

O'REILLY®

Deutsche
Ausgabe

Data Mesh

Eine dezentrale Datenarchitektur entwerfen



Zhamak Dehghani

Vorwort von Martin Fowler

Übersetzung von Jochen Christ und Simon Harrer

Inhalt

Cover

Titel

Impressum

Widmung

Inhalt

Vorwort

Vorbemerkung der Übersetzer

Einleitung

Prolog: Fallstudie

Teil I Was ist Data Mesh?

1 Data Mesh im Überblick

Ergebnisse

Veränderungen

Prinzipien

Domain Ownership

Data as a Product

Self-Serve Data Platform

Federated Computational Governance

Zusammenspiel der Prinzipien

Data Mesh kurz erklärt

Daten

Operative Daten

Analytische Daten

Entstehungsgeschichte

2 Das Prinzip Domain Ownership

Domain-driven Design
Strategisches Design
Archetypen von Domänendaten
Quellenorientierte Domänendaten
Aggregierte Domänendaten
Konsumentenorientierte Domänendaten
 Einführung von Domain Ownership
Verantwortung wandert flussaufwärts
Verknüpfung von Modellen
Mythos Single Source of Truth
Daten-Pipelines als Implementierungsdetail
 Zusammenfassung
 3 Das Prinzip Data as a Product
 Product Thinking
Grundlegende Usability-Attribute von Datenprodukten
 Einführung von Data as a Product
Domänen verantworten Datenprodukte
Eine neue Sprache
Daten als Produkt statt als Asset
Vertraue, aber prüfe nach
Daten und Logik als Einheit
 Zusammenfassung
 4 Das Prinzip Self-Serve Data Platform
 Data-Mesh-Plattform im Vergleich
Autonome domänenorientierte Teams als Zielgruppe
Autonome und interoperable Datenprodukte als Kernkonzept
Operative und analytische Funktionen in einer integrierten Plattform
Generalisten als Hauptnutzende

Dezentrale Technologien

Domänenagnostisch

Platform Thinking

Wertschöpfung durch autonome Teams

Wertschöpfung mit autonomen und interoperablen Datenprodukten

Reduktion des Cognitive Load

Skalierung der Bereitstellung von Daten

Förderung einer Innovationskultur

Einführung einer Self-Serve Data Platform

API first

Optimierung für Generalisten

Simplify

High-Level-APIs für Datenprodukte

Nutzerorientierung statt Feature-Orientierung

Klein anfangen und mitwachsen

Zusammenfassung

5 Das Prinzip Federated Computational Governance

Systems Thinking

Gleichgewicht zwischen Autonomie und Interoperabilität

Dynamische Topologie als Standardzustand

Automatisierung und verteilte Architektur

Föderalismus

Föderales Team

Leitlinien

Policies

Anreize

Automatisierung

Standards als Code

Policies als Code

Automatisierte Tests

Automatisiertes Monitoring

Einführung von Federated Computational Governance

Delegation der Verantwortlichkeit an Domänen

Integration der Policy-Ausführung in jedes Datenprodukt

Automatisierung und Monitoring der Policies

Bestimmung von Polysemen

Messung des Netzwerkeffekts

Mut zu Veränderung

Zusammenfassung

Teil II Warum Data Mesh?

6 Der Wendepunkt

Die großen Erwartungen

Die große Kluft

Skalierung: Begegnung einer neuen Art

Nichts ist so beständig wie der Wandel

Diskrepanz zwischen Investition und Rendite

Zusammenfassung

7 Nach dem Wendepunkt

Auf Veränderungen angemessen reagieren

Alignment von Fachlichkeit, Technologie und analytischen Daten

Integration der operativen und analytischen Welt

Dezentralisierung von Datenänderungen

Beseitigung unbeabsichtigter Komplexität

Agilität trotz Wachstum

Beseitigung zentralisierter und monolithischer Komponenten

Reduzierung des Koordinierungsaufwands von Pipelines

Reduzierung des Koordinationsaufwands für Data Governance

Förderung der Autonomie

Verbesserung des Kosten-Nutzen-Verhältnisses

Komplexitätsreduktion durch die Datenplattform

Product Thinking überall

Organisationsübergreifender Datenzugriff

Zusammenfassung

8 Vor dem Wendepunkt

Evolution der analytischen Datenarchitekturen

Erste Generation: Data-Warehouse-Architektur

Zweite Generation: Data-Lake-Architektur

Dritte Generation: multimodale Cloud-Architektur

Eigenschaften der analytischen Datenarchitektur

Monolithisch

Zentralisiert

Technologieorientiert

Zusammenfassung

Teil III Wie entwirft man eine Data-Mesh-Architektur?

9 Logische Architektur

Schnittstellen für die Bereitstellung analytischer Domänendaten

Design der operativen Schnittstelle

Design der analytischen Datenschnittstelle

Domänenübergreifende Abhängigkeiten von Analysedaten

Datenprodukt als Architekturquantum

Strukturelle Komponenten eines Datenprodukts

Bereitstellung und Konsum von Daten

Discovery- und Observability-APIs

Die drei Ebenen der Datenplattform

Dateninfrastruktur-Ebene

Datenprodukt-Ebene

Mesh-Ebene

Beispiel

Eingebettete automatisierte Policies

Datenprodukt-Sidecar

Datenprodukt-Container

Control-Port

Zusammenfassung

10 Datenplattform

Entwurf der Plattform anhand von User Journeys

User Journey der Data Product Developer

Incept, Explore, Bootstrap, Source

Build, Test, Deploy, Run

Maintain, Evolve, Retire

User Journey der Data Product Consumer

Incept, Explore, Bootstrap, Source

Build, Test, Deploy, Run

Maintain, Evolve, Retire

Zusammenfassung

Teil IV Wie entwirft man Datenprodukte?

11 Entwurf eines Datenprodukts anhand von Affordances

Datenprodukt-Affordances

Merkmale einer Datenproduktarchitektur

Der Einfluss von komplexen adaptiven Systemen auf den Entwurf

Emergentes Verhalten aus einfachen lokalen Regeln

Kein zentraler Koordinator

Zusammenfassung

12 Daten konsumieren, transformieren und bereitstellen

Daten bereitstellen

Anforderungen der Datennutzer

Entwurfseigenschaften für das Bereitstellen von Daten

Entwurf

Daten konsumieren

Archetypen von Datenquellen

Lokalität des Datenkonsums

Entwurf

Daten transformieren

Programmatische vs. nichtprogrammatische Transformation

Datenflussbasierte Transformation

Machine Learning als Transformation

Zeitabhängige Transformation

Entwurf

Zusammenfassung

13 Daten finden, verstehen und kombinieren

Daten finden und verstehen

Gefunden werden dank Selbstregistrierung

Finden des globalen URI

Semantische und syntaktische Modelle verstehen

Vertrauen schaffen mit Datengarantien

Untersuchung der Form von Daten

Lernen mit Dokumentation

Entwurf

Daten kombinieren

Anforderungen

Traditionelle Ansätze

Entwurf

Zusammenfassung

14 Daten verwalten, regeln und beobachten

Lebenszyklus verwalten

Entwurf

Manifest

Daten regeln

Entwurf

Standardisierte Policies

Integration von Daten und Policies

Verknüpfung von Policies

Daten beobachten

Entwurf

Zusammenfassung

Teil V Wie führt man Data Mesh ein?

15 Strategie und Umsetzung

Sollten Sie Data Mesh schon jetzt einführen?

Data Mesh als ein Teil der Datenstrategie

Data Mesh Execution Framework

Fachlich getriebene Umsetzung

Iterative Ende-zu-Ende-Umsetzung

Evolutionäre Umsetzung

Zusammenfassung

16 Organisation und Unternehmenskultur

Veränderungen

Kultur

Werte

Anreize

Intrinsische Motivation

Extrinsische Motivation

Struktur

Annahmen über die Organisationsstruktur

Zuschnitt von Datenprodukten

Menschen

Rollen

Weiterbildung

Prozess

Wesentliche Prozessänderungen

Zusammenfassung

Index

Über die Autorin

Über die Übersetzer

Kolophon

Das Prinzip Data as a Product

Eine seit Langem bestehende Herausforderung bei analytischen Datenarchitekturen sind die hohen Hürden und Kosten der Datennutzung: *Auffinden, Verstehen, Vertrauen, Erforschen und schließlich Konsumieren von hochwertigen Daten*. Es gibt zahlreiche Studien, die diese Probleme aufzeigen. Ein aktueller Bericht, »The State of Data Science 2020« (<https://oreil.ly/S8XMz>) von Anaconda, einem Unternehmen für Data-Science-Plattformen, kommt zu dem Ergebnis, dass fast die Hälfte der Zeit eines Data Scientist für die Datenaufbereitung (das Laden und Bereinigen von Daten) aufgewendet wird. Wird dieses Problem nicht angegangen, wird es sich mit Data Mesh noch verschärfen, da die Anzahl der Stellen und Teams, die Daten bereitstellen, sprich der Domänen, zunimmt. Die Verlagerung von Data Ownership in die Hände der Domänen wirft wichtige Fragen in Bezug auf Zugriffsmöglichkeiten, Benutzerfreundlichkeit und Standardisierung auf. Eine weitere Isolierung der Daten und eine schlechtere Usability sind mögliche unerwünschte Folgen des ersten Prinzips von Data Mesh, der *Domain Ownership*. Das Prinzip *Data as a Product* geht auf diese Bedenken ein.

Mit dem zweiten Data-Mesh-Prinzip, *Data as a Product*, wird *Product Thinking* auf Daten angewendet, um solche Usability-Probleme zu beseitigen und die User Experience der Datennutzenden (Data Scientists, Data Analysts und alle weiteren) zu verbessern. *Data as a Product* setzt voraus, dass die von den Domänen bereitgestellten analytischen Daten wie ein Produkt behandelt werden und dass die Nutzerinnen und Nutzer dieser Daten wie Kundinnen und Kunden behandelt werden – als glückliche und zufriedene Kundinnen und Kunden. Darüber hinaus unterstreicht »Daten als Produkt« die Vorteile von Data Mesh, indem es den Wert der Daten eines Unternehmens durch eine drastische Erhöhung des Potenzials für die geplante und ungeplante Nutzung freisetzt.

In seinem Buch *INSPIRED* (<https://oreil.ly/mzv7u>) liefert Marty Cagan, ein prominenter Vordenker auf dem Gebiet der Produktentwicklung und des

Produktmanagements, überzeugende Beweise dafür, dass erfolgreiche Produkte gemeinsam drei Merkmale aufweisen: Sie sind *praktikabel* (engl. *feasible*), *nützlich* (engl. *valueable*) und *benutzerfreundlich* (engl. *usable*). Das Prinzip *Data as a Product* definiert ein neues Konzept, das *Datenprodukt*, das standardisierte Eigenschaften verkörpert, um Daten *nützlich* und *benutzerfreundlich* zu gestalten. Abbildung 3-1 demonstriert das visuell. In diesem Kapitel werden diese Merkmale vorgestellt. Kapitel 4 beschreibt, wie man die Erstellung von Datenprodukten *praktikabel* macht.

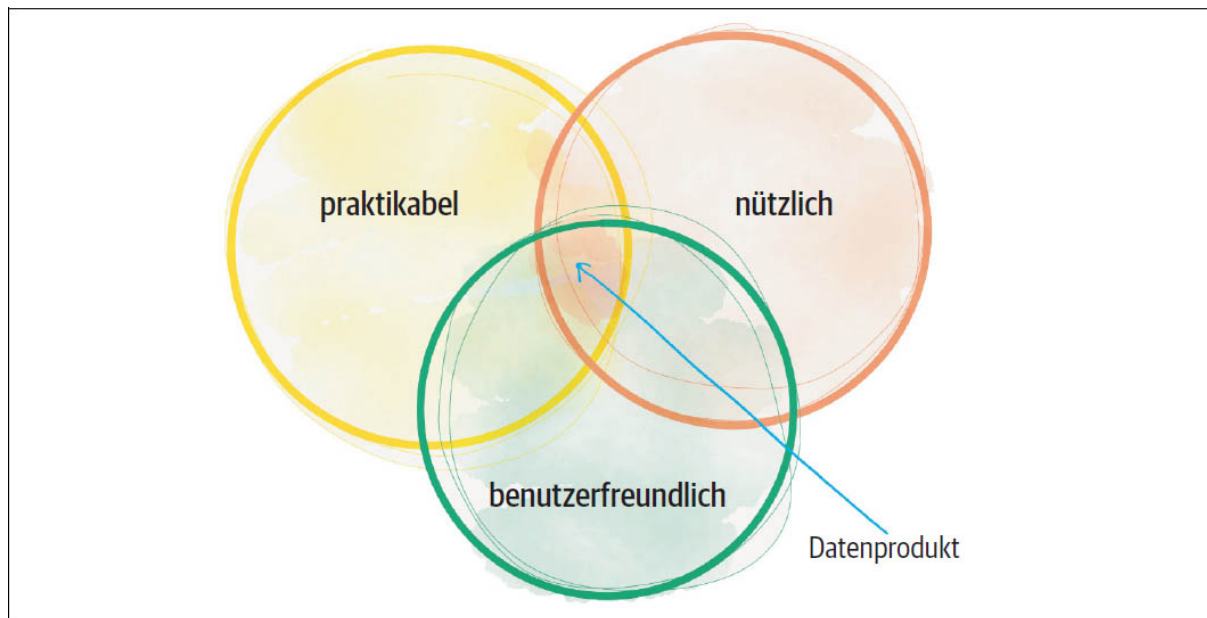


Abbildung 3-1: Datenprodukte liegen im Schnittpunkt von Marty Cagans Merkmalen erfolgreicher Produkte.

Damit Daten ein Produkt sind, müssen sie bestimmte Regeln befolgen und eine Reihe von Merkmalen aufweisen, damit sie genau in den Schnittpunkt des Venn-Diagramms von Cagan (Benutzerfreundlichkeit, Praktikabilität und Nutzen) fallen. Damit Daten ein Produkt sind, müssen sie *nützlich* sein – für sich selbst und in Zusammenarbeit mit anderen Datenprodukten. Sie müssen *Empathie* für ihre Nutzerinnen und Nutzer zeigen und für ihre Benutzerfreundlichkeit und Integrität sorgen.

Ich gebe zu, dass die Behandlung von Daten als Produkt nicht einfach so geschieht. Daher führt dieses Prinzip neue Rollen ein, wie *Domain Data Product Owner* und *Data Product Developer*, die dafür verantwortlich sind, Datenprodukte zu erstellen, bereitzustellen und zu bewerben, während sie gleichzeitig die spezifischen objektiven Merkmale des Datenzugriffs, der Qualität, der Benutzerfreundlichkeit und der Verfügbarkeit während der gesamten Lebensdauer der Datenprodukte sicherstellen. Kapitel 16 wird weitere Einzelheiten über diese neuen Rollen behandeln.

Im Vergleich zu früheren Paradigmen kehren Daten als Produkt die Verantwortung um. In Data-Lake- oder Data-Warehouse-Architekturen liegt die Verantwortung für die Erstellung von qualitativ hochwertigen und integren Daten beim zentralisierten Datenteam. Data Mesh verlagert diese Verantwortung direkt an die Quelle der Daten. Dieser Übergang ist nicht nur bei Data Mesh zu beobachten; in den letzten zehn Jahren hat sich der Trend zur *Linksverschiebung* bei Tests und im Betrieb durchgesetzt, da die Lösung von Problemen billiger und effektiver ist, wenn sie nahe an der Quelle erfolgt.

Ich gehe sogar so weit zu sagen, dass das, was in einem Mesh bereitgestellt wird, nicht nur *Daten* sind; es ist ein *Datenprodukt*.



Bei *Data as a Product* geht es darum, Product Thinking auf die Art und Weise anzuwenden, wie Daten modelliert und bereitgestellt werden. Dies ist nicht zu verwechseln mit dem Verkauf von Produkten.

Im Folgenden möchte ich erläutern, wie wir Product Thinking auf Daten anwenden können.

Product Thinking

In den letzten zehn Jahren haben sich erfolgreiche Unternehmen entschieden, ihre interne Betriebstechnologie wie ein Produkt zu behandeln, ähnlich wie ihre externe Technologie. Sie behandeln ihre internen Entwicklerinnen und Entwickler als Kundinnen und Kunden und ihre Zufriedenheit als Zeichen des Erfolgs. Insbesondere in zwei Bereichen hat sich dieser Trend durchgesetzt: in der Anwendung von Produktmanagementmethoden auf interne Plattformen, wodurch interne Entwicklerinnen und Entwickler schneller in der Lage sind, Lösungen auf diesen Plattformen zu erstellen und zu hosten (z.B. Spotify Backstage (<https://oreil.ly/B1fwB>)), und in der Behandlung von APIs als Produkt, um APIs zu erstellen, die auffindbar, verständlich und leicht testbar sind, um eine optimale Developer Experience zu gewährleisten (z.B. Square APIs (<https://oreil.ly/gG9eL>)).

Die Anwendung des *Product Thinking* auf interne Technologien beginnt damit, dass man sich in die internen Kunden (d.h. die Entwicklerkolleginnen und -kollegen) hineinversetzt, mit ihnen zusammenarbeitet, Nutzungsdaten sammelt und die internen technischen Lösungen im Laufe der Zeit kontinuierlich weiterentwickelt, um die Benutzerfreundlichkeit zu verbessern. Erfolgreiche

digitale Organisationen stellen erhebliche Ressourcen und Aufmerksamkeit für die Entwicklung interner Tools bereit, die für die Entwicklerinnen und Entwickler und letztlich für das ganze Unternehmen hilfreich sind.

Seltsamerweise fehlt bei Big-Data-Lösungen häufig diese Empathie, d.h. die Behandlung von Daten als Produkt und ihrer Nutzenden als Kundinnen und Kunden. Operative Teams betrachten ihre Daten immer noch als Nebenprodukt des Betriebs und überlassen es jemand anderem, z.B. dem Datenteam, sie aufzusammeln und zu *Produkten* zu recyceln. Bei Data Mesh wenden im Gegensatz dazu die Domänenteams Product Thinking konsequent auf ihre Daten an und streben nach der besten User Experience für die Daten.

Nehmen wir wieder das Beispiel von Daff: Eine seiner wichtigsten Domänen ist die **Media-Player-Domäne**. Sie liefert wichtige Daten, z.B. welche Lieder von wem, wann und wo gespielt wurden. Für diese Informationen gibt es mehrere verschiedene Hauptnutzer. Das **Media-Player-Supportteam** ist beispielsweise an Near-Realtime-Events interessiert, um Fehler, die die User Experience der Kundinnen und Kunden beeinträchtigen, schnell zu erkennen und den Service wiederherzustellen oder um auf eingehende Anrufe beim Kundensupport sachkundig zu reagieren. Das **Media-Player-Designteam** hingegen ist an aggregierten Play-Events interessiert, die eine Aussage über die Entwicklung der Hörerinnen und Hörern über einen längeren Zeitraum hinweg machen, um die Media Player mit ansprechenderen Funktionen zu optimieren und die User Experience insgesamt zu verbessern.

Mit einem fundierten Verständnis dieser Anwendungsfälle und der Informationen, die die anderen Teams benötigen, stellt die **Media-Player-Domäne** zwei verschiedene Arten von Daten als Produkte für den Rest des Unternehmens bereit: *Play-Events*, die als kontinuierlicher Event-Stream veröffentlicht werden, und aggregierte *Play-Sessions*, die als einzelne Dateien in einen Object Store gelegt werden. Das ist Product Ownership, angewandt auf Daten.

Abbildung 3-2 zeigt die Datenprodukte der Media-Player-Domäne.

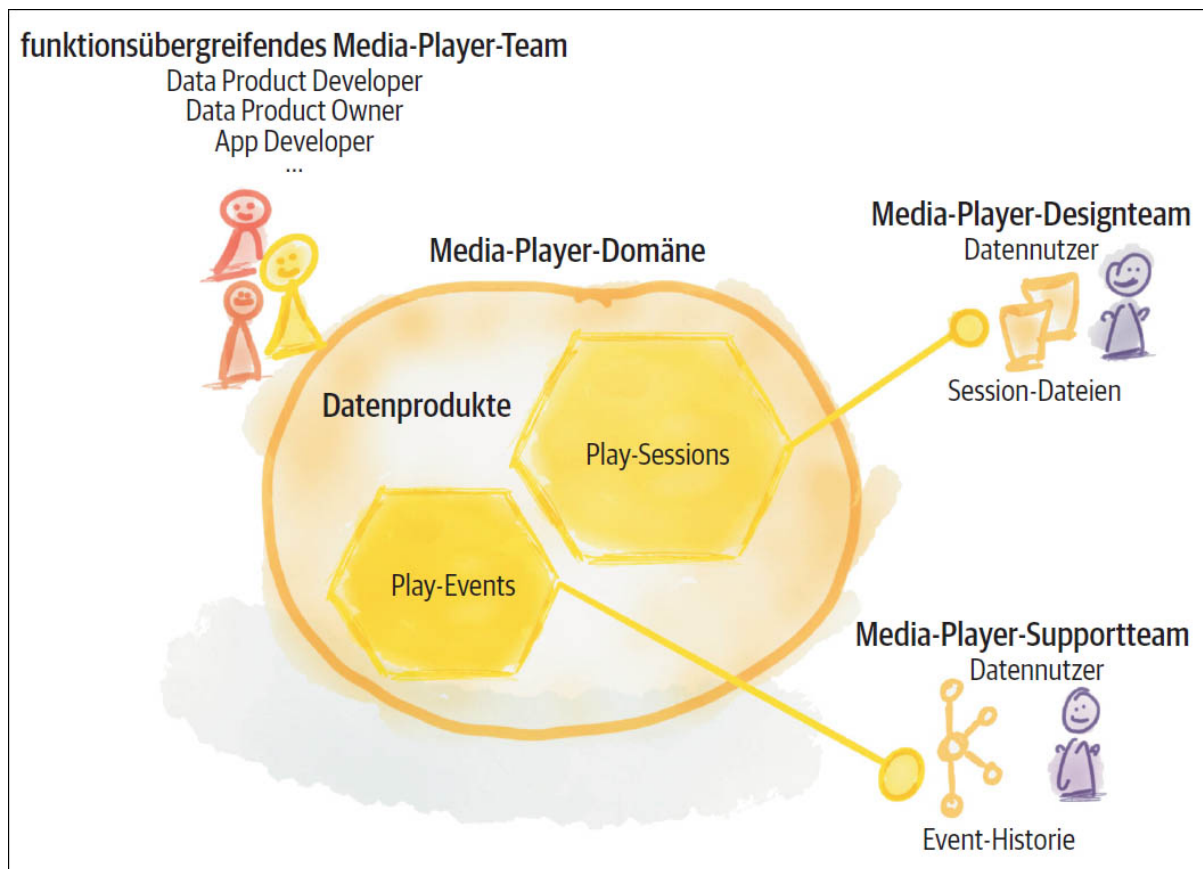


Abbildung 3-2: Beispiel für Datenprodukte

Wie Sie sich vorstellen können, lassen sich die meisten Product-Ownership-Methoden auf Daten übertragen. Daten haben jedoch eine Besonderheit: Der Unterschied zwischen Datenprodukten und anderen Produkten liegt in den vielfältigen Verwendungsmöglichkeiten von Daten, d.h. in den Möglichkeiten, bestimmte Daten mit anderen Daten zu kombinieren und letztendlich in Erkenntnisse und Maßnahmen umzusetzen. Data Product Owner wissen, welche Anwendungsfälle für ihre Daten aktuell bekannt sind, während ein großer Teil der zukünftigen Anwendungsfälle für die heute produzierten Daten noch unbekannt ist und vielleicht außerhalb ihrer Vorstellungskraft liegt.

Dies gilt vor allem für quellenorientierte Domänen und weniger für konsumentenorientierte Domänen. Die quellenorientierten Daten erfassen die Realität der fachlichen Interaktionen und Ereignisse, so wie sie stattgefunden haben. Sie können auch irgendwann in der Zukunft noch verwendet, transformiert und neu interpretiert werden. Quellenorientierte Datenprodukte müssen ein Gleichgewicht zwischen den unmittelbar bekannten und den unbekanntem Anwendungsfällen herstellen. Sie sollten sich daher bemühen, die Realität des Unternehmens so genau wie möglich in ihren Daten abzubilden, ohne dabei zu viele Annahmen über die Verwendung der Daten zu treffen. Zum Beispiel ist die Erfassung aller **Play-Events** als kontinuierliches, detailliertes

Journal eine gute Wahl. Sie ermöglicht zukünftigen Nutzerinnen und Nutzern, andere Transformationen vorzunehmen und neue Erkenntnisse aus den heute erfassten Daten abzuleiten.

Dies ist der Hauptunterschied zwischen der Gestaltung von Datenprodukten und der Gestaltung von Softwareprodukten.

Grundlegende Usability-Attribute von Datenprodukten

Meiner Meinung nach gibt es eine Reihe von nicht verhandelbaren grundlegenden Eigenschaften, die ein Datenprodukt aufweisen muss, um als *benutzerfreundlich* zu gelten. Diese Merkmale gelten für alle Datenprodukte, und zwar unabhängig von ihrer Domäne oder ihrem Archetyp. Ich nenne sie *grundlegende Usability-Attribute von Datenprodukten*. Jedes Datenprodukt muss diese Merkmale aufweisen, um Teil des Mesh zu sein. Abbildung 3-3 listet die Usability-Attribute von Datenprodukten auf.

Beachten Sie, dass sich diese Merkmale der Benutzerfreundlichkeit auf die Personen beziehen, die Daten nutzen. Sie sind nicht dazu gedacht, die technischen Eigenschaften zu beschreiben. Teil IV, »Wie entwirft man Datenprodukte?«, geht auf die technischen Eigenschaften von Datenprodukten ein.

Die in diesem Abschnitt aufgeführten grundlegenden Merkmale sind eine Ergänzung zu dem, was bisher auch unter dem Akronym FAIR-Daten (<https://oreil.ly/2BB7V>) zusammengefasst wurde – Daten, die den Grundsätzen der *Auffindbarkeit* (engl. *Findability*), *Zugänglichkeit* (engl. *Accessibility*), *Interoperabilität* (engl. *Interoperability*) und *Wiederverwendbarkeit* (engl. *Reusability*) entsprechen.¹ Zusätzlich zu diesen Prinzipien stelle ich Merkmale vor, die notwendig sind, damit verteilte Data Ownership funktioniert.

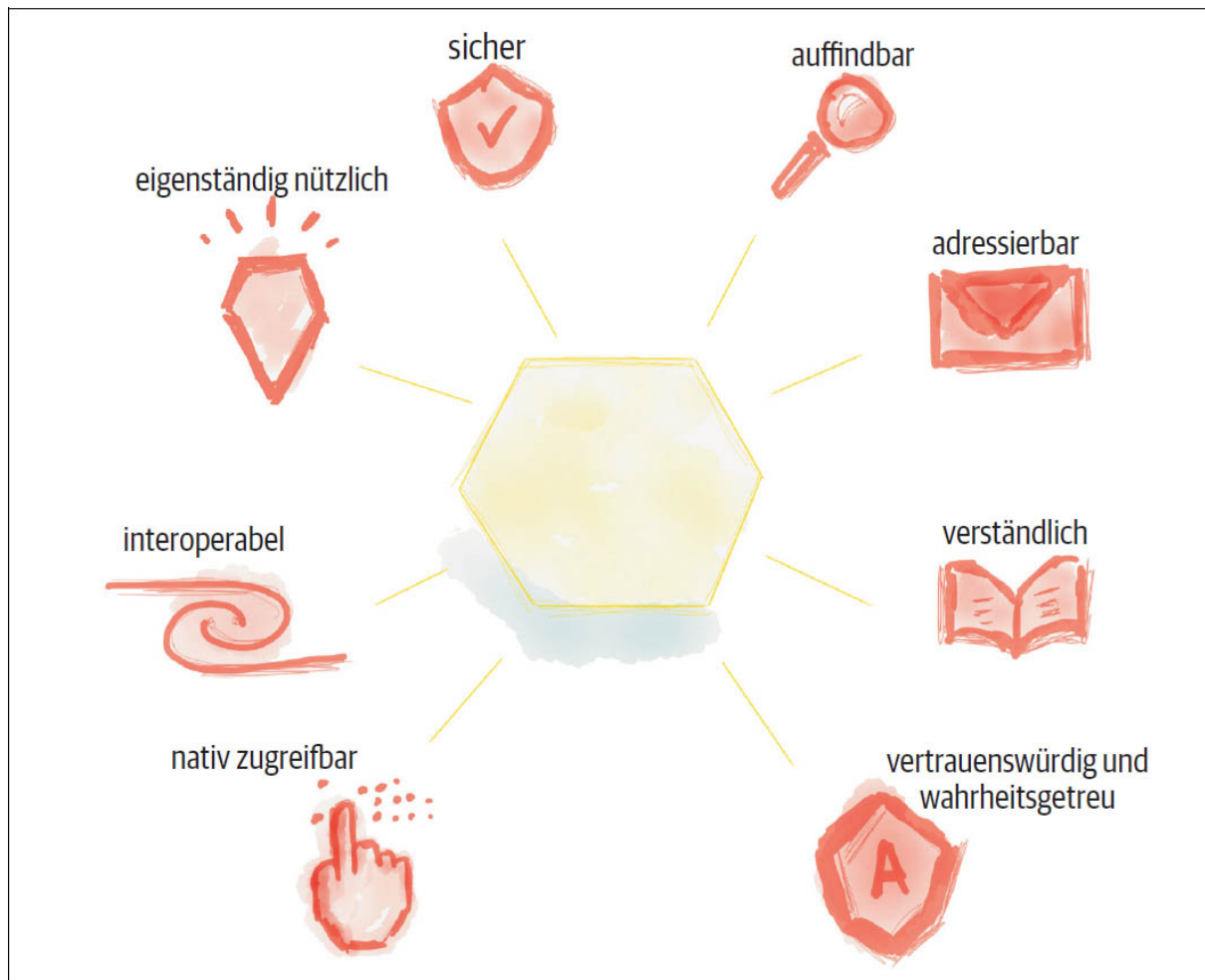


Abbildung 3-3: Grundlegende Usability-Attribute von Datenprodukten

Versetzen wir uns in die Lage der Datennutzerinnen und Nutzer und gehen wir diese Attribute durch.

Auffindbar

Zwei der wichtigsten Eigenschaften guten Designs sind Discoverability und Verständlichkeit. Discoverability: Ist es möglich, überhaupt herauszufinden, welche Aktionen möglich sind und wo und wie man sie ausführt? Verständlichkeit: Was bedeutet das alles? Wie soll es genutzt werden? Was bedeuten all die verschiedenen Bedienelemente und Einstellungen?²

– Don Norman

Der allererste Schritt, den Datennutzerinnen und -nutzer unternehmen, besteht darin, die ihnen zur Verfügung stehende Datenwelt zu entdecken, zu erkunden und zu suchen, um »das Richtige« zu finden. Daher ist Discoverability eines der ersten Usability-Attribute. Die Nutzer von Daten müssen in der Lage sein, die verfügbaren Datenprodukte zu erkunden, die gewünschten Datensätze zu

suchen und zu finden, sie zu prüfen und Vertrauen zu gewinnen. Eine klassische Umsetzung der Discoverability ist ein zentrales Verzeichnis oder ein Katalog, in dem die verfügbaren Datensätze mit zusätzlichen Informationen über den jeweiligen Datensatz, die Owner, den Standort, Beispieldaten usw. aufgeführt sind. Häufig werden diese Informationen nachträglich vom zentralen Datenteam oder dem Governance-Team kuratiert.

In Data Mesh wird die Discoverability durch eine Linksverschiebung erreicht, bei der das Datenprodukt selbst Informationen zur Discoverability enthält. Data Mesh berücksichtigt also die dynamische Topologie des Mesh, die sich ständig weiterentwickelnden Datenprodukte und die enorme Menge der verfügbaren Datenprodukte. Es stützt sich daher darauf, dass die einzelnen Datenprodukte ihre Discoverability-Informationen während ihres gesamten Lebenszyklus (Erstellung, Deployment und Betrieb) auf standardisierte Weise bereitstellen. Jedes Datenprodukt teilt kontinuierlich seine Datenquelle, seine Owner, Laufzeitinformationen wie Aktualität, Qualitätsmetriken, Beispieldatensätze und vor allem Informationen, die von seinen Konsumenten beigesteuert werden, wie z.B. die wichtigsten Anwendungsfälle und Anwendungen, die durch seine Daten ermöglicht werden.

Teil IV, »Wie entwirft man Datenprodukte?«, wird die technische Ausgestaltung der Discoverability von Datenprodukten behandeln.

Adressierbar

Ein Datenprodukt hat eine dauerhafte und eindeutige Adresse für den programmatischen oder manuellen Zugriff. Dieses Adressierungssystem berücksichtigt die dynamische Eigenschaft der Daten und der Mesh-Topologie. Es muss berücksichtigen, dass sich viele Aspekte der Datenprodukte weiterhin ändern werden, während es gleichzeitig eine kontinuierliche Nutzung sicherstellt.

Das Adressierungssystem berücksichtigt unter anderem die folgenden sich ständig ändernden Aspekte eines Datenprodukts, wobei der Zugriff auf das Datenprodukt über seine dauerhafte, eindeutige Adresse erhalten bleibt:

Semantische und syntaktische Änderungen in Datenprodukten

Schema-Evolution.

Kontinuierliche Bereitstellung neuer Daten

Partitionierungsstrategie und Gruppierung von Daten, die einem bestimmten Zeitpunkt (oder Zeitfenster) zugeordnet sind.

Neu unterstützte Arten des Datenzugriffs

Neue Wege der Serialisierung, Darstellung und Abfrage der Daten.

Änderungen von Laufzeitinformationen

Zum Beispiel Service Level Objectives, Zugriffsprotokolle, Debug-Logs.

Die eindeutige Adresse muss einer globalen Konvention folgen, die es allen Benutzerinnen und Benutzern erleichtert, programmatisch und einheitlich auf alle Datenprodukte zuzugreifen. Das Datenprodukt muss über ein adressierbares Aggregate Root (<https://oreil.ly/D4M9x>) verfügen, das als Einstieg zu allen Informationen über ein Datenprodukt dient, einschließlich seiner Dokumentation, Service Level Objectives und der eigentlichen Daten, die es bereitstellt.

Verständlich

Sobald ein Datenprodukt gefunden wurde, besteht der nächste Schritt darin, es zu verstehen. Das bedeutet, dass man Syntax und Semantik lernt.

Jedes Datenprodukt liefert semantisch kohärente Daten: Daten mit einer bestimmten Bedeutung. Jede Datennutzerin und jeder Datennutzer muss diese Bedeutung verstehen: welche Art von Entitäten das Datenprodukt kapselt, wie die Beziehungen zwischen den Entitäten sind und welche benachbarten Datenprodukte es gibt.

Um auf das Beispiel der **Media-Player-Events** zurückzukommen, so sollte man leicht verstehen, was ein Player-Event ausmacht: **User**, durchgeführte **Play Actions**, **Time** und **Location** ihrer Aktion und **Feedback**, das die Aktion ausgelöst hat. Man sollte leicht verstehen, welche Arten von Aktionen verfügbar sind und dass es eine Beziehung zwischen einem Hörer oder einer Hörerin, die ein Player-Event auslöst, und einem **Subscriber** aus der verwandten **Listener-Domäne** gibt. Datenprodukte liefern eine formale Darstellung solcher *Semantiken*.

Neben dem Verständnis der Semantik müssen die Datennutzerinnen und Datennutzer auch verstehen, wie genau die Daten vorliegen. Wie werden sie serialisiert, und wie kann man auf die Daten zugreifen, wie ist die Syntax der Abfrage? Welche Arten von Abfragen sind möglich, und wie können die Datenobjekte gelesen werden? Sie müssen das *Schema* der zugrunde liegenden *Syntax* der Daten verstehen. Diese Informationen werden idealerweise durch Beispieldatensätze und Beispielcodes ergänzt. Beispiele in Verbindung mit einer formalisierten Beschreibung der Daten erleichtern das Verständnis.

Darüber hinaus sind dynamische und interaktive Dokumente wie Computational Notebooks (<https://oreil.ly/k0TmE>) hervorragende Hilfsmittel, um die Geschichte

eines Datenprodukts zu erzählen. Solche Notebooks enthalten sowohl eine Dokumentation der Daten als auch den Code für die Verwendung der Daten mit einem unmittelbaren Feedback, das das Ergebnis visualisiert.

Und schließlich ist das Verstehen auch ein sozialer Prozess. Wir lernen voneinander. Datenprodukte erleichtern die Kommunikation zwischen ihren Nutzenden, damit sie ihre Erfahrungen austauschen können und erfahren, wie sie das Datenprodukt nutzen.

Um ein Datenprodukt zu verstehen, sollte man nicht auf fremde Hilfe angewiesen sein. Datenprodukte eigenständig verstehen zu können, ist ein grundlegendes Merkmal der Benutzerfreundlichkeit.

Vertrauenswürdig und wahrheitsgetreu

[Vertrauen ist] eine zuversichtliche Beziehung zum Unbekannten.³

– Rachel Botsman

Niemand wird ein Produkt verwenden, dem sie oder er nicht vertraut. Was bedeutet es also, einem Datenprodukt zu vertrauen, und, was noch wichtiger ist, was ist notwendig, um zu vertrauen? Um diese Frage zu klären, verwende ich gern das von Rachel Botsman vorgeschlagene Konzept des Vertrauens: *die Brücke zwischen dem Bekannten und dem Unbekannten*. Ein Datenprodukt muss die Lücke schließen zwischen dem, was Datennutzende sicher über die Daten wissen, und dem, was sie nicht wissen, aber wissen müssen, um ihnen zu vertrauen. Während die vorherigen Merkmale wie Verständlichkeit und Discoverability diese Lücke bis zu einem gewissen Grad schließen, braucht es viel mehr, um den Daten für die Nutzung zu vertrauen.

Die Datennutzerinnen und -nutzer müssen sich darauf verlassen können, dass das Datenprodukt wahrheitsgetreu ist, also dass es den Sachverhalt des Unternehmens korrekt wiedergibt. Sie müssen einschätzen können, wie genau die Daten die Realität der eingetretenen Ereignisse widerspiegeln und wie hoch die Wahrscheinlichkeit ist, dass die Aggregationen und Projektionen, die aus den geschäftlichen Fakten erstellt wurden, wahrheitsgemäß sind.

Ein Beitrag zur Vertrauensbildung besteht darin, Service Level Objectives (<https://oreil.ly/41TpM>) (SLOs) der Datenprodukte zu gewährleisten und zu kommunizieren.

Zu den Datenprodukt-SLOs gehören unter anderem:

Interval of Change

Wie oft Änderungen in den Daten wiedergespiegelt werden.

Timeliness

Die Zeitspanne zwischen dem Auftreten einer Tatsache und dem Zeitpunkt, an dem sie den Datennutzern zur Verfügung steht.

Completeness

Der Grad der Verfügbarkeit aller relevanten Informationen.

Statistical Shape of Data

Verteilung, Umfang, Volumen usw.

Lineage

Der Weg der Datentransformation von der Quelle bis hierher.

Precision und Accuracy

Der Grad der korrekten fachlichen Abbildung im Laufe der Zeit.

Operative Eigenschaften

Freshness, Verfügbarkeit, Performance.

Beim traditionellen Datenmanagement ist es durchaus üblich, fehlerhafte Daten zu extrahieren und zu importieren, das heißt, sie spiegeln nicht unbedingt die Realität des Unternehmens wider und sind schlichtweg nicht vertrauenswürdig. Darauf konzentriert sich der Großteil der Arbeit zentralisierter Daten-Pipelines, nämlich auf die Datenbereinigung nach dem Einlesen.

Im Gegensatz dazu bringt Data Mesh den fundamentalen Wandel mit sich, dass die Data Product Owner die Qualität und Vertrauenswürdigkeit – spezifisch für ihre Domäne – als wesentliches Merkmal ihres Datenprodukts kommunizieren und gewährleisten müssen. Dies bedeutet, dass die Datenbereinigung und die automatisierten Datenintegritätstests zum Zeitpunkt der Erstellung eines Datenprodukts erfolgen müssen.

Die Angabe der Datenherkunft und der Lineage – also des Pfads der Daten, woher sie stammen und wie sie hierhergelangt sind – als Metadaten zu jedem Datenprodukt hilft, weiteres Vertrauen in das Datenprodukt zu gewinnen. Die Nutzerinnen und Nutzer können diese Informationen auswerten, um festzustellen, ob die Daten für ihre speziellen Bedürfnisse geeignet sind. Ich bin der Meinung, dass, nachdem das Vertrauen in Datenprodukte aufgebaut wurde, weniger Bedarf besteht, Vertrauen durch investigative und detektivische Methoden bei der Durchsicht des Lineage-Pfads aufzubauen. Dennoch wird Lineage in einigen Szenarien wichtig bleiben, z.B. bei der Post-Mortem-Analyse, bei der Fehlersuche, bei Audits zur Einhaltung von Policies und bei der Bewertung der Eignung von Daten für das ML-Training.

Nativ zugreifbar

Je nach Reifegrad der Organisation gibt es ein weites Spektrum an Personas von Datennutzerinnen und -nutzern, die Zugriff auf Datenprodukte benötigen. Das Spektrum umfasst ein breites Profil, wie in Abbildung 3-4 dargestellt: Data Analysts, die mit der Auswertung von Daten in Tabellenkalkulationen vertraut sind, Data Analysts, die mit Abfragesprachen vertraut sind, um statistische Modelle von Daten in Form von Visualisierungen oder Reports zu erstellen, Data Scientists, die Daten kuratieren und strukturieren und Data Frames verwenden, um ihre ML-Modelle zu trainieren, und Entwicklerinnen und Entwickler von datenintensiven Anwendungen, die einen Event-Stream oder Pull-basierte APIs erwarten. Dies ist ein ziemlich breites Spektrum an Personas mit ebenso unterschiedlichen Erwartungen hinsichtlich des Zugriffs und des Lesens von Daten.

Es besteht ein direkter Zusammenhang zwischen der *Usability* und der Einfachheit, mit der Datennutzende mit ihren gewohnten Tools auf die Daten zugreifen können. Daher muss ein Datenprodukt verschiedenen Personas den Zugriff im jeweiligen nativen Datenformat ermöglichen. Dies kann durch eine polyglotte Speicherung der Daten oder durch die Entwicklung mehrerer Adapter zum Lesen der Daten realisiert werden.

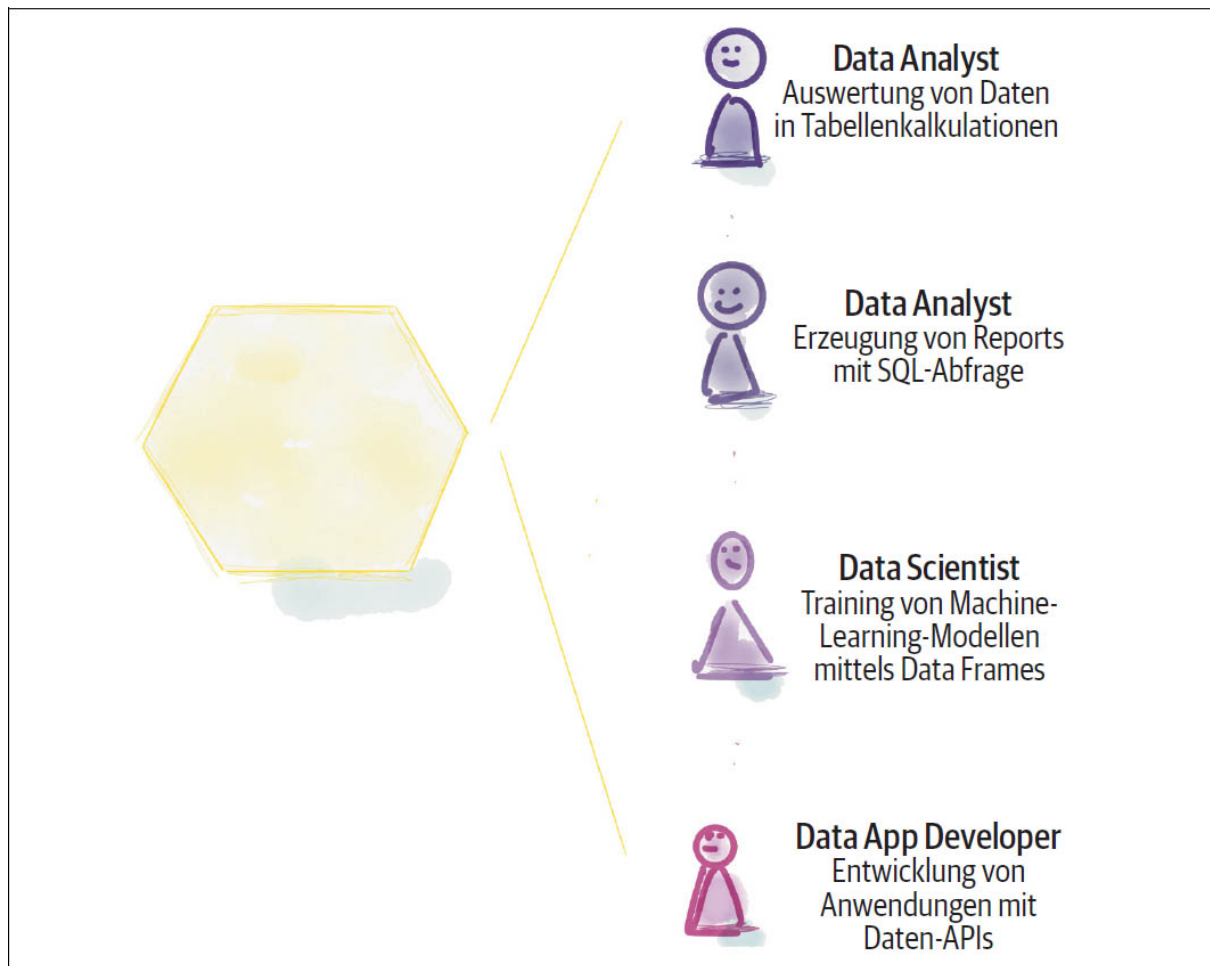


Abbildung 3-4: Beispiel für ein breites Spektrum von Nutzerinnen und Nutzern eines Datenprodukts mit unterschiedlichen Zugriffsmustern

Das Datenprodukt **Play-Events** muss beispielsweise SQL-Abfragen unterstützen, um den nativen Zugriffsmodus von Data Analysts zu erfüllen, sowie die Veröffentlichung eines Event-Streams für datenintensive Anwendungen und spaltenbasierte Dateien für Data Scientists.

Interoperabel

Einer der wichtigsten Aspekte in einer verteilten Datenarchitektur ist die Fähigkeit, Daten über Domänen hinweg zu korrelieren und sie auf sinnvolle Weise miteinander zu verknüpfen: mittels Joins, Filtern und Aggregationen. Entscheidend für eine effektive Kombinierbarkeit von Daten über Domänen hinweg ist die Einhaltung von Standards und Regeln zur Harmonisierung, die eine einfache *Verknüpfung von Daten* über Domänen hinweg ermöglichen.

Hier sind einige Dinge, die Datenprodukte standardisieren müssen, um Interoperabilität und Kompatibilität zu fördern:

Feldtypen

Ein gemeinsames, explizit definiertes Typensystem.

Polysemer Identifikator

Universelle Identifizierung von Entitäten über alle Datenprodukte hinweg.

Globales Adresssystem für Datenprodukte

Eine eindeutige globale Adresse, die jedem Datenprodukt zugewiesen wird, idealerweise mit einem einheitlichen Schema, um die Verknüpfung mit verschiedenen Datenprodukten zu erleichtern.

Gemeinsame Metadatenfelder

Beispielsweise für den fachlichen Zeitpunkt und den Verarbeitungszeitpunkt.

Schema-Linking

Fähigkeit zur Verknüpfung und Wiederverwendung von Schematypen, die von anderen Datenprodukten definiert wurden.

Data-Linking

Möglichkeit der Verknüpfung oder Zuordnung zu Daten in anderen Datenprodukten.

Schemastabilität

Ansatz zur Entwicklung von Schemata, der die Abwärtskompatibilität berücksichtigt.

Nehmen wir zum Beispiel die Handhabung von polysemen Identifikatoren. Bei Daff ist *Artist* ein zentrales Modell, das in verschiedenen Domänen auftaucht. Ein *Artist* ist zwar die gleiche globale Entität, hat aber in jeder Domäne unterschiedliche Attribute und wahrscheinlich auch unterschiedliche Bezeichner. Das Datenprodukt **Play-Events** erfasst eine Künstlerin oder einen Künstler anders als die Domäne **Payment**, die sich um die Rechnungen und Zahlungen für die Tantiemen kümmert. Um Daten über eine Künstlerin oder einen Künstler über verschiedene Datenprodukte hinweg zu korrelieren, muss sich Daff darauf einigen, wie ein *Artist* global und über alle Datenprodukte hinweg identifiziert werden kann.

Kapitel 5 behandelt die Frage der globalen Standards und Protokolle, die auf jedes Datenprodukt angewendet werden. Interoperabilität ist die Grundlage eines jeden verteilten Systems, und Data Mesh ist da keine Ausnahme.

Eigenständig nützlich

Ich denke, es ist ziemlich offensichtlich, dass ein Datenprodukt nützlich sein muss, es sollte einen inhärenten betriebswirtschaftlichen Wert haben. Wenn sich ein Data Product Owner keinen Wert des Datenprodukts vorstellen kann, ist es vielleicht besser, es nicht zu entwickeln. In diesem Zusammenhang sei darauf hingewiesen, dass ein Datenprodukt einen Datensatz enthalten sollte, der für sich genommen *nützlich* und *sinnvoll* ist, auch ohne dass er mit anderen Datenprodukten verknüpft und korreliert wird.

Natürlich gibt es immer eine weitergehende Aussage sowie Erkenntnisse und Werte, die sich aus der Korrelation mehrerer Datenprodukte ableiten lassen. Wenn jedoch ein Datenprodukt für sich allein keinen Wert bereitstellt, sollte es nicht existieren.

Auch wenn dies offensichtlich klingen mag, gibt es ein häufiges Antipattern bei der Migration von einer Data-Warehouse-Architektur zu Data Mesh: Die direkte Zuordnung von Warehouse-Tabellen zu Datenprodukten kann zu Datenprodukten ohne Wert führen. Im Data Warehouse gibt es Glue-Tabellen (auch Faktentabellen genannt), die die Korrelation zwischen Entitäten verbessern. Dabei handelt es sich um Identitätstabellen, die Bezeichner einer Art von Entität einer anderen zuordnen. Solche Identitätstabellen sind für sich genommen nicht sinnvoll oder wertvoll, solange sie nicht mit anderen Tabellen verknüpft sind. Sie sind lediglich mechanische Implementierungen zur Erleichterung von Joins.

Im Gegenteil, es gibt keine Datenprodukte, die nur dazu dienen, Informationen über das Mesh zu korrelieren. Optimierungen wie Indizes oder Faktentabellen müssen von der Plattform automatisch erstellt werden und werden vor den Datenprodukten versteckt.

Sicher

Datennutzerinnen und Datennutzer greifen sicher und unter Wahrung der Vertraulichkeit auf ein Datenprodukt zu. Datensicherheit ist ein absolutes Muss, unabhängig davon, ob die Architektur zentralisiert oder verteilt ist. In einer verteilten Architektur wie Data Mesh wird die Zugriffskontrolle jedoch durch das Datenprodukt direkt in der Verarbeitung von Daten, also dem *Zugriff*, dem *Lesen* oder dem *Schreiben* validiert. Die Berechtigungen können sich dynamisch ändern. Sie werden während des Zugriffs auf Datenprodukte kontinuierlich ausgewertet.

Der Zugriff auf ein Datenprodukt ist nicht unbedingt binär – also die Frage, ob eine Person die Daten sehen kann oder nicht. In vielen Fällen kann man zwar die

eigentlichen Datensätze nicht einsehen, man hat aber möglicherweise ausreichende Berechtigungen, um die Form der Daten zu sehen und sie anhand ihrer statistischen Merkmale auszuwerten.

Berechtigungen können zentral definiert werden, werden aber zur Laufzeit von jedem einzelnen Datenprodukt ausgewertet. Datenprodukte folgen der Praxis der Security Policy as Code (<https://oreil.ly/zH1k2>). Das bedeutet, dass Security Policies so geschrieben werden, dass sie versioniert, automatisch getestet, deployt und überwacht sowie automatisiert ausgewertet und kontrolliert werden können.

Eine Policy, die als Code beschrieben, getestet und gewartet wird, kann verschiedene sicherheitsrelevante Belange zum Ausdruck bringen, z.B. die folgenden:

Berechtigungen

Wer oder was (Systeme und Menschen) auf das Datenprodukt zugreifen kann und wie die Datennutzer darauf zugreifen können.

Verschlüsselung

Welche Arten der Verschlüsselung (*on disk*, *in memory* oder *in transit*) mit welchem Verschlüsselungsalgorithmus verwendet werden, wie man die Schlüssel verwaltet und wie man die Auswirkungen im Fall von Sicherheitsverstößen minimiert.

Vertraulichkeitsgrad

Welche Arten von vertraulichen Informationen, z.B. personenbezogene Daten, persönliche Gesundheitsinformationen usw., das Datenprodukt enthält.

Aufbewahrungsfristen

Wie lange die Informationen aufbewahrt werden müssen.

Regulierungen und Vorschriften

DSGVO, CCPA, fachspezifische Vorschriften, vertragliche Vereinbarungen.

Einführung von Data as a Product

Bei der Arbeit mit meinen Kunden habe ich festgestellt, dass sie den Data-Mesh-Prinzipien mit großer Offenheit gegenüberstehen und sich oft fragen: »Warum bin ich nicht selbst darauf gekommen?« oder gelegentlich sagen: »Wir haben etwas Ähnliches gemacht, aber nicht ganz dasselbe.« Die Prinzipien scheinen

intuitive und ganz natürliche nächste Schritte auf dem Weg der technologischen Modernisierung ihrer Organisation zu sein, einer Modernisierung der operativen Aspekte von Organisationen, z.B. des Übergangs zu Domain-orientierten Microservices und der Behandlung interner Dienste wie APIs als Produkte.

Ein Gefühl des Unbehagens stellt sich jedoch ein, wenn sie sich eingehender mit der Frage befassen, was tatsächlich erforderlich ist, um die Transformation in Richtung Data Mesh durchzuführen. In meinen Gesprächen mit ersten Data-Mesh-Implementierern stellte ich fest, dass sie zwar die Grundsätze und ihre Absicht, diese umzusetzen, verbalisieren, aber dass die Implementierungen stark von den vertrauten Techniken der Vergangenheit beeinflusst sind.

Aus diesem Grund habe ich mich entschlossen, eine Reihe von Denkanstößen für den Übergang sowie einige pragmatische Schritte aufzunehmen, um die Unterschiede zwischen dem bestehenden Paradigma und dem echten Verständnis von *Daten als Produkt* deutlich zu machen.

Ich lade Sie ein, weitere Ideen zu ergänzen, die mir entgangen sind.

Domänen verantworten Datenprodukte

In den letzten zehn Jahren haben sich die Teams immer mehr von funktional getrennten zu funktionsübergreifenden Teams entwickelt. Die DevOps-Bewegung hat dazu geführt, dass die Lücke zwischen der Entwicklung und dem Betrieb von Unternehmensdiensten geschlossen wurde und dass funktionsübergreifende Entwicklungs- und Betriebsteams gebildet wurden. Die kundenorientierte Produktentwicklung hat das Design und die Produktverantwortung näher zu den Entwicklerinnen und Entwicklern gebracht.

Die Einführung von analytischen Daten als Produkt ergänzt die Liste der bestehenden Verantwortlichkeiten funktionsübergreifender Domänenteams und erweitert ihre Aufgaben:

Data Product Developer

Die Rolle, die für die Entwicklung, Bereitstellung und Wartung der Datenprodukte der Domäne verantwortlich ist, solange die Datenprodukte existieren und verwendet werden.

Data Product Owner

Die Rolle, die für den Erfolg der Datenprodukte einer Domäne verantwortlich ist, indem sie einen Mehrwert liefert, die Datennutzerinnen und Datennutzer zufriedenstellt, ihre Nutzerzahlen erhöht und den Lebenszyklus der Datenprodukte aufrechterhält.

Definieren Sie diese Rollen für jede Domäne und weisen Sie den Rollen eine oder mehrere Personen zu, je nach Komplexität der Domäne und der Anzahl ihrer Datenprodukte.

Eine neue Sprache

Ein häufig verwendeter Begriff in der Datenanalyse ist *Ingestion*, d.h. der Empfang von Daten aus einer Upstream-Quelle – oft einer nicht vertrauenswürdigen Quelle, die Daten als Nebenprodukt ihrer Tätigkeit erzeugt hat. Es ist nun die Aufgabe der Downstream-Pipeline, die Daten aufzunehmen, zu bereinigen und zu verarbeiten, bevor sie zur Wertschöpfung genutzt werden können.

Data Mesh schlägt vor, die Begrifflichkeiten bei der Verwendung analytischer Daten von *Ingestion* zu *konsumieren* zu ändern. Der feine Unterschied besteht darin, dass die Upstream-Daten bereits bereinigt, aufbereitet und bereitgestellt werden, sodass sie einfach konsumiert werden können. Der Sprachwechsel schafft einen neuen kognitiven Rahmen, der besser mit dem Prinzip des Bereitstellens von Daten als Produkt übereinstimmt.

In diesem Zusammenhang muss auch der Begriff *Extraction*, der in ETL (*Extract, Transform, Load*) verwendet wird, kritisch bewertet werden. Die Extraktion suggeriert eine passive Rolle des Anbieters und eine aufdringliche Rolle des Konsumenten. Wie wir wissen, führt das Extrahieren von Daten aus einer operativen Datenbank, die nicht für eine externe Nutzung optimiert ist, zu allen möglichen pathologischen Kopplungen und einem fragilen Design. Stattdessen können wir die Sprache anpassen mithilfe von Begriffen wie *veröffentlichen*, *bereitstellen* oder *teilen*. Dies bedeutet, dass die Implementierung der Datenbereitstellung vom Zugriff auf Rohdatenbanken auf die bewusste Bereitstellung von Domain-Events oder Aggregaten verlagert wird.

Inzwischen haben Sie wahrscheinlich bemerkt, dass ich den Schwerpunkt auf die Sprache und die von uns verwendeten Metaphern lege. George Lakoff (<https://oreil.ly/3YZ26>), Professor für Kognitionswissenschaft und Linguistik an der University of California in Berkeley, zeigt in seinem Buch *Metaphors We Live By* (Metaphern, mit denen wir leben) auf elegante Weise, welche Folgen es hätte, wenn wir unsere Sprache um das Konzept des *Arguments* herum verändern würden, vom *Krieg* zum *Tanz*. Stellen Sie sich die Welt vor, in der wir leben würden, und die Beziehungen, die wir pflegen würden, wenn wir als Tänzerin oder Tänzer ein ausgewogenes und ästhetisch ansprechendes Argument vortragen und unsere Ideen und Gefühle durch das schöne und gemeinschaftliche Ritual des Tanzens zum Ausdruck bringen würden, anstatt ein

Argument zu gewinnen oder zu verlieren und die Schwachpunkte eines Arguments anzugreifen. Diese unerwartete Umgestaltung der Sprache hat tiefgreifende Auswirkungen auf das Verhalten.

Daten als Produkt statt als Asset

»Daten sind ein Asset.« »Daten müssen wie ein Asset verwaltet werden.« Dies sind die Phrasen, die unseren Sprachgebrauch im Big-Data-Management beherrschen.

Die Metapher des *Assets* (engl. für »Vermögenswert« oder »Sachgut«), die für *Daten* verwendet wird, ist nicht neu. Immerhin hat TOGAF (<https://oreil.ly/oekyG>), ein Standard der Open Group für Enterprise-Architecture-Methoden und -Frameworks, seit Jahrzehnten explizit »Data Is an Asset« (<https://oreil.ly/Z6Y0y>) als ersten Grundsatz seiner Datenprinzipien festgeschrieben. Obwohl dies oberflächlich betrachtet eine eher harmlose Metapher ist, hat sie unsere Wahrnehmung und unser Handeln eher negativ geprägt, z.B. unser Handeln in Bezug auf die Art und Weise, wie wir Erfolg messen. Nach meinen Beobachtungen haben Daten als Asset dazu geführt, dass der Erfolg anhand von *Vanity-Metriken* gemessen wird – Metriken, die uns gut aussehen lassen oder sich gut anfühlen, sich aber nicht auf die Leistung auswirken. Beispiele hierfür sind die Anzahl der Datensätze und Tabellen, die im Data Lake oder Data Warehouse erfasst sind, und das Datenvolumen. Auf diese Kennzahlen stoße ich in Unternehmen immer wieder. *Data as an Asset* fördert die Aufbewahrung und Speicherung von Daten, anstatt sie bereitzustellen. Interessanterweise folgt auf das TOGAF-Prinzip »Data Is an Asset« unmittelbar das Prinzip »Data Is Shared«.

Ich schlage vor, die Metapher auf *Daten als Produkt* zu ändern und einen damit verbundenen Perspektivenwechsel vorzunehmen, z.B. die Messung des Erfolgs durch die *Adaptionsrate*, die *Anzahl der Nutzenden* und deren *Zufriedenheit* mit den Daten. Dies unterstreicht *das Teilen von Daten* im Gegensatz zur Aufbewahrung und Abschottung der Daten. Es legt auch den Schwerpunkt auf eine kontinuierliche Pflege, die ein Qualitätsprodukt verdient.

Ich lade Sie ein, andere Metaphern und Vokabeln zu identifizieren, die wir ändern müssen, um ein neues System von Begriffen für Data Mesh zu schaffen.

Vertraue, aber prüfe nach

Das Prinzip *Data as a Product* beinhaltet eine Reihe von Praktiken, die zu einer Kultur führen, in der man standardmäßig auf die Gültigkeit der Daten vertrauen

und sich darauf konzentrieren kann, die Eignung der Daten für den jeweiligen Anwendungsfall zu prüfen.

Zu diesen Praktiken gehören die Einführung einer Rolle für die langfristige Data Ownership eines Datenprodukts, die für Integrität, Qualität, Verfügbarkeit und andere Eigenschaften des Datenprodukts verantwortlich ist, die Einführung des Konzepts eines Datenprodukts, das nicht nur Daten bereitstellt, sondern explizit auch objektive Kennzahlen wie Aktualität, Aufbewahrungsfrist und Genauigkeit enthält, und die Schaffung eines Datenproduktentwicklungsprozesses, der die Tests der Datenprodukte automatisiert.

Heutzutage ist *Data Lineage* in Ermangelung dieser Praktiken nach wie vor ein entscheidender Faktor für das Schaffen von Vertrauen. Man hat heute keine andere Wahl, als anzunehmen, dass Daten nicht vertrauenswürdig sind und eine detektivische Untersuchung der Datenherkunft erfordern, bevor man ihnen vertrauen kann. Dieses mangelnde Vertrauen ist das Ergebnis der großen Kluft zwischen Datenanbietern und Datennutzern, die darauf zurückzuführen ist, dass die Datenanbieter die Nutzer und ihre Bedürfnisse nicht kennen, dass sie nicht langfristig für die Daten verantwortlich sind und dass es keine automatisierten Kontrollen gibt.

Das Prinzip *Data as a Product* zielt darauf ab, eine neue Kultur aufzubauen, weg von einer *Schuldvermutung* hin zum russischen Sprichwort *Vertraue, aber prüfe nach*.

Daten und Logik als Einheit

Machen wir einen kleinen Test.

Wenn Sie das Wort *Datenprodukt* hören, was kommt Ihnen in den Sinn? Welche Form? Was enthält es? Wie fühlt es sich an? Ich kann Ihnen versichern, dass ein Großteil von Ihnen an Dateien oder Tabellen denkt, an Spalten und Zeilen, an irgendeine Form von Speichermedium. Es fühlt sich statisch an. Es ist akkumuliert. Sein Inhalt besteht aus Bits und Bytes, die die Fakten repräsentieren, vielleicht sogar schön modelliert. Das ist intuitiv. Schließlich ist ein *Datum* per definitionem ein »Stück Information«.⁴

Diese Sichtweise führt zu einer Trennung von *Code* und *Daten*, in diesem Fall zu einer Trennung des Codes, der die Daten verwaltet, erstellt und bereitstellt. Durch diese Trennung entstehen verwaiste Datensätze, die mit der Zeit verfallen. Im großen Maßstab erleben wir diese Trennung als Data Swamps (<https://oreil.ly/6ixTI>) – einen verwaahlsten Data Lake.

Data Mesh ändert diesen dualen Modus von Daten und Code zu Daten und Code als eine architektonische Einheit, eine deploybare Einheit, die strukturell vollständig ist, um ihre Aufgabe zu erfüllen, nämlich die Bereitstellung der qualitativ hochwertigen Daten einer Domäne. Das eine kann ohne das andere nicht existieren.

Die Koexistenz von Daten und Code als eine Einheit ist kein neues Konzept für diejenigen, die sich mit Microservices-Architektur (<https://oreil.ly/cKogS>) beschäftigt haben. Die Entwicklung operativer Systeme ist zu einem Modell übergegangen, bei dem jeder Dienst seinen Code und seine Daten, seine Schemadefinition und seine Upgrades selbst verwaltet. Der Unterschied zwischen diesen operativen Systemen und Data Mesh ist aber die Beziehung zwischen dem *Code* und seinen *Daten*. Im Fall der Microservices-Architektur dienen die *Daten* dem *Code*. Sie halten den Zustand aufrecht, damit der Code seine Aufgabe erfüllen und die Funktionen bereitstellen kann. Im Fall eines Datenprodukts und Data Mesh ist diese Beziehung umgekehrt: Der *Code* dient den *Daten*. Der Code transformiert die Daten, hält ihre Integrität aufrecht, regelt ihre Policies und stellt sie schließlich bereit.

Beachten Sie, dass die zugrunde liegenden Infrastrukturkomponenten, die den Code und die Daten hosten, getrennt bleiben.

Zusammenfassung

Das Prinzip *Data as a Product* ist eine Antwort auf die Herausforderung der Datensilos, die sich aus der Übertragung der Data Ownership auf Domänen ergeben könnte. Es ist auch ein Wandel in der Datenkultur hin zu *Datenverantwortung* und *Datenvertrauen* am Entstehungsort. Das ultimative Ziel ist es, Daten einfach *benutzbar* zu machen.

In diesem Kapitel wurden acht nicht verhandelbare grundlegende *Usability-Attribute* von Datenprodukten erläutert, darunter *Discoverability*, *Adressierbarkeit*, *Verständlichkeit*, *Vertrauenswürdigkeit*, *nativer Zugriff*, *Interoperabilität*, *eigenständig nützlich* und *Sicherheit*.

Ich habe die Rolle des *Data Product Owner* eingeführt – jemand, der die Daten der Domain und ihre Nutzer genau kennt –, um die kontinuierliche Data Ownership und die Verantwortlichkeit für Erfolgskennzahlen wie *Datenqualität*, *Lead-Time der Datennutzung* und allgemein *Zufriedenheit* durch Net Promoter Score (<https://oreil.ly/iFctM>) zu gewährleisten.

In jeder Domäne gibt es einen *Data Product Developer*, der für die Erstellung, Pflege und Bereitstellung der Datenprodukte der Domäne verantwortlich ist. Data Product Developer arbeiten mit den anderen Anwendungsentwicklerinnen und -entwicklern in der Domäne zusammen.

Jedes Domänenteam kann ein oder mehrere Datenprodukte bereitstellen. Es ist auch möglich, neue Teams zu bilden, um Datenprodukte bereitzustellen, die nicht in eine bestehende operative Domäne passen.

Das Prinzip *Daten als Produkt* führt zu einem neuen Weltbild, in dem Daten vertrauenswürdig sind und mit großem Verständnis für ihre Nutzerinnen und Nutzer erstellt und bereitgestellt werden. Der Erfolg wird an dem Wert gemessen, den sie liefern, und nicht an ihrer Größe.

Dieser ehrgeizige Wandel muss als organisatorische Transformation betrachtet werden. Ich werde die organisatorische Transformation in Teil V dieses Buchs behandeln. Außerdem ist eine zugrunde liegende unterstützende Plattform erforderlich. Das nächste Kapitel befasst sich mit der Plattform, die Daten als Produkt möglich macht.

A

Abfragesprachen, Daten 269

Abhängigkeiten, Data-Mesh-Ansatz 191, 193, 199, 325

Abstrahieren von Komplexität

 durch Automatisierung 104

 Verbesserung des Kosten-Nutzen-Verhältnisses 160

Adressierungssystem 77

Affordances

 Daten beobachten, debuggen und auditieren 196, 201, 305–309

 Daten bereitstellen 243–260

 Daten finden, verstehen, vertrauen 196, 201, 275–285

 Daten kombinieren 286–295

 Daten konsumieren 260–267

 Daten transformieren 268–272

 Entwurf von Datenprodukten anhand von 233, 235–242

 Lebenszyklus verwalten 237, 297–300

 Rollen von 237–238

aggregierte Domänenendaten 63, 261

Agilität im Datenmanagement, Notwendigkeit von 145, 155–159

analytische Daten, Beziehung zu operativen Daten 53–54, 142–144

Anreize, zur Umgestaltung der Governance 127–128

Anreize bei der Umstellung auf Data Mesh 352–355

Anwendungsfälle

- die mit der bestehenden Plattform umsetzbar sind 325
- fachlich getriebene Umsetzung von Data Mesh 325
- APIs *siehe* Schnittstellen
 - a posteriori investigative Aufklärung 276
- A-posteriori-Kuratierung und Konsolidierung 276
- Archetypen von Datenquellen 261–264
- architektonische Veränderungen 46
- Architekturquantum 233, 235
 - siehe auch* Datenprodukte
 - Auditierung, Datenproduktarchitektur 306
- ausgleichende Feedback-Schleife 116
- automatisierte Notebooks 285
- Automatisierungen, Plattform
 - automatisierte Tests und Monitoring 131
 - Data Governance 119, 122, 128–131, 157
- Autonomie
 - Automatisierung der Governance zur Steuerung der Domänen 158
 - in Domain Ownership 353
 - für domänenorientierte Teams 110, 189
 - dynamisches Gleichgewicht mit globaler Interoperabilität 115–119
 - Erhaltung der Agilität trotz des Wachstums 158
- B**
- Berechtigungsverwaltung und Identifizierung 83, 302, 303
- Bereitstellung von Daten 58
 - Datenprodukte 199–201
 - domänenorientiert 185, 188–192, 262
 - als Motivation für den Data-Mesh-Ansatz 353
 - Skalierung 100
- bewegungsbasierter Wandel 347–349
- BI (Business Intelligence) 166
- Bitemporale Daten 251–256
- Bounded Context 61, 67
- Brooks, Fred, »No Silver Bullet – Essence and Accident in Software Engineering« 154

Building Evolutionary Architectures (O'Reilly) 192

Business Intelligence *siehe* BI

C

CDAOs (Chief Data and Analytics Officers) 369

Change Data Capture 262

Code

 Komponente eines Datenprodukts 194–198

 Policies als 130, 158, 198, 206, 300

 Standards als 129

 Transformation 268–272

Cognitive Load, Reduktion 99

Computational Governance 128–131

Consumer-Aligned Domänendaten *siehe* konsumentenorientierte Domänendaten

Container, Datenprodukt 208

Context Mapping 61, 67

Control-Port 208, 259, 301

D

Daff, Inc., fiktives Beispiel 27–41

DAG (Directed Acyclic Graph) 174, 194, 267

Dark Data 177

Data as a Product 53, 71–89

 Anwendung von Product Thinking auf Daten 323

 Datenquantum 185, 192

 Evolutionsphasen 332–333

 Feedback-Schleifen in 153

siehe auch Datenprodukte

Data Expectations 280

Data Governance 300–305

 Enabling-Teams 361

 Moderation 123

 neue Rollen 369

 Policy-Standardisierung 302–304

siehe auch Federated Computational Governance

- Verringerung der Koordination zur Erhaltung der Agilität angesichts des Wachstums 157–158
- Data-Governance-Mitglieder 216
- Data-Lake-Architektur 143
 - Evolution der analytischen Datenarchitektur 167–168
 - Konvergenz mit Data Warehouse 169
- Data Lineage 129, 306, 307, 308
- Data Marts 166
- Data Mesh 45–56
 - analytische Daten 54
 - Einführungsbeispiel 27–41
 - Ergebnisse 150–161
 - Haupttreiber 140–147
 - logische Architektur 185–209
 - operative Daten 53
 - Organisation *siehe* organisatorische Perspektive
 - Prinzip Data as a Product 49, 53, 71–89
 - Prinzip Domain Ownership 48, 57
 - Prinzip Federated Computational Governance 50, 111–136
 - Strategie für die Einführung von 317–321
- Data-Mesh-Architektur
 - Datenplattform 187
 - Data-Mesh-Strategie 320
 - logische 185–209
 - Datenplattform 201–205
 - Datenprodukt als Architekturquantum 192–201
 - eingebettete automatisierte Policies 205–209, 300
 - Schnittstellen zur Bereitstellung von Domänendaten 152, 185, 188–192, 262
- Data Mesh Execution Framework 321–344
 - bewegungsbasierter Wandel 347–349
 - evolutionäre Umsetzung 328–344
 - fachlich getriebene Umsetzung 322–327
- Data Ownership *siehe* Ownership
- Data Platform Developer 217
- Data Platform Product Owner 217, 368

- Data Product Developer 85
 - Abstraktion der Komplexität für 104
 - Rollen in der Data-Mesh-Organisation 368
 - Unterstützung durch die Plattform 101
 - User Journey der 217–227
- Data Product Owner 216
 - Rolle im Domänenteam 122, 367
 - Vorteile von dauerhaften 324
- Data-Warehouse-Architektur 143
 - Entwicklung von analytischen Datenarchitekturen 166
- Daten
 - Ansatz für die Schnittstellengestaltung 190
 - Beziehung zwischen operativen und analytischen Daten 53–54, 142–144, 152–154
 - Data Mesh – Annahmen zur Omnipräsenz und Diversität 149
 - Definition als analytische 45
- Datenabhängigkeiten, Data-Mesh-Ansatz zu 191, 193, 199, 325
- Daten bereitstellen, Entwurf von Datenprodukten 243–260
 - Entwurfseigenschaften 246–259
 - Komponenten für 260
- Datendemokratisierung 370
- Dateneigentum 58
- datenflussbasierte Transformation 270
- Datengarantien 154, 280–283
- datengetriebenes Arbeiten 146
- Dateninfrastruktur-Ebene 203, 213
- Daten kombinieren 286–295
- Daten konsumieren 260–267
- Datenmodelle 279–280
- Daten-Pipelines *siehe* Pipelines, Daten
- Datenplattform 187
 - Ebenen der 213
 - Self-Serve Data Platform 53
- Datenplattformarchitekturen 213–232
 - Data-Mesh-Strategie 320

- monolithische 171–175
- Multicloud-Datenplattform 265
- Plattformabhängigkeiten 199
- Self-Serve Data Platform 187, 333–334, 338
- Teams als Plattformteams 360
- User Journey der Data Product Consumer 227–232

Datenproduktarchitektur

- Auditierung 306
- Bereitstellung 243–260
- Debugging 306
- Discoverability 275–279, 285
- Fehlersuche 307, 308
- Governance 300–305
- Kombination 286–295
- Konsum 260–267
- Lebenszyklus verwalten 297–300
- Observability 201, 305–309
- Transformation 268–272
- Untersuchbarkeit 275–278, 284
- Verständlichkeit 275–278, 279–280, 285
- Vertrauenswürdigkeit 275–278, 280–283

Datenprodukte

- als Architekturquanten 192–201
- andere Datenprodukte als Datenquellen 263
- Architektur von *siehe* Datenproduktarchitektur
- Container 208
- Datengarantien 280–283
- Grenzen in der Organisation für 363–367
- Komponenten 193–199
- Komponenten zum Bereitstellen von Daten 260
- Kontrolle über den Datenkonsum 267
- lokaler Input als Datenquelle 264
- Manifest 299
- und Microservices 228

- organisatorische Data-Mesh-Strategie 319
- Rolle der Feedback-Schleifen 116–117
- Rolle des Owners 122
- Semantik bei multimodalen Daten 246–247
- siehe auch* Affordances
- Transformation 268–272
- Verantwortung für die Discoverability 276, 277–279, 285
- Zunahme der Nutzung als Maßstab für den Erfolg 354
- Datenprodukt-Ebene 204
 - Datengarantien 281
 - globale URI-Ermittlung 278
 - in der User Journey der Data Product Developer 220–227
 - Rolle des Datenprodukt-Containers 208
 - Rolle in der Datenplattform 213
- Datenprodukt-Manifest 298–300
- Datenprodukt-Sidecar 207
 - Ausführung von Policies 301
 - Discoverability 285
 - eingebettete automatisierte Policies 129, 187–188
 - Embedded Computational Policies 207
 - integrierte automatisierte Policies 242
- Datenqualität
 - in einer dynamischen, skalierenden Datenumgebung 145
 - Grenzen der Zentralisierung 177
 - Metriken für Datengarantien 281
 - Verantwortung im dezentralen Data Mesh 123, 132
 - Verarbeitungszeit für Widerrufe 257
- Datenquantum 155, 235
 - siehe auch* Datenprodukte
- Datenquellen, Archetypen von 261–264
- Datenschutz und Einwilligung 124, 259, 304
- Datensouveränität 58
- Datenstrategie *siehe* Strategie
- Daten transformieren 268–272

DDD (Domain-driven Design) 59–62, 176–177

Debugging 306

deklarative Modellierung des Zielzustands 104, 298

Demokratisierung durch Daten 370

der menschliche Faktor bei der Datenkombination 295

The Design of Everyday Things (Norman) 236

Dezentralisierung

- Agilität trotz Wachstum 155
- als Ziel und Resultat einer Data-Mesh-Transformation 46
- Entwurfseigenschaften für das Kombinieren von Daten 288–289
- Fehlen eines zentralen Orchestrators 242
- siehe auch* Self-Serve Data Platform

Die Struktur wissenschaftlicher Revolutionen (Kuhn) 55

differentieller Datenschutz 284

Discoverability und Suche

- kombinierbare Daten im Typsystem 288
- Schnittstellen (APIs) für 196, 277, 279
- Standards als Code 129
- Untersuchbarkeit 284
- Verantwortlichkeiten von Datenprodukten 275–279, 285

Diskrepanz zwischen Investition und Rendite 146

Dokumentation, um Datenprodukte zu verstehen 284

Domain-driven Design *siehe* DDD

Domain Ownership 48, 52, 57

- Archetypen 63–66
- Datenprodukt *siehe* Data Product Owner
- Domain-driven Design 59–62, 176–177
- Evolutionsphasen 330
- Fitnessfunktionen 338
- Schnittstellen zur Bereitstellung von Domänendaten 185, 188–192
- Selbstbestimmung 353
- Übergang zu 67–69
- Zusammenführung von Fachlichkeit und Daten 151

Domänen

- aggregierte Domänendaten 65, 261
- Ausrichtung der Geschäftstätigkeit auf analytische Daten 151
- Automatisierung der Governance zur Kontrolle 157
- Delegation der Verantwortung an 132
- dynamisches Gleichgewicht zwischen Autonomie und Interoperabilität 115–119
- konsumentenorientierte Domänendaten 261
- Lokalisierung von Datenänderungen in Domänen 154
- quellenorientierte Domänendaten 261, 262–263
- Zusammenarbeit zwischen operativen und analytischen Daten 153
- domänenagnostische Datenplattform 98, 265
- Domänen-Events, Bereitstellung 262
- domänenorientierte Datensemantik 155
- domänenorientierte Observability-Daten 309
- Domänenteams
 - Autonomie für 101–102, 110, 146, 189
 - Rolle des Data Product Owner 122, 367
 - als Stream-Aligned-Teams 358
- domänenübergreifende analytische Datenabhängigkeiten 191
- Domänenvertreter, föderales Team 122
- dynamisches Gleichgewicht zwischen Domänenautonomie und globaler Interoperabilität 115–119
- dynamische Topologie als Standardzustand 119

E

- Early Adopters von Data Mesh 316
- Ebenen der Datenplattform, logische Architektur 201–205
- EDGE 328
- einfache, lokale Regeln für Datenprodukte 241
- eingebettete automatisierte Policies 205–209, 300
- Enabling-Teams 361
- Entstehung von Mesh-Eigenschaften durch einfache, lokale Regeln für Datenprodukte 241
- Erwartungen an Daten 140–142, 280
- ETL-Prozess (Extract, Transform, Load) 143
- eventgetriebene Bereitstellung von Daten 262

- eventgetriebene Microservices 156
- Events, Speichern und Bereitstellen 256
- Eventual Consistency des Mesh, Erhaltung der 250, 258
- evolutionäre Umsetzung 328–344
 - Data-as-a-Product-Evolutionsphasen 332
 - Federated-Computational-Governance-Evolutionsphasen 335–336
 - Fitnessfunktionen 336–340
 - Legacy-Migration 340–344
 - mehrphasiges Entwicklungsmodell 328–336
 - Self-Serve Data Platform 333–334
- Expansions- und Wachstumsphase, evolutionäre Umsetzung 331, 332, 334, 336
- Explorations- und Einführungsphase, evolutionäre Umsetzung 331, 332, 334, 335
- Extraktions- und Reifephase, evolutionäre Umsetzung 331, 333, 334, 336
- extrinsische Motivationen, für Belohnungen 354

F

- Fachexperten, in einem föderalen Team 122
- fachlicher Zeitpunkt
 - Bitemporale Daten 251–254, 256, 257, 264
 - zeitbewusste Transformation 272
- fachlich getriebene Umsetzung 322–327
- FAIR data 75
- Federated Computational Governance 50, 53, 111–136
 - als Enabling-Teams 361, 369
 - Anwendung der Föderation auf das Governance-Modell 120–128
 - Automatisierung 128–131
 - Einführung einer 131–134
 - Evolutionsphasen 335–336
 - Fitnessfunktionen 340
 - Policies *siehe* Policies Rolle der logischen Architektur 187–188
 - Systems Thinking, angewandt auf 114–120
- Feedback-Schleifen 116–117, 153
- Fehlerbehandlung, Data Mesh 256, 258
- Fehlersuche 307, 308

Fitnessfunktionen, geführte Evolution mit 336–340

föderales Team 121–123

Form der Daten, Untersuchung der 284

Forrester, Jay W. 118

fortlaufende Datenmigration 266

Fowler, Martin 47, 253

Führungskräfte und Entscheider 315

funktionsübergreifende Teams 52

G

Garbage Collection, Feedback-Schleifen als 117

Geschäftsperspektive

- Datentechnologie im Mittelpunkt 315

- Initiativen und Anwendungsfälle der Datenstrategie 319, 325

- Komplexität als Treiber für die Einführung von Data Mesh 39, 145

globale Anreize 128

globale Control-Port-Funktion 259

globale Interoperabilität 115–119

globale Policies 53, 126, 130

globale Standards 112

globaler URI, für Datenprodukte 293

Governance, Data *siehe* Data Governance

große Kluft 142–144

H

Hebel 118, 127

Herkunft der Daten, und Skalierung 144

Hyperlinks, Beziehungen definiert durch 291

I

Infrastruktursicht

- Abhängigkeiten 193

- domänenagnostische Datenplattform 265

- und Data-Mesh-Ziele 46

Innovationskultur 100, 106

Input-APIs 196

Input-Ports 200, 260, 262, 263

Interoperabilität 81

 dynamisches Gleichgewicht mit 115–119

 Einfluss auf Data Governance 112

intrinsische Motivationen, für Anreize 353

iterative Ende-zu-Ende-Umsetzung 327

K

komplexe adaptive Systeme, Einfluss auf den Entwurf 240–242

Komplexität

 Data-Mesh-Einführung 39, 145, 150–155

 Grenzen der monolithischen Architektur 172

 inhärent bei der Erstellung von Software 154

 Reduktion von unbeabsichtigter 154, 250

siehe auch Abstrahieren von Komplexität

Konfiguration des Datenflusses, lokal generiert 241

konsumentenorientierte Domänendaten 63, 66, 261

KPIs (Key Performance Indicators) 337

Kuhn, Thomas, Die Struktur wissenschaftlicher Revolutionen 55

Kultur, in der Organisation 29–30

 Anreiz 352–355

 kultureller Wandel 349–352

 Menschen 367–372

 Prozess 372–374

 Struktur 355–367

 Veränderung von 347–349

L

Lakehouse-Architektur, Business Intelligence 143

Lakeshore Marts 167

Lebenszyklusmanagement für Daten 237

Legacy-Migration 340–344

Lineage 80

logische Architektur 185–209

- Datenprodukt als Architekturquantum 192–201
- domänenorientierte Schnittstellen für die Bereitstellung von Daten 152
- Ebenen 201–205
- eingebettete automatisierte Policies 205–209, 300
- Rolle der Governance 187–188
- Schnittstellen zur Bereitstellung von Domänendaten 185, 188–192, 262
- traditionelle Datenarchitektur 171
- Logs, Observability-APIs 307, 308
- lokale Anreize, Einführung von 127
- lokale Entscheidungsfindung 154
- lokaler Input als Datenquelle 264
- Lokalität der Datenkonsumierung 265–266
- lose Kopplung, Kombinierbarkeit im Data Mesh 293

M

- Machine Learning *siehe* ML
- Manifest, Datenprodukt 298–300
- MapReduce 156
- Marktplatz für Daten 102
- Meadows, Donella
 - »Leverage Points: Places to Intervene in a System« 118
 - Thinking in Systems 115
- mehrphasiges Entwicklungsmodell 328–336
 - evolutionäre Umsetzung 328–336
 - Fitnessfunktionen 336–340
- Mehrwert aus Daten
 - mittels Autonomie und Interoperabilität 102
 - Befähigung von Teams 101–102
 - durch übergreifende Verknüpfung 351
 - durch Reduktion des Cognitive Load 99, 103–105
 - Verhältnis von Wert aus Daten zu Investitionen erhöhen 159–161
- Mesh-Ebene 204–205
 - Entwurf der Data Governance 301
 - globale URI-Ermittlung 278

- Monitoring des Mesh 226
- Rolle der Discoverability 277
- Rolle des Datenprodukt-Containers 206
- Rolle in der Datenplattform 214
- Metadaten 198 309
- Metriken, Observability-APIs 308
- Microservices 151
- ML (Machine Learning)
 - die Rolle der analytischen Daten bei 142
 - als Transformation 271
- monolithische Architektur 171–175
 - Grenzen in einem komplexen Datenumfeld 172
 - organisatorische Perspektive 174–175
 - technische Partitionierung 177
 - Ziele von 171
- Multicloud-Datenplattform 265
- multimodale Cloud-Architektur 169
- multimodale Daten 246–247
- Multisided-Plattform, Data Mesh als eine 102

N

- native Domänenendaten *siehe* quellenorientierte Domänenendaten
- negative Feedback-Schleife 116
- Netzwerkeffekt, Messung des 38, 133
- nichtprogrammatische Transformation 269–270
- Norman, Don, The Design of Everyday Things 236
- »No Silver Bullet – Essence and Accident in Software Engineering« (Brooks) 154
- Nutzer 238
 - Bedarf an bereitgestellten Daten 243
 - Nutzer analytischer Daten 243

O

- Objectives and Key Results *siehe* OKRs
- Observability 201, 305–309
 - domänenorientierte Observability-Daten 309

- und Interoperabilität 105
- Observability-Daten 307–309
- Schnittstellen 129, 196, 307–308
- strukturierte und standardisierte Daten 309
- OKRs (Objectives and Key Results) 354
- OLAP-Systeme (Online Analytical Processing) 54
- OpenLineage 96
- OpenTelemetry 96
- operative Perspektive
 - Beziehung zwischen operativen und analytischen Daten 53–54, 142–144
 - und Data-Mesh-Ziele 46
 - Daten aus dem operativen System konsumieren 261–263
 - Schnittstellengestaltung 190
 - Überwachung des Zustands des Mesh 305
- operative Systeme als Datenquellen 262–263
- Optimierung für Menschen oder Maschinen 294
- organisatorische Perspektive 139–181, 313–375
 - Alignment als Teil der Data-Mesh-Strategie 320
 - domänenorientierte Organisation 316
 - Execution Framework 321–344
 - Kultur und Organisation *siehe* Kultur
 - Strategie für die Einführung von Data Mesh 314–321
 - traditionelle Datenarchitektur *siehe* traditionelle Datenarchitektur
 - Wendepunkte *siehe* Wendepunkte
- Output-APIs 196
- Output-Ports 199, 201
- Ownership-Modell
 - Data Product Owner 324, 367
 - langfristige 364
- Oyserman, Daphna 349

P

- Pais, Manuel 356
- Peer-to-Peer-Ansatz 156

Peer-to-Peer-Datenbereitstellung 352

Perfektion der Verwaltung, Bereitstellung und Nutzung analytischer Daten 353

physische Implementierung von Schnittstellen 203, 209

Pipelines, Daten 67

- datenflussbasierte Transformation 270
- Directed Acyclic Graph (DAG) 194, 267
- Reduzierung der Koordinierung von 154

Platform Architect 122

Platform Product Owner 368

Platform Thinking 98–107

Plattformabhängigkeiten 199

Policies 187–188

- automatisiertes Monitoring 131
- automatisierte Tests 131
- Automatisierung 122
- als Code 130, 158, 198, 206, 300
- Einbettung der Ausführung von Policies in jedes Datenprodukt 132, 205–209
- Föderalismus in der Governance 126–127
- globale 53, 126, 130
- Integration mit Daten 304
- Konfiguration mit Control-Port 208, 259, 301
- lokale 126
- Standards *siehe* Standards und Standardisierung
- Verknüpfung 304

Polyseme 68, 129, 133

Ports *siehe* Schnittstellen

positive Feedback-Schleife 117

positiver Netzwerkeffekt 38, 331

Post-Mortem-Analyse 306

privilegierte Operationen, Control-Port 209

produktorientiertes Arbeiten 324

programmatische Transformation 269–270

Projektbudgets 324

Q

- quellenorientierte Domänendaten 63, 64–65
 - Daten konsumieren 261
 - operative Systeme als Datenquellen 262–263

R

- Read-only-Zugriff 258
- reaktive Verarbeitung ständiger Veränderungen, bitemporale Daten 254
- Realtime-Entscheidungen 145
- relationale Schemata 289
- Risikomanagement, Data Mesh 133

S

- Schnittstellen (APIs) 129
 - als Code 196–197
 - Data-Mesh-Ansatz 190
 - domänenorientierte Bereitstellung von Daten 185, 188–192, 262
 - physische Implementierung der 203, 209
 - Standardisierung von 208
- Schnittstellenverträge, Daten und Services 151
- Self-Serve Data Platform 50, 53, 91–110, 187
 - Data Mesh Platform Thinking 98–107
 - Datenplattform *siehe* Datenplattformarchitekturen
 - Ebenen der 204–205
 - Evolutionsphasen 333–334
 - Fitnessfunktionen 338
 - Innovationskultur 100
 - Skalierung der Datenbereitstellung 100, 105–106
- semantisches Modell, Datenprodukt 279
- semantische Verknüpfung 293
- Service Level Objectives *siehe* SLOs
- Sicherheit *siehe* Berechtigungsverwaltung und Identifizierung
- Sidecar *siehe* Datenprodukt-Sidecar
- Skalierbarkeit, Entwerfen für 240

Skalierung der Datenbereitstellung 100

Skalierung von Unternehmen

- Bedeutung der Autonomie des Domänenteams für 189
- Grenzen monolithischer Architekturen 175
- in Bezug auf Volume, Velocity und Variety 37
- als strategischer Wendepunkt beim Übergang zum Data Mesh 144

Skelton, Matthew 356

SLOs (Service Level Objectives) 79–80, 258

Software Architecture: The Hard Parts (O'Reilly) 192

Softwareentwicklungsmethoden 316, 324

Source-Aligned Domänendaten *siehe* quellenorientierte Domänendaten

Souveränität 58

soziotechnisches Paradigma, Data Mesh als ein 48

Speziallösungen, als Herausforderung für die Datenplattform 323

Standards und Standardisierung

- globale Standards 123
- integrierte Standardisierung mittels Sidecar 242
- Kommunikationsprotokolle 208
- Observability 309
- Policies 207, 302–304
- Schnittstellen (APIs) für 208
- Sidecar 208
- Standards als Code 129

Star Model 345–347

Strategie, Einführung von Data Mesh 317–321

- Execution Framework *siehe* Data Mesh Execution Framework
- Kriterien für die Einführung von Data Mesh 313–317
- Wendepunkte 139–147

Stream-Aligned-Teams 358

Synchronisierung, Reduzierung und Betonung der Lokalisierung von Datenänderungen 154, 293

syntaktisches Modell, Datenprodukt 279

Systems Thinking in der Data Governance 114–120

T

Teams

- aufgabenorientierte Teamaufteilung 179–180

- Autonomie für 110, 146, 189

- föderales Team 120, 121–123

- funktionsübergreifende 52

- hyperspezialisierte Data Engineers in monolithischen Silos 174–175

- Moderation 123

- Plattform 360–361

- Rolle des Data Product Owner 367

Team Topologies 356–363

technisch partitionierte Architektur 177

technologieorientierte Datenarchitektur 177–180

Technologieperspektive

- Ausrichtung auf die Abteilungen (als Domänen) 151

- auf Data Mesh 46

- Entscheidung über Data Mesh 315

- monolithische Architektur 173

Teil versus Quelle der Wahrheit für Daten 258

Tests zur Build-Zeit 101

Tests zur Laufzeit 101

Thinking in Systems (Meadows) 115

TOGAF (The Open Group for Enterprise Architecture) 86

Traces (Lineage), Observability-APIs 307, 308

traditionelle Datenarchitektur 165–180

- Evolution 165–170

- Kombinierbarkeit 289–293

- limitierende Eigenschaften von 170–180

- monolithische Architektur 171–175

Transformation als Code 194–196

Transformationscode 101

Typsystem, Bestimmung von verknüpfbaren Daten 290–293

U

unbeabsichtigte Komplexität, Reduzierung mit Data Mesh 154, 250

Untersuchbarkeit, Datenproduktarchitektur 275–278, 284, 285

unveränderliche Daten 248–250, 258

Ursachenanalyse, Logs für 307

Usability-Attribute 75–83

V

veränderliche Daten 248–250

Veränderung

 Berücksichtigung in einem komplexen Geschäftsumfeld 145

 Dimensionen der 46–48

 Entwerfen für 239, 250, 252

 bei der Umstellung der Organisation auf Data Mesh 347–349

Verarbeitungszeitpunkt

 bitemporale Daten 251–256, 264

 zeitbewusste Transformation 272

Versatz zwischen Verarbeitungszeitpunkt und fachlichem Zeitpunkt, Reduzierung mit
 bitemporalen Daten 254, 255

Verschlüsselung, Standardisierung bei der Governance 303

Verständlichkeit

 Datenproduktarchitektur 275–278, 279–280, 285

 Dokumentation für 284

verteilte Datenarchitektur

 Automatisierung 119

 Bedeutung von unveränderlichen Daten 248–250

 Data Mesh *siehe* Data-Mesh-Architektur

 Datenprodukt *siehe* Datenproduktarchitektur

 Domain Ownership 151

 Typsystem 290, 291, 293

Vertrag, Daten 281

Verträge, Daten und Services 49

 APIs im Sinne der definierten Verträge 237

 APIs nach definierten Schnittstellenverträgen 196

 für lokale Entscheidungen 154

 und Metadaten 198

- für native Zugriffsmodi 155
- operative Systeme als Datenquellen 262
- Verringerung der Pipeline-Koordination durch 157

Vertrauenswürdigkeit der Daten

- Datengarantien 154, 280–283
- Datenproduktarchitektur 275–278
- größeres Vertrauen als extrinsischer Anreiz 355

Vielfalt der Datenquellen, Herausforderung bei der Skalierbarkeit 144

W

Weiterbildung 370–372

Wendepunkte beim Übergang zum Data Mesh

- Datenarchitektur vor dem Wandel 165–180
 - Evolution der analytischen Datenarchitekturen 165–170
- Einführungsbeispiel 28–39
- Folgen des Wandels 149–161
 - Agilität trotz Wachstum 155–159
 - auf Veränderungen reagieren 150–155
 - Verhältnis von Wert aus Daten zu Investitionen erhöhen 159–161
 - Zusammenfassung der Ziele von Data Mesh 161

strategisch und organisatorisch 139–147

- die große Kluft 142–144
- Diskrepanz zwischen Investition und Rendite 146
- Erwartungen an Daten, Beispiele von 140–142
- Skalierung in Bezug auf Umfang, Geschwindigkeit und Vielfalt 144
- Umgang mit Komplexität, Ungewissheit, Volatilität 145

Werte, Data Mesh

- Automatisierung zur Erhöhung der Geschwindigkeit und Qualität der Datenbereitstellung 352
- Ergebnisse 46
- Datenprodukte sind veränderlich, dauerhaft und unabhängig 351
- für analytische Daten sind alle zuständig 350
- Gleichgewicht zwischen lokaler Datenbereitstellung und globaler Interoperabilität 352
- glückliche Nutzer 351
- Kultur bei der Einführung von Data Mesh 350–352

Mehrwert durch übergreifende Verknüpfung 351

Peer-to-Peer-Datenbereitstellung 352

Wertschätzung der Bedeutung von Daten 351

Wertschöpfung, Entwerfen für 240

Widerrufe, Reduzierung mit bitemporalen Daten 253, 256, 257

Windowing 254

Z

zeitbewusste Transformation 272

Zeit versioniert alle Aspekte eines Datenprodukts, bitemporale Daten 254

Zentralisierung

Reduzierung mit Data Mesh 149

traditionelle Neigung zu 175–177

Zielzustand, Datenprodukt 298

Zusammenarbeit zwischen Teams 356–358

Zustände, Speichern und Bereitstellen 256

zwei Datenwelten 143