning of this journey, we start at our well-built home city, where we design the strategy. Everything here is neatly organised in different buildings, connected by straight paved roads. A well-designed data strategy should look this way - clear, consise and specific. Nevertheless, to reach our final destination must leave the safe confines of our home behind and venture out into the wilderness. This is where the rubber meets the road. Here, what stands between us and the final goal is the "forest of implementation", where the use cases we designed commence their development cycles. This can be the most treacherous and difficult obstacle in our journey. This is why DELIVERY is a critical phase of a successful strategy. A half-good plan that is executed is better than a perfect plan which collects dust on some hard drive (more on this in my conversation with Nicolas Averseng). The question mark corresponds to the river between strategy and implementation - an invisible barrier where data projects and

products fail. You'll learn how to navigate this river in the third part of this book.

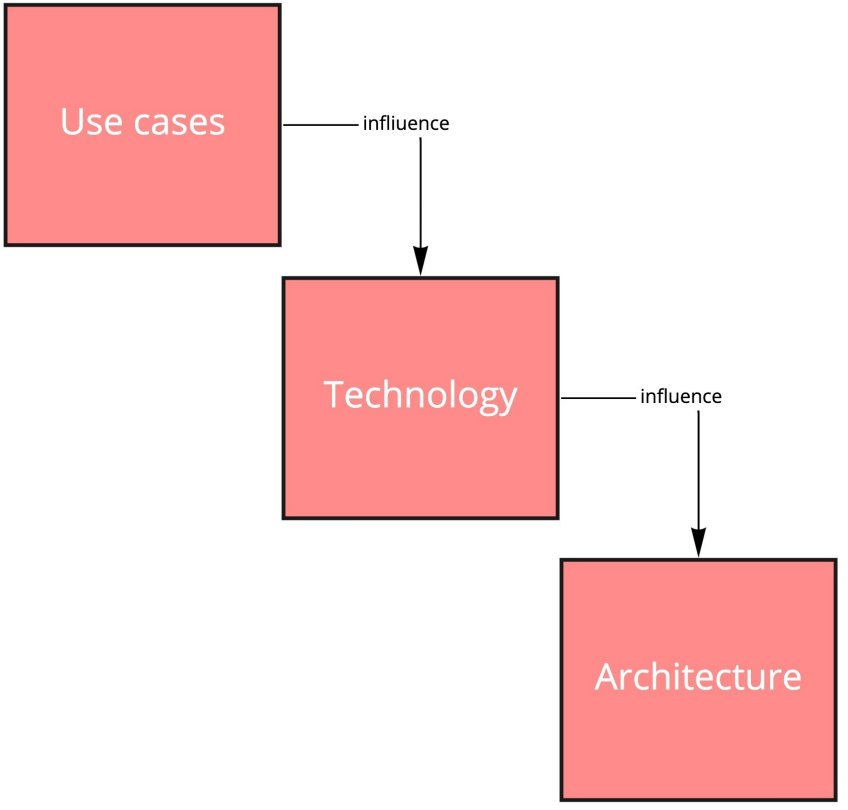
Now, armed with those analogies we can dive into the main concepts of data strategy design.

The Influence Cascade

During the CSA you should have already experienced how interrelated data strategy elements can be. Modifying the design in one area can have dramatic effects on another. If we don't consider such unforseen effects we might end up with a disjointed, confusing and unfocused strategy, that becomes obsolete in the first weeks of implementation. These effects are potentially exacerbated by the inherently cyclical nature of data work: we often need to do experiments, without a clear idea if they would eventually succeed in delivering value or not. We need to often operate on assumptions only[*]. In my conversation with Datentreiber's Martin Szugat, advocates for the use of an "experimentation" phase before commiting to implementation (equivalent to a lighthouse project, more on this in a bit).

I have termed this concept of interconnected, and sometimes unpredictable effects the "Influence Cascade". Changes in requirements are common, and any recommendation dependent on them needs to be adjusted everywhere downstream. Here's how this cascade looks like:

[*]This is also described in DELIVERY, where I talk about CRISP-DM and similar frameworks for agile data.



The Influence Cascade: changes in use cases influence technology, and changes in technology influence architecture decisions.

Imagine we're developing a use case for detecting abusive content on social media. Our initial idea might be to create a text classifier*, which is trained on a corpus of tweets and outputs a predicted class (for example abusive/non-abusive). We then proceed with evaluating and commiting to a technology stack supporting this product. In this case a good approach

*If you are interested in the different types of machine learning, such as classification have a look at this resource

would be to use Python (since there's a great number of Natural Language Processing (NLP) related open source packages in Python's ecosystem), and use a Support Vector Machine (SVM) as a classification algorithm. Once this is decided we can design the appropriate backend architecture for this tech stack to run on. For example, we can store text data in a NoSQL database (such as MongoDB), since it is a good option for document-oriented storage.

What is a tech **stack**? This is a common word to summarise the different major architecture and technology components of an organization. It is derived from the idea that most systems like this can be summarised as a hierarchical collection of layers feeding into each other - like a stack of books.

So far so good! But suddenly the business development team realizes that there are too many similar solutions on the market, and that we need to pivot. It turns out there's a niche available for the same product but focused on video data rather than social media texts. We become quickly excited again but will eventually realize that the whole stack needs to be redesigned from scratch (even if the problem statement, and target audience remain mostly unchaged). The fact that now we need to work on video data makes the technology tooling completely different (SVM probably is not a good idea for an algorithm, and neither is MongoDB for data storage). Now, storing the raw video files on AWS S3 and using Convolutional Neural Networks (CNNs) sounds like a much better idea.

You can see how any changes in high-level decisions regarding data products can have significant (sometimes unforseen), consequences down the line. At every step in the data strategy design we need to be mindful of the Influence Cascade, but since we are following a StratOps approach, this should be expected and budgeted for. This is the reason why we'll now

start by designing the use cases, and only then consider architecture and technology.

Use Cases

Deliverables:

- An exhaustive list of potential use cases (based on brainstorming sessions)
- A smaller list of use cases that are filtered based on feasibility and priority
- Metadata for the smaller list (i.e. what technologies and architecture elements are needed, what levels of data access and quality and others)
- A document outlining the budgets and resource requirements
- A roadmap ready for implementation

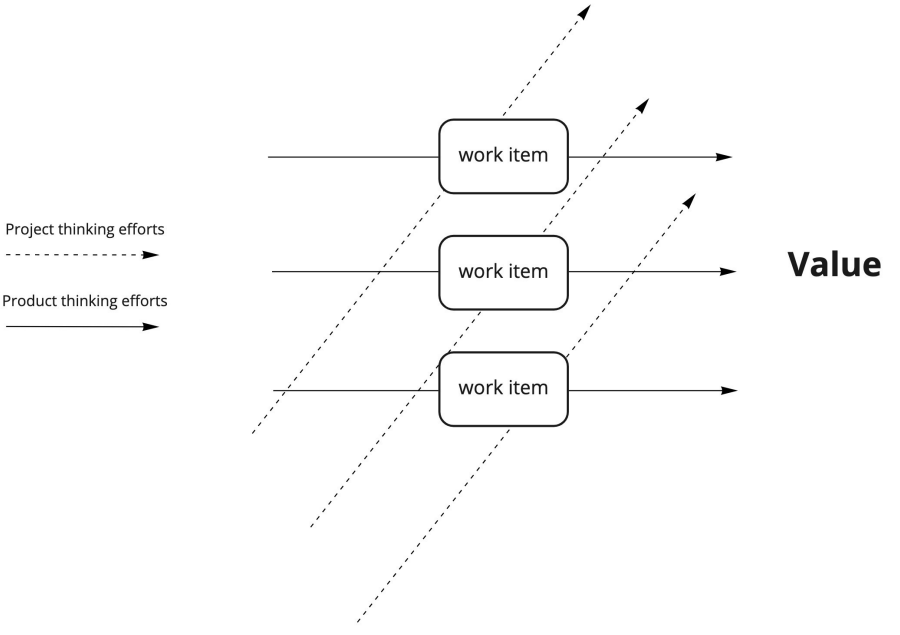
In my conversation with Martin Szugat I gained an essential insight into why many data initiatives remain unsucessful, and why that's hard to change. A big contributing factor is the lack of a certain *mindset*: product thinking. To understand it better we can contrast it with the prevalent way of thinking: project thinking, the differences should become clear:

Project thinking	**Product thinking**
Focused on process	Focused on outcome
Hard to measure	Easy to measure
Hard to design roadmaps	Easy to design roadmaps

The project thinking mindset dominates in more inefficient, bureaucratic teams. There, a large part of the workforce tends to focus on optimising processes and workflows, rather than immidiately contributing with value. Being always focused on the bottom line will ensure the team pulls in the same direction and builds the right features.

Additionally, as the old adage goes,you can manage what you measure. The success of a “project” tends to be much harder to measure, as compared to that of a “product”. And finally, both of those benefits or product thinking allow for the ease of creation of roadmaps. It’s much easier to plan ahead when you are building a concrete product, rather than abstract processes.

Let’s have a look at a visual representation of those differences:



Project versus Product Thinking visualized.

Product thinking tries to get to value as fast as possible with optimal use of effort, while resources can easily be unfocused in a project thinking setting. Needless to say, adopting a "product" mindset is a better idea and I'll be approaching the use case design from this standpoint.

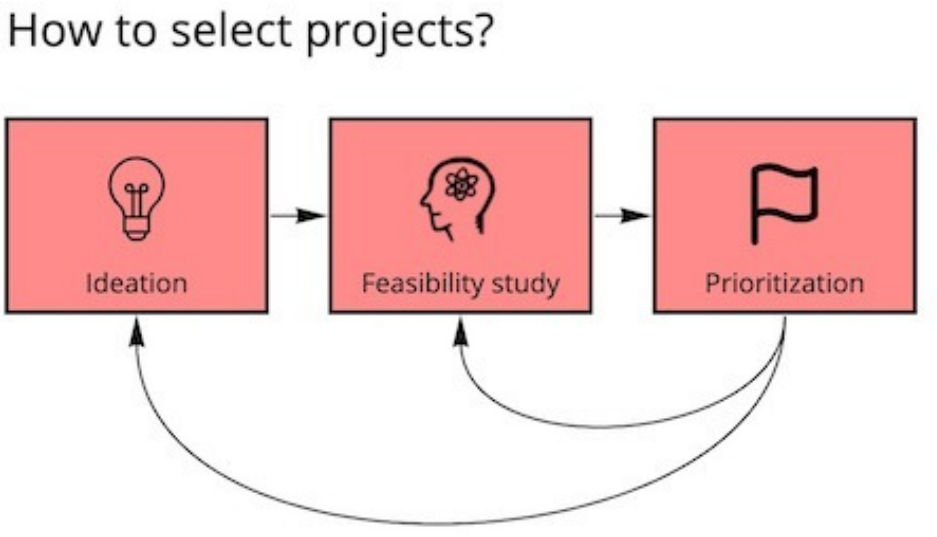
> **Design thinking** is essential for developing a good product-focused mindset. A good definition is available on interaction-design.org: *Design thinking is a non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test*. They go further and define five common verbs that are associated with the process: Empathize, Define, Ideate, Prototype and Test. I would go a step further and argue that design thinking is more than a process: it's a way of thinking, which is particularly powerful when reasoning about difficult to grasp problems (the black boxes we covered in `SYSTEMS 101`). Another term for those is "wicked problems".

Coming up with ideas on what to do with data is fun, but the data strategist should always be aware of the danger of *pilotitis*. While it's easy to come up with many ideas, the challenge lies selecting the ones with the highest impact, and also what are feasible for the resources available. With unlimited resources (including time) we would be able to build any and all products we want. Unfortunately this is not the reality we live in, and we always need to operate under constraints (which can be even more limiting in the case of data products, due to raised expectations and often larger budgets).

> **The cold start problem**. Since the scope of a data strategy is often expansive, getting started can be a daunting task. Stakeholders might be reluctant to make an extensive investment before seeing tangible results.

> In such situations it's a good idea to run a "lighthouse project" first. The task is to start with a simple to do, yet impactful use case, to demonstrate the potential value of data. In the optimal scenario that this project turns out to be a success, the data strategist can use both the learnings, and new-found trust to proceed with other use cases. Additionally, there's a high likelihood that at least some of the components developed for the lighthouse project can be reused further down the line.

Our end goal is to determine *what* we should build, and support this decision with estimates of its value, technical feasibility and resource constraints. To achieve this we go through three steps: ideation, feasibility study and prioritisation:



Project selection

The steps should be executed sequentially, but as you can see from the

feedback arrows between them, sometimes we need to go back and forth. For example, we might come up with a great idea - one that is also feasible from a technical perspective - but it might still fall short of the final prioritised list due to resource constraints. In that case we need to go all the way back to the ideation phase to gather new ideas. While it may seem like a setback to go retreat to the drawing board, in reality we save ourselves a lot of pain down the line (data products tend to be dangerous when not executed properly) and avoid falling victim to sunk cost fallacy.

Ideation

While it might be true there's no shortage of ideas in data work, *good* ideas are hard to come by. What do I mean by *good*? And, perhaps more importantly, how can we compare two (or more) data use case ideas? This question might seem vague and subjective at first, but it turns out to be surprisingly quantifiable. But before we go on to select ideas, let's first learn what is the best way to gather them: workshops.

Most of the deliverables in DD can be completed by conducting a series of workshops and interviews. In DESIGN, the amount of interaction with non RACI team members is reduced - we are often left to our own devices to design the data strategy. One notable exception, however, is the use case ideation element, together with the closely associated ones (feasibility study, impact assessment, and prioritisation). Those are ideally done in a wider group in a workshop format.

> What *is* a workshop? A colleague of mine from my consulting days used to joke that this term is so overused recently that it mostly just means

> a longer meeting. To me, a workshop is a *collaborative* meeting, with a clear agenda, objectives and facilitation, focused on a topic that can't be resolved in any other way.

Since this is one of the most important tools in the arsenal of a data strategist, it makes sense to spend some time describing its key elements and defining attributes.

Format: Workshops are typically structured as in-person, or remote (via digital whiteboards such as Miro, Mural or Google's Jamboard) meetings. These can be split into chunks of several hours each, if needed across several days. Since workshops can be mentally (and sometimes emotionally) intense, frequent breaks are necessary. A standard workshop session is around two hours, with two ten minute breaks.

Participants and **key roles**: As a rule of thumb there should be around five to seven atendees. Any more and the session can become difficult to manage (also schedule). Any less, and we might not gather all the potential view points and ideas. We are aiming for a diversity of opinions and skills (the more T-shaped people[*], the better). The role normally assumed by the data strategist is that of a *facilitator*. You can think of this role as a referee in a football match - their job can be deemed successfully executed when their presense goes mostly unnoticed. Since this person is mostly engaged in leading the workshop, an additional person is required to take notes and document the proceedings (you will refer to those often later). For client projects this role is oftentimes assigned to the data strategist, while the facilitator is nominated internally. The final key seat at the table is reserved for a decision maker. You should always try to get this person in the room,

[*] People possesing both a broad skillset, and deeper specialisation within a single area.

or at least have someone with a clear decision-making mandate delegated to them present in the sessions. The reason is that workshops are often decision-making focused, and can be disruptive (hopefully in a positive way) activities - thus often involving political, comittment and budgeting issues. In light of this, to mitigate against potential future roadblocks, make sure to have the decision maker in the room. This will also increase the amount of trust, and you'll need a lot of it during the DELIVERY phase.

Elements: There are hundreds (probably thousands) of workshop designs available for you to pick from. A quick look at the Miroverse (Miro's user-generated, curated collection of workshop templates) can show you that. This can seem overwhelming to a novice workshopper, but rest assured - the most essential elements of a workshop, similarly to those of a data strategy, can always be reused and adjusted for other purposes. If you learn how to do those select few, you would be able to adapt and deliver in various situations (often the workshop dynamics and goals change as you go!), and able to customise workshops to specific use cases quickly[14].

For this work you can use many of the deliverables from the DD most importantly the data and use case audits. There are many approaches to ideating data projects, and from my experience methods from design thinking transfer very well into the data domain[*]. We are going to use a workshop format for the ideation sessions, so make sure to read the aside. Let's go through the main elements of an ideation workshop:

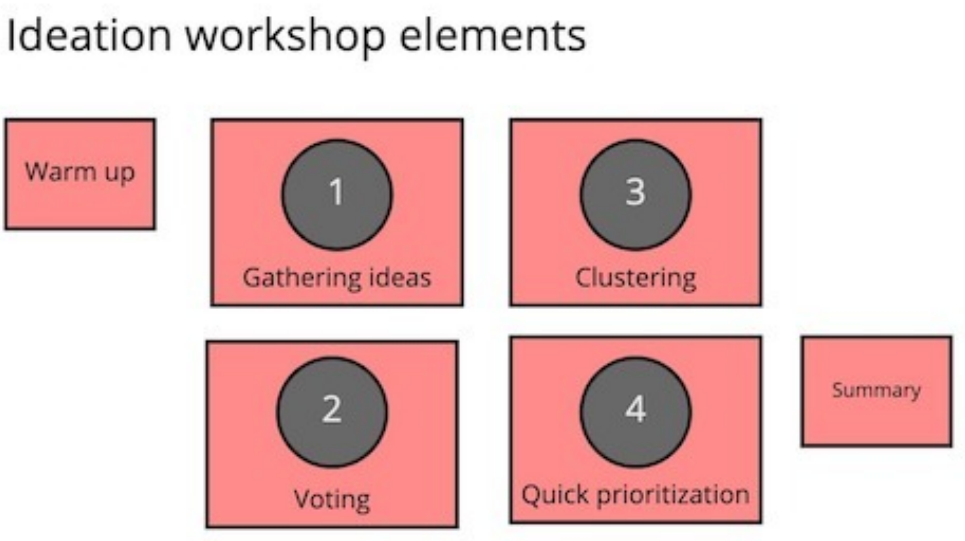
Goals and deliverables: At the end we want to have enough materials gathered so we can make those ideas happen. The deliverables need to be informative enough for technical requirements gathering and generation of design documents and requirements. The most important final deliverable is a list of *prioritised* potential products/projects.

[*] This has now growing into the field of "data thinking".

Format: In this case a larger number of participants is preferrable. After all, we do want to generate ideas - and the more people are there, the more ideas we can discover.

Participants: The most important property of the participating team is that it's diverse. People from different departments (and seniority levels) can provide valuable ideas. As we mentioned before, you would want to have a decision maker involved. For data projects in particular it's also important to have people who would be the consumers of the data products present as well.

Elements: It's essential that all elements of the workshop are clear to understand and have a good flow between them. Have a look at the diagram below for the structure of a sample data ideation workshop.



Ideation workshop elements.

You shouldn't underestimate the warm-up section. This is absolutely necessary for people for several reasons. One, they need to get used to the

software (if you are using a digital white-boarding solution, such as Miro. A second reason is for them to get focused on the task at hand. And finally, the third reason is that they go into the workshop with a positive mood (this is especially important for more politically challenging workshops, such as the ones in the CSA). There are a ton of warm-up templates that you can borrow, normally called "ice-breakers". Check out the resources in the Miroverse again for inspiration.

> When working in uncertain environments, a good way start and get the lay of the land is to do somehing called **anchoring**. You really just need to start somewhere, even if eventually during the design process you end up somewhere completely different. By working through such challenges you will obtain the information you need. A good anchor for use cases in my experience is first seeing what human workers are doing - generating a map of their processes. After that's done, you can have a look at any automation potentials and their respective values over this journey. Start automating a human-led process first.

After the warm-up the participants gather ideas with sticky notes. This activity should take around 15 minutes - it's safe to assume the most important ideas should come out rather quickly. If you attempt to spend more time on this, the focus of the participants can wander as well. The third session is where the participants vote on the ideas gathered. There are different methods for voting, but I would recommend anonymous voting where each participant has three votes at their disposal. This can take around ten minutes. In the final step the ideas are clustered together into similar topics, or common challenges they address - here the workshop data strategist needs to be quite active and ask a lot of clearing up quesions..

This is important to focus the ideas a bit further and eliminate any possible redundancy. And finally, the clusters get prioritized by using a tool such as a two-by-two Impact-Effort matrix (more on this later in PRIORITIZATION).

In conclusion to the ideation workshop a good practice is to summarise the results and share them with the team, while the data strategist uses them to contunue the use case design.

Feasibility Study

The ideation part is one of the most exciting elements of data strategy. There we are able to work with different people and show them what's possible by using data. There should be a lot of excitement generated! We'll need this in the following work where things start to become more complex. Technical projects are notoriously difficult to estimate in terms of work required and probability of a successful outcome, especially from a business point of view. Some features can be seen as hard to do, but very simple in practice, and the other way around. The data strategist now needs to take the generated list of use cases and determine to the best of their ability which ones are feasible, and which ones not.

An example that I often use to illustrate how some ideas can seem easy to implement at a first glance, but are actually tricky comes from the field of web development. If we are looking at the visual features of a digital product, let's imagine an online banking website, such as PayPal, we might think that adding buttons to an interface is relatively straightforward. In many cases this can be true, but often times we would underestimate the actual complexity even of this simple task. Sure we can add the button, but the wiring behind it can be tremendously complex - it really depends on what

this button should be doing.

Let's get back to the realm of data products. There are two main factors which can influence whether a given product is feasible. We need to look at each use case that made the ideation list from two different perspectives.

Architecture and Technology Constraints

Estimating the feasibility of a product (or project) from a technology and architecture perspective can be challenging. At this point stakeholders often face the cold start problem that I covered earlier. To be able to estimate topics related to technology, the organization needs to have established technology in the first place. Moving beyond the buzzwords and blank statements, such as "we need to develop an AI powered product for storage warehouses" requires a lot of technical know-how, and even more importantly - real world experience. Both are hard to come by in an organization of low digital maturity. This is one reason why consultants can be very valuable in estimating the feasibility of a project, especially from a technical perspective when constraints need to be estimated.

You should note that here how architecture and technology go hand in hand. All too often feasibility studies conducted within organisations neglect to take into account architectural constraints. Those can be grouped in two groups:

Legacy projects: Most organisations have an existing codebase, which can consume, process or create data. Any new initiatives need to take this into account, and build on top of it (or redo some of the components). This can impose severe constraints on your work. Even if there's no codebase related to the data itself, there are often other systems with which your newborn data systems need to interact.

Greenfield projects: In this case most of the technical constraints are related to stability and performance, for example concepts such as RTO and RPO[*].

Despite huge advances in cloud-first infrastructure and associated tooling, making this scale and work properly in data projects remains a challenge. This is showcased by the abundance and salaries of positions of data engineers, cloud engineers and data architects - their skills are probably the most sought-after in the whole data industry. In terms of technology, we can be constrained by what is currently available in terms of the on-the-shelf tools on the market. A good example of this is trying to build self-driving cars in the 90s. At that point people already started to see the promise of neural networks for Computer Vision (CV) tasks, but nobody would be able to use them even on large amount of images, let alone on real time video object detection in a moving vehicle. Other technological constraints also affected this use case, for example the lack of good internet speeds and coverage to transmit all the data (and support the latencies required by the use case), or the bare computing power in the car itself. Nowadays with 5G networks and edge computing those constraints have been removed and the self-driving car use case has become now feasible.

There are many examples how seemingly innocuous requirements can derail data product development. Unfortunately, it's widely thought that improvements in performance scale linearly with hardware. The unforgiving reality is that often we would require to rethink the whole data pipeline and architecture to just shave off several hundred millisecods in application latency.

[*] RTO stands for Recovery Time Objective (how long can the application be down) and RPO for REcovery Point Objective (the time span database backups should cover).

Folks at the large tech-first companies such as Google know that many of the problems that they need solving arise just from the sheer scale of datasets that they had to work with, which are only growing further with time - sometimes exponentially so. Data scientists and engineers in such environments are forced to invent new ways of working with large amounts of data, completely throwing away the old ones - from inventing the map-reduce algorithm to Kubernetes. The data strategist should be aware that some use cases might require custom architecture development, which cannot be simply purchased as an on the shelf service, and would require even additional maintanenance.

Resource constraints

Now let's go into the second large influencing factor in a feasibility study - resources. This is such an extensive topic, that deserves its own book, but I'll cover the main points. Beyond the obvious items such as laptops, servers, internet speed, office space (and coffee/pizza availability), we can safely assume that the largest operating cost and complexity in data projects comes from people. We can further break this down into their skills, experience, and salary, with the last one also a moot point, which can readily be bundled into the other operating costs of the business. Some use cases, especially ones which require a complex orchestration of services, and managing mission-critical infrastructure (such as the aforementioned self-driving cars), can require not only knowledge of technologies and frameworks, but also just pure on the job experience. More senior engineers would not only know *what* needs to be done, but more importantly *how*. By working with experienced engineers, you'd be able to avoid a lot of technical debt and unnecessary complexity in the code. The lack of such people can be a tremendous constraint on what projects are feasible to pursue. The same goes for skills themselves. If the current stack in the organisation (that we

discovered in the CSA) is built on top of an arcane or niche technology, such as KOBOL or Elixir (as is often the case of legacy banking systems), it would not be feasible to switch to Python and readily take advantage of cloud services for that matter, since most of the SDKs[*] for AWS, GCP or Azure are in written in more common scripting languages, such as Python.

Data constraints

And finally, whenever we are making a data-centric product we are always limited by data. Here we can mostly refer back to the data audits in CSA: presence of data (are all measurements we need available), data size, data cleanliness and data bias. To that we can also add one extra dimension how data can make or break a project - the data format.

In my O'Reilly book, *Python and R for the Modern Data Scientist: The Best of Both Worlds*[15] I made the argument that the format of the data is essential for what we can do with it. This concept is also covered by some great data thinking-inspired projects, such as the AI Ideation Cards (by AIxDesign). Let's see several different data formats to understand how they constrain us. Note that this list is by no means exhaustive, but these are by far the most popular data formats (beyond the normal tabular numeric format that everyone is used to at this point):

Text data: Collections of documents, such as tweets, blog posts, news articles, scientific articles, PDFs of invoices and others.

Image data: Satellite imagery, celebrity photos, animal photos. Videos can also be part of this category.

Time series data: Anything with a timestamp on it, for example financial transactions or sensor data.

[*] Software Development Kit: a packaged collection of sofware tools that help developers write software for a specific system or purpose.

Spatial data: Anything which can be mapped on a coordinate system (i.e. elevator locations for subway stations).

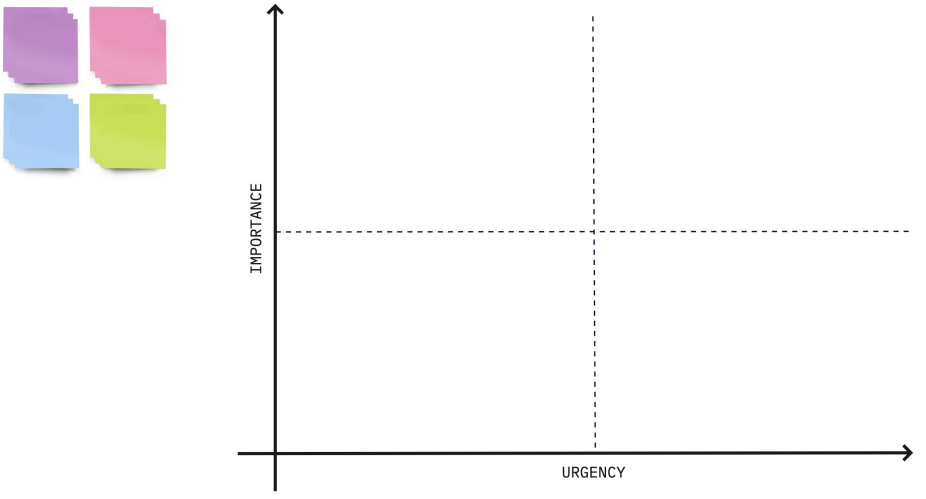
> **Compliance as a constraint**. This is an additional constraint to which we need to pay attention when designing the `USE CASES`. This is of heightened (and often crucial) importance in highly regulated domains, such as healthcare, the military and transportation. We need to ensure the use cases follow ethical and legal guidelines.

All those different formats require very different technologies and supporting architectures (another example of the `INFLUENCE CASCADE`). Additionally, for more advanced use cases such as machine learning, many of the algorithms will not have the same performance on them, or even would readily work outside of the box. For example for time series it might be more useful to use an Long short-term memory (LSTM) neural network, and for text data - SVM.

Now that you know what the common constraints are, you can use them in the following sections on use case prioritisation and planning. We want to have a funnel - we should start with many possible projects, but end up with just a few that we can proceed with successfully (at least to the best of our knowledge at the time of those sessions).

Prioritization

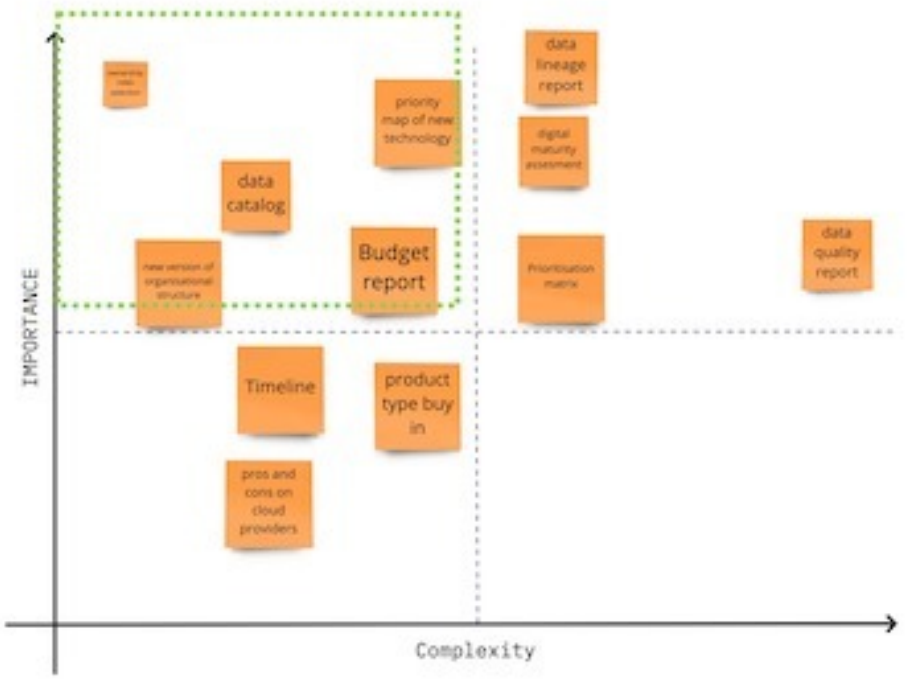
The hard part in prioritization is choosing the right metrics. The format is straightforward, any two by two matrix will do:



2x2 matrix

Typically one of the metrics indicates the “importance” of the use case - this can also be substituted for “impact”, “business value” or something similar. On the other axis we can use “urgency”, “effort required” or “feasibility”. Then we can group the different use cases into the different quadrants (again a good idea to do this a workshop setting). The results that we should focus on first will appear in just one of the four quadrants, for example into the one which is very important and very urgent (top right) - and we should start with those.

Here’s a filled prioritization matrix:



Filled prioritization matrix

Data Architecture and Technology

Deliverables:

- Design and requirements documents for target data architecture
- Documentation on selected technology stacks (supporting the prioritized use cases)

As we covered in the INTRODUCTION, there's a considerable amount of technical knowledge that a data strategist needs to have. In no other element of data strategy this is more required than in ARCHITECTURE AND TECHNOLOGY.

The main purpose of this section is to lay down the *approach* for guiding an organization in terms of architecture and technology. Thus I won't go into detail on the different technical terms, there's more than enough resources on that (linked wherever suitable). What's more important for a data strategist is how they they are selected, how they fit together - and most importantly - how can they support the value-generating use cases.

We can differenetiate our approach (like we did when estimating constraints) between two scenarios: *greenfield* and *legacy*. In the former scenario, the data strategist's job is to help design an architecture and technology strategy from scratch, since there's none in place. In the latter, the strategist needs to ensure the new recommendations can be embedded or

connected to existing systems properly. Almost always there will be some legacy system (whether it's part of the data stack, or another system with which the data stack needs to interact with) to take into account. This second scenario is arguably harder to operate in, since it adds more complexity and constraints to our recommendation. The good work in understand in which scenario you are operating, and the necessary information to make qualified decisions should be available from the CSA.

Let's take a step back and do some definition setting. We hear the words architecture and technology quite often, but it's rare that we stop and pause to reflect what they actually mean. If you ask five different people in tech to provide you with definitions, you will get five different answers. This reflects the broad nature of those concepts. Also, finally we can make use of the nice analogies we covered in the beginning of this chapter.

> *Why AWS?* The technical examples in this book are mostly focused on Amazon Web Services (AWS). While there are certainly other cloud providers, and some of them are in several areas more advanced, I selected AWS since in my opinion (at the time of writing) they have the most diverse portfolio of services, supporting many use cases. In any case, almost almost any service in AWS has a corresponding alternative in Google Cloud Platform (GCP) or Microsoft Azure. As mentioned, I don't want to go into detail in all the services, and how to use them, but if you need more information an excellent resource is the AWS Cookbook - it's full to the brim with recipes you can use for your data work in the cloud.

Data Architecture: Everything which is not technology, which has a supporting role for the successful operation of technology. You can also imagine architecture as the scaffold of the building, which supports the other

elements. Examples of data architecture components are data lake, data mart, data warehouse, ingestion, and presentation layers, and others.

- Oil analogy: Pipes, wells, vessels.
- Kitchen analogy: Fridge, oven.

Technology: Those are the tools that process and deliver data. Those can be data cleaning scripts, glue code pulling data from third-party APIs[*], enrichment queries to knowledge bases, processing for ML training, ML algorithms, ML APIs, and others.

- Oil analogy: Filters, valves, extraction chemicals.
- Kitchen analogy: Knives, blenders, forks, and spoons.

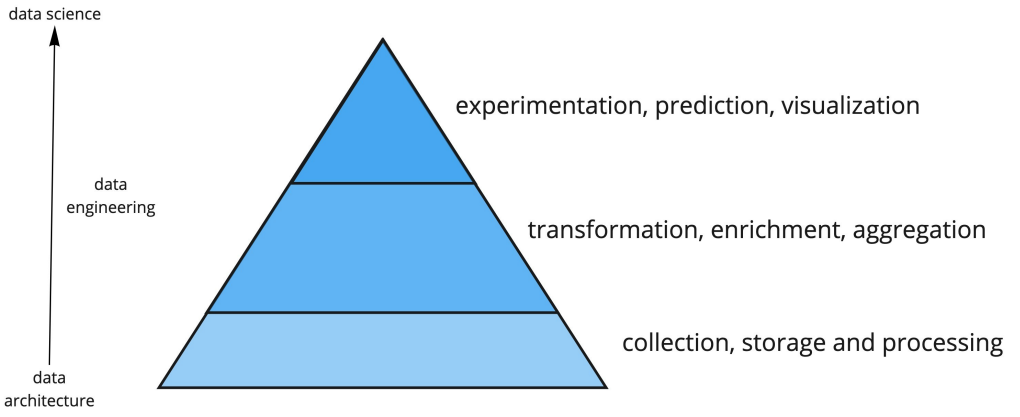
With the definitions out of the way we can confidently proceed to designing the architecture and technology elements of the data strategy. Here we'll use the word *target*. As you remember in `GAP ANALYSIS`, in data strategy we're always striving to achieve a goal, a target state. This is essential to always keep in mind when working on architecture and tehnology, since it's very easy to get bogged down in unnecessary details. The most important outcome from this part of data strategy is preparing the deliverables and the shortest path to achieving them with minimal effort.

Target Data Architecture

Let's with the why (Amadeus Tunis explains why this is an essential skill in `INTERVIEWS`). Why do we care about data architecture? There's no better way

[*]Stands for Application Program Interface. This is the standard way to expose a piece of software to other software systems. An abstraction lawyer for software-to-software communication.

to visualise this than Shopify's Data Science Hierarchy of Needs. Here's a simplified version:



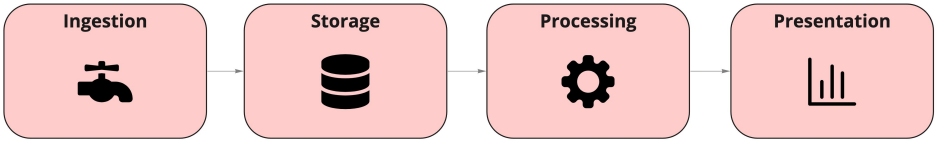
The data science hierarchy of needs.

This diagram puts the three pillars of technical data work in context. Without the solid foundation of data collection, storage and processing the whole pyramid would break down, long before we manage to successfully deploy more advaned data science use cases at scale, such as prediction. As we go up the pyramid more data engineering tasks such transformation and enrichment take advantage of the data architecture, and finally enable the diverse data science use cases. This shows how without a good data architecture, more advanced work is not possible, at least at scale.

System design is an essential skill of a data architect. Understanding and mapping out how different architectural components fit together as a whole is difficult to learn from books - it is mostly obtained from experience. Still there's one practical book which shows the way of thinking of a good software system designer: *"System Design Interview: An Insider's Guide"* by Alex Xu.

So how do we go about designing a data architecture? This is a step that can often feel tricky to a data strategist, who might have more of a business background. In no area of data science, there's more technical jargon, coupled with an ever-expanding explosion of services and frameworks, as in data engineering and architecture. I would argue that for the fundamental work of a data strategy design engagement, a data strategist should be able to complete most of the work without additional assistance from a more technical perspective. Still, especially in the legacy scenario, where the complexity is higher, advice from a data engineer or architect is probably unavoidable.

At the highest abstraction level, for any scenario, in almost all organizations the data architecture can always be broken down to those elements (layers)[*]:



As many common complex topics, the best way to manage the complexity of a larger task is to break it down into smaller, more manageable chunks - this will also help us get started more easily. We can breakdown any architecture (the *stack*) into its different component layers.

Ingestion

In this layer data enters the system. There can be multiple sources here, and ideally this collection of data should be mostly automated. Example data

[*]Here the concept of abstraction layers we discussed in the Systems 101 section proves useful.

sources include third party APIs, databases, data from IT systems (such as CRMs), scraped data and others.

> Key patterns: automation, orchestration, scraping, third-party data
> Key terms: APIs, Airflow, Kafka, Kubernetes, AWS Glue

Storage

The data from the ingestion layer needs to be stored in an organized way. In the old days when data volumes where lower, and most of the data was generated by internal systems, the most common place to store data were databases. Nowadays the volume, velocity and variety (as covered in `SYSTEMS AUDIT`) has increased exponentially. Specifically the volume and variety of the data require it to be stored in different ways and systems. For example image data due to its size should be stored on a data lake (such as AWS S3, or Azure Blob Storage), and time series data in a specialised database such as InfluxDB.

Often when you are designing architectures and trying things out costs are a worrysome factor. A nicely structured, informative and interactive educational resource on the topic is available on the AWS Well-Architected Labs website.

A common pattern in data engineering is the concept of a “single source of truth” (SSOT). This means that the raw data should always be stored somewhere in the system, and any downstream processing should be stored separately. Like this we can ensure data quality and add some redundancy to the system in case of errors in processing.

> Key patterns: data lake, data warehouse, databases
> Key terms: S3, Redshift, RDS

Processing and Enrichment

Another relatively recent development is that in order for this large amount of unstructured data to be useful for the business (let's say as an input to an ML model), it needs to be processed. Example processing steps include deduplication, feature engineering, dimensionality reduction and others. In most systems this is done by the orchestration of different scripts and services (by using AWS Lambda, Apache Airflow, and/or AWS Step Functions).

> Key patterns: enrichment, deduplication, disambiguation
> Key terms: AWS Lambda, AWS Data Brew, AWS Step Functions, Apache Airflow

Presentation and Delivery

A data architecture is meaningless if it does not support a good use case. And each good use case has a goal in mind - the results of all of this data ingestion, storage and processing need to be consumed by another system. This system can be a backend application which is a part of the standard IT architecture, or an external user which needs to use the results of the data product. An example for the latter is a ML model endpoint which is then exposed via a frontend.

Key patterns: DevOps, DataOps, MLOps, self-service analytics
Key terms: API Gateway, AWS CodeCommit, AWS Fargate, SparQL

Finally, we can use all the layers and combine them together in our target architecture.

Target Technology

If the architecture is the "skeleton" of a data project, technology represents the "muscles". There are several angles through which we need to select the technology that supports a data strategy.

By far the most important one is to make sure that any technology choice has its main reasoning rooted in a solid use case selection. Even the shiniest, coolest, most modern technology is useless if decoupled from a successful use case. Technologists like to work with cool tools, and on cool stacks. As I mentioned previously, human resoures are one of the biggest constraints on what we can achieve when implementing a data strategy. If possible we should use more modern and interesting tools., normally, they are also better at what they do. A practical way to achieve a balance between cool and useful technologies this is to have a combination of projects, where the engineers can switch between working on "cooler" stacks, and something which has more of a "legacy" nature. People love to have autonomy at the workplace, and most bring some valuable experience from previous positions. In order to take maximum advantage of this it's sometimes good to let them decide which tech should be used. We'll return to this point when discussing how to successfully deliver a data strategy in DELIVERY.

Of course, we should always use the best tool for the job, but sometimes there needs to be a compromise in order to reduce technical debt and complexity down the line. For example, let's say that 80% of the data stack in the organisation is written in Python, using the standard Python data packages, such as `numpy`, `pandas` and `scikit-learn`. But there's a team member who is working on more statistical use cases, for which R could be a better choice. Should they be encouraged to write R code in this case? The answer is - it depends. If we are planning to have a larger team which is focused on stats, or this person can do almost all of their work safely in isolation - maybe this is a good idea. But if other people need to understand, or even edit the code - maybe we should make sure that the stack is consistent, and use Python for the analytics stack as well. Use the best tool for the job, but remember to take the rest of the stack in consideration.

Technology Selection Criteria

Many of the choices that need to be made in the course of a data strategy have a markedly "qualitative" rather than "quantitative" flavour to them. Those decisions are more gray, and require a nuanced approach. In terms of selecting technologies, here are recommended selection criteria:

Fit for purpose: This one should go without saying, but the tech we use should fit the purpose (see the "use case driven" point).

Ease of use: The learning curve ideally should be low. You'd be surprised how quickly this can become an issue if you are working within deadlines and other constraints, or have a less experienced data team.

Popularity: Normally how popular a package is is directly correlated with how good it is (with some notable exceptions).

Open source vs. Proprietary: Nowadays this is not even up for discussion. Except for some niche cases (such as when you are working with GIS*), for data projects you should always go for an open source solution.

Maintenance and future proofing: This is probably the most overlooked factor when selecting packages. It is especially relevant if you have selected open source technologies for your stack. One of the few negatives of using open source software is that you probably won't have good support, and any issues that come up during development have to be addressed by your or your team. Thus it's essential to use packages which have good community, and stable future. Stay far away from abandonware!

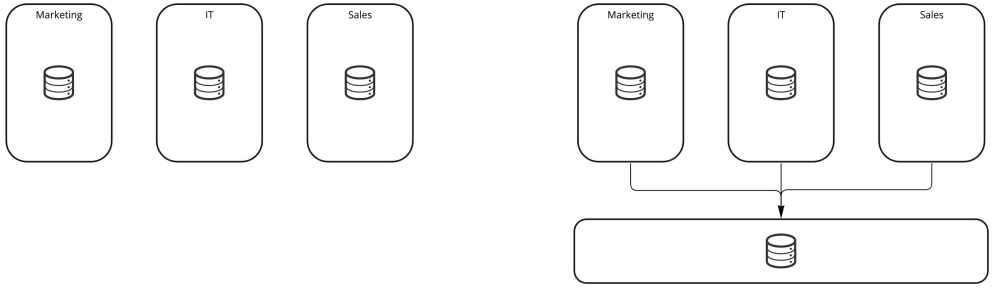
* Geographic Information System.

Data Governance

Deliverables:

- Design of target state of data governance
- Documentation on data governance requirements

As the organisation grows, and the usage of data becomes more established and widespread, a new source of complexity arises. Access to the data and the technology required to operate it (resource provisioning) for different teams needs to be managed.

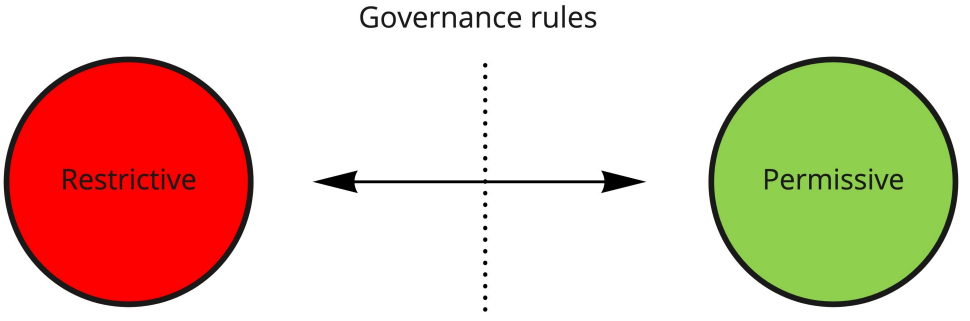


Data siloes: bad governance vs. good governance.

An immediate reaction to the need of governance is that ideally all people in an organisation should have access to all of the data. This sounds good in theory, but one needs to think of a few things first to realise the

impracticality of this suggestion. There's not just one type of access to data - there is at least access to view ("read access") or edit ("write access"), who gets which? Also should information on payroll, or other personal information (which is readily available and necessary for operations of the accounting department) be available across all organisations? How about truly global enterprises which have offices and operations of differing significance, credential and compliance requirements in different regions in the world, subject to different laws and security environments? How about external consultants and contractors, to which data should they have access? This makes clear why there's need to define at least some basic rules of governance.

The concept of data security is almost completely tied in with the concept of governance, that is if governance rules are set up correctly, security would automatically be taken care of. Additionally, this also falls in the ream of the traditional IT strategy and operations, so it's out of scope of what we are covering.



Tension diagram for governance.

Governance should be balanced between rigidity and flexibility, and completely attached to the use cases pursued. Different teams will have access

to different data. As a rule of thumb it's safe to assume that more data access than necessary is better than less, with the notable exception of personally identifiable or otherwise sensitive data.

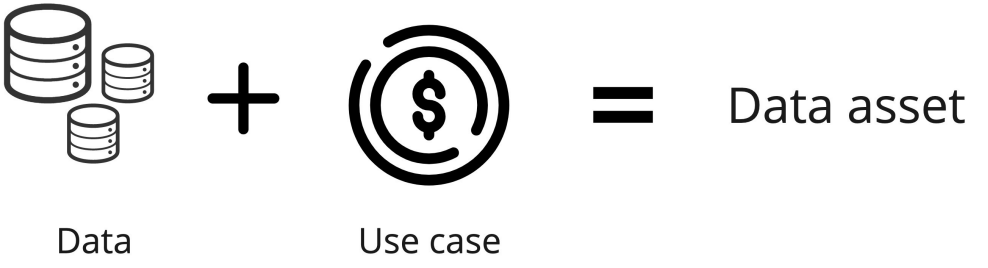
Different departments would require different software / hardware. I believe it's a great idea to have a lot of flexibility in terms of what hardware and software people use across the organisation (whether it's their operating system or choice of a text editor or browser). This will of course sometimes result in some issues, since the data projects developed in the organisation would need to work on different platforms and machines, and also be able to be developed on those. Still, this is normally not a big issue and the pros mostly outweigh the cons.

When it comes to the cloud that the organisation provides, it's good to stick to one cloud provider. Nowadays the three incumbents (AWS, Azure and GCP) share almost an identical set of services which can cater to even the most specific data use cases. One important point to make here is that data science and engineering teams should not have absolutely the same privileges in tooling.

Vendor lock-in: A common situation that the data strategist should be aware when making recommendations, especially in terms of cloud providers, is vendor lock-in. Some services can seem very useful in the first place, but we have to weigh that against future risks. We don't want the organization to be in a situation when the service is not suitable anymore, and migration to another one is difficult or expensive.

Data Assets

There are different definitions of what a data asset is, but for the uses of this book, we'll definite it as the combination between a dataset (or collection of datasets) and a use case.



Defining a data asset

One of the most challenging elements in a data strategy is to inspire the members of an organisation to change how they view data. Much is said about being "data driven", and this can only be achieved if we view data as an asset[*]. Doug Laney in his great book *Infonomics: Monetise, Manage & Measure Information*[16] tells a compelling story on how modern enterprises treat things like printers, monitors and window blinds as "assets", but not their data. This is why we should look at our datasets always with a use case in mind, and if we have done our work well in the previous section of use

[*] In my conversation with Amadeus Tunis, he mentioned that data can only have *relative* value, not absolute as other assets.

cases, it should become clear how this view can have monetary gains - which is one of the main driving factors behind the motivation of becoming truly a data-driven organisation.

BSG Data. This is a concept that can help design architecture and technologies strategies. It stands for the different classes of data assets - Bronze, Silver, and Gold.

The concept of using metals to describe the state of data is already used in data engineering. You can learn about it from articles by leading companies such as Teradata and Databricks[a]. This is good use, but I thought, can we use this when discussing data architecture and technology? It turned out it can, and in my conversations with clients, it has worked very well so far. So what are those BSG data assets?

Bronze: This is the data that enters the system. It's data in its rawest, atomic form - without any processing. It usually is of large quantity, little quality, and of limited business use. Focusing on just collecting these data is dangerous, since it's of limited use in this form - this strategy is called "boiling the ocean", and is exactly as productive as it sounds.

Silver: This is the data that is the result of processing and enrichment steps. This data is also the input to ML algorithms. It usually is smaller in size and of intermediate value. It can also be the input to self-service analytics systems, where data analysts can build the presentation layer.

Gold: This is the data presented to the end-user (whether internal or external). It could be the data points at a dashboard from the presentation layer or the results of a machine learning algorithm. It is of the least amount but the highest quality and importance for a business.

[a] Teradata's article is here and Databricks here.

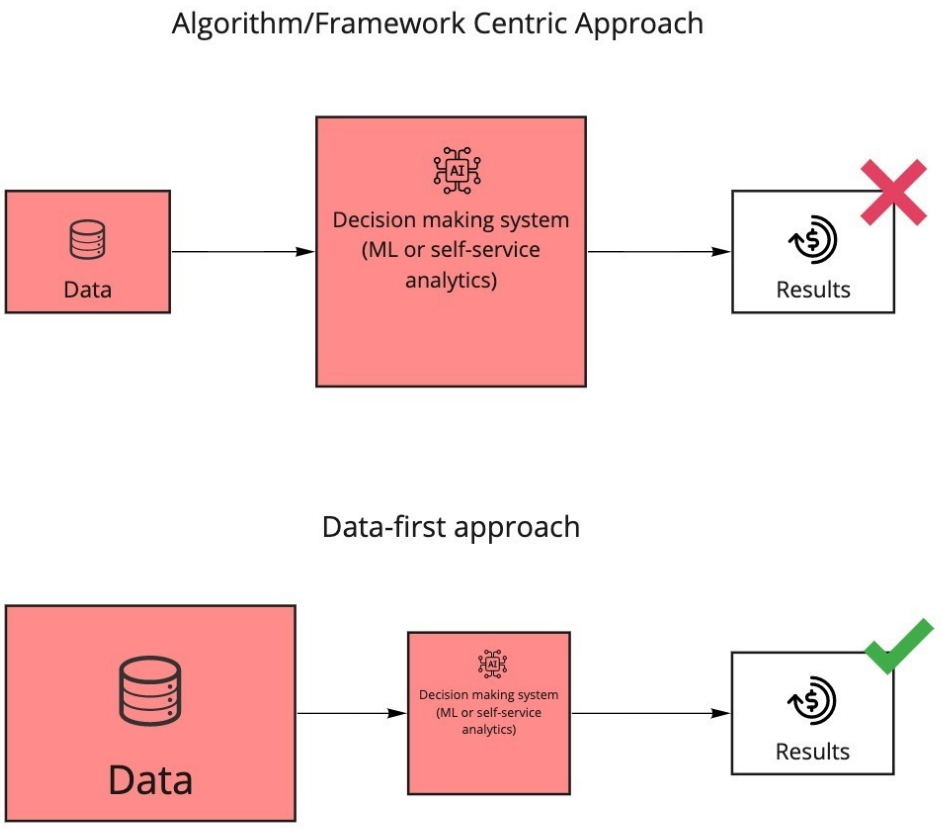
With those concepts in mind, we can look at the whole. The technology is used to modify the data (enrich, process or cook) until it gets to the Gold layer, while the architecture hosts the technology.

Data Lineage

A common phrase in the field of data science and engineering is "garbage in - garbage out"[17]. It's mostly used in the machine learning context. In the case of training ML models it often occurs that data scientists tend to spend an inordinate amount of time on fine tuning a model, or trying to solve the problem by changing the algorithm completely (often going for a shiny new one, or a different framework altogether). While such changes might improve the model performance with a few percent (in the best case), the highest gains of performance occur when better quality and quantity of data is supplied to the algorithm. The best algorithm on the planet would not save the project if the data is garbage. In non-ML applications, wrong data can also have disastrous consequences - weekly financial reports based on the wrong numbers can spell catastrophe.

One might ask themselves, why does this focus on the wrong thing happen so often? The reason is that gathering more and better data can be tedious, and requires a lot of thinking, coupled with domain knowledge (which in turn requires better communication with experts).

Recently, there's been a lot of talk of "data-centric AI", spearheaded by the famous Andrew Ng[18]. Let's visualise this focus on data in the diagram below:



Data-centric

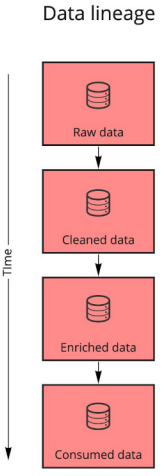
The size of the components is a proxy for their importance in the approach. When we place equal, or even greater, importance on data than algorithms, we can improve the quality of our data-driven solutions. Of course, once those efforts are finished, we should go back and try more tech-oriented improvements on our products, such as hyperparameter tuning or trying different algorithms.

A common theme in the book is to always be aware of the context in which

you and your team operate (as we covered during DD). In some cases, even minuscule improvements in performance of your data products can have tremendous positive (and negative) consequences for the whole business line. In those cases, and also depending on the resources you have available at your disposal, it might make sense to invest heavily in optimisation projects.

In light of this, we need to have a better understanding of the data "lineage", in order to succeed in providing value through our products. Before we dive into how to investigate the lineage in a data concept - let's use the favorite tool of this book - an analogy. The word "lineage" is mostly used in two contexts - for royal families, and in biology. I believe the second example can be closer to what we're trying to achieve here. There's a class of newly born cells, called "stem cells". Those have the unique ability to differentiate in all kinds of other cells which comprise living organisms - from tissue cells, to blood cells and neurons. The history of the gradual morphing of a stem cell into each of those is called "cell lineage". Slowly, a relatively bland in features cell can become a wonderfully specialised cell.

Data behaves in a similar way. In its origin it is mostly useless, lacking features and utility. With time (and correct data processing) it can serve a purpose. The history of data as it changes with time is called data lineage. Let's illustrate this in a figure and then explain why this is important to clarify:



Data lineage

Any dataset's lineage can be split into those several categories. Also note how this relates closely to the BSG concept we covered earlier, with the different categories mapping to the lineage steps:

- Raw data
- Cleaned data
- Enriched data
- Consumed data

In between those categories there might be many intermediate steps, and those differ based on the data format.

So why should we care about data lineage? Here we circle back to the "garbage in, garbage out" metaphor. At this point you should know that raw data is rarely useful for downstream applications - it tends to arrive with a lot of errors and duplication. Moreover, with every processing step there are

new opportunities to make mistakes, especially if there are manual, human-involving steps in between - those can introduce tremendous bias and error. Such errors can be hard to detect, and this is why it's important to have a clear picture of the complete data lineage, and equally as important: who are the people responsible for the different parts of it. As part of the data strategy we need to interview them and make sure that the information we have on the data, and the assumptions on its processing are correct - so that we have an up-to date documentation on the lineage. Note that this work fits nicely with the concepts of data dictionary in the `SYSTEMS AUDITS`.

When we are going through the different steps of a data lineage, new ideas in how we can process the data can come. These new data points are called "derived data", and the process of using this insight to improve data products is called *feature engineering*[19].

Data Ethics and Privacy

Any system that is impacting human lives in any meaningful way has to be held to some standard of ethics. In this section I have grouped ethics and privacy together, since the solutions to both often go hand in hand, so it makes sense to cover them at the same time. We can go through the issues that a data strategy needs to address regarding ethics in privacy based on two angles, descriptive and prescriptive.

Use cases that are of purely descriptive nature in terms of ethics are easier to tackle, since there's no concept of a black box involved. The most important consideration here is to make sure that any decisions that are based on the datasets used follow ethical guidelines, which should be readily established. The second use case, that of prescriptive analytics is harder to tackle, since

it can be more of a black box, so the decisions are done (or at least heavily influenced by machine learning systems). We can do several things here:

Human in the loop: make sure that the results of prescriptive systems are audited, or have active participation by humans

Ethical design: ensure that the way prescriptive systems are built is representative. The most common issue here is that of using biased datasets.

Explainability: use explainability methods on top of prescriptive systems to automatically deduce what they are learning and what are the main factors in their decision making process

> **Explainable AI (XAI)**. The black boxiness of ML systems has been an issue ever since they were conceived. This issues has been exacerbated further nowadays since we have ML systems being deployed in more life critical applications, and moving away from the realm of traditional tech companies. Think how received a wrong prediction for what YouTube video to watch next impacts a person's life, as compared to a machine learning system for cancer detection. Also the complexity of machine learning models used has risen - since as a rule for most modern machine learning applications, such as in the computer vision or text domain, more complex algorithms are used (more neural networks than linear models). Those more complex algorithms also tend to be less explainable.
>
> XAI can be seen as an architectural layer on top of a traditional system. There are different methods that can be used here, and some of the more popular ones are LIME and SHAP. I can refer the interested reader to the excellent *Interpretable Machine Learning* by Christoph Molnar for a deeper dive into XAI[20].

GDPR has been a moment of reckoning for most organisations, which now

see the need to store personal data securely. The simplest in this case solution would be to store no data at all, or delete it after some time. Still, customer data is essential for most modern businesses, and many use cases in data and AI revolve around such datasets. Moreover sometimes dropping the data would not avoid ethical or privacy issues completely, since some other features (or a combination thereof) might correlate with the privacy-infringing one, and we might find out about this much later than we would like to.

Thus we need to find a way to both be compliant, but still keep the data. There are two methods to achieve this: anonymisation and pseudonymisation. Anonymising the data means making sure that a single individual cannot be identified based on the data. This means making sure that fields such as name, address, age, gender and other similar ones are taken care of appropriately. Pseudonymisation refers to the idea that we can do this work without destroying the analytical value in the data, and use it for our purposes - the best of both worlds. The concept of “federated learning” falls into this latter category[21].

Operating Model

Deliverables:

- A design of an ideal operating model structure
- A list of recommendations for achieving target operating model state

Organisational Structure

While there are several methods as to how to proceed in setting up an initial structure for data work, a common one is the establishment of a Center of Excellence (CoE). Some organisations, even if they have existing functions with data ownership and responsibilities, would still prefer to start fresh.

The CoE typically starts with a smaller group, around 15 people, with different roles, working on more than one pilot project. Here are the reasons why such a setup makes sense:

Low management complexity: The group is smaller and (normally) has one leader who reports externally.

Low change to the existing structures: The amount of change necessary to the organisation is kept to a minimum. By using a CoE the other depart-

ments and teams can continue to be staffed and operate in the same way as before the start of the data strategy initiative.

A fresh start: A brand new structure is created, and those people haven't worked together before. This has always positive effects, since many of those people would have a new way of looking at things, and the office politics would be at a minimum.

An agile way to create an experiment: Since this is a small change (see point number two), and the team is small in size as well (see point number one), the success of this team is relatively easy to observe. Thus it would also be easier to implement adjustments if necessary.

Now you understand why a CoE is a good idea, but what other models exist, and how do they compare? Let's have a look.

Data Team Models

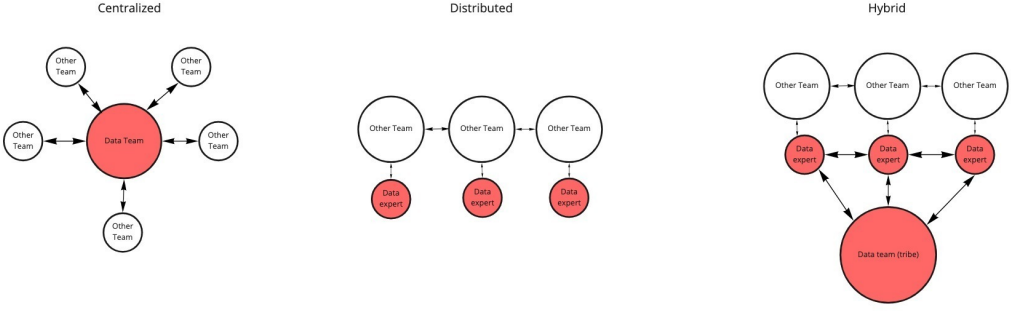
Data teams are a bit different from traditional software teams in how they operate. For software teams it's more accepted to function in a more isolated role with more or less limited interactions with other functions, but for data teams this is often not enough. Data teams need to understand the business use case even more, and the products they develop are often consumed more internally. There are several models that an organisation can use to structure how their data teams interact with the wider organisation.

Centralised: In this model the data team functions closer to a traditional software team, and work daily together. This is great for team spirit and cohesion, but can be an issue when the projects require more interaction with other departments. This can be seen as "in-house consulting" or Center

of Excellence (CoE) format. This is also naturally much easier to set up than the other option, and hence is the more commonly occurring one.

Distributed: A more ambitious, and arguably better suited for data work setup is the distributed one. In this case every data person works in a separate team most of the time. This is the other extreme, but very useful when data in the organisation is more focused on descriptive rather than predictive use cases, and data has more of a support, than leading role.

The answer again lies in the middle. The best approach is to have a **hybrid** between the two and alternate flexibly. This takes the most effort to pull of successfully, since the analytics managers need to make sure to balance the workloads and culture carefully. The different models are shown in the figure below:



Data team models

To end this section I want to offer one final advice. Any operating model is better than none. Clear structure can tremendous improve productivity, because the relationships and responsibilities of data team members are clear. Another important point to consider is that some of those models are more difficult to pull off than others, and would be suitable for organisations of higher digital maturity. The amount of trust that you have gained within

the organisation will be essential to get enough confidence in changing operating models, so the labours in the DD part of the book will finally bear fruit here.

Change Management

As we previously discussed, the hardest thing to change in an organisation is not the technology, but the people around it and how they operate. Still, it has to be done, and in this subsection we'll go through some methods in doing exactly that. This field is broadly known as change management and this is the term we'll use from now on this topic as well.

Change hierarchical structures: Much is made of having a "flat hierarchy" nowadays, so much so that it's an ever-present item in the perks section of job descriptions. I don't think this is a good idea, and this shift represents a knee-jerk reaction to the bureaucracies of old - the pendulum just swings to the other extreme, instead of finding a productive middle ground. *Some* hierarchy is necessary. A good rule of thumb is that for every 4-7 people to add a manager; then for every 4-7 managers you need a department head and so on up the chain. The reason for this specific number is that this is the approximate number of direct reports that one person can lead effectively: both from a technical, and personal perspective.

Change skills and diversity: Everybody knows that diverse teams are more successful, but why? The fundamental reason is that diversity allows for different view points - thus there are many more paths to solve a complex problem available. The same concept applies to skills.

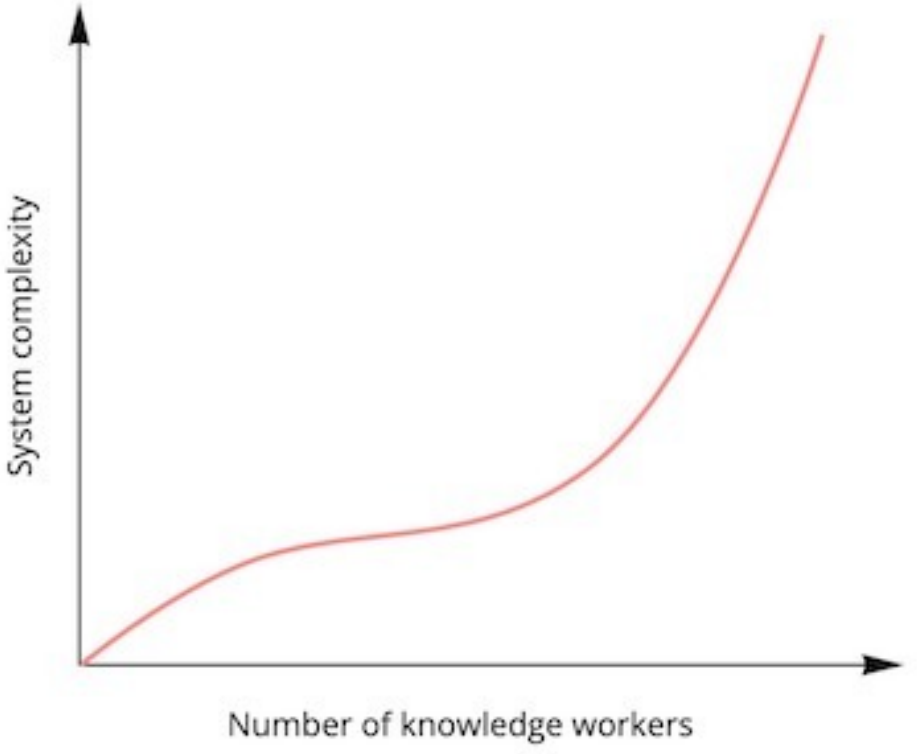
Take into account career progression: One fundamental reason behind the large churn numbers in tech is that managers and decision makers

regard the careers of employees as static and unchanging. The reality is that those are almost always in flux, and an employee should *always* be growing, and this has to be taken into account not only when discussing compensation and titles, but also when distributing tasks.

Communication: This topic is discussed in more detail DELIVERY.

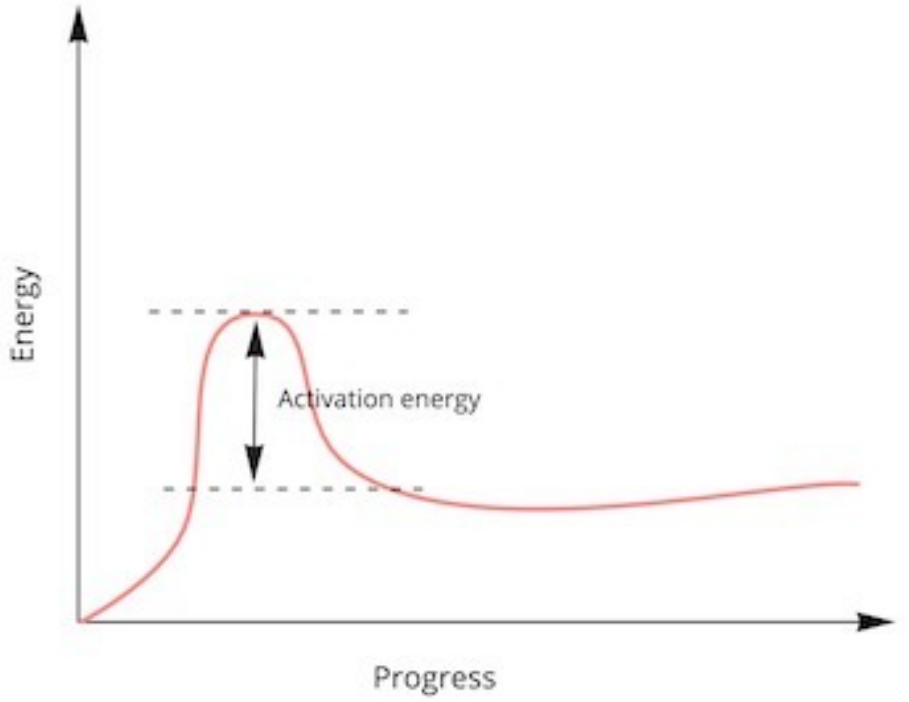
You can look at data strategy as a process of planting valuable seeds that need fertile ground to grow. Someone said, "culture eats strategy for breakfast", and I can't agree more. Even if you design and deliver a perfect data strategy, it's pointless if not accepted by the organisation at large. This difficulty in accepting the strategy is the hardest task for a data strategist to address. This is not a book on social psychology - and unfortunately there's no substitute for real-world experience in the case of this complex and challenging topic. Still, I'll attempt to give you a few starting points, so that at least you can start making solid progress.

Before we start with this task we need to think why changing a working culture is so difficult, so we can address those challenges head on. After all there's a complete field of work dedicated to this topic, amply named "change management", along thousands of books. Unsurprisingly, we can probably reduce the main reasons for difficulty to the human factor. Any organisation comprising of people is a good example of a Complex Adaptive System (CAS from SYSTEMS 101) . The complexity of this system increases non-linearly (that is - very fast) with increase of number of workers. This is further influenced by the composition of the said workforce - knowledge work tends to be even more complex (harder to learn, teach and automate). Any complex system is hard to change, mostly because it functions as a black box - it's inner workings are hard to deduce, and therefore influence and change.



Complexity - team growth curve

A further issue in complex systems is that they tend to contain feedback loops, whether positive (amplifying) or negative (inhibiting). The most important property of a feedback loop is that once it's set in motion it's hard to reverse. Its momentum can be tremendously strong, even if we do all the tight things. A useful concept that aligns well with CAS in large organisation is *activation energy*:



Activation energy

This is a concept that I borrow from chemistry. Imagine you are back in your chemistry class in school. You are given a beaker and two transparent, liquid reagents that need mixing. The teacher tells you that the resulting mixture should be of green color. Naturally, we start by adding the first reagent. Once it's all in the beaker, we slowly start to add the second one. To our disappointment, nothing happens - the mixture between both substances continues to be transparent, no matter how much of the second reagent we add. This is frustrating, since our chemical calculation show that both mixtures should combine readily. What is the issue? The problem lies that in order for the chemical reaction to occur, we need to add some energy to the system. Once the reaction is over that hurdle, it can occur (see the diagram above). For this to happen, we need to add a third reagent, whose only

purpose is to *lower the activation energy*, so that the reaction can commence. Once we do it, the two transparent mixtures will combine and result in a beautiful green color!

Masterclass. The most direct way to up-skill the organisation is to conduct a series of masterclasses on data. The audience for this event (or series of events) should be pulled from across the wider organisation. You can view this as a series of lectures on the utility of data, covering the fundamentals of the most important topics, such as data assets, machine learning, business intelligence, data quality and others. A great reference for this work is the book *The Art of Data Science: A Guide for Anyone Who Works With Data* by Roger Peng and Elizabeth Matsui[22].

Operating model changes. The way the data team is set up to work can have a great positive impact on the data literacy of the organisation. This team needs to spend extra effort in being helpful to other teams, and being in close contact with them, constantly looking for ways to help the business. One can go even a step further in embedding data team members in different divisions for a time to achieve those goals. We covered this already in the beginning of the section.

Coursework. There are numerous digital and remote-friendly courses for fundamentals of data science, and data literacy available online. Providing structured access to them for different departments can be very useful.

Product demos. Data team members need to serve as evangelists for data in the wider organisation. The work that those teams produce is often seen as opaque, and results and impact not easy to understand - this underscores the importance of making frequent product demos to a wider audience.

Hackathons (Datathons). Organising events where different groups within the company can work with data team members can be tremendously

beneficial not only for making the company more data literate and generate excitement for the technology, but sometimes deliver real business value by piloting the best of the projects developed in this format.

Use case ideation sessions. To foster collaboration between the data teams and the rest of the organisation another good idea is to make everyone more involved in the decision process on what should be done with data. The specifics of such data ideation workshops are detailed further in the following Use Cases subsection.

Roadmap

Deliverables:

- Documentation of budget required (cloud, office, salaries, FTE estimations and auxililary metrics)
- Short, mid- and long-term strategic roadmap of initiatives (use cases, target architecture and technology and governance implementations)

Budget and Scope

For this element we assume that the use cases are already selected and prioritised, and we have the basic layers of the data architecture mapped out. Those should also contain the different technologies present in each layer. The only missing information here is an assumption on the volume and type of data that we are expecting the system to ingest and process. The best way to do this extrapolation is to base it on existing datasets. Let's say we have 3 million text files available for our main NLP use case. We expect this dataset to grow by a million more texts every month.

We can now take all this information together to come up with the cloud costs necessary, since nowadays most of the data engineering work is done

in the cloud. There are several costs that we need to take into account: Storage, Compute and Service Costs. Fortunately for us, there are very sophisticated calculators made available by all the major cloud providers[*]. By knowing how much data is expected you can calculate the amount of storage you need. The compute you can base on the technology that you'll use (calculate the requirements for InfluxDB for handling the amount data for example), and finally service provisioning should also be straightforward to estimate. If we take our NLP use case again as an example, and that an essential feature of the product would require to obtain the sentiment of those pieces of text, we can use the AWS Comprehend tool. This at the time of writing would cost $0.0001 per unit. Now you can think how many texts you have and can come up with an estimate of your costs for this service.

Now we can do the rest of the resource estimation. Beyond the obvious things that a tech company needs to budget for (such as laptops and other hardware), most of the cost is based on salary. And the salary itself is based on three things: the seniority, skillset and market (location). Now you can appreciate why it makes sense to make a budget for the headcount only at this stage of the data strategy. We need to be certain of the use cases and the architecture and technology to be able to determine what types of people we would need. A good rule of thumb for the amount of people necessary is the concept that I have termed "atomic team", which we'll go through in more detail later in the operating model section. Almost any use case is achievable by a diverse team of 4 to 7 people in size. You can couple the roles, skills and experience levels of such a team together with adjacent cost factors such as recruitment and onboarding, to estimate the human resource cost for your budget.

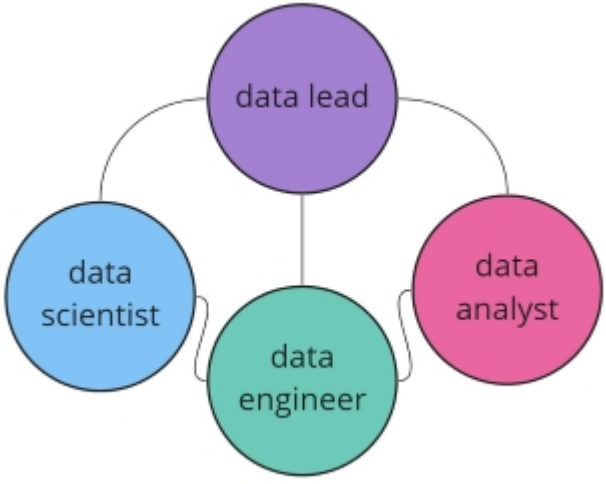
You might wonder why I chose the number 7 for the maximum team size.

[*]For the AWS pricing calculator go here, for GCP here and for Azure here.

This is arguably the number of people that can work with one leader. If you increase the team, you should have an additional leader, and then it makes more sense to have two groups.

The Atomic Team. I like to think about project chunks that a small team can accomplish. I would call this the "atomic team". The concept of *atomic* is used to describe the smallest possible abstraction level (from ancient Greek). There's an added cultural dimension that can make this concept a bit easier to memorize.

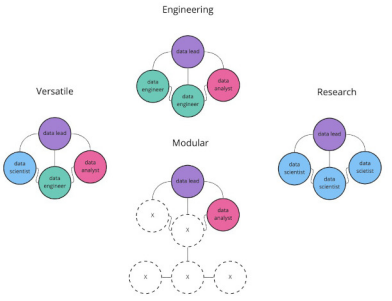
Let's have a look at the following diagram to see what an atomic team is:



The Atomic Team

It is comprised of four members, covering the main roles in data (there are many resources detailing what different roles are responsible for, I recommend Borek and Prill[23]). It also has the needed hierarchy (ownership) role. Most of the data projects can be broken down into tasks that a single team of this size can accomplish. If you cannot break down the work in

such a task, then you have to rethink your approach. Of course, there are variations to this team; let's have a look:

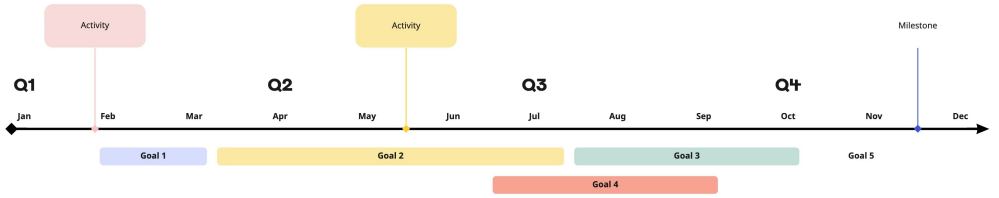


The flavors of an atomic team

Here you can see that we can use different configurations (but not too different) to adapt to the needed work. For example, some more engineering-heavy projects or data projects might require an increased engineering effort in the productisation phase. The same goes for more research projects, where we might have a team of data scientists and a lead. Finally, you can see the *modular* configuration that shows flexibility to add other roles, such as a data architect and a bigger team (but no bigger than 7).

Timeline

Despite the fact that a data strategy shouldn't be seen as a static plan, at least a fundamental roadmap is needed. Especially in the short and medium terms. In the diagram below you can see an example roadmap:

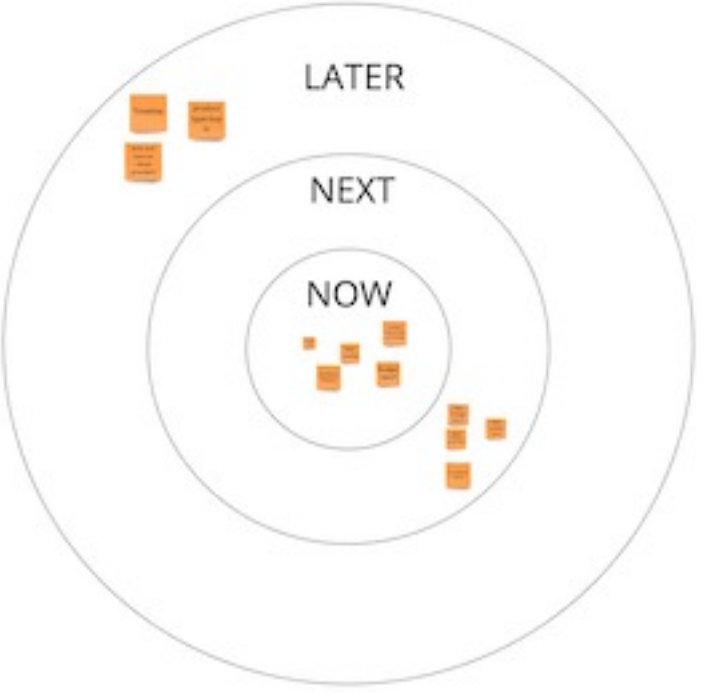


Timeline

This is an example of a Gannt chart. It consists of a timeline, and several swim lanes underneath it. The latter can be overlapping, especially if the data strategy needs to be followed bu multiple teams (which is often the case). The scale of the timeline can vary, but the most common one is that of quarterly planning. This caters to short-term and medium-term goals simultaneously. The most important thing to consider here is the need to add the regular check ins (also check out `IMPACT ASSESSMENT` in `DELIVERY`).

This roadmap can be further customised. For example, one thing which I did with a client of mine which was in the rapid growth phase, is to add the Human Resources (HR) to the timeline. The client wanted to see at which timepoints which additions would be necessary (together with their roles of course). This was very useful for the HR department to be able to plan their recruiting activities, and is a good example how a good data strategy can and should be aligned to the other departments in an organisation.

If you are wondering where to start with a roadmap, a practical initial exercise is to fill out a bull's eye diagram like the one below. Adding work elements to just the three sections should be relatively straightfoward and you can proceed to build on that:



Summary

With the DESIGN phase of the data strategy, we are done with our recommendation and roadmap. This is the final deliverable before the implementation phase - remember the StratOps approach.

We followed a MECE approach and covered the key elements. First with the one which generates value (remember we always should focus on the end goal) - USE CASES. We gathere ideas and prioritized them based on feasibility. After this we designed target DATA ARCHITECTURE and TECHNOLOGY to support those use cases. Their fuel, the data, is managed in the set of policies we defined in DATA GOVERNANCE. The non-technical element of data strategy is addressed in designing the teams and processes in OPERATING MODEL. Finally, we prepare the most critical deliverable of a data strateegy - it's ROADMAP, supported by a budget and scope.

By finishing this section, we are ready with the "static" part of data strategy. In the next one we'll make sure that it get's delivered successfully, and not suffer the same fate as many strategic documents in the past - laying unused.

Part III: Delivery

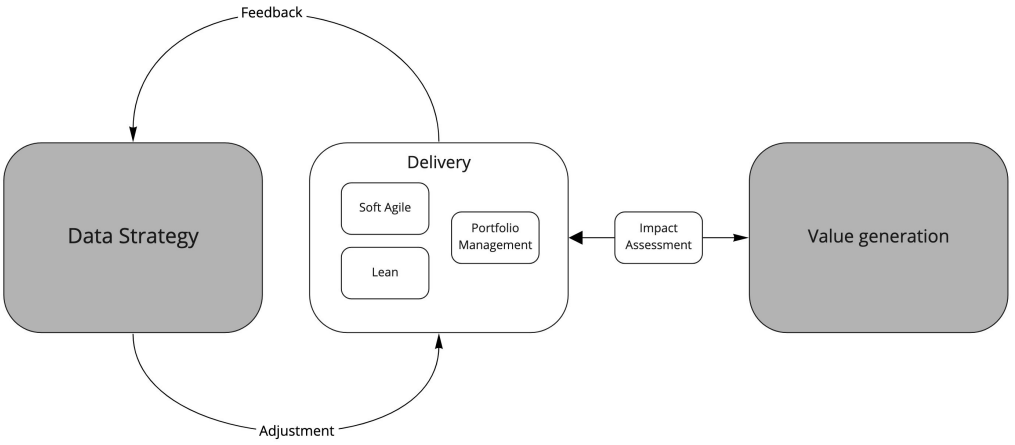
Soft agile—Implementation forest—Lean data—The knowledge factory—Impact assessment—Portfolio management

Overview

> "Well done is better than well said."
> –Benjamin Franklin

A good plan that's easy to act on is better than a perfect one that nobody wants to follow. Even if in the first two parts of the data strategy process (DUE DILIGENCE and DESIGN) we managed to formulate and prepare a great strategy, all those efforts can eventually prove to be in vain if the result is not adopted and acted upon within the wider organisation. Much of this is due to the fact that any organization is a complex adaptive system, full of operational and communication complexity. When executing a data strategy you'll need to rely heavily on many of the concepts and methods from SYSTEMS 101. As the rubber meets the road, our designs and assumptions will be challenged. We'll need to adjust continiously, while staying focussed on delivering value, by following StratOps principles.

This is how DELIVERY relates to designed data strategy:



StratOps approach to data strategy delivery.

It sits between the designed data strategy and the generation of value. The arrows pointing in both directions show that this is a two way process of constantly adjusting based on feedback. Naturally, the concept of "value" needs to be made explicit for this to work, this is why I dedicated a whole section to this - `IMPACT ASSESSMENT`. This element is used to measure value generation at different quality gates (more on this later in the chapter). `DELIVERY` relies on two main elements - `SOFT AGILE` and `LEAN DATA`. Explaining them and how to apply them to our purposes is the main goal of this part. `PORTFOLIO MANAGEMENT` provides a high-level view of all initiatives, so that decision makers can also have an opportunity to adjust the strategy as it goes.

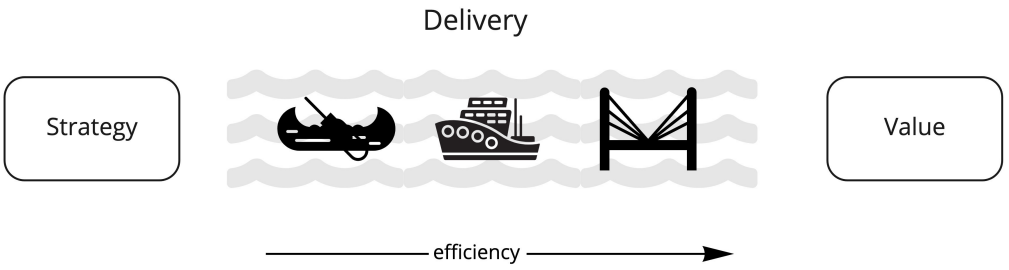
It is at this stage when most flexibility is required by the data strategist, since the complexity of applying the strategy can be staggering. This can also be a frustrating experience, since it is here that we often discover mistakes and wrong assumptions we made during the first two parts of the process. It takes a certain amount of courage to admit to those, but this is

essential if we are to successfully adapt. You'll be well advised to remember - no two organisations are the same, and all are continiously evolving, your impact won't stay constant either.

Technical jargon. In DELIVERY you'll be interacting with people who haven't neccessarily been participating in the other data strategy phases. For example those could be other members of the technology department, such as frontend developers. Thus you need to pay special attention to the use of technical jargon. Specialised fields often have a significant barrier to entry for newcomers. Much of it comes from the inherent complexity of the work, but some is due to the overuse of technical jargon. This is perhaps more true for technical fields than anything else. The usage of specialised concepts, such as "data lake" or "clustering" are the bread and butter of data scientists and engineers, but in the wider business context they can form a significant roadblock to a successful data strategy execution. A simple solution is that those words are just avoided all together, but unfortunately this is not always possible. In those cases some up-skilling work is completely justified. One idea is to always include a glossary of terms in any documentation, and making sure that all the people involved are aware of them. When explaining technical terms it's also useful to focus on their *function* rather than just descriptions and definitions - borrow the analogies from DESIGN.

Our end goal is to modify, extend and steer a complex system into a new, more profitable direction. At this stage we can make use of our journey analogy once again. Moreover, what I'll cover maps very well to different data maturity levels, so if you have done a thorough job in CSA you'll know what to expect here in DELIVERY. You can visualise our task as taking our

designed strategt and crossing the currents of a river, with the value on the other side:



Taking the journey analogy even further.

We can think of three different ways to cross it, some less efficient (but quick!), others more so (but also more expensive and time-consuming to build). Our first option is to attempt to cross the river in a small boat. This is how organizations of low digital maturity go about implementing strategies, and is the most inefficient and prone to errors way. A small boat is likely to capsize, relies on the muscle power of the crew only, and can be a victim of the whims of the winds, with little control over where it ends up. The use cases in such delivery scenarios rarely see the light of day (this is what Harvinder Atwal calls “laptop data science”, or in my words *pilotitis*), and the team members often burn out and leave. Clearly, a better way is to build a ship which ferries back and forth between the opposing shores. This is faster and more stable, but relatively inefficient: you are still heavily limited in your resources (i.e. how many people can be on the boad, and how much they can transport with them). The final option is where we want to be. We have to build a bridge. Successful data-driven companies, which are at the end of their digital transformation ascent, such as Google or Amazon, have bridges between their strategy and implementation efforts- this is why their efficiency and efectiveness is regarded as the gold standard in data strategy,

and the target of many gap analyses (remember the previous part). These organisations are so data-centric, that even this analogy would do them a disservice. A more apropriate way to term what they have achieved is “data highways”. A recommended reading on how they achieve this is O’Reilly’s “Software Engineering at Google”[24].

Let’s summarise those delivery scenarios in the following table:

Delivery analogy	Description
Boat	Small pilot projects, no overall strategy. Ad-hoc delivery.
Ship	Stable delivery and good operations, but no scale.
Bridge	Continious development under an overarching strategy. Feedback loops measuring value.

Two popular and battle-tested methodologies will help us build the bridge between strategy and value: lean and agile. They address different issues and should be used at the same time. I’ll use the North Star concept to explain them[*]. Here’s how their implementation looks when successfull:

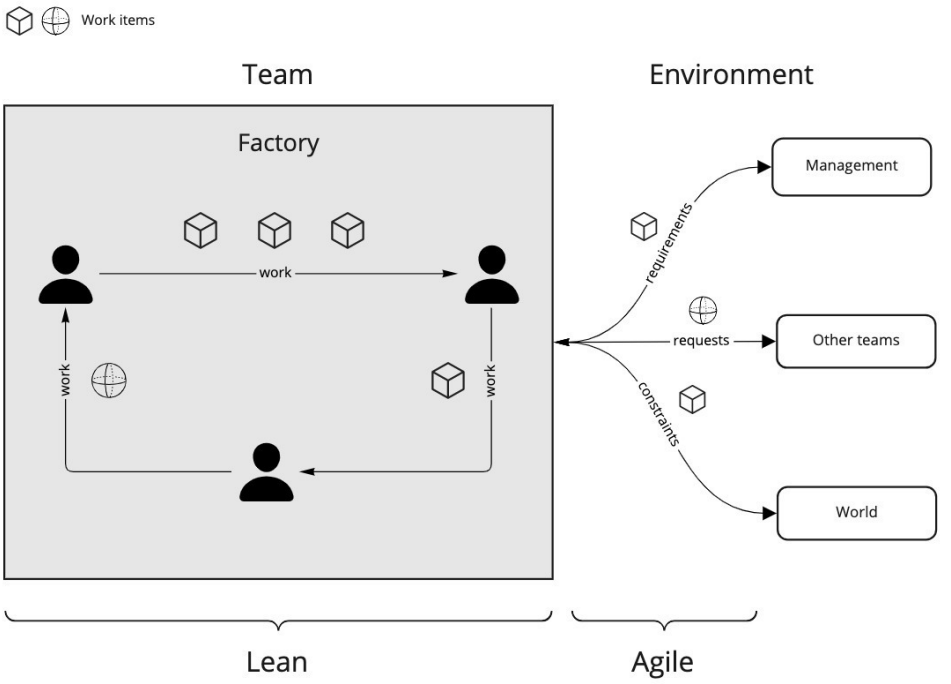
Lean: The data teams operate as a factory. There are clear targets, conveyor belts, automation, specialised labour and others. Everything keeps moving and the factory reliably churns product of measurable value. In case of issues replacement parts are available. No bottelenecks and limited waste.

Agile: The factory can adapt to a changing environment and requirements. If the need arises to change a feature of the product developed by the factory, or even replace it with a completely different one - agile allows the factory to do so without decreasing value output - on the contrary, small adjustments

[*] If you want to go deeper into them good references are *“Agile Project Management with Kanban”* (Eric Brechner) and *“The Lean Six Sigma Pocket Toolbook”* (Michael L. George).

can vastly scale the output.

The following diagram shows how the two methodologies relate to each other. While lean allows the factory to be productive, agile ensures it responds to the environment appropriately.



How Lean and Agile relate to each other.

While it might seem obvious that we want to use those frameworks to deliver the data strategy, there are hidden dangers. We can easly fall victim to a cargo cult, explained below.

Cargo Cults

Similar to the disease *pilotitis* that we covered in `DUE DILIGENCE`, cargo cults

> are the second common disease among large (and small) organisations, preventing them of executing on their strategies.
>
> Towards the end of the Second World War, many pacific islands have been affected adversely because of the armed engagements between the US and Japanese armies. At some point there were enough resources available to the US military that they started to supply these islands with food and essential items. For this purpose, the Americans constructed basic makeshift airports. This went on for quite some time, but after the end of the war, the supply lines trickled down to zero, and the locals were left alone. For their societies there was little understanding of aircraft technology, and they associated it with the delivery of supplies. Thus they promptly created airports and airplanes from materials they could find. Of course, those creations were far from functional, but nevertheless the locals believed their presence will automatically lead to supply delivery. Unfortunately for them, this did not occur.

We might feel far away from the pacific islands, but those dangers are all arounds us, especially once we try to adopt new methodologies. Many leaders across organizations believe that we can simply transplant apparently successful methods from other companies and expect immediate results. Large technology companies in particular are a common source of inspiration due to their massive success in innovating and spreading data products. It's useful to remember that those organizations have been focusing on digital products from day one, thereby greatly reducing the complexity of the task. They did not have much of the communication issues that any non-tech organization is bound to have, hindering work between technical and non-technical teams. Agile and lean methods can be even destructive, since they generate a lot of noise that can be mistaken for productivity, and open

up new options for micromanagement and miscommunication.

A plane from the 424th Bombardment Squadron. Non-functional replicas of such aircraft was construted in vain by the locals.

As long as we don't apply those methods blindly (this goes for the data strategy framework itself), and adjust them to the specific needs, resources and circumstances of our organization we are good to go.

Soft Agile

Soft Agile Theory

Does agile work for data projects? There are many different answers to this question. Most of us in the data community are wary of creating a cargo cult by transplating this from software development. For me, and many leading data strategists[*], the right answer is *yes, but with adjustments*. Those adjustments led me to name the flavor of agile that I recommend SOFT AGILE.

At the moment we find ourselves in a similar situation as when the agile methodology itself rose to prominence in the 1980s. Practitioners in the field, such as data scientists, engineers and strategists realise that the most widespread methodology at the moment (agile development) is not completely appropriate for the tasks at hand. This is similar how many years ago agile development replaced the predominant methodology at the time - waterfall. With the evolving needs of the industry there might be the temptation to completely get rid of the current methodology and look for a brand new replacement - but as was the case before, I believe it makes more sense to adjust the agile process to fit data projects better instead.

Let's look at the fundamental principles of agile, from the original document - the Agile Manifesto. The four main pillars fit perfectly well with data. Here's how they map to our designed data strategy:

[*] Check out the conversations with June Dershewitz and Noah Gift in INTERVIEWS for their opinion on this.

1. **Individuals and interactions over processes and tools**. This is captured in our efforts in making interactive sessions in USE CASES and work on upskilling in OPERATING MODEL. Culture eats strategy for breakfast!
2. **Working software over comprehensive documentation**. When we were preparing the USE CASES the essential metric for selection and prioritization was value. Documentation is important, but secondary to that. This shows again the product vs. project thinking comparison.
3. **Customer collaboration over contract negotiation**. Value generation in data products is always tied to customers, whether they are internal or external.
4. **Responding to change over following a plan**. This is the StratOps approach we have adopted in INTRODUCTION.

The main ideas map out well, but how about the specific principles of agile (also a fundamental part of the manifesto)? Here the situation changes, and I'll show you how SOFT AGILE differs:

Satisfying customers through early and continuous delivery of valuable work. The first part of this sentence fits very well. A good delivery idea is to show results of our data initiatives early so we can get stakeholder buy-in, and dispel skepticism (those are the lighthouse projects from USE CASES). Now, the second part is more difficult - the value of data projects depends on numerous factors, many outside of our control. For example, the idea of data drift[25]. Changes in the incoming datasets start to degrade the model performance. This was widespread during the COVID-19 pandemic, where shopping behavior changed dramatically overnight, rendering many predictive models obsolete. This is why offering a constant value in data projects is more challenging and the data strategist has to *manage expectations* - an essential feature of SOFT AGILE.

Breaking big work down into smaller tasks that can be completed quickly. Useful advice for any knowledge work, we'll take it.

Recognizing that the best work emerges from self-organized teams. This applies to data teams even more than to software, since adequate decisions on the project can be mostly done only by the implementers who understand the data. This is the fundamental principle behind homeostatic project management, a soft agile tool that is covered in DELIVERY METHODS.

Providing motivated individuals with the environment and support they need and trusting them to get the job done. Also extremely valuable for data projects. There are many sources of pressures on data teams (scope creep, changing requirements and others) that need to be addressed. Additionally, good embedding of the teams within the organisation is essential.

Creating processes that promote sustainable efforts. Not very useful for us since most projects will differ (beyond the architecture) and new processes must be developed for each.

Maintaining a constant pace for completed work. Not applicable, since effort estimation for data projects is very variable.

Welcoming changing requirements, even late in a project. Not applicable, because of the INFLUENCE CASCADE.

Assembling the project team and business owners on a daily basis throughout the project. Useful, since often the business owners will have a different perspective on the data.

Having the team reflect at regular intervals on how to become more effective, then tuning and adjusting behavior accordingly. Applicable if not done in a very rigid fashion. While it's important to have retrospectives, their cadence should be timed with achieved milestones, so difficult to have

pre-determined, set in stone intervals.

Measuring progress by the amount of completed work. Not applicable, since the data projects don't have an absolute value, just relative one. Progress is measured by business value.

Continually seeking excellence. Applicable, self-explanatory.

Harnessing change for a competitive advantage. Applicable, self-explanatory.

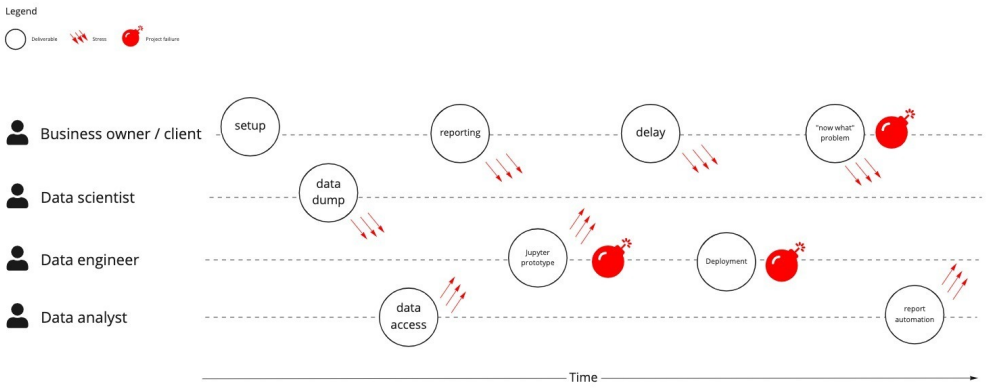
Now that we know what are the main principles of soft agile we can apply the *via negativa* method once again. Illustrating how agile fails for data projects is the best way to learn it.

The Implementation Forest

Any person who has some experience working in data and analytics knows that the reasons for complexity of such projects are numerous, and I can fill a whole separate book recounting them. But the best way to illustrate the issues is with a story. Many of you will recognize most if not all patterns in this story, and probably it won't bring back good memories. Still, in DELIVERY METHODS I'll provide tools to help you ensure this remains just a story.

Remember the concept of "atomic team" from DESIGN? Let's take one such typical data team and assign them to a data project. The data scientist, engineer and analyst are responsible for the implementation and report to the business owner, who is also responsible for the project management. This is a brand new data project, starting from scratch with a new exciting use case. The team has received the brief and is ready to go, and they start with a two week sprint.

The diagram below gives you an overview of what happens next:



The implementation forest in action.

We have the different roles on the left side, and time progresses from left to right. The deliverables are shown in white circules, the red arrows correspond to stress to the system and the bombs, appropriately, for project failiure. Let's play the scenarion through the different points in time to see what's going on.

Setup

Data projects can fail already at the start, when we are still setting up the project. The first possible mistake is to have requirements which are not specific or clear enough. I have witness whole centers of excellence setups starting out with the basic requirement: "let's do an AI for X", and you can substitute X for anything, such as improving the patient outcome, warehouse operations or new sales funnel performance. If you don't specify your goals properly, even with the best of teams the most that you can hope to get out of the data initiative is for them to dig a perfect hole in the wrong yard. Also, pivoting mid-way through the project is not a viable strategy. As you have seen in the `INFLUENCE CASCADE` any changes have big consequences,

and the project setup (in terms of people, hardware, software, access to data) needs to take care of that.

Data dump

As the project starts, a common mistake from the setup becomes apparent. The data scientist has set-up their local development enironment and are ready to go, but they realize they don't actually have access to data. Now they need to find the right responsible person or team who is owning that data, and even if they do they can find out they first need to ask their project manager to do negotiation. Data governance policies are rarely set-up in a flexible manner in large organisations, so you can imagine this can take weeks. Even if this is not the case, they need to talk to the data engineer to provide them with a dump of the data so they can at least start working. This is also rarely added to sprints as a ticket, and the data engineer instead of working on their own tasks needs to drop everything and export a dataset for the scientist, since otherwise they wouldn't be able to work.

Data access

At the same time, the business owner has asked the data analyst to provide some initial reports on the data. Similarly to the data scientist, they realise they don't have access to the data and get in touch with the data engineer as well. Unfortunately for the data engineer, a data dump is not enough for them. The analyst needs real-time access via a self-service analytics tool. More ad-hoc work for the data engineer.

Notebook prototype

After obtaining a data dump from the engineer, the data scientist has been working on creating a prototype ML model. They developed locally on their machine and are ready to provide the code for deployment. Usually, the data engineer is also responsible for this, and unfortunately again (at this point

you should start to see why good data engineers are so saught after) they realize they need to recode the modeling code from R to Python Harvinder[26] refers to this issue as the "throwing over the fence problem".

Deployment

At this point, finally the data engineer has the modeling code written in the language they need. After deployment the team realises that the data which is coming in to the ML system is different, both in terms of format and quality. Now there's a need to set up the MLOps infrastructure and database connections (the scientist has been working with flat files) to ensure continious learning and performance monitoring. The product team also complains, since the several seconds prediction latency might be noticed by the users, and that it needs to drop below one second. This news change the whole development process and the whole infrastructure and modeling code need rethinking.

The so what problem

Now we are reaching the final, and perhaps most difficult challenge. Even if the project has somehow survived till this point, they have this final hurdle to face. They realize that they have built and deployed a good model, but make it accessible only via an API endpoint, and not user interface. The product team realises that they don't have enough technical people on their side to consume this API. An additional problem is that it starts to be unclear which system needs to use this product, and how are the results stored. Finally, business users are distrustful of the accuracy of the model, and start to ask about details in understanding the model. The data scientist realises that they should have created an XAI layer on top of the model to convince the business users. You can see that even with a good finished product there are new challenges that can arise that can derail everything, and we pay the

price for the bad setup - and the lack of data strategy.

This story is admittedly rather bleak, and many of you will have had similar experiences. Still, with the right tools most of those problems are easily avoidable. I'll provide them in DELIVERY METHODS, but before this let's have a look at the other framework for delivery.

Lean Data

Lean Data Theory

The lean methodology has it's origins in industrial manufacturing - the legendary japanese Toyota car making factories, with the Toyota Production System[27]. It's a bit further away from data science than Agile, since the latter is very wide-spread in software. We should take this into account, the application of lean methods in data science can seem a bit more abstract than Agile, requiring more effort from the data strategist. This methodology has been greatly popularized in the technology world by the Lean Startup movement[28].

Reducing waste is at the core of lean. As I explain the concept, you'll realize that it fits well the *via negativa* method: instead of an optimistic goal - "how can we have more resources", we are focused on a more pessimistic, yet actionable one - "how can we do more with the limited resources that we have".

The Knowledge Factory

Which gap does lean allow organizations to bridge? To answer this question, we need to understand the golden standard of a lean organization, something I call the "knowledge factory".

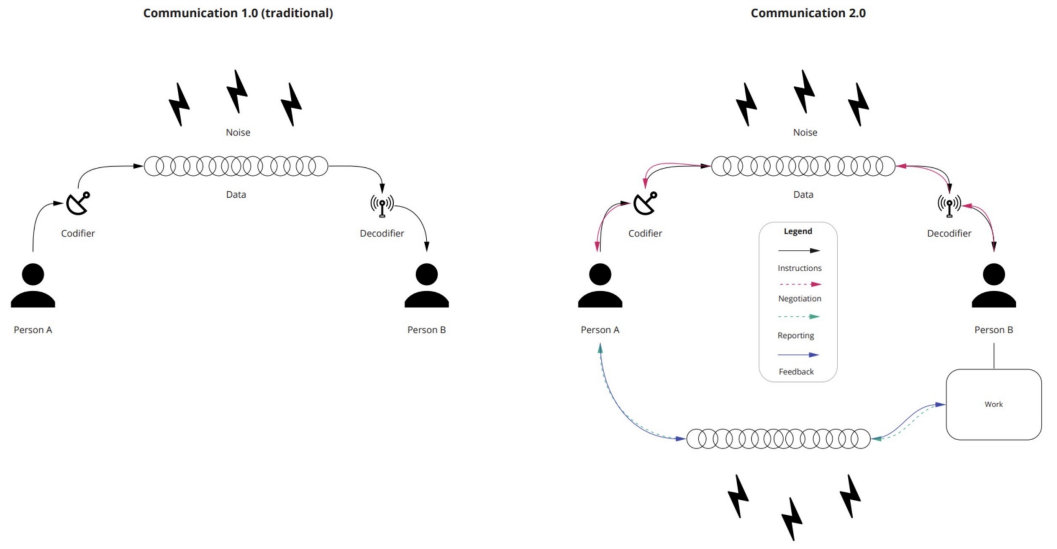
Since a lot of the inspiration behind the lean methodology comes from the industrial sector, it makes sense to answer a common question on the topic: how different is knowledge work from more manual labour? This is one of the most covered topics, arguably the sole purpose behind the whole field of operations research and management science. I'll aim to cover this in a few brief paragraphs.

First we should talk about what makes knowledge work different. There are several elements to this, that can neatly fall into several buckets:

Complexity of the work. The world "complexity" itself is confusing. There is a whole field of research dedicated to the topic, and this is a frequent topic of discussion among academics. For the purposes of our work, we can define a complex piece of work as one that cannot be understood at 100% at all points in time. And this is the very nature of knowledge work. It is often the result of enormous mental effort, frequently committed by a group of people, over a sustained period of time, and is always changing[^complexity_frontier]. This makes it almost impossible for a single individual to understand it fully, and they have to treat some elements of it as a black box. This is the number one reason why it's hard to build a knowledge factory.

Challenging communication. With the growing specialisation of the knowledge work in general, it becomes less and less common to have employees with the renaissance-man skillet - where they have achieved mastery across a widely different set of areas. Nowadays it's more common to have a group of specialists, that function together as a whole. And with the increased specialisation, those new fields bring their own terminology, and their work can become harder to explain from specialist of even adjustment in application areas, let alone those widely distant. This relates very closely to the increased complexity of the work - more complex work

is by definition harder to explain. It's clear that good communication is essential for productivity, and thus the highly complex data work can be a victim of communication difficulties.



Communication complexity

As one of the most influential thinkers in the field of organisational management, Peter Drucker wrote: "What gets measured, gets measured"[29], we are often confronted with the difficulty in measurement within data projects. A first question that you might have is, what do we actually want to measure in data projects? Let's provide two examples: measurement of how long a certain task takes, and how successful it's product can be.

Measuring how long a data task takes. Traditional software projects themselves are very tricky to be estimated, but for most of those cases there's some kind of deterministic outcome desired, and one can normally rely on the successful sequence of building of components on top of each other, much like building a house. For data projects this is very different, since the very possibility of building some of the components is in question,

and they are much more prone to failure due to the *probabilistic* nature of the work. For example, how do we estimate how long it would take us to finish a machine learning project (from conception to production) when we are not even sure that we can successfully build one based on the data we have? And related to that - how do we then estimate confidently when there is a high chance that the data that's collected is not of sufficient quality, and we have to re-collect it? In a software project if a component breaks or is unavailable, normally we would have quite good variety of replacement parts. This is not the case for data projects.

Measuring of success of data projects. This is another tricky issue, which has also been similarly challenging for traditional IT projects. Even if we manage to successfully implement and deploy data projects to production, how can we measure their success? How do we know that we actually succeeded? Some of you might say - what do we care - the project works. But in larger organisations, where millions of dollars are spent on data projects, it's simply not enough. We have to come up with an estimate on whether the data work yields concrete results. I'll show you how to do this in `IMPACT ASSESSMENT`, since we have to apply it to the data strategy itself.

Now that we know the typical challenges that make data projects (and knowledge work in general) harder to manage, what are some potential solutions to them? Here are the advice points that any analytics manager or strategist should keep a note of:

- Measurement of how long a data project (or a subtask of it) takes.
- Have shorter iteration cycles
- Add the uncertainty in the planning
- Check-in more often
- Pay attention to cross-functional synchronisations

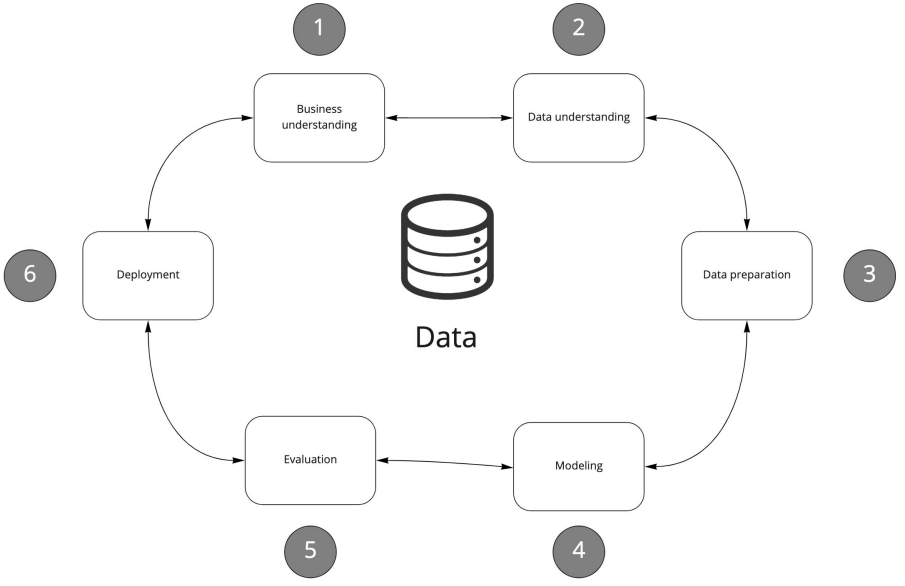
- Measuring of success of data projects.
- Make sure the data project is tightly integrated within the larger IT architecture. This should also be baked-in into the design of the data project from day one.
- Work closely with the consumers of data projects and estimate the ROI with them.
- Adjust the measurement frequently based on feedback from various stakeholders.

Now we can go through the ideal state of an organisation relying on knowledge work. So what is the knowledge factory then? The knowledge factory is the holy-grail of knowledge work. All of us can probably agree that a physical factory can be optimised to almost perfection (especially if it's largely machine focused, such as the new Tesla factories). There will always be some waste in the operations, due to the nature of the universe, but those inefficiencies would not be based on the design of the system and it's components. All elements are functioning based on their specifications, their interactions are perfectly synchronised, and there are zero bottlenecks and delays. Now, try to image having the same concept for knowledge work - that is what I call the "knowledge factory". This is probably an unattainable goal, but still one to aspire to achieve.

The only thing which is now left for us to do is to see how we can bridge the gap between the North Star of the knowledge factory and the current state. And the best place to do this, as you by now should have been accustomed to in this work, is the application of *via negativa* principles - let's have a look at what the typical sources of waste (inefficiencies) in data projects are and try to eliminate them.

Delivery Methods

CRISP-DM



CRISP-DM process (modified)

Knowing the hidden dangers of implementing a cargo cult, we should proceed with caution when applying agile to data projects.

Kanban

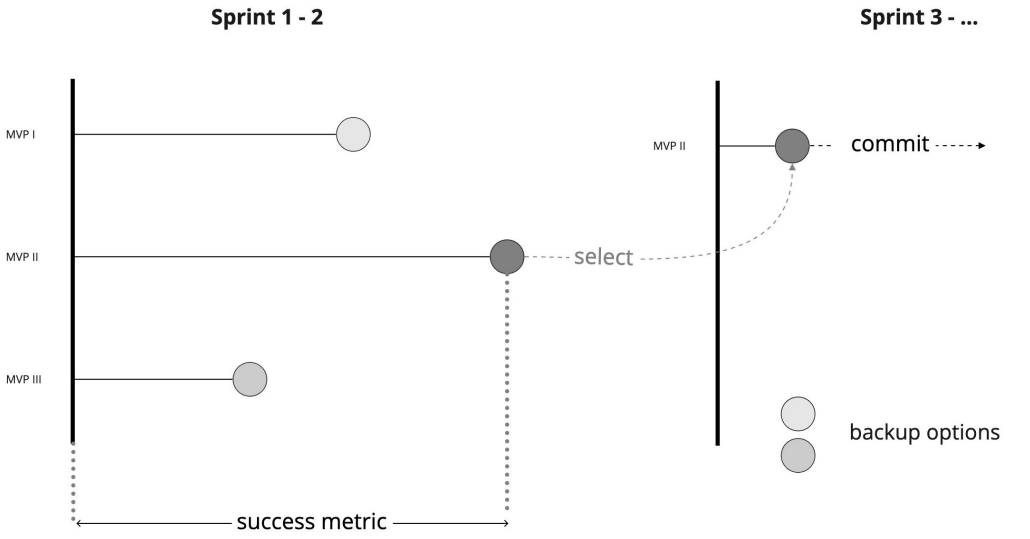
Scrum and Scrum of Scrums

SAFe

Shotgun MVP

There's a method to address complexity when deciding for which use case to go while still alleviating the uncertainty behind planning MVPs - the Shotgun MVP. As a first step, several MVP initiatives are launched simultaneously (staffed to the bare minimum needed). Thus, all possible approaches are covered. This process continues for several sprints until a clear frontrunner becomes apparent - based on a "success" metric. There is flexibility in defining this metric, and possible examples include performance metrics (i.e., model accuracy) and solution complexity (such as the workforce needed to complete further development iterations). After this MVP (MVP II in our case) is selected, resources are committed fully to it for the following sprints while keeping the other options as backup plans or possible enhancements for downstream work.

I call this concept "Shotgun MVP" and it's illustrated below:



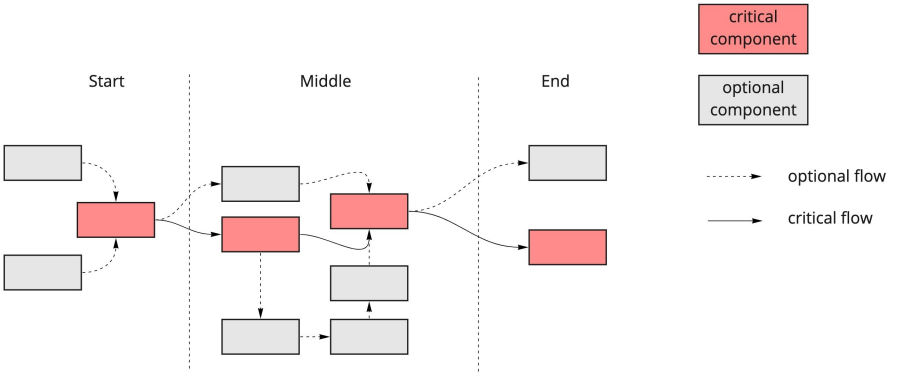
Shotgun MVP

Closing the Loop

When facing complex products (which is often the case in data), it can be overwhelming to decide on how to start. One book which has been a great inspiration is *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*[30]. This rapid iteration approach is very useful in dealing with the overwhelming complexity of the task at hand.

I have summarised my interpretation of the rapid ideation approach in the "closing the loop" concept. I like this name better since the name can be sufficient in explaining it.

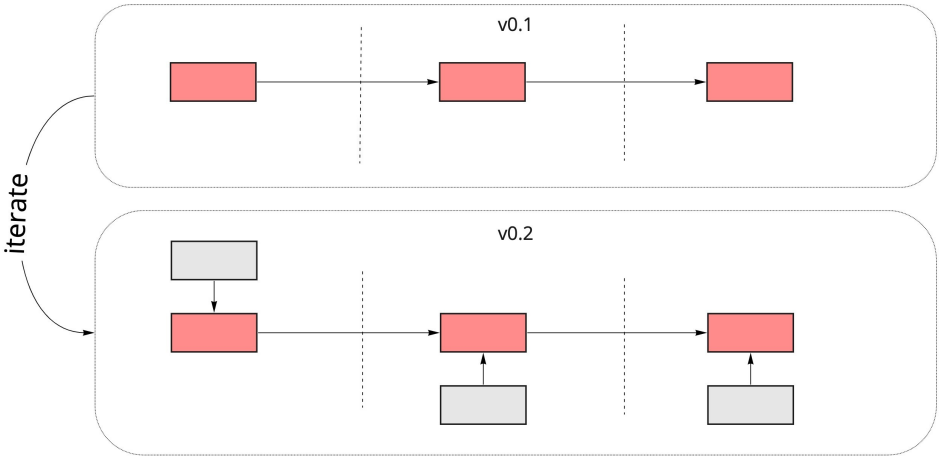
Let's first have a look at how a complete product can look like, at the figure below:



Closing the loop I

Most of the time we can visualise a digital product by mapping its elements onto a customer journey. This journey often consists of several phases, in this case start, middle and end. For example user authentication and landing pages are elements normally found at the start, while payment processing is often the last step of a flow through a product.

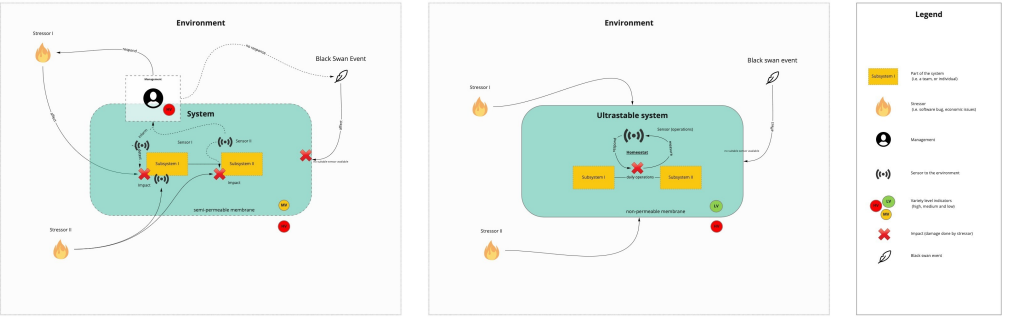
We can add more detail to this representation by colouring splitting the critical and optional components, and separate the different flows between them as well. This separation results in the most critical, end-to-end elements and associated flows. We can take them and close the loop:



Closing the loop II

With this approach in mind, instead of tackling the complete product, we decide to focus on the most necessary parts, and their relationships first. Only then we can iterate and add the other elements. At that point, however, that would be a much simpler task since the foundations are present (albeit rudimentary). An additional property of using the "closing the loop" approach is that having a basic end-to-end process implemented, we can already deploy our solution to customers and measure its success.

Homeostatic Project Management



Homeostatic project management.

Sources of Waste

In the beginning of this part of the book we visualised a common scenario showing where data projects fail - the "implementation forest". In that single scenario there are several things going wrong at the same time, so it makes sense to take a step back and structure the causes more. Harvinder Atval in his brilliant book *Practical DataOps: Delivering Agile Data Science at Scale*[31] has done a great job in explaining the sources of waste which are typical for data projects. Lean revolves around removing those sources of waste

We can start by having a look at the traditional sources of waste in manufacturing, and then cover the specific translation into data. This mapping can help us understand the concepts better. In traditional manufacturing the sources of waste are:

Overproduction: Basically having too many things. It might sound a bit counterintuitive to think having too much of a given product is a bad

thing. But most products nowadays are complex combinations of different components - and having too much of one can cause delays in processing downstream.

Inventory: Bad tools are a force multiplier for decrease in productivity.

Motion: If a worker needs to walk an extra step than necessary, efficiency decreases.

Defects: Sub-par products that need to be discarded - this wastes valuable time and resources.

Waiting: This corresponds to underutilisation of human resources.

Transport: If moving different components in the pipeline is too slow, it can become a bottleneck.

It should be relatively straightforward to imagine how those occur in an industrial setting, but how about knowledge work? Let's have a look at what Atval proposes:

Partially done work: This relates to the *pilotitis* problem that we described in Part I. It's easy to start new projects, but to make them actually usable requires much more effort.

Unreproducible work: the "it works on my machine" problem. Non-existent or rudimentary dependency management and absence of DevOps and automation.

> **The lab notebook**. The documentation of data projects, especially those which are heavy on the exploration and modeling work is by necessity different from that of traditional software. My time in a molecular biology lab as an undegrad gave me an idea that has proven useful in my projects: we should use shared lab notebooks. The idea of a lab notebook is that it is an

immutable, continiusly updated log of your work, including experimental results and notes. The most important thing is that it needs to be time stamped, and shared with your team. Have a look at an example below:

> There are many software tools to assist with this, including Confluence and SharePoint.

Defects: No code is perfect and there are always edge cases in software.

Vague definition of done: The result of communication and planning issues. Since data projects tend to be more complex than traditional software and their novelty introduces a multitude of different opaque terms, it can be harder to write good requirements. This is one of the main reasons for the "so what" problem in the Implementation Maze that we described in the beginning of this part.

Multitasking: This relates to the idea that human productivity tends to dramatically dropped if attention has to be split between different tasks at the same time. This is often the result of "scope creep" - additional, unplanned work being assigned to the team during a sprint.

Extra features: Instead of focusing on the fundamental components of a data system and only then iterating, the engineering team attempts to build everything at once. This can result in delays, frustration and other issues. A solution to this source of waste is described later, called "Closing the Loop".

Waiting. This can translate almost one to one from the industrial setting to data. Especially in larger organisations it's common to have more tedious processes and inefficient data governance, resulting in long waiting times for the data team to even proper tooling, let alone access to data itself.

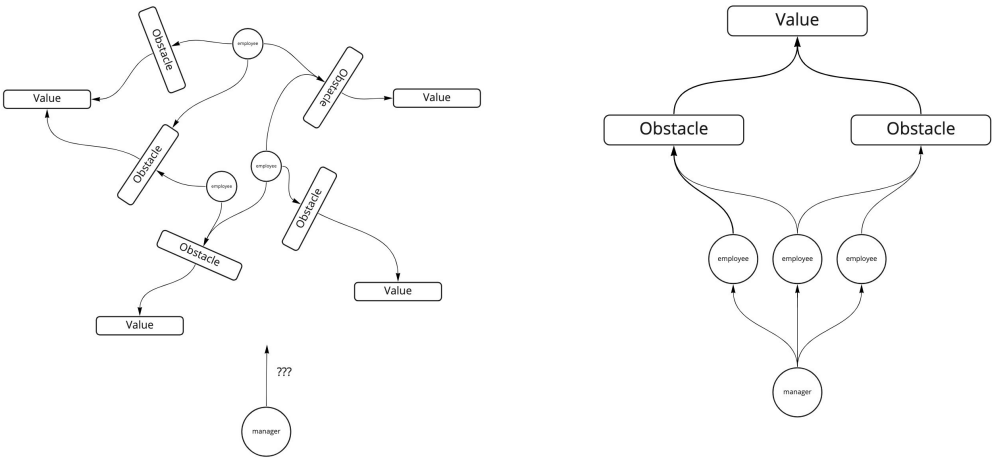
Lack of knowledge sharing: The result of siloing of data team work and members. In larger and less efficient organisations the worst form of this issue appears - different teams working on the same thing, reinventing the wheel in isolation.

Extra processes: This concept relates to the idea of "soft agile" that I described in the previous section. It occurs when the balance between rituals and work is leaning too much on the former than the latter, normally because of the cargo cult effect.

Extra motion: A common source of waste in data projects, especially by more junior members is overcomplicating the system. For example it's very easy to install an open-source package, and then just use one function from it. This can seem productive at the beginning, but introduces unseen technical debt in the system, since this package needs to be taken into account for dependency management and containerisation (see the "unreproducible work" source of waste).

Lack of documentation: Unfortunately in many cases documentation is still seen more as an afterthought of work instead of actual ticket. This introduces tremendous fragility in the system, which can acutely be felt when a senior member leaves the team, taking valuable, and even critical knowledge of the system with them.

Heuristics



Reality versus expectations.

Simple and easy-to follow rules often prove more effective, especially if the organisation which is trying to implement the data strategy is at a low data maturity stage. Before it's members can commit to an extensive (yet often complex) strategy, they need to first warm up to the ideas and processes. They also need to see results quickly, to stay motivated that the chosen path is right. A Harvard Business Review article on strategic heuristics explains this idea well. In this article the authors explain how Napoleon innovated in the art of managing a war during the conflict with Russia in the 19th century. At that time the lines of communication between the generals and the front-line soldiers was virtually non-existent - this was quite some time ago before the invention the telephone. Wars started to become also more complex, even because of the sheer size of the armies involved. The relay of

orders and information to and from the generals was mostly accomplished by messengers - often riding on horses. It's not difficult to imagine how such inefficient communication channels result in confusion - by the time the information has arrived, it is already outdated. It's at best irrelevant - and at its worst - dangerous. On the battlefield this can make all the difference between victory and defeat, perhaps even more so than the size and quality of the armies involved. To avoid this, Napoleon issued his troops with a set of simple heuristics: in the case of total communication breakdown, go into the direction of fire and take the high ground.

Heuristics can be useful for people at the executive level as well. I have often observed analytics leaders applying basic rules to their work, especially in terms of time management. For example when asked how they prioritize different work areas, such as hiring, outreach and development work, they would often come up with a rule of thumb percentage, such as 10-15-75% in this case.

Let's use another example to additionally illustrate this point. After years of efficient growth, the electric car company Tesla started to experience issues. Some of those were due to the difficulties in innovating in the space, and the need to invent radically new technologies and production methods, but some were due to simple communication issues and human bureaucracy. Noticing this, the CEO Elon Musk decided to take decisive action. He wrote a letter to all employees listing a few simple heuristics. Some of them included guidance on how to take decisions - if one needed to get the permission of more than one senior person, then something is wrong, and Musk had to be informed.

The challenges of modern enterprises are rarely of the magnitude and severity as the Napoleonic wars, or have the complexity of Tesla, but still the same

basic rules apply. To take the first steps in a data strategy implementation in a large organisation, basic initial heuristics can be of great service.

Adaptive System Design

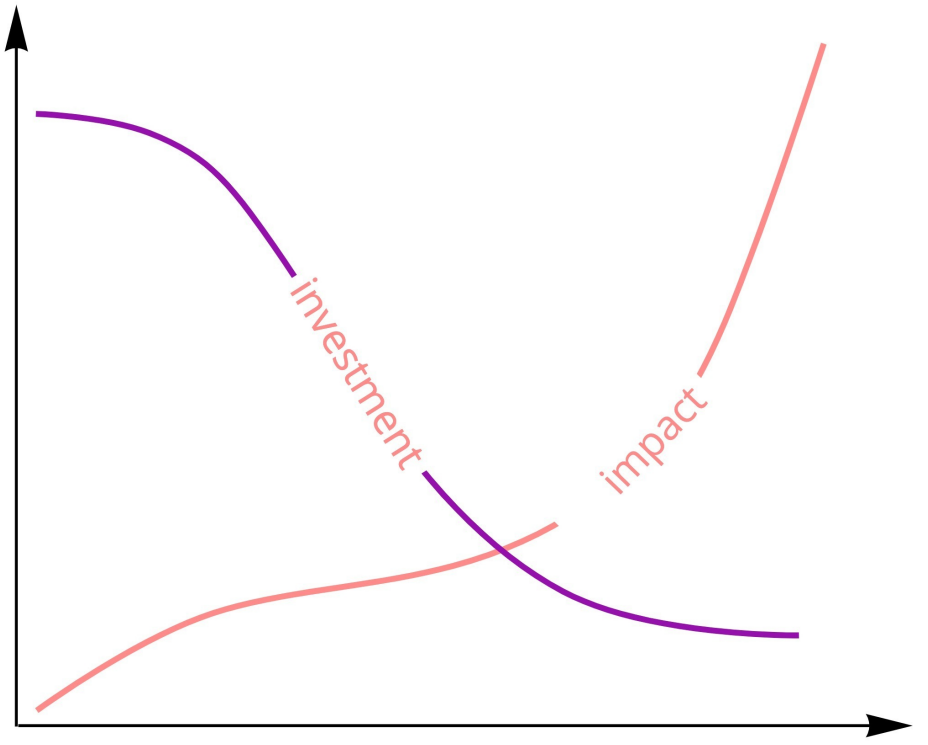
Viable System Model

Value Stream Mapping

Impact Assessment

Before we wrap up the book, there's one final question which we need to answer. Yes, we have designed and delivered a data strategy to the organisation, but how can we measure it's success? Of course, StratOps and agility have been guiding principles throughout this book, and this final section will follow that pattern.

Before we dive into how we actually can asses the results of the data strategy delivery, I would like to elaborate on why it's a good idea to do so. Have a look at the following graph, to appreciate the source of frustration on the executive level when it comes to data projects:

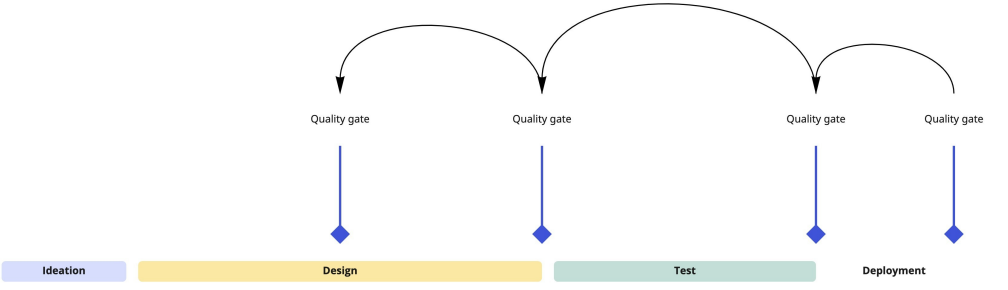


Investment - impact curve.

Basically, things take time. This is the opposite graph of *pilotitis* (in Part I), and actually a healthy view of a successful data strategy initiative, especially for large organisations. We are always building the foundations first, and it would naturally take time before the results of the investment start to show tangible results. Thus a time, normally around the middle of the data strategy initiative, comes where the investment has been tremendous, yet the results are barely noticeable. There's another great drawing that is common, where a person is digging a tunnel to get to the diamonds, and slowly gets more and more frustrated, just to quit meters away from the target. Thus it's important to put trust in the team, measure the right things so that the impact is assured.

Countless books have been written on how to measure the success of of technical projects, some methods better than others. I would say that for data projects a lot of similar logic applies as the one to traditional software projects, with some caveats that we covered in the Agile and Lean sections.

Metric	Description
Time to Market	How quickly the product is deployed in production
ROI	How much money the product makes as a return on investment
Ramp time	Time for a new hire to become productive
Deployment number	How frequent are deployments
Actionable insights delivered	self-explanatory



Quality gates of impact assessment.

Portfolio Management

In DUE DILIGENCE, while gathering information during the CSA, a major focal point was to determine all data-related use cases in the organization - its analytics portfolio. Even small organizations have a diverse pool of existing use cases, or data-generating/consuming systems - let alone the larger ones. From the USE CASES section we have probably added even more use cases to the strategy, filling the roadmap. The complexity of such a landscape requires us to dedicate time in structuring it. This activity falls under the scope of portfolio management. Here's a good definition from indeed.com:

Portfolio management oversees all projects and programs in an organization, but focuses on the overarching goals and how projects and programs align to those goals.

Why do we need it? Here are the main motivating factors for this element of data strategy:

Visibility: It's essential to have an overview on all data activities in the organization. Not only does this make all other management activities easier, this also ensures that well running projects also get some additional recognition as a reward.

Alignment and coordination: When a wider audience in the organization has access to information on the inititiatives, they would be able to ensure that all portfolios don't have contradicting goals, but work together. Such an overview makes it much easier to plan new initiatives down the line,

and align the different timelines into a general organization-wide roadmap and manage budgets and resources. And finally, any potential dependencies (technology, architecture, resources etc.) are also taken into account - so there's consistency as well.

Not reinventing the wheel: If there's no overview, a common issue for data analytics portfolios is doing the same thing all again. Good example can be, creating a custom data processing pipeline, instead of using one already built for another use case.

Each use case will need several points. Here's a table providing a specific example:

Data point	Example
Topic	Customer 360
Technology	Python, seaborn, PowerBI, scikit-learn, flask
Architecture	S3, Refshift, Sagemaker
Data	CRM data, tabular data
Roles	John Doe, data strategist; Jane Doe, data scientist
Status	In progress
Metrics	Click-through-rate, churn rate

Summary

In the third phase of the Data Strategy process we learned how to ensure its successful delivery. Hopefully the reader understands that it's not enough to create a static document describing the plan - it needs more work to be implemented, and due to the huge complexity of the real world application, it also needs to co-evolve, and be continuously re-adjusted based on the circumstances. In many ways this is the hardest part of the data strategy process - this is where the rubber meets the road, and where the true impact of this work is realised.

We learned what are the key contributing factors responsible for encountering difficulty in applying data strategy work. Some of those factors are political, and others - operational and organisational. Before we can alleviate their effects, we need to identify the source, and as we mentioned this is probably the hardest work that a data strategist needs to do, since it involves the most complex system of all - the system of humans.

We went through the two fundamental frameworks in how to deliver a data strategy - agile and lean. The origins of those approaches lie in other industries, namely software development and industrial manufacturing, and they do need some adjustment to become applicable in the field of data. But still, when applied with care they can bring tremendous benefits in the successful delivery of a data strategy.

[^complexity_frontier] Anna Filippova from dbt has a great example of this in her *"Complexity: the new analytics frontier"* article, that you can read here.

Conclusion

As my parting thoughts I want to do a full circle and go back to the Zen saying in the beginning of this book. The *Elements of Data Strategy* is just the finger pointing to the moon - not the moon itself. It is up to you, the dedicated practitioner to take this work and make it your own - and come up with new, and better ways to do data strategy.

As you emark on this journey I have just one ask of you. As the last years have shown, there are still great challenges facing us - pandemics, global conflict, inequality. All against the backdrop of climate change. If you can, with your data work, try to use it for good, and the benefit of the rest of us. A small drop makes a canyon.

Boyan Angelov
Berlin, 2022

Interviews

Nicolas Averseng

BA: Boyan Angelov
NA: Nicolas Averseng

Nicolas Averseng is the founder and CEO of YOOI, a data analytics and management platform. He has an extensive experience as a CTO and other leadership positions in a variety of industries, solving challenges with data.

BA: Nicolas, it's such an honor to be able to discuss data strategy with you. I think you have one of the most forward-thinking visions on the field, and I'm sure the readers can learn a lot from our conversation! Let's start with a simple question. How did you end up being involved in data strategy?

NA: My background is in engineering, and I spent most of my career on the software vendor side, and I had a chance to work for smaller and larger companies. I was always able to mix being 50% in the "cave" building software and then, at the same time, work closely with customers. I always enjoyed being technical, but I would say that while I love technology, I also have an issue in doing it just for its sake. I always like making stuff people use and this is what drives me. I got into data through a company that I joined 15 years ago, an innovator in the monitoring space. Monitoring is all about real-time capture of vast amounts of information, and analyzing them versus historical knowledge and rules, in order to support decisions, often in real-time. One of the things we have been pioneering is around

monitoring business processes, which is now known under the fancy term "operational intelligence". Four years ago, I joined a consulting company as CTO. This company was mostly staffed by a large number of data scientists, and I saw my job as reinforcing and developing the technology side further. Eventually, we wanted to help people with not only the data science part of the work, but also the other elements of data strategy: the data platform, architecture and production environments, pipelines, and so on. I built the technical teams there while working with large enterprises - also with the same angle as what I mentioned - helping them avoid the mistakes of thinking too much about the "how" without having a clear view on the "why". With this experience, I started YOOI, aiming to solve this problem.

BA: This is a very interesting point. I would agree that the most frustrating thing in data I see is when clients have invested a ton into people and resources - and spend it on digging a perfect hole, but in the wrong place.

NA: Yes - and this relates to why there are so few successful companies successful with ML. They don't manage to deal with the most basic - but also hardest - part, starting with the why. Nothing else matters if you don't start with what you want to achieve and the business question. It might seem obvious to many of us in the field, but this is a very common issue. People would often come to me as a technical person: "We need to do X." And my first reaction is always, "Why?". The first thing in a data strategy is aligning the people on the ultimate goal and success criteria. Only when we have that can we engage people around that same objective. Only then can you work on the other elements, such as technology, processes and culture. This potential misalignment is indeed the leading cause of failure. A good illustration is what has happened often with data lakes: enterprises have invested a lot to build infrastructure and then expect that it will solve all their issue and then might struggle to actually build and deploy ML use

cases. To make an analogy, if you have a big enough hammer, in which you have invested, everything starts to look like a nail. And in the end, you might find out what you really need is a screwdriver.

BA: Can you define what a data strategy is in your view?

NA: Data strategy is the way to define what to do with data, in order to support the business strategy of the company. It should always be focused on supporting the business goals. A data project should have the same goals and metrics of failure as a regular software project. Of course, there are additional dimensions to data projects, such as the data itself, which add to the complexity and uncertainty.

BA: So, do you see some significant differences between software and data projects?

NA: There are some, but the biggest one is the uncertainty of data - this proves to be challenging to many people and organizations, as it makes the whole value chain more complex, more fragile. Another related source of uncertainty is how to involve people in the process. This relates to one of my favorite concepts in software, the three U's: usable, useful, and used. While this concept applies to all software projects, it is especially challenging for data products. In the beginning, let's say people do sales forecasting "manually". They do a great job, but it does not scale, and they cannot focus on multiple product segments. The job, in this case, is to build a good enough model that can automate this part of the work. But once done, they discover that they don't know how to deploy and use this model effectively because they did not set a measurable goal or think about how to make it actionable within their business process. They forgot why they were doing this work in the first place.

BA: Couldn't agree more. If you have a solid "why," the rest of the data

strategy work takes care of itself. Let's now talk about YOOI and the purpose of your offering.

NA: I want to talk and expand on the three U's first; this will help explain the purpose of YOOI. Why I like this concept is because it's so simple. Often people forget that people need to use whatever they built. How is all of this going to be used by people? How will it be integrated into real processes and drive real decisions? Even if you put a model in production, people would still not use it because it behaves like a black box. This is also why you need to be build trust and metrics in your data projects, and engage users all along to make sure they understand and buy into those. This is why we built YOOI - we see it as a cockpit for data strategy. A place for the team to align all those different dimensions, connect the dots and make sure the technology is meaningful. Of course, you can always hire a great consultant like you -

BA: *laughs*

NA: - but all the great work might end up sitting somewhere on SharePoint, gathering dust and not being used. People will then lose track of why the work is done, and also, you can't repeat this data strategy work every year. It has to be a living, continuously learning system. What happens to the data strategy in a month when a new technical requirement comes up? The world is changing, and this is something we need to accept. At the same time, we have to keep everyone aligned on the same goals, and this is why you need a tool for this. Our tool is a combination of process and visibility elements. It allows to make sure when the ideas are proposed, there's sufficient information to make decisions on selection, budgeting, and technology. This is a view of the complete value chain. Now there are so many different tools available from the cloud that make doing data projects easy. What is still missing is this monitoring part, bridging the gap with project execution.

You can learn more about YOOI on the official website. Follow Nicolas on LinkedIn.

Summary

- Start with the why
- Data strategy has the same goals as the business strategy
- Prioritize, monitor, and focus on delivery

Noah Gift

BA: Boyan Angelov
NG: Noah Gift

About: Noah Gift is the founder of Pragmatic A.I. Labs. Noah Gift lectures at MSDS, at Northwestern, Duke MIDS Graduate Data Science Program, the Graduate Data Science program at UC Berkeley, the UC Davis Graduate School of Management MSBA program, UNC Charlotte Data Science Initiative and University of Tennessee (as part of the Tennessee Digital Jobs Factory). He teaches and designs graduate machine learning, MLOps, A.I., Data Science courses, and consulting on Machine Learning and Cloud Architecture for students and faculty. These responsibilities include leading a multi-cloud certification initiative for students. (source: https://noahgift.com)

BA: I was listening to your recent podcast on DataFramed, and I loved it. While listening, I was thinking - this person certainly has unique opinions that need to be heard. I like your skepticism, and I believe this is even more important in a field such as ours, where have more than enough buzzwords.

Maybe we can start with your background. How did you end up in data?

NG: I first started in TV and film. They offered me a full-time job, and that was it, the start of my career. But I just wanted a little more than that, and I always wanted to make sure I got a degree, and I decided to go to Cal Poly San Obispo. I was interested in being a professional athlete, even perhaps going to the Olympics, playing professional basketball. This is why I studied nutritional science. I thought that was a good degree to learn more about performance. And what was good about it is that nutrition science really is a form of data science. All the courses you take, such as organic chemistry, are very architectural in nature. Anatomy, physiology, and even dissection can be seen as data science. You are inspecting the body and looking at the parts, and seeing what they do. I also did experimentation on my own body, centrifuged my blood, took doses of Vitamin C. After this, I briefly pursued being a professional athlete. I was in the process of training after college to play basketball, not NBA level, but lower-end tier. But then also applied for a job at Caltech in Information Technology, since I thought that I probably won't make too much money doing the sport. I spent a few years there, learning a bunch of stuff about Unix and Linux, learned Python. Right after I spent three years there, I decided to go back to the film industry. A lot of the stuff I learned at Cal Poly helped me make film pipelines. Film pipelines and data engineering are essentially the same thing! After this, I moved to the Bay area and worked at startups for roughly ten years there. Since then, I've been consulting, teaching, writing books.

BA: This is a fascinating background. It makes me think a lot about systems thinking. The way you make those unique parallels between different fields - film, sports, and data.

NG: Yes! And even machine learning is very similar to film because we've been doing distributed computing in film for a very long time. You have to

set up each job on a different node, and each does a separate piece, and in the end, you must combine them.

BA: Interesting! My following questions are related to the challenges in management consulting, working with large European companies. One can even say that some of those companies are even further behind their American counterparts, in terms of their digital transformation journey. I believe the remedy for this is to design a data strategy. As a part of this process, a data strategist such as myself would be sent to the company and start to help them with what type of people they need, on which use cases they should work, and so forth. Can you tell me your thoughts on this? Why do some big companies fail and others succeed?

NG: I think the issue is that many organizations hire academics - research-oriented people. With research, you are not focused on production. You are essentially hiring the wrong people to work for the company. They might even be solving the wrong problem. Don't get me wrong - it's great to have researchers available in some situations, but most companies need operations. I think it is important to have a data strategist. Ultimately with MLOps and data engineering, the thing you're building is not a model or data but a pipeline. It's almost like the name of the discipline itself is incorrect - if you say, "Hey, I'm doing data engineering", ultimately you mean you're building a data pipeline. And it's the same with machine learning. So what is a pipeline? It's dynamic; it can expand and contract or react to different things. So you have to build the capability to respond to things dynamically. That's the opposite of a researcher. Research operates on a fixed problem that's constrained to a lab environment. The pipeline should constantly improve and produce results.

BA: And it should be measurable - this is another crucial property. Interesting, I know of two common analogies about data work that I also reference in

the book. One is this oil processing one, where we talk about data gathering and enrichment. The other one is the kitchen analogy. Where you have the raw data in terms of ingredients, and you have the recipe. What I find strange in the approach of larger companies is hiring some people on high salaries and telling them - let's do AI in X, where X can be anything. They make the team, set up the roles, provide some data, and say, "let's go, talk to you when the results are ready". But do you think we can do better than this when planning data projects in a large organization? How would you even start to think about such complex work?

NG: I would say that the oil pipeline analogy is pretty good. I think oil processing exploded in significance in the 1920s when cars as a means of transportation started to become more popular. Imagine a geologist in Texas in those early days. They come to the site and start drilling - oil squirting out of the ground. That could be good enough - if you are a researcher. You could say: "Look at this, we found some oil!". They're just taking the oil in a bucket and throwing the thing in the pot, spilling dirt everywhere. The result of this is that maybe enough oil for running a car will be produced once a day. That's a metaphor for how data science works nowadays. Now, what's the opposite? We can build an oil refinery. That is a complete platform for oil production, with people and components working on different parts subsystems. The result of this is that we can produce many more unites per day, to run our cars. It's the complete opposite. If we look at an organization that wants to be successful, the first part is they need to realize that the majority of people will be like that person, digging the hole in the ground and making a mess with the oil spilling everywhere. Instead, you should use a platform, like AWS Sagemaker, Azure ML Studio, Google Vertex AI, maybe a third-party tool. Use that platform and have everything standardized. The goal isn't to play around with the oil. The goal is how

many gallons of gas we produce per day. Similarly, if you force the data scientist to work inside a platform that has strict rules, then you've already made it much more likely they'll be successful. A second component is also if your oil refinery is producing diesel, and all your vehicles are unleaded gas, well, you're making the wrong type of fuel. That's the other part of the problem. There must be requirements that are mapped to the executives in the company. Even if you're using a platform, you have to make sure the people build the right thing.

BA: This reminds me of another idea I'm developing - starting with the end goal in mind. One of the worst ways for data science projects to fail is when a bunch of intelligent and hardworking people spending a ton of effort on the wrong thing. Nobody asked why. We can use all the platform tools, let's say AWS Sagemaker. We can make a pipeline - but in the end, the business unit often says - ok cool, we have this AI, but so what? How are we going to use this? What happens now? Sometimes we would find out that instead of exposing our model through an API, all that was needed for the business was to provide a scored Excel file. A good example of perfectly building the wrong thing. Another issue here is not planning for the integration part of a data project - how would that fit into the overall IT infrastructure and who the consumers of that solution will be. How does the end result connect to other systems? Do you think those issues are a failure of planning, a lack of skilled workers, or a strategic problem?

NG: Well, let's go back to what data and machine learning engineering are. They represent the ability to respond to change by getting and reacting to feedback loops. The issue is when you are just building one thing without the capacity for change. For example, I like to do Jujutsu. This is a pretty interesting martial art. In theory, the goal there is when someone attacks you, you submit them. To achieve this goal, you have to respond to events

dynamically. Let's say someone jumps on top of you, then you get out and do something else. The ability to react dynamically to any situation is essential. I think that's the issue with the data field. The goal isn't to do something static - it is to have a feedback loop to respond to the business. The feedback should happen much quicker than it does now. A good example solution is to show prototypes once a week.

BA: Another idea of tackling such projects is building a very basic pipeline first, but end to end. Then you should get feedback from the stakeholders and commit to further work on the different components of this pipeline. Another follow-up question I have is, admittedly very tricky one - about project management and data science. What do you think of the combination of agile development and data? Is there a better way? How do you think data projects should be planned and executed?

NG: I think a lightweight agile process is pretty hard to complain against. What I mean by lightweight - every week, you demonstrate where you're at. You have components divided into tasks that are maybe 4 hours or 8 hours apiece, and then you do those tasks each week. I think that's enough - no need to go more complex. The problem when people go more complex, it can becomes a cargo cult, another analogy I like. The cargo cult of martial arts is aikido. Aikido is much more like scrum, where everything is staged into specific, scripted rituals. This would never happen in reality. You can't expect a certain sequence of events, static and frozen in time. I've never seen this work well in organizations.

BA: Let's dig deeper into this. Even if we do a lightweight agile, how would we communicate our process to senior stakeholders in a big company, who might be expecting something else, who think our processes are not rigid enough? Do you have some advice on that? How can this fit with a traditional business?

NG: I think the three components of a successful project are a weekly demo, tasks assigned in a lightweight ticket system, and a spreadsheet showing the quarter's plan. That's it. And the demo is what the product managers would show to the CEO. This demo should just be good enough (like you said end to end). Then you can get feedback immediately. In this case, you can quickly fix significant issues, avoiding unnecessary work.

BA: Another question I have is the word "pragmatic". I heard you use it on several occasions. Could you elaborate on it? How can we be more pragmatic in this work?

NG: I think pragmatic means being ruthless about efficiency. For example, let's say we have a system that barely works that took several years to build. The person who did most of the building would very much like to keep it the same. The right thing to do is to clean as much as possible - imagine a pull request where 25% of the codebase is deleted while the system continues to work. This is pragmatic. Nothing's precious; whatever is needed to improve the system should be done. Working only on things that matter - that's pragmatism.

BA: Do you think that knowledge work can be automated? Where does the future go of our field? Do we lose creativity in what we do? What skills do you think are most important right now for data people to remain relevant?

NG: I would say that it's surprising that people think that AutoML and such tools won't get better with time. Even very famous people would think that it doesn't work. Let's look at anything that happened in the last 50 years. Once you start automating anything, it will always get better with time. A great example is the film industry. When we first started editing, we had 3/4 inch tapes, and they were analog. You had to dissolve with three decks, using three different machines. You have the source tape, the destination

tape, and the black tape. And now, with my laptop, I can just click a button that says “dissolve”. Of course, everything gets automated! Still, there’s art. Editing is very creative. Such work will remain - the creator must provide their signature. If you’re talented as a data person, you should be excited about all of this happening because you’ll become more impactful with the work you do.

You can learn more about Noah on his website at noahgift.com, or connect on LinkedIn.

Summary

- Hire operations-oriented data people
- Build pipelines and use platforms
- Be pragmatic

June Dershewitz

BA: Boyan Angelov
JD: June Dershewitz

About: June Dershewitz is a Data Strategist at Amazon Music. Before this she spent 20+ years in driving data and analytics strategies for industry-leading companies, including Fortune 500 corporations and tech startups. She is also serves as Board Char at the Digital Analytics Association. You can connect with her on LinkedIn.

BA: Let’s start with your story. How do you end up in data? It’s a question I always ask since there are so many diverse backgrounds in our field.

JD: I got my start a very long time ago, with a bachelor's degree in theoretical math. That was in the very beginning of the internet. After that, I got a job working for a mathematician who was building a website for math teachers, students, and professors to talk to each other. I got to do many things on that research project - essentially becoming a front-end engineer. I got the chance to understand how the internet works, which was really exciting and new at the time. Eventually, I decided it was time to move into the corporate world in San Francisco.

BA: Why San Francisco?

JD: Well, I'm originally from Oregon, and I love the west coast. I had been living in Philadelphia, so I felt the need to go to a large city again. It was in 1999, the middle of the first dot-com boom. And I figured I could get a job! I started applying to front-end engineer positions. At one company I was asked whether I would like to become a data analyst. They told me I had the combination of skills necessary to become a great analyst - software engineering and math. I accepted! I realized that I loved it, even though it wasn't the vision I had for myself originally. That was a start of a very long career in data. Since 1999, I've worked with data as an individual contributor, building and leading teams of data people both, on the brand side and as a consultant.

BA: Those were very early days in data science. I assume there were no data scientists back then?

JD: No, they were called statisticians! Indeed, I ran across quite a few people who would consider themselves statisticians, who today probably would call themselves data scientists.

BA: Being in the field for such a long time, do you think companies know more now how to do data projects than before? The technology has

advanced quite a bit, but how about the more strategic part?

JD: It's frustrating to see the same problems over and over that we keep repeating and not figuring out. But I think we can build on ideas much faster than before and iterate on them. An example of this would be A/B testing, which a company would employ to optimize business outcomes. We'd like to think that the dot-com organizations figured this out already, and any competitive company out there is maximizing their investment. Well, that can be true to a certain extent, but they certainly didn't invent it. These methods have existed even before the internet. For example, it was being done by advertising companies to measure the effectiveness of direct mail. Now we can just do it with much more ease, and we can do it faster.

BA: Interesting. Operationally, we probably still have the same problems regarding how people understand data. It might even be harder nowadays. My next question is on the title of a "data strategist". What do you think about that? I know some companies use similar titles, such as data translator. Is this a widely accepted and understood role at this point?

JD: Not really. I think that data-related job titles have always been somewhat of a pain point. Throughout my career, I've at times cared more or less about the job title. The job title a person holds sometimes is important and sometimes not. Early in my career, I was making a move from an individual contributor to a consultant. Before I started, the company's co-founder called me, telling me he was working on the business cards and would like to know my job title. I was thinking - perhaps VP of something? He said, ok - vice president of analytics. And that was my job title from then on. But when you work for a 14 person company and you have the job title VP of Analytics, it means you're going to do everything. And I generally feel that way about data-related job titles as they have evolved over time, especially with the "data scientist" one. Usually when I talk to other leaders

of data organizations, when they talk about their staff makeup, the data scientists, data engineers, and others - they're usually admit those roles mean different things in different organizations. On one extreme, you might have a company where any person who touches data is called a data scientist. And then in another one, you might have so many specific different job titles where you'll have a data scientist, research scientist, ML engineer or data analyst. There's no right or wrong. I think that data strategist and data strategy are malleable terms that we can use to mean different things. I don't think they will become standard terms to describe a specific job function in the company. I can contrast them with a title such as a "data engineer", which I think is very specific and tangible.

BA: Yes, "data engineer" is already quite an established one. But it's safe to assume that the role of data strategist has always existed before as well, probably under a different name. Someone must have been taking care of the "translator" duties in the organization.

JD: This reminds me how in 2019 I was an invited speaker at a conference in San Francisco called "Marketing Analytics in Data Science". The title is quite specific. When I went to it, I was surprised to find that everyone around me actually was working in data science for marketing analytics! That same year I decided to run a panel for the conference on building a company-wide data strategy. I wrote a summary, and then I went looking for speakers - everyone I got was a chief data officer. It turned out into a super interesting discussion. But as we were preparing for it in advance, we all had the conversation: what are we actually talking about here? What is a business-wide data strategy?

BA: This is a perfect time for me to ask you for a definition of data strategy. Do you have one?

JD: I've found several that I would mash together into one: data strategy

is a vision for how a company will manage and use data to generate value for the business and the customer. This is still broad but could be broken down even further. For example, what data do we need? How are we going to source it? How are we going to collect it? How are we going to store it? What technology we'll use? Who will we share it with? What are the policies for data? How will we use the strategy, in what areas of our business, and to what ends? How would we know it's working? And if we're doing it right, what kind of value is it generating? How do we describe and quantify this value to the business or the value to the customer of all of the time and money that our data teams spend on working with data, trying to serve the business?

BA: This is great. Now we start to talk about the specific elements of data strategy. I now have a question on whether a data strategy is something static, such as a PowerPoint deck, or is it more of a continuous function that someone is performing? I've had clients ordering giant slide decks, only for them to be buried somewhere, never to be seen again.

JD: It depends. Let's say you are a data person at a company that isn't yet sold on the value of data. You have a tough task in front of you because it's all about education and convincing executives to fund your efforts. Because if you don't have any funding, you're always going to be at the bottom of the barrel. Your work can be an afterthought, and that's not where you want to be. Let's say there are a few people who do data work throughout the organization, but they're doing it at a really low level, mostly on unconnected pilot projects. But if you could show results, you can use this base to form a team or even multiple teams. And the more you do with data, the more you can say you're using data successfully across the organization. Say we're using data successfully in Marketing, but we haven't necessarily gotten a full value out of what we're doing with the data in

Product. So you decide to build a Product Analytics team. And then perhaps you can see how you can support the Sales team with data or insight. And then, at the more advanced stage, you would be looking across the entire company, and you're collecting and managing all the kinds of data that matter to the business. In the end you'll be able to turn around and use that to generate value for the business and the customer in all the ways where it matters. I think that depending on the stage the company is at, you're going to see different variations of this process.

BA: Who do you think the customer of a data strategy is? How far down, up, or sideways does this document need to be used in the organization?

JD: It depends on the org structure. It's never going to be perfect. I'm sure we can spend a whole hour just talking about different kinds of org structures for data people and the pros and cons of each choice. But I think as long as you understand what you're striving for, you can compensate for the weaknesses of any kind of org structure. I believe data strategy can work best when incorporated into company-wide strategic planning. So if you set annual goals about what you want to accomplish, hopefully, some of them will be quantitative and require support and participation from data people. Even if it's basic business optimization, it's meaningful as long as it helps grow the business.

BA: I agree. I don't think you can separate data strategy from business strategy and hope for good results. How iterative should a data strategy be? Should it be more of a living document or more of a static roadmap?

JD: People could discuss the value of long-term planning versus the effort spent on implementation, but I see value in it - as long as it's combined with shorter-term plans that are directly related to execution. I think that a well-thought-out three to five-year plan is a great idea. This can show

where the organization's data efforts are today and the vision for where it wants to be way off in the future. Still, I don't think you can't just set it once and forget about it. The strategy will get stale after a while, but it should be able to serve you well long enough so that you can generate annual plans under the umbrella of the larger, three to five-year one. And from there, you can set tangible and specific quarterly targets. You always need the five-year north star guiding you. I've found, especially with data science and machine learning projects, that they can easily meander. You need to reinforce the focus, even if it shifts over time. You can plan quarterly and build on top of your knowledge, but everything should be aligned with the longer-term plan.

BA: What is the most important thing for a company with low data maturity to tackle first?

JD: I think that, that as a business, they should have a clear understanding of where they're going to get the most business value from their investment. This is a good starting point to do the first proof-of-concept project.

BA: A good point, but how do we estimate this business value?

JD: Let's take the example of business intelligence people. Unfortunately, they tend to get undervalued, especially now when there's this separation between them and data scientists in some companies. Data scientists are often perceived as more impactful, but that's not necessarily the case. If you take away the people making the dashboards, there's going to be a considerable gap - an unmet need. As a business, we always have to think about how much time and energy we need to spend on creating and maintaining dashboards. It shouldn't be a hundred percent, but it can't be zero either. Let's say it's zero - we're just not going to make any dashboards. We're going to invest in data scientists solely instead. Then what happens is the data

scientists get asked to create dashboards because (surprise), people still need dashboards! And then, the data scientists become unhappy because this wasn't in their job description, and they might leave the organization altogether.

BA: How do you think about managing data projects? Does agile work for data? How do we go about estimating tasks and resources?

JD: I do think agile works. Of course, estimating how long something will take is always difficult. And especially if you've got something big and ambiguous and have nothing built yet. In that case, you're not going to be very good at estimating how long things will take. As a project develops, you'll better understand what is worth pursuing and what is not. This skill will take time to learn. At some point, you can refer to your experience - for example, the team compositions and skills, knowing who to involve, and it does become easier. I think in the beginning, you'll be able to estimate things that are only one to three months out. And then, when it comes to six months or a year out into the future, you really might not have much of a clue. You might know the result you're after, but you wouldn't have a good amount of information to estimate how long this will take or even who needs to be involved at what level. I've seen in the past chronic underestimation of data engineering effort for data science work. Also some confusion about roles - for example, what should a data analyst do? How about a data scientist? You often won't have the luxury of bringing in people with all the right skillsets to contribute at all times. This might slow you down because you might have a data scientist who's also asked to be a data engineer, and it might not be their core skillset, or they might be doing it, but as a result, they are not writing high-quality code. And so then you'll need to have someone come in later on to fix the problems that were made because they weren't an expert. I have also seen a lot of issues with trust-building with leaders who

funded or sponsored a data project. A data scientist might be motivated, but there might be a lack of clear understanding of what the business will get at the end. What can happen then is the business leader can ask: “what have you done for me lately?”. And if data scientists have gone off working on things just because they are novel and exciting, they can get in trouble. Meandering away from the business purpose or not communicating enough back to the people who have funded it can easily backfire.

BA: I agree. Doing cool things just for the sake of learning can backfire. As a data scientist, one might think this is smart, but as long as the work delivers no value, it’s useless. Can you tell me the biggest reason for data projects to fail nowadays?

JD: I think it all started with the whole “sexiest job of the 21st century“ article. I think this oversold the field and made it seem like snake oil. How will you actually set data scientists for success when you don’t clearly understand the value they will deliver? And I think modern data science in terms of how it fits into a larger organization is better understood now. It’s been around long enough so that people can ask and answer the question, “what have you done for me lately”.

BA: Yeah, so to paraphrase a little bit: you would say that a lot of the issue with data project success would be high expectations? Leaders think they can just put a data science wizard on the project, and everything will fall into place.

JD: Yeah, exactly. One approach I’ve seen that I think works fairly well: start with a small proof of concept project with a short turnaround time. Then show its value. If you don’t do this before a further longer-term investment commitment, you might end up with a wasteland.

BA: In my book, I call it pilotitis - the disease of doing pilot projects only.

How do we ensure such smaller projects are successful in the medium term?

JD: This is not easy. One thing you can do is set goals for pilots. For example, we'll finish it by this date, and it will do those exact things. This way, you keep its scope limited. After this, you can show that it has all the features you feel are essential and there's widespread usage on the receiving end. I don't think a data scientist could do this alone. Having a product manager involved is important for scoping, gathering requirements, user acceptance, testing, and keeping a backlog of feature requests. So it's not really data science work, but this is necessary for creating something like a long-term program.

BA: It sounds like we do need this person in the middle. It doesn't matter if they're called a data strategist or a product manager.

JD: You can name this function a "technical program/product management". I think you need someone who has that kind of mindset to treat the system being built as something that evolves over time and is only successful if it's adopted, used, and supports the business outcomes.

BA: What specific skills would you say this person should have?

JD: A product manager is a big generalized job right now. And the product manager for something that is directly facing customers of a business might be different than a product manager for something else, such as a recommendation engine. They can be one step removed from the end customer who is receiving the recommendations. But, still, I think some of the same skills apply. I think having an excellent understanding of why a product is built and articulating that. Always have a solid customer focus, know for whom the product is designed, and ensure users can use it successfully. This person should also know where the project will be in the next quarter and align on the long-term vision.

BA: Exactly. The most frustrating thing I've seen in my career is brilliant people building the wrong product that nobody wants to use.

JD: Yeah. And people might make different choices. It's often a case of taking the product in direction A or direction B, with trade-offs, and in each case. If you only have a person involved who cares about the novelty and complexity of the system they're building, they may make one choice that is not necessarily what the customers need. And if you chose instead a simple approach that is not as technically sophisticated but what leads to a better business outcome - it may be the right choice.

Summary:

- Always align with the business objective
- Start small in one area, prove the value and grow further
- Have a longer-term data strategy and short term, agile plan at the same time
- Always be focused on the end-user when building data products

Martin Szugat

BA: Boyan Angelov
MS: Martin Szugat

BA: Let's begin at the beginning - how did you end up in data?

MS: I started my career already during school. After many manual jobs, I decided I would prefer to use my brain more (laughs). Since I liked playing video games, the next obvious step was to start programming. My father

had a client looking for programmers, and this is how I started coding. I also started writing for magazines, such as the Visual Basic magazine. I dove deep into the .NET area and started teaching other people. Around that time, I was also one of the first people in Germany to become an expert in the whole XML topic, which would be the origin of data in my career. For my studies, I initially studied computer linguistics and philosophy but switched to bioinformatics.

I also wrote some books. One of them was about social software. Social media didn't exist then, and people mainly meant blogs and wikis by this term. I also had the idea to start a company with my bioinformatics professor but decided against doing that and joined UnternehmerTUM instead, intending to meet like-minded people. We created a social media agency with one of them, doing digital collectible games (now you would call those NFTs, so that was way ahead of its time) and Facebook apps. After several years of this, I wanted to go back to doing data because of my background in bioinformatics. I had never had the chance to apply those skills before, and nobody was talking about ML or AI at the time. At that time, I started Datentreiber with the idea of putting all my experience into one venture - combining data and business. I also saw how many companies fail in the topic and saw an opportunity to help and improve their processes.

BA: It seems like you had a very diverse experience. What essential skills you gained during this time are valuable to you now? What did you learn from doing bioinformatics or running your own company?

MS: The skill which stands out to me is learning the design thinking approach while working with IDEO. Discovering design thinking was a life-changing experience because afterward, I applied this design thinking mindset to all my ventures and projects, and currently in consulting. From bioinformatics, I learned something quite important when thinking

about models. The people in bioinformatics also got this wrong. Back then, Support Vector Machines were trendy, and the scientists wanted to solve everything with them, including how genes worked and other topics like that. But the biochemists proved that most of those models were wrong. They did real-world experiments and tested the modeling work against real-world data. This was called the "ivory tower" syndrome - bioinformaticians at the time were rarely working wth someone in biochemistry or molecular biology. Bioinformaticians wrote software for bioinformaticians. Avoiding this condition is something I learned the hard way.

BA: I've seen nowadays that people try to put PhDs in a room with MBAs and see what kind of ideas come from it. Not sure if that's such a great idea, but it sounds like a better approach.

MS: Yes, and this is why I decided not to create a company with that professor. I've seen companies full of PhDs. If you ask them who will make the sales, they have no answer - they think they are different and don't need it. In reality, you need some sales, marketing, and HR.

BA: Let's now talk about the title of a "data strategist." What do you think about that? Is it necessary? Are there better titles?

MS: You have to always distinguish between the title and the responsibility. Different titles can have a similar responsibility - whether they are a data strategist, an analytics strategist, a Head of Data, or a Chief Data Officer. There should always be someone, especially in bigger companies, responsible for designing, executing, and monitoring a data strategy. And by data strategy, I always speak about data and analytics strategy - you can't separate the two. From my perspective, there's no "AI strategy." AI is part of analytics, and analytics is part of the data strategy. Going back to my previous point, sometimes you'll have a CDO saying they own the strategy

or another role. That can be a data strategist, depending on the company's size. You should spend money on hiring such a person!

BA: Exactly. Depending on the organization's size, you might need different people at different levels. Especially at the very top, you need someone with this analytical skillset who manages all use cases. As you said before, miscommunication between technical and business people is common, and that's why you need a responsible person to translate between the two.

MS: Yes. Another essential responsibility this person must carry is ensuring the data strategy is aligned to the business strategy. There should be a strategy for all data and analytics initiatives and investments, and they must be responsible, also, for killing projects or use cases that are not contributing to the business objectives. This goes more into project portfolio management. For example, there are a lot of projects that fail because of issues with data quality or availability. A data strategist needs to take the responsibility to check the data sourcing, collection, and quality initiatives and ensure that down the line, let's say in three years, the data is available so that they can implement the use case. This happened to one client project a few years ago, and that use case could not be implemented since the data were simply missing, and nobody paid attention to this.

BA: It sounds like there was just no plan, no strategy. Sometimes executives believe you can just hire some people, give them a broad target and let them work. All of this is done without doing the essential homework - checking that everything is in place. I agree this is one of the most critical attributes for a data strategist. What other skills are necessary?

MS: I think the best data strategists have a technical background in data science, analytics, or a related field. If people just come from the business perspective, they lack the skills and analytical thinking. If they studied

economics or something similar, they simply have a different way of seeing and perceiving things. Data scientists from physics or biology have this analytical thinking trained, which is very hard to get.

BA: Can you elaborate further? Do you mean a scientific mindset, experimental and hands-on thinking?

MS: Yes, but not only. Most importantly, they realize that everything is simply an assumption. A strategy itself is one big assumption. A great book on the topic I recommend is "Good Strategy, Bad Strategy." The author has a lot of strategy consulting experience and is a professor at UCLA. His first advice was to keep in mind that strategy is just an assumption and always needs to be tested. You first design the strategy, and after this, check whether it works out.

BA: This reminds me of the saying that all models are wrong, but some are useful.

MS: Yes, exactly!

BA: I want to play the devil's advocate here. While a data strategist needs to have a scientific mindset, I think it's equally important to be good at dealing with ambiguity. This skill set is essential for communication and dealing with more political issues, which are common with clients. Ambiguity is also a part of any data strategy since, as you said, no strategy is perfect, and many assumptions need to be made during data strategy design. For example, when estimating budgets and resources, you need to be comfortable providing concrete numbers, even if it's not clear what they are for.

MS: Yes, and there are multiple levels of assumptions. An essential element of a data strategy is defining the data products you want to build. Each is also based on assumptions, and you must have an experimental approach

to making them. With such a mindset, you can become a data strategist. Still, there are a few other very useful skills such as mediation, moderation, and communication skills. I would still say you can train all those skills, but changing the mindset of people is hard.

BA: This connects nicely to my next question. How do you train people for such work? I know this is a big topic for Datentreiber.

MS: Yeah, as I mentioned before, design thinking is the most crucial method to be learned. At the beginning of the training, we are primarily focused on teaching the basic topics. For example, what's the difference between descriptive and predictive analytics, and what's machine learning. What's AI, and what's not AI. It's essential to focus on the fundamentals first. There's too much buzzwordy content out there, and you can notice that people spend too much time on LinkedIn. So this is the first level. At a second level, we train people in our data strategy design kit and other methods we have developed based on our experience.

BA: Do you also train the people how to teach other people themselves? It's an essential part of the job of a data strategist to "train" C-level and business people. After all, they also spend some time on LinkedIn and probably need to be "un-trained" a bit first.

MS: Yes! We've learned a lot, especially in the past year, that one thing you should do before you work on the data strategy design in a series of workshops is to have a training session. You can introduce the business people to the basics first (such as the difference between a metric and KPI). We noticed that the following workshops work much better if the people had training before, because otherwise during the workshops you'll have a lot of discussions about the definitions of things. Sometimes people talk about the same things with different words. Another issue that can also arise if

people have no training before is unrealistic expectations. In one case, I remember one of the clients wanted to build an Alexa-like system for a car workshop. I already knew this would be hard. The Alexa team at Amazon numbers in the hundreds, and still, the product has issues. This is closer to science fiction than reality!

BA: Yes, expectation management is critical, and you must re-educate. Another question I have is about the success of data strategy in general. You've been doing this for a while now. Could you tell us how C-level people think about data strategy in 2022? Are they excited about it, see it as an essential activity, or are they skeptical?

MS: I think this depends on the company. When they talk about data strategy, they mean more about data architecture design. Or they just might want a PowerPoint presentation for the management board to get the data team financed. Others don't want to do a data strategy but want a "very concrete plan of how to create value with data and analytics" instead. But then, I would call this a data strategy (laughs)! Some other ones even don't want to call it a data strategy. They would say that "strategy" is reserved for the management board. It should also not be named "data" since the IT department should be responsible for it. So, in that case, we would call it a "MarTech Concept" or something like that. But that's, in fact, a data strategy.

BA: Right. People still want it and understand why it's necessary, even if it's hidden under different names. Do you think data strategy as a PowerPoint deck? Do you believe organizations know what needs to happen after that, how to implement things, and measure success?

MS: I think this happens only in organizations with low digital maturity. Executing a data strategy is the only way to know whether it is good. Those

should focus on doing more pilots and experimentation. I have seen this issue quite a few times. For example, one client requested we build a so-called KPI driver tree with them, the value driver tree. This would help them understand the relevant metrics and set the objectives correctly. We did this for several months, and after finishing, they were pretty happy and realized this value. Still, I had to remind them that this is a good start, but it is still just the first step of a long data strategy.

The more focused and smaller the data strategy is, from my experience, the higher the likelihood it will be successful. We have also advised clients on an overall company-wide data strategy. Still, we encountered the problem that it became too superficial - it becomes that PowerPoint with a lot of vague texts, such as "employees should treat data as an asset," or "all our departments should utilize data in a way which is aligned with our business objectives." While those statements are undoubtedly true, they are valid for any company - you can just copy and paste this text. What most companies struggle with is creating a holistic data strategy, which is a long-term one. It needs to be executable and have checkpoints where you can measure its success and adjust if necessary - see if it works out. This is a real strategy.

BA: How do you ensure the clients trust you with an expensive data strategy and that it delivers results? One way is to run a prototype and show results as quickly as possible. But still, a data strategy costs money, and the benefits can become apparent much later.

MS: What helps again is to ensure that the data strategy is not superficial but detailed. For example, a large corporation might decide to go into making more personalized offers and products. It's then assumed that having the products more personalized will make their customers buy more of them. You can test this assumption by doing a small A/B test - get 50 people who get the personalized offering and 50 who don't - and compare

the results, seeing if the average revenue per customer increased. You can do such experiments pretty quickly in the beginning. Like this, you can test one fundamental assumption of your data strategy before executing it and wasting a lot of money.

BA: Yes, this makes sense. Another one of my favorite questions is how do you plan for things you can't plan for? How do you ensure that a data strategy does not get derailed, for example, when one of the prioritized use cases doesn't work out? And how do we deal with expectations relating to this? Do you have any advice here?

MS: There are multiple things you can do. After designing the strategy or product, the next phase should not be to start the execution or implementation immediately. You should have this experimental phase instead. There you build prototypes, research, and try to falsify your assumptions. This is another critical thing I learned during my training - always identify your most critical beliefs and ruthlessly test them. A term for this is RAT - Risky Assumptions Test. You can do those for any specific product but also the overall strategy. Then you make sure all your RATs are eliminated. Only after that do you start working on the engineering part.

A second thing you can do is just accept that many of your assumptions will just be wrong. If you understand this, the logical consequence is that a strategy is never done. It's something fluid: after the first draft, you need to test it and perhaps entirely through it away. Or maybe just modify it a little and then retest again. It doesn't work if you just hire someone to do the data strategy, deliver it to you and then forget about it.

BA: Can we now discuss an important concept - data assets and products. Can you define what a data asset is?

MS: I use the term data asset to describe a data source with a precise value

for the business. This implies a data product, some form of analytics, or whatever you applied to this data, to extract and analyze information from it. And if this information then leads to better decisions, actions, and results - reaching the objective in the end.

That's a great definition! How about data products versus data projects? What's the difference between the two? Data products must be different from other products, such as clothing.

MS: The answer to this question depends on who you're talking to. If you're talking to business people, they might think about data products as packaging the data itself and selling it to other companies. When you speak to old-school data scientists like me, a data product can be defined as the outcome of a data mining process, where you apply analytics to data. Even an ad-hoc research paper can be considered a data product with this definition. The definition is different nowadays: a data product is closer to a software product. It's the data, and the analysis software, whether automated or semi-automated, which is ideally scalable and reusable.

BA: So, by that definition, a machine learning model exposed via an API to serve predictions would be a data product?

MS: Yes, but it can also have a graphical interface. It can be a dashboard or an application generating business reports. It's shocking how often, even nowadays, generating business reports is done manually, by hand. We have this one client, and they have so many people generating reports, and that's their whole job. After generating them, they just sit around at a SharePoint somewhere; who uses them is not clear.

BA: Why do you think such inefficiencies are still so widespread? Is building data products a challenge in larger companies rather than startups?

MS: Yes, of course. In larger organizations, you already have a lot of people

who have been doing such manual work in the past. In one case, we were working with a pharma company, and they had to create a study on how time influences the effectiveness of drugs and whether that can lead to potential side effects. Each year they would have thousands of new drugs, and hundreds of people are doing this analysis. Many of them would have ad-hoc scripts and just copy and paste from each other without centralized solutions or templates. Startups rarely have this since they have too much pressure to survive, have fewer people, and often have the luxury of developing greenfield data products, which is much easier.

BA: In this case, if you were to automate such a process and create a data product, how do you ensure that the people trust it? Especially in such sectors, this is a big topic. I can imagine that even if it's inefficient, it can still be perceived as more trustworthy since many humans are involved, rather than a centralized black box.

MS: Now we're getting back to the whole design thinking topic, which is why it's so important - not only when you're designing data product, but the data strategy itself. A central theme in design thinking is using the users' or stakeholders' point of view. The best way to do this in data strategy is to involve them in the design process. If they have a seat at the table and can share their point of view with you on a whiteboard (making it more tangible and visual), they can express themselves so that other people understand it. People with a higher degree of understanding will trust and accept the strategy.

This goes in both directions. Also, for data scientists is vital to understand the business process. Otherwise, they might design a solution for the wrong problem. There are a lot of examples where there's a perfect solution to the wrong problem out there. This survey by Eric Siegel confirmed that many models are not deployed just because they don't fit the business processes.

This happens because the technical people have no understanding of the business. If you had no idea how a car works, you wouldn't enter it.

BA: But you can't understand everything simultaneously, right?

MS: Of course. It depends on the person. Some people are very comfortable when they just have a rough understanding. All they need to know is that the car is secure. They can just enter the vehicle and feel safe. Others need a much deeper understanding of the cars' inner workings to feel safe. The only way to know what people you are dealing with is to start to work with them in a workshop. Many potential problems can be avoided if you co-design and co-develop data products and strategies.

BA: So, how can one go about learning design thinking? I think it's still a skill not widely known beyond certain circles.

MS: This is a good question. One prominent misconception about design thinking is that it's a specific method. It's much more than that - you can call it a mindset. You think in designs and design things from a user perspective, always in T-shaped teams or teams full of T-shaped persons. You just need all those different perspectives. I would say it's just not enough to do one training or a five-day design sprint course. That certainly helps, but you need much more real-world practice and experience to master this truly.

I remember my first contact with design thinking. There was a presentation about it. It sounded quite superficial, and only when I had this practical project with IDEO, where it was shown much clearer how this mindset is shaped, that I truly embraced it. I think it's much more important to work with other people who have already applied design thinking to projects and exercise together. From my experience, many workshops that we did, were much more successful if the people had already done some design thinking beforehand. Otherwise, they might be in for a hard time - especially the part

where you need to think from the user perspective. Exercising this empathy might sound trivial, but it's the hard part!

Summary:

- Learn and embrace design thinking
- Keep the strategy granular
- Have an experimentation phase between the strategy and execution

Amadeus Tunis

Appendix

List of acronyms and abbreviations

AI Artificial Intelligence

API
Application Programming Interface

AWS
Amazon Web Services

BI Business Intelligence

CAS
Complex Adaptive System

CNN
Convolutional Neural Network

CoE
Center of Excellence

CSA
Current State Analysis

CV Computer Vision

DD Due Diligence

DMA
Digital Maturity Assessment

EDA
Exploratory Data Analysis

FTE
Full-time Equivalent
GA Gap Analysis
GCP
Google Cloud Platform
GIS Geographic Information System
LSTM
Long short-term memory
ML Machine Learning
NLP
Natural Language Processing
RACI
Responsibility and Assignment Matrix
SDK
Software Development Kit
SSOT
Single source of truth
ST Systems Thinking
SVM
Support Vector Machine
XAI
Explainable Artificial Intelligence

Notes

Introduction

1 Arnold, Ross D., and Jon P. Wade. “A definition of systems thinking: A systems approach.” Procedia computer science 44 (2015): 669-678.

2 Borek, Alexander, and Nadine Prill. Driving Digital Transformation Through Data and AI: A Practical Guide to Delivering Data Science and Machine Learning Products. Kogan Page Publishers, 2020.

3 Box, G. E. “All models are wrong, but some are useful.” Robustness in Statistics 202.1979 (1979): 549.

4 Hewitt, Eben. Technology Strategy Patterns: Architecture as Strategy. O’Reilly Media, 2018.

5 Meadows, Donella H. Thinking in systems: A primer. chelsea green publishing, 2008.

6 Siegenfeld, Alexander F., and Yaneer Bar-Yam. “An introduction to complex systems science and its applications.” Complexity 2020 (2020).

Alignment with Business Strategy

7 Borek, Alexander, and Nadine Prill. Driving Digital Transformation Through Data and AI: A Practical Guide to Delivering Data Science and Machine Learning Products. Kogan Page Publishers, 2020.

Current State Analysis

8 Taleb, Nassim Nicholas. Antifragile: Things that gain from disorder. Vol. 3. Random House, 2012.

9 Maister, David H., Robert Galford, and Charles Green. The trusted advisor. Free Press, 2021.

10 Brackett, Michael, and Production Susan Earley. "The DAMA Guide to The Data Management Body of Knowledge (DAMA-DMBOK Guide)." (2009).

11 Dunning, David. "The Dunning–Kruger effect: On being ignorant of one's own ignorance." Advances in experimental social psychology. Vol. 44. Academic Press, 2011. 247-296.

Gap Analysis

12 Taleb, Nassim. "The black swan: Why don't we learn that we don't learn." NY: Random House (2005).

13 Rumelt, Richard P. "Good strategy/bad strategy: The difference and why it matters." Strategic direction 28.8 (2012).

Use Cases

14 Courtney, J. "The Workshopper Playbook–How to Become a Problem-Solving and Decision-Making Expert." (2020).

15 Scavetta, Rick, and Boyan Angelov. Python and R for the Modern Data Scientist. O'Reilly Media, 2021.

Data Governance

16 Laney, Douglas B. Infonomics: how to monetize, manage, and measure information as an asset for competitive advantage. Routledge, 2017.

17 Kilkenny, Monique F., and Kerin M. Robinson. “Data quality:“Garbage in–garbage out”.” Health Information Management Journal 47.3 (2018): 103-105.

18 Hajij, Mustafa, et al. “Data-Centric AI Requires Rethinking Data Notion.” arXiv preprint arXiv:2110.02491 (2021).

19 Zheng, Alice, and Amanda Casari. Feature engineering for machine learning: principles and techniques for data scientists. “ O’Reilly Media, Inc.”, 2018.

20 Molnar, Christoph. Interpretable machine learning. Lulu. com, 2020.

21 Yang, Qiang, et al. “Federated learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning 13.3 (2019): 1-207.

Operating Model

22 Peng, Roger D., and Elizabeth Matsui. The Art of Data Science: A guide for anyone who works with Data. Skybrude consulting LLC, 2016.

Roadmap

23 Borek, Alexander, and Nadine Prill. Driving Digital Transformation Through Data and AI: A Practical Guide to Delivering Data Science and Machine Learning Products. Kogan Page Publishers, 2020.

Overview

24 Winters, Titus, Tom Manshreck, and Hyrum Wright. Software engineering at google: Lessons learned from programming over time. O'Reilly Media, 2020.

Soft Agile

25 Hoens, T. Ryan, Robi Polikar, and Nitesh V. Chawla. "Learning from streaming data with concept drift and imbalance: an overview." Progress in Artificial Intelligence 1.1 (2012): 89-101.

26 Atwal, Harvinder. Practical DataOps: Delivering Agile Data Science at Scale. Apress, 2019.

Lean Data

27 Dekier, Łukasz. "The origins and evolution of Lean Management system." Journal of International Studies 5.1 (2012): 46-51.

28 Reis, Eric. "The lean startup." New York: Crown Business 27 (2011): 2016-2020.

29 Drucker, Peter Ferdinand. Classic Drucker: essential wisdom of Peter Drucker from the pages of Harvard Business Review. Harvard Business Press, 2006.

Delivery Methods

30 Knapp, Jake, John Zeratsky, and Braden Kowitz. Sprint: How to solve big problems and test new ideas in just five days. Simon and Schuster, 2016.

31 Atwal, Harvinder. Practical DataOps: Delivering Agile Data Science at Scale. Apress, 2019.