

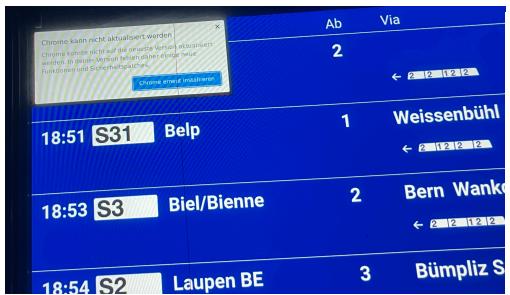
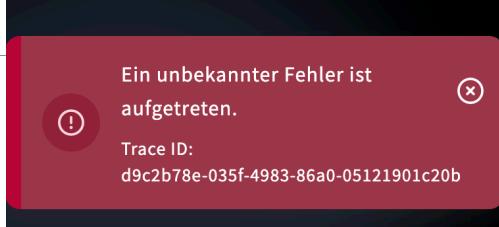
Einstieg

Internet of Bugs



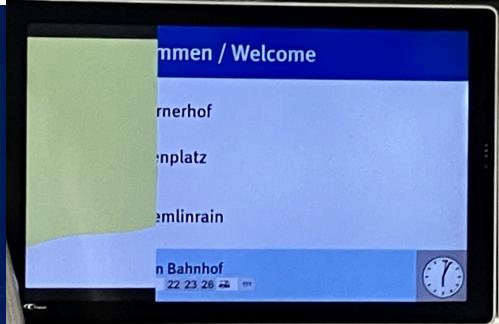
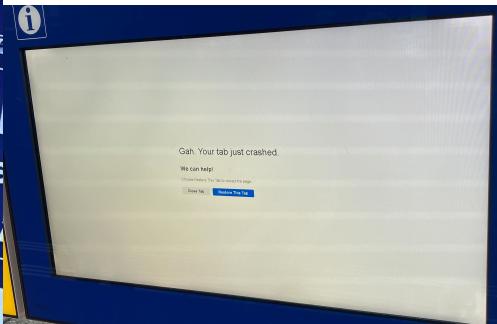
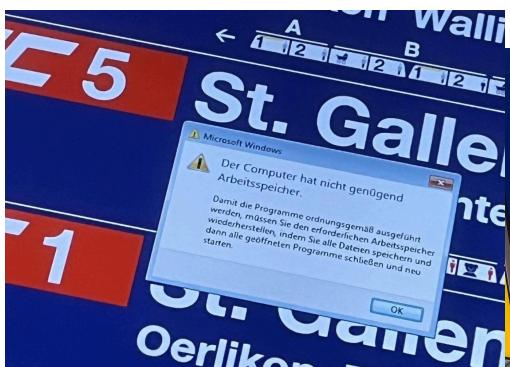
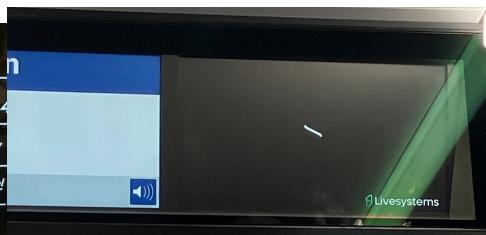
Service Unavailable - DNS failure

The server is temporarily unable to service your request. Please try again later.
Reference #11.84161502.1741077553.50f924e
<https://errors.edgesuite.net/11.84161502.1741077553.50f924e>



 Genève Aéroport
about an hour ago

 Suite à une panne informatique, des retards sont à attendre au départ et à l'arrivée.
Nous sommes désolés pour les désagréments et faisons tout notre possible pour rétablir la situation.
Nous demandons aux passagers de contacter leur compagnie aérienne. ... See more



Auf Grund technischen Problemen bleibt die Filiale momentan geschlossen



Logo DIE POST

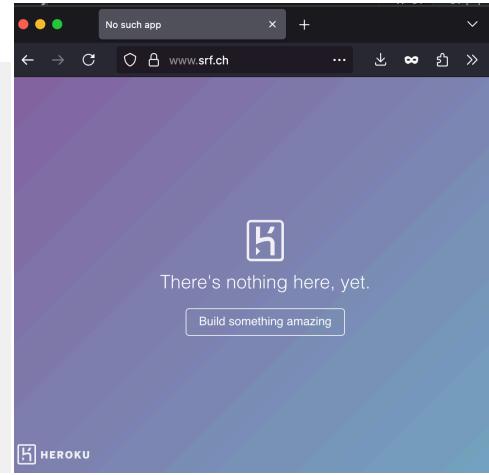
Falsche E-Mail mit Betreff «Lorem ipsum»

Lieber Herr Hirter

Sie haben gestern Abend von uns gegen 19 Uhr eine E-Mail mit dem Betreff «Lorem ipsum» erhalten.

Wir sprechen jetzt wieder Klartext – versprochen! Die betreffende E-Mail ist auf einen internen technischen Fehler zurückzuführen und Sie können sie direkt in Ihren digitalen Papierkorb verschieben. Bitte entschuldigen Sie daraus entstandene Unannehmlichkeiten.

Freundliche Grüsse
Ihre Post



Equalizertyp Studiotechnik

Gespart 78 kg CO₂

Zustand Gebraucht

AUF DEUTSCH ÜBERSETZEN

Used Phonics studio equalizer for sale. It offers a range of controls for precise sound shaping and is perfect for both recording and live applications.

Brooke Siren Systems (BSS) DPR-402 Stereo Compressor Limiter

9. Okt. 2025, 08:17 Uhr

Sofort-Kaufpreis
299.00

SOFORT KAUFEN

PREIS VORSCHLAGEN

ZU FAVORITEN HINZUFÜGEN

Lieferung
Paket B-Post, CHF 20.50
Abholung durch Käufer in 1202 Geneve, CHF 0.00

Verkäufer
 T36 99.6%

TEILEN **ÄHNLICHEN ARTIKEL VERKAUFEN**

- EJPD: Kostenexplosion bei IT-Projekt: Finanzdelegation schlägt Alarm
- VBS: Schweizer Armee ohne krisensichere Logistik bis 2035,
- Armee-Debakel:300-Millionen-Projekt seit Monaten suspendiert
- Kantonsverwaltung: Wegen fehlerhafter Software braucht es mehr Haftplätze
- Polizei: Berner Polizisten beklagen sich über die neue IT

Crowdstrike



-- <https://www.srf.ch/news/international/crowdstrike-softwarefehler-der-tag-an-dem-die-it-weltweit-verruckt-spielte-ein-ueberblick>

Google

Incident affecting API Gateway, Agent Assist, AlloyDB for PostgreSQL, Apigee, Apigee Edge Public Cloud, Apigee Hybrid, Cloud Data Fusion, Cloud Firestore, Cloud Logging, Cloud Memorystore, Cloud Monitoring, Cloud Run, Cloud Security Command Center, Cloud Shell, Cloud Spanner, Cloud Workstations, Contact Center AI Platform, Contact Center Insights, Data Catalog, Database Migration Service, Dataform, Dataplex, Dataproc Metastore, Datastream, Dialogflow CX, Dialogflow ES, Google App Engine, Google BigQuery, Google Cloud Bigtable, Google Cloud Composer, Google Cloud Console, Google Cloud DNS, Google Cloud Dataflow, Google Cloud Dataproc, Google Cloud Pub/Sub, Google Cloud SQL, Google Cloud Storage, Google Compute Engine, Identity Platform, Identity and Access Management, Looker Studio, Managed Service for Apache Kafka, Memorystore for Memcached, Memorystore for Redis, Memorystore for Redis Cluster, Persistent Disk, Personalized Service Health, Pub/Sub Lite, Speech-to-Text, Text-to-Speech, Vertex AI Online Prediction, Vertex AI Search, Vertex Gemini API, Vertex Imagen API, reCAPTCHA Enterprise

Multiple GCP products are experiencing Service issues.

Incident began at **2025-06-12 10:51** and ended at **2025-06-12 18:18** (all times are **US/Pacific**).

Previously affected location(s)

Johannesburg (africa-south1), Multi-region: asia, Taiwan (asia-east1), Hong Kong (asia-east2), Tokyo (asia-northeast1), Osaka (asia-northeast2), Seoul (asia-northeast3), Mumbai (asia-south1), Delhi (asia-south2), Singapore (asia-southeast1), Jakarta (asia-southeast2), Multi-region: asia1, Sydney (australia-southeast1), Melbourne (australia-southeast2), Multi-region: eu, Multi-region: eur3, Multi-region: eur4, Multi-region: eur5, Warsaw (europe-central2), Finland (europe-north1), Stockholm (europe-north2), Madrid (europe-southwest1), Belgium (europe-west1), Berlin (europe-west10), Turin (europe-west12), London (europe-west2), Frankfurt (europe-west3), Netherlands (europe-west4), Zurich (europe-west6), Milan (europe-west8), Paris (europe-west9), Global, Doha (me-central1), Dammam (me-central2), Tel Aviv (me-west1), Multi-region: nam-eur-asia1, Multi-region: nam10, Multi-region: nam11, Multi-region: nam12, Multi-region: nam13, Multi-region: nam3, Multi-region: nam5, Multi-region: nam6, Multi-region: nam7, Multi-region: nam8, Multi-region: nam9, Montréal (northamerica-northeast1), Toronto (northamerica-northeast2), Mexico (northamerica-south1), São Paulo (southamerica-east1), Santiago (southamerica-west1), Multi-region: us, Iowa (us-central1), South Carolina (us-east1), Northern Virginia (us-east4), Columbus (us-east5), Dallas (us-south1), Oregon (us-west1), Los Angeles (us-west2), Salt Lake City (us-west3), Las Vegas (us-west4)

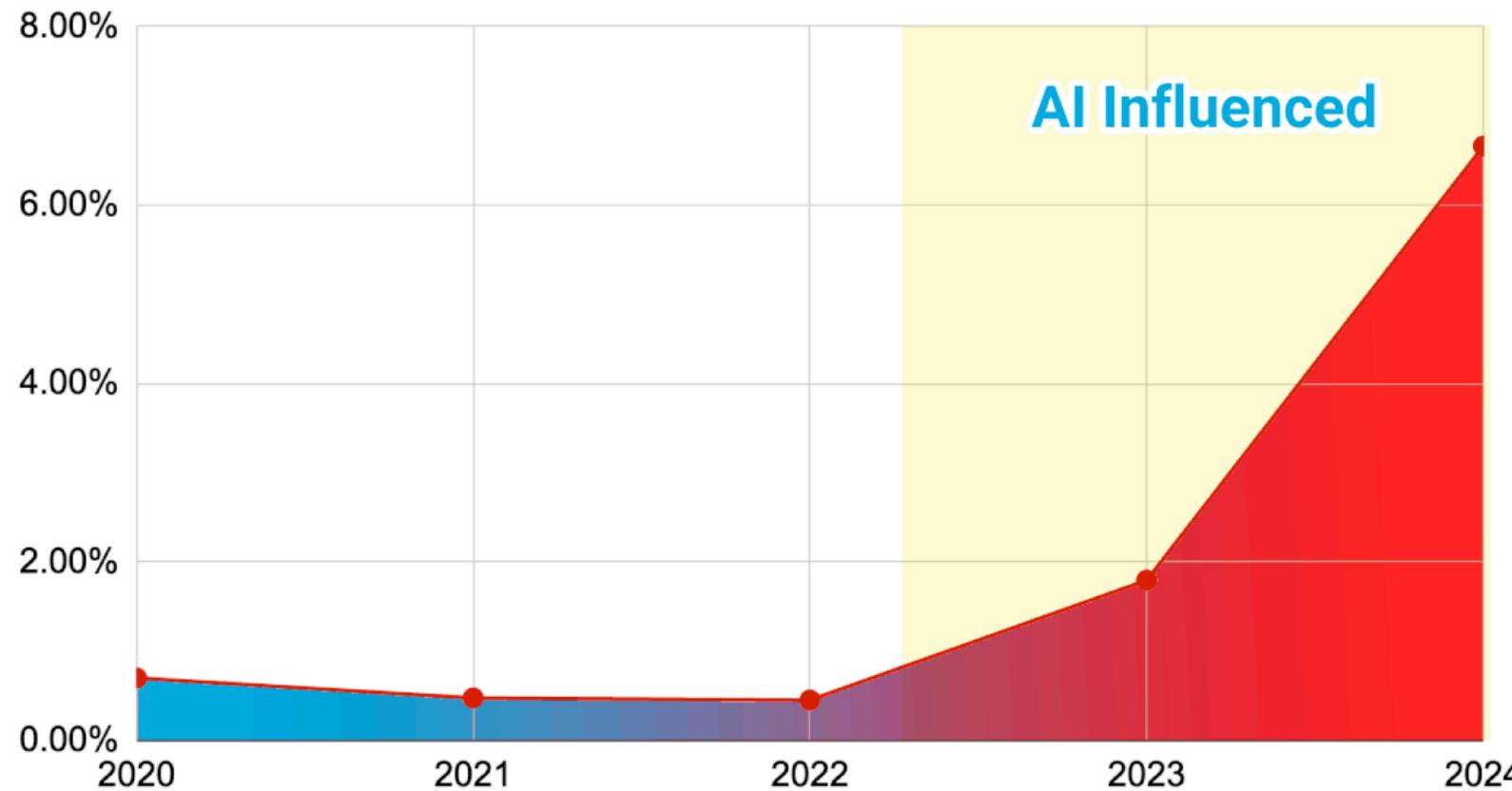
<https://status.cloud.google.com/incidents/ow5i3PPK96RduMcb1SsW>

AWS 20.10.25

Between **11:49 PM PDT** on October 19 and **2:24 AM PDT** on October 20, we experienced increased error rates and latencies for AWS Services in the **US-EAST-1** Region. Additionally, services or features that rely on US-EAST-1 endpoints such as **IAM and DynamoDB** Global Tables also experienced issues during this time. At **12:26 AM** on October 20, we identified the trigger of the event as **DNS resolution** issues for the regional DynamoDB service endpoints. [...] As we continued to work through EC2 instance launch impairments,[...] connectivity issues in multiple services such as **Lambda, DynamoDB, and CloudWatch**. [...] By **3:01 PM**, all AWS services returned to normal operations. [...]

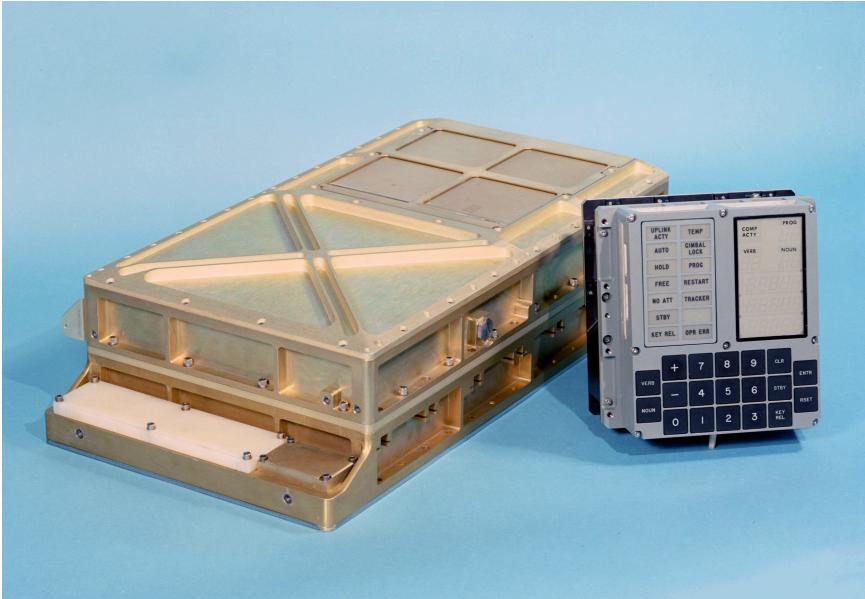
-- <https://health.aws.amazon.com/health/status>

Percent of Commits Containing a Cloned Code Block



-- https://www.gitclear.com/ai_assistant_code_quality_2025_research

Ab 1961: Margaret Hamilton, Apollo Guidance Computer



"I remember thinking, Oh my God, it worked," the pioneering software engineer tells TIME. "I was so happy. But I was more happy about it working than about the fact that we landed."

-- <https://time.com/3948364/moon-landing-apollo-11-margaret-hamilton/>

Anforderungen an (moderne) Software

- the problems of achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society
- the difficulties of meeting schedules and specifications on large software projects
- the education of software (or data systems) engineers

-- SOFTWARE ENGINEERING, Report on a conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 7th to 11th October 1968

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>

Agiles Manifest

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

2001, <https://agilemanifesto.org/>

Anforderungen an Software

Software soll den Kunden Mehrwert bringen

- Software soll zuverlässig sein
- Neue Features sollten schnell umgesetzt und nutzbar sein

Zuverlässigkeit

- Hohe Verfügbarkeit
- Skalierbarkeit
- Im Katastrophenfall sollen die Systeme schnell wiederhergestellt werden können
- Soll funktionieren, auch wenn Teile des Systems Offline sind (Resilienz)
- Kostengünstig
- Einfach
- Updates müssen einfach eingespielt werden können

Softwarekrise

Softwaresysteme werden immer komplexer

“[The major cause of the software crisis is] that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”

-- Edsger Dijkstra: The Humble Programmer

<https://www.cs.utexas.edu/~EWD/ewd03xx/EWD340.PDF>, 1972

Software Engineering vs Software Architecture vs Software Development

Software engineering is the application of an empirical, scientific approach to finding efficient, economic solutions to practical problems in software

(Farley, 2022, S.4)

The goal of software architecture is to minimize the human resources required to build and maintain the required system

(Martin, 2018)

- Übergang zwischen Software Entwicklung, Software Architektur und Softwareentwicklung ist fliessend.
- Grundsätzlich sollen alle Beteiligten in allen Bereichen bewandert sein.

Kommunikation

Mehrere Personen arbeiten am selben Softwareprojekt

- Fachkräftemangel
- Ausbildung ist sehr herausfordernd
- Wissenstransfer

Lernen

- Iteratives und inkrementelles Arbeiten
- Feedback
- Empirisches und experimentelles Arbeiten

(vgl. Farley, 2022, S.4)

Komplexität "managen"

- Modularity & Separation of Concerns
- Cohesion & Coupling
- Abstraction

(vgl. Farley, 2022, S.5)

Production Is Not Our Problem

- Softwareentwicklung ist meistens Kreativarbeit
- Die Herausforderung der "Produktion" existiert kaum

Space X Starship

[How Not to Land an Orbital Rocket Booser, 2017 WOW! Watch SpaceX Catch A Starship Booster In Air, 2024](#)

Finanzierung: ca 3 Mrd. Dollar

Apollo-Programm: 1958 bis 1969, inflationsbereinigt: **163 Mrd. Dollar** (ohne Mercury und Gemini)

Generative KI

- LLMs können inherent nur durchschnittliche Antworten generieren
- Je nach Fragestellung kann das hilfreich sein oder auch nicht
- Längere generierte Texte scheinen oft auf den ersten Blick sehr gut, bei genauerer Betrachtung aber inhaltsleer, inkorrekt und übermäßig umfangreich.
- Offensichtlich generierte Texte stoßen beim Empfänger oft auf starke Ablehnung.
- Datenschutztechnisch ist die Verwendung von LLMs sehr heikel.

in der Software-Entwicklung

- Die Kontextfenster sind oft zu klein für Software-Architektur.
- Bei Code werden oft Features implementiert, die nicht gefragt wurden oder übermäßig komplizierte Lösungen entworfen.
- Der Nutzen durch die schnelle Code-Generierung wird durch längeres Debugging und Refactoring oft zunichtegemacht.
- Diskutieren von Architekturideen kann sehr hilfreich sein.

in der Bildung

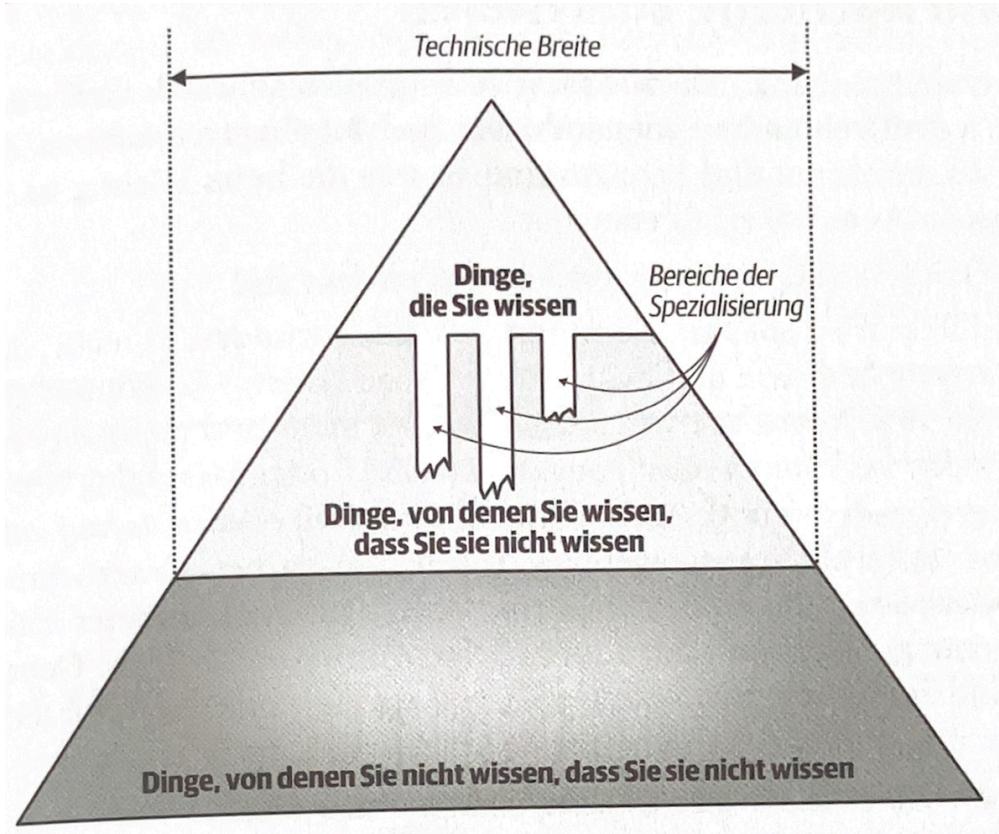
- Wenn die inhaltliche Korrektheit wichtig ist (was sehr oft der Fall ist), muss der KI-Output vollständig überprüft werden.
- Voraussetzung dafür ist ein vollständiges Verstehen des Inhalts.
- In der Ausbildung ist dies meistens nicht der Fall.
- Der Lerneffekt ist gering, echtes Verständnis entsteht, wenn man sich intensiv mit einer Materie auseinandersetzt.
- Um auf dem Arbeitsmarkt erfolgreich zu sein, ist kritisches Denken essenziell.
- Ein Studium ist das ideale Umfeld, dies zu lernen.

vgl. <https://www.golem.de/news/produktivitaetssabotage-ki-muell-kostet-unternehmen-millionen-2509-200417.html>

Konkrete Empfehlungen

- KI verwenden für:
 - Analyse von Dokumenten(-sammlungen)
 - Boilerplate-Code
 - Formulierungen und Textkorrektur
 - Entwurf von Lösungsstrategien
- KI nicht verwenden für:
 - Schreiben von ganzen Dokumenten, Arbeiten, E-Mails
 - Schreiben von ganzen Funktionen und Klassen
 - Entwurf von ganzen Architekturen

Lernen



(Richards, 2021, S.29)

Iteratives und inkrementelles Arbeiten

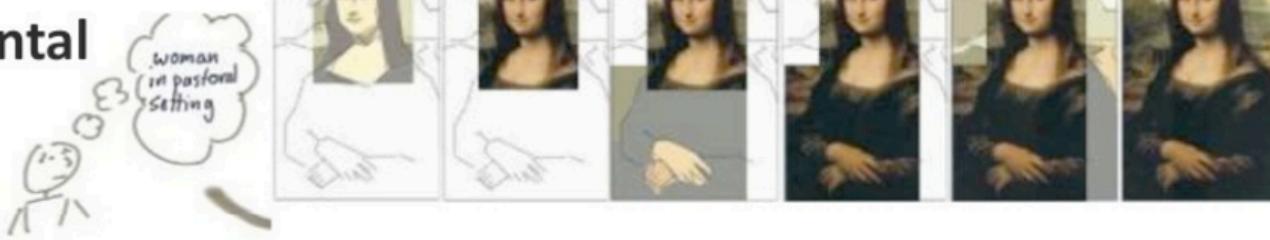
Iterative



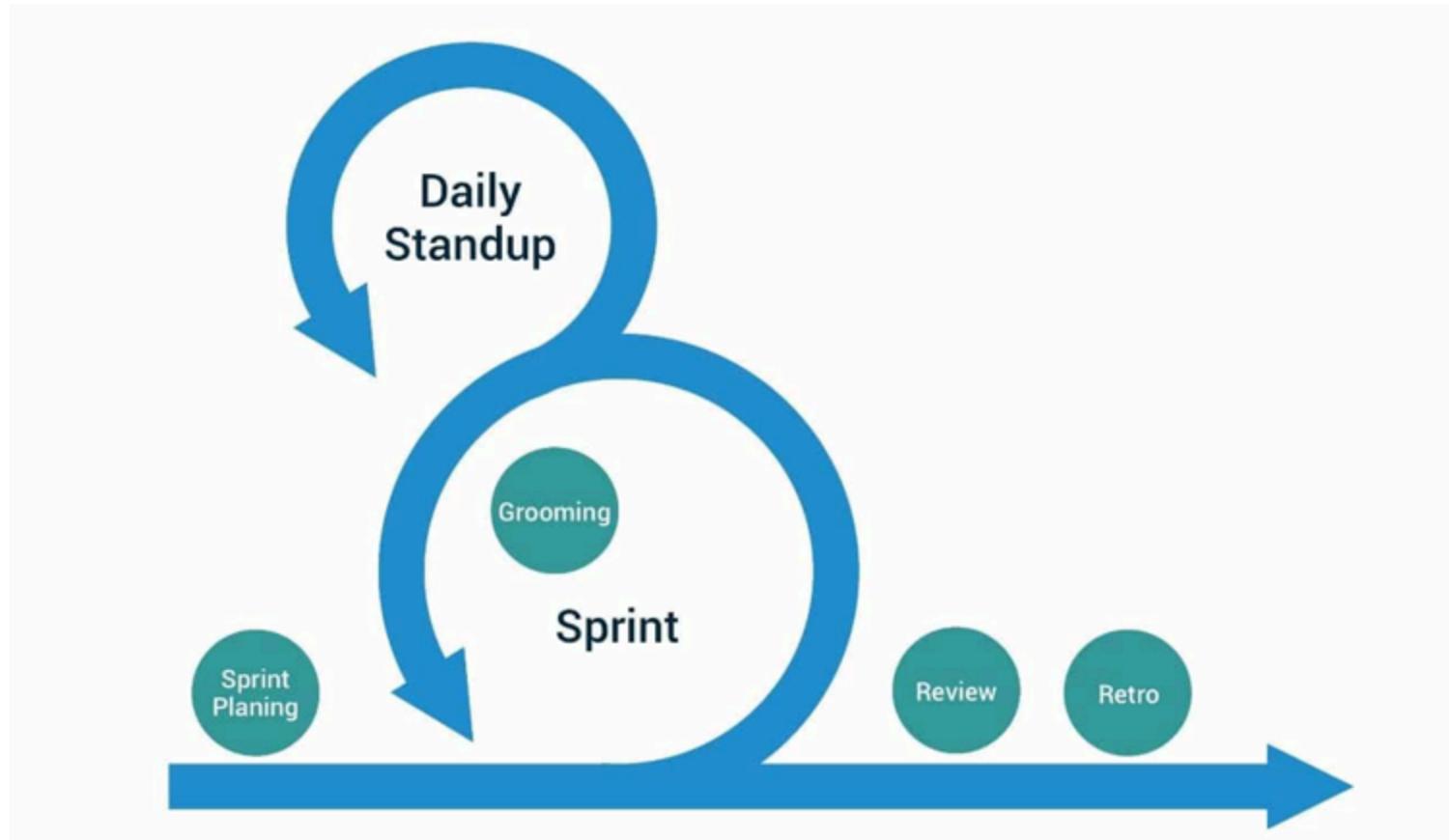
Incremental



Iterative &
Incremental



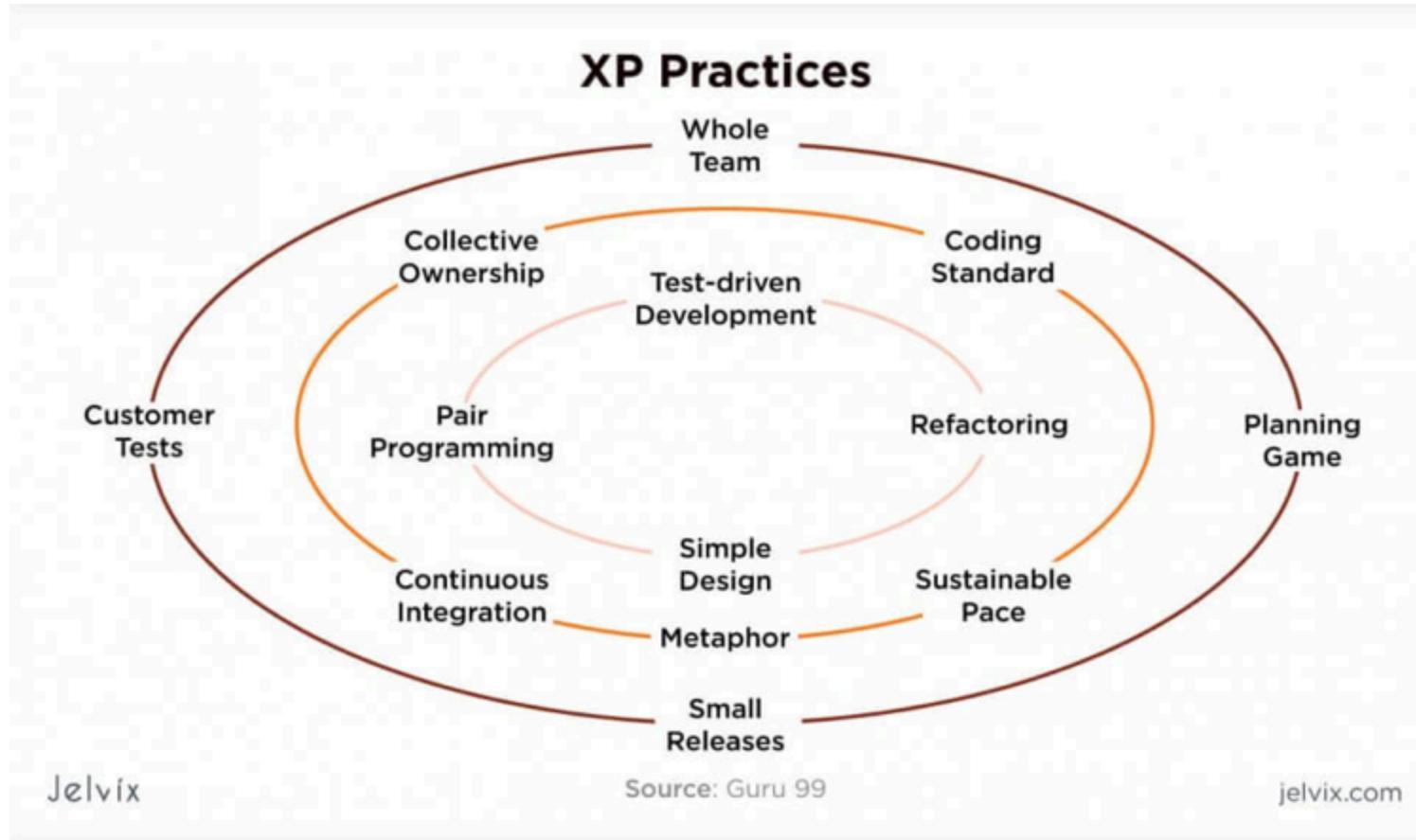
Iterationen



Embrace Change

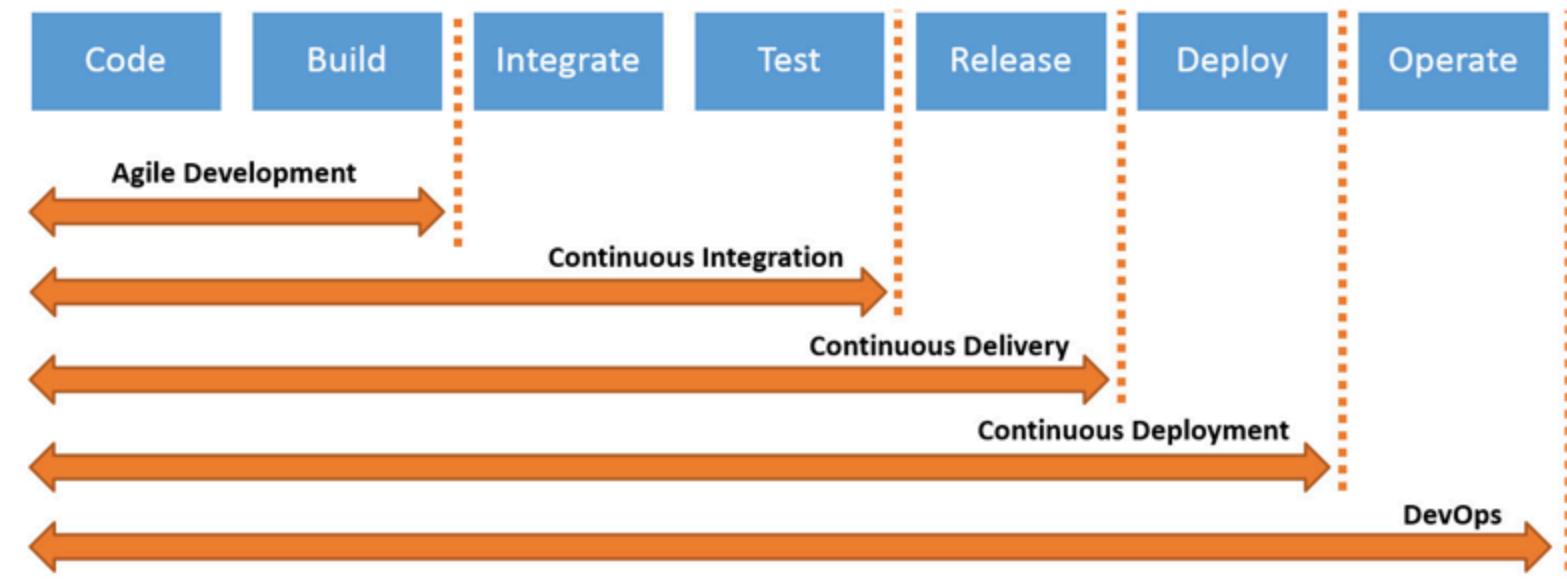


Extreme Programming



Feedback

CI/CD



Continuous Integration

- **Kein Branching**, alle Änderungen werden von allen Teammitgliedern **mehrmals täglich ** in den Master Branch eingeccheckt.
- Dieser Branch ist **jederzeit lauffähig**
- Dadurch werden die **Releases vereinfacht**
- Eine sehr hohe, **automatische Testabdeckung** ist zwingend

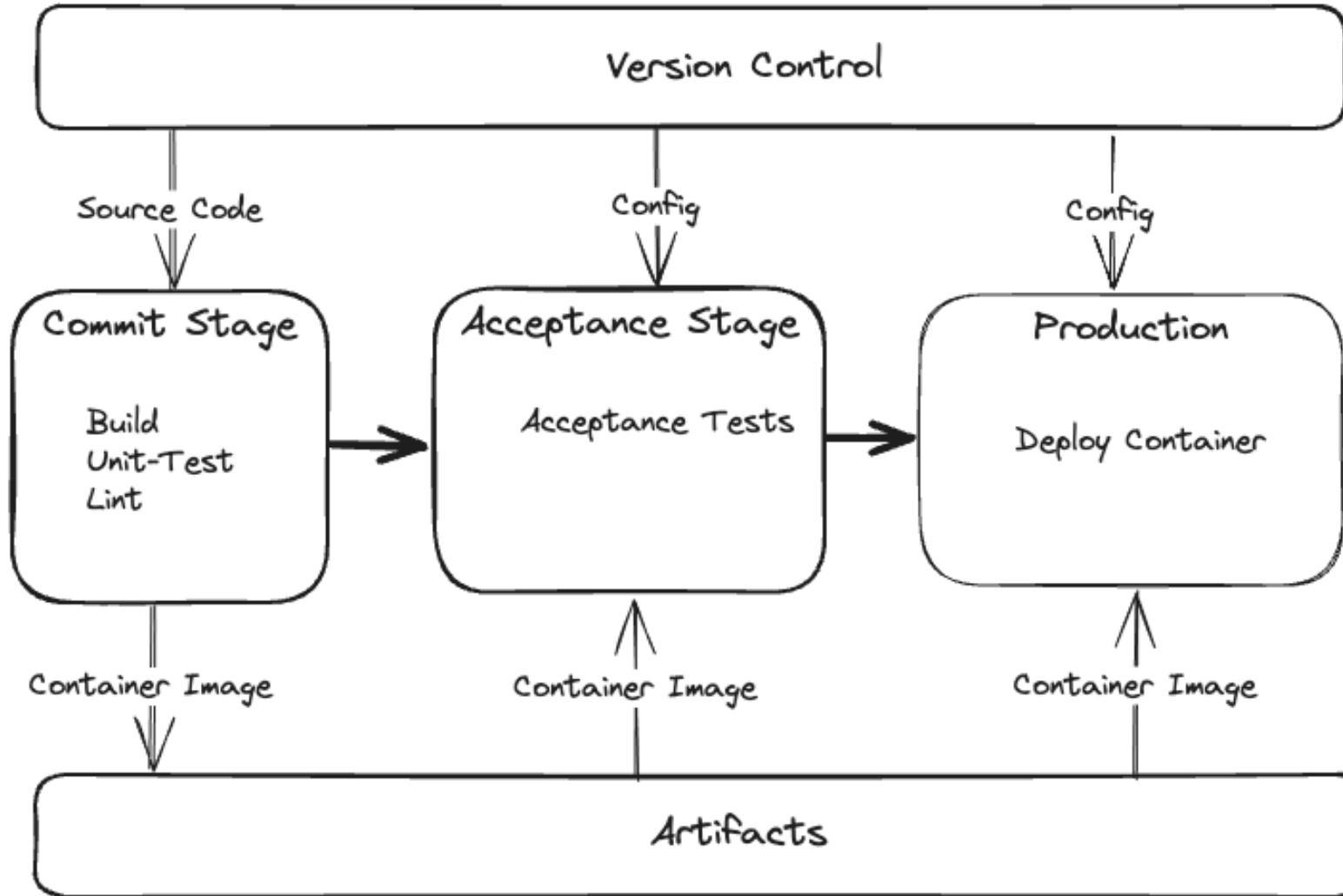
Continuous Deployment

- Ziel: **Releases werden vereinfacht**
- **Time to market ist kürzer**, neue Features sind sofort verfügbar
- Durch automatisierte Deployments ist der Aufwand initial höher, anschliessend jedoch sehr klein
- **Higher quality, Better products**
- Kaum mehr Release-Stress, **Happier teams**

<https://www连续部署.com/>

Modern Software Engineering

Deployment Pipelines



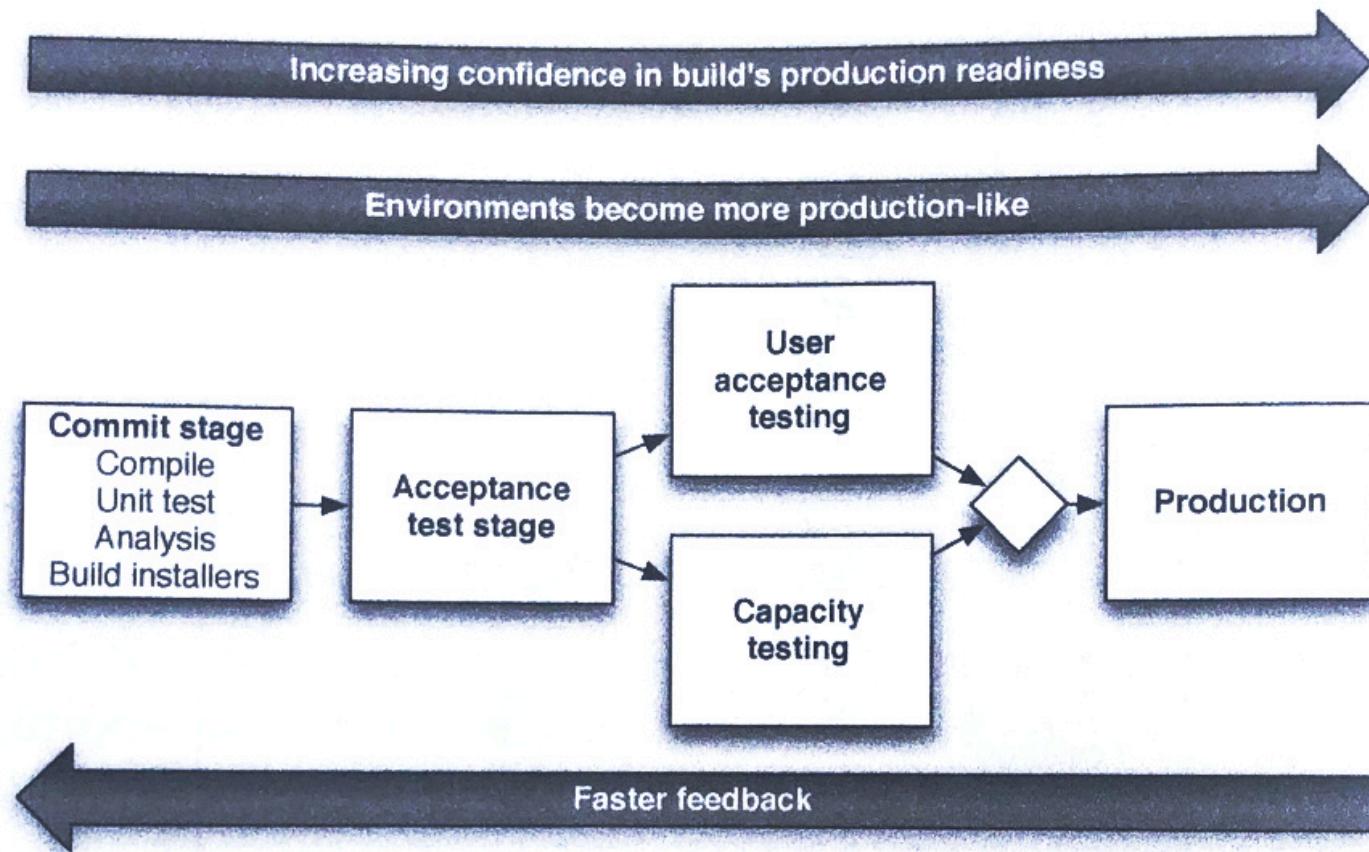
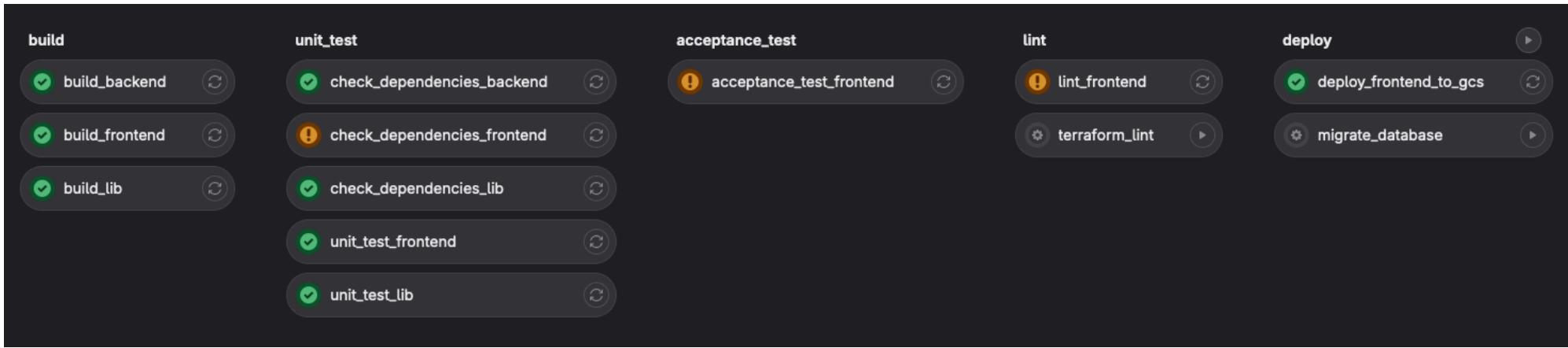
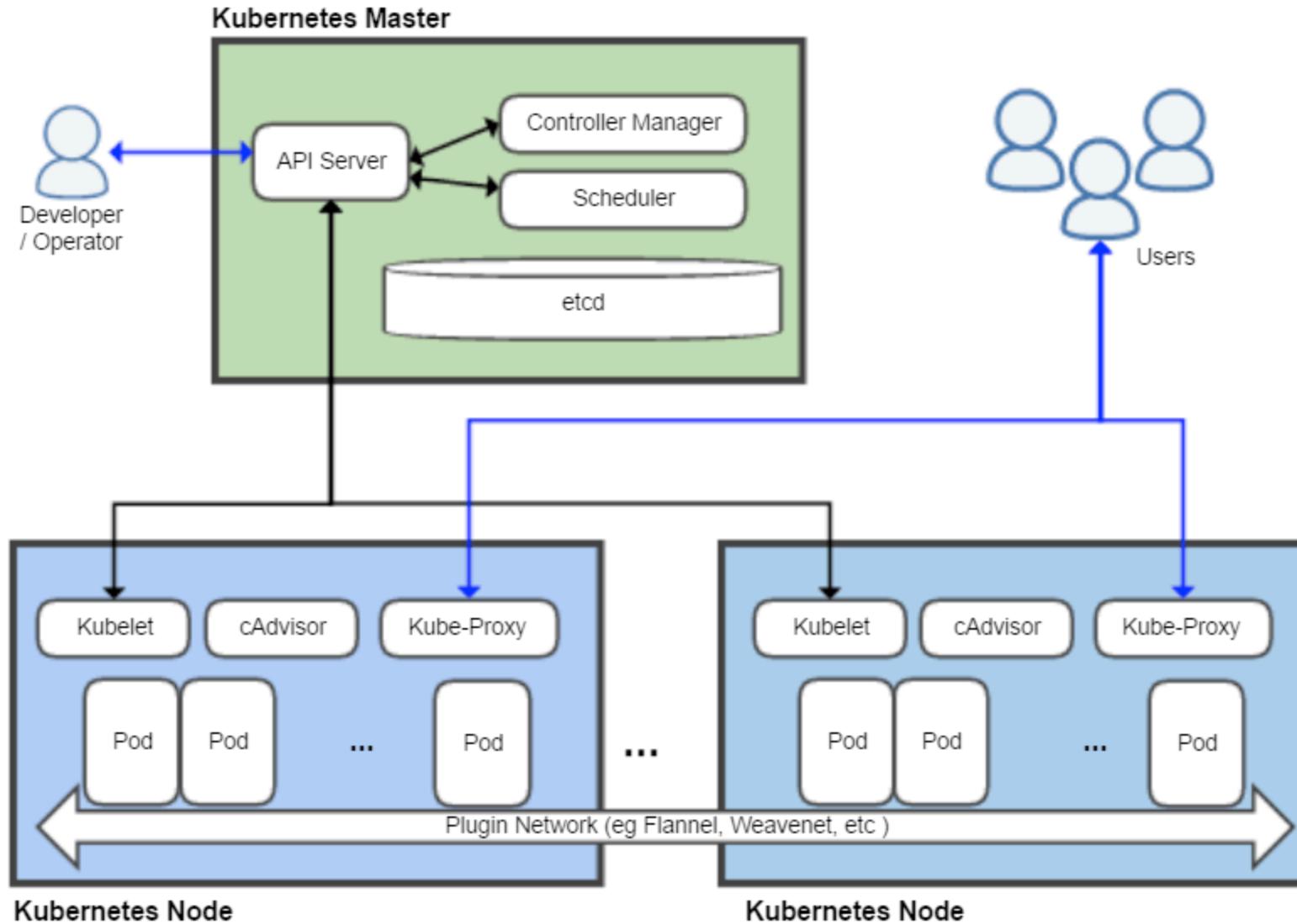


Figure 5.3 *Trade-offs in the deployment pipeline*



Kubernetes



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

Quellen

- [Youtube: Continuous Delivery - Deployment Pipelines](#)
- Jez Humble, David Farley (2010): Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley Signature Series (Fowler)

Empirisches und experimentelles Arbeiten

Rule 1. You can't tell where a program is going to spend its time. Bottlenecks occur in surprising places, so don't try to second guess and put in a speed hack until you've proven that's where the bottleneck is.

Rule 2. Measure. Don't tune for speed until you've measured, and even then don't unless one part of the code overwhelms the rest.

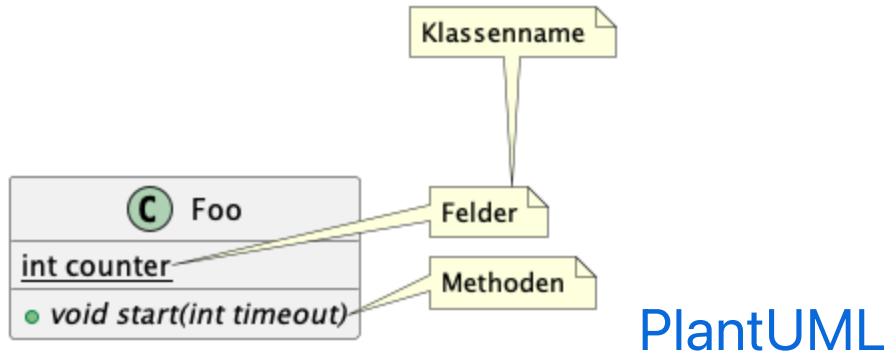
Rule 3. Fancy algorithms are slow when n is small, and n is usually small. Fancy algorithms have big constants. Until you know that n is frequently going to be big, don't get fancy. (Even if n does get big, use Rule 2 first.)

Rule 4. Fancy algorithms are buggier than simple ones, and they're much harder to implement. Use simple algorithms as well as simple data structures.

Kommunikation

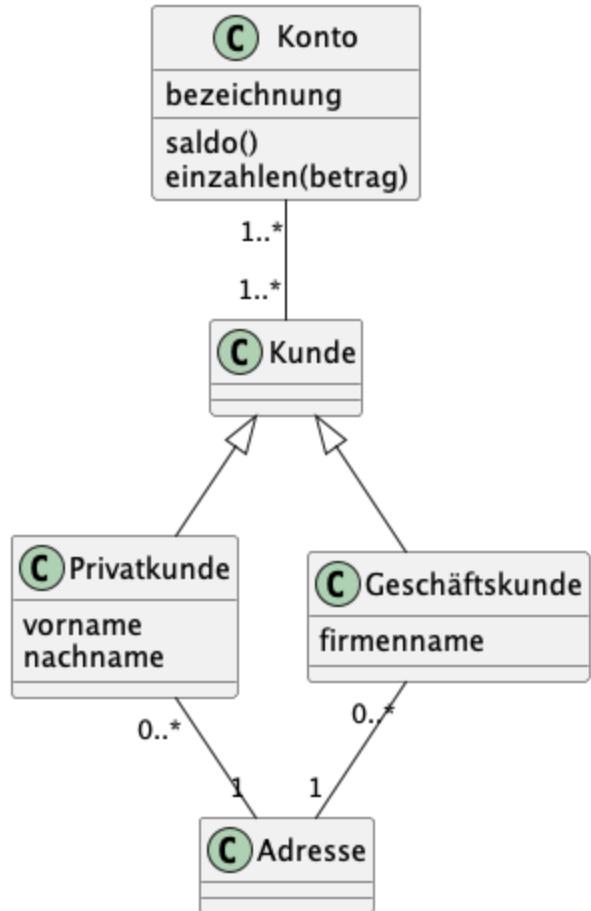
Domain Driven Design

UML Klassendiagramm



PlantUML

UML Klassendiagramm



PlantUML

```
@startuml
class Konto {
    bezeichnung
    saldo()
    einzahlen(betrag)
}

class Kunde {}

class Privatkunde {
    vorname
    nachname
}

class Geschäftskunde {
    firmenname
}

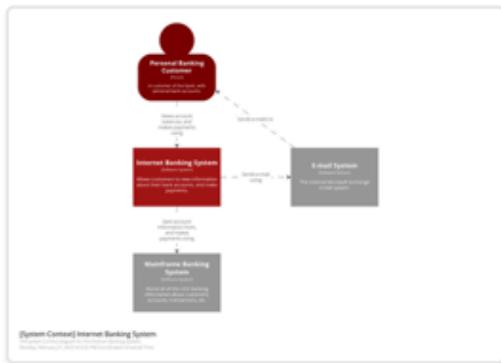
class Adresse {}

Kunde <|-- Privatkunde
Kunde <|-- Geschäftskunde

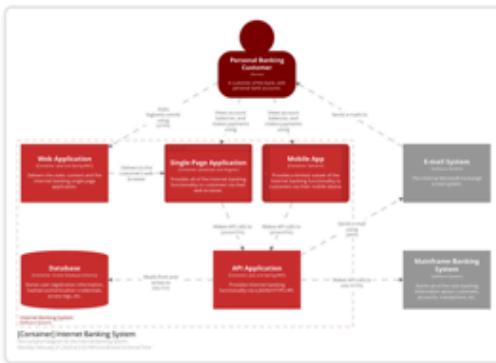
Privatkunde "0..*" -- "1" Adresse
Geschäftskunde "0..*" -- "1" Adresse

Konto "1..*" -- "1..*" Kunde
@enduml
```

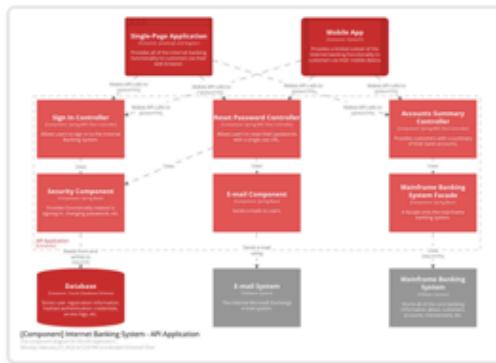
C4 Model



Level 1: A **System Context** diagram provides a starting point, showing how the software system in scope fits into the world around it.



Level 2: A **Container** diagram zooms into the software system in scope, showing the high-level technical building blocks.



Level 3: A **Component** diagram zooms into an individual container, showing the components inside it.

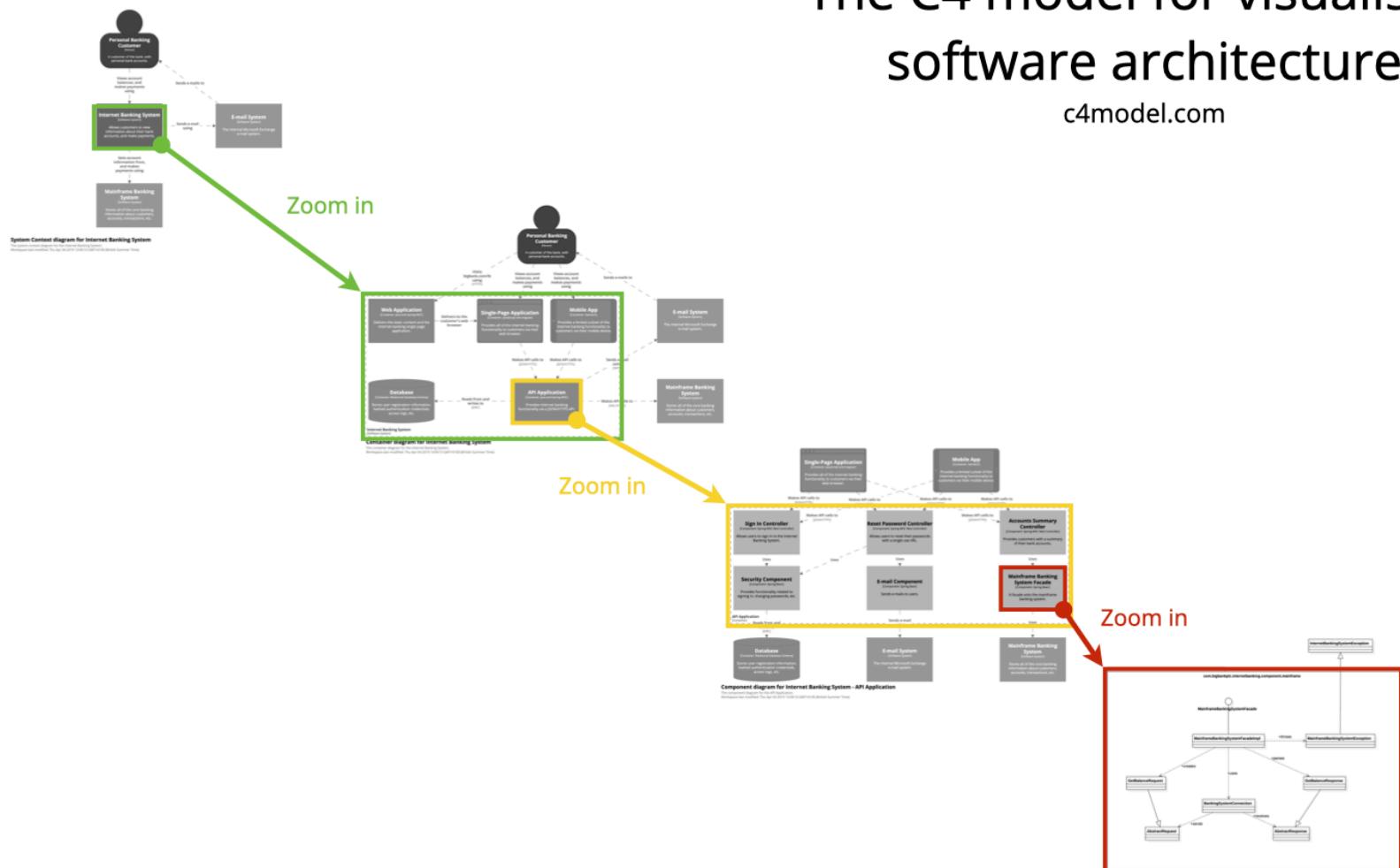


Level 4: A **code** (e.g. UML class) diagram can be used to zoom into an individual component, showing how that component is implemented.

<https://c4model.com/>

The C4 model for visualising software architecture

c4model.com

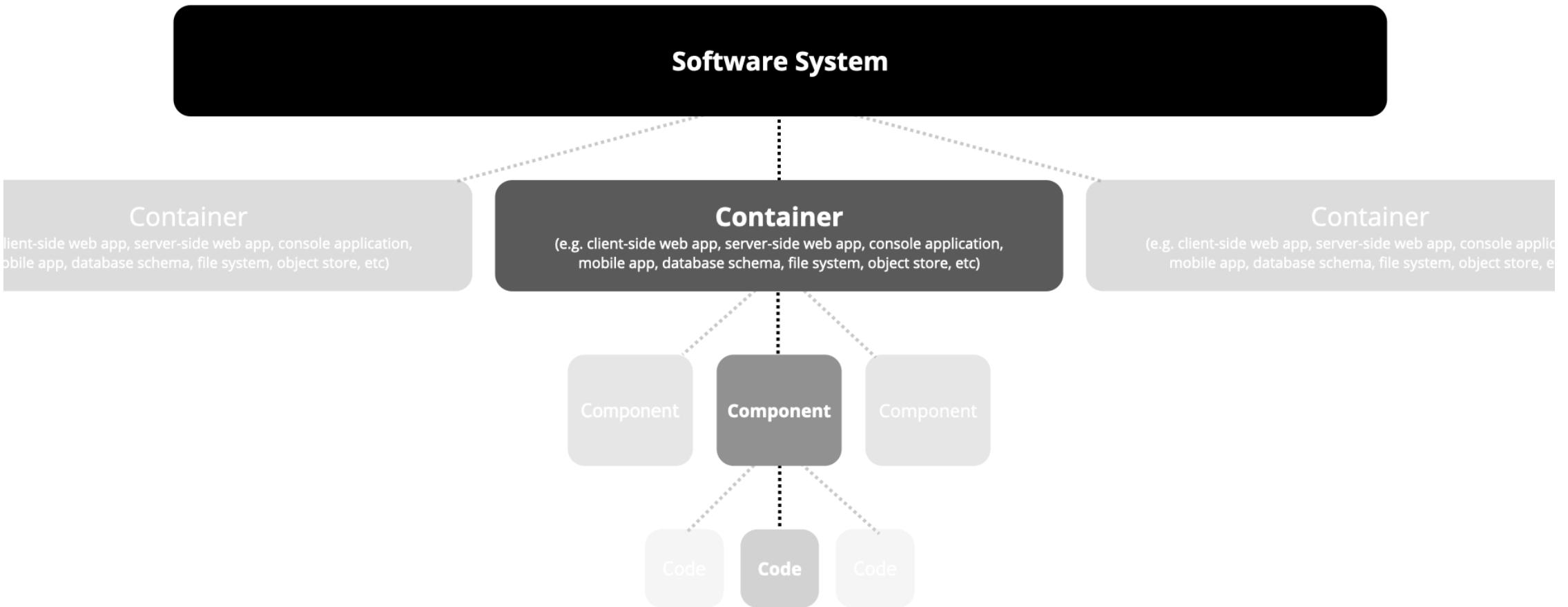


Level 1
Context

Level 2
Containers

Level 3
Components

Level 4
Code



A **software system** is made up of one or more **containers** (applications and data stores), each of which contains one or more **components**, which in turn are implemented by one or more **code** elements (classes, interfaces, objects, functions, etc).

Architectural Decision Records

```
# <!-- short title, representative of solved problem and found solution -->

## Context and Problem Statement

## Considered Options

## Decision Outcome

### Consequences
```

- <https://github.com/adr/madr/blob/4.0.0/template/adr-template-bare-minimal.md>
- <https://github.com/adr/madr/blob/4.0.0/template/adr-template-bare.md>

Templates

- Nygard: <https://github.com/joelparkerhenderson/architecture-decision-record/blob/main/locales/en/templates/decision-record-template-by-michael-nygard/index.md>
- MADR: <https://github.com/adr/madr/blob/4.0.0/template/adr-template.md>

Tools

- <https://github.com/npryce/adr-tools>
- <https://github.com/opinionated-digital-center/pyadr>

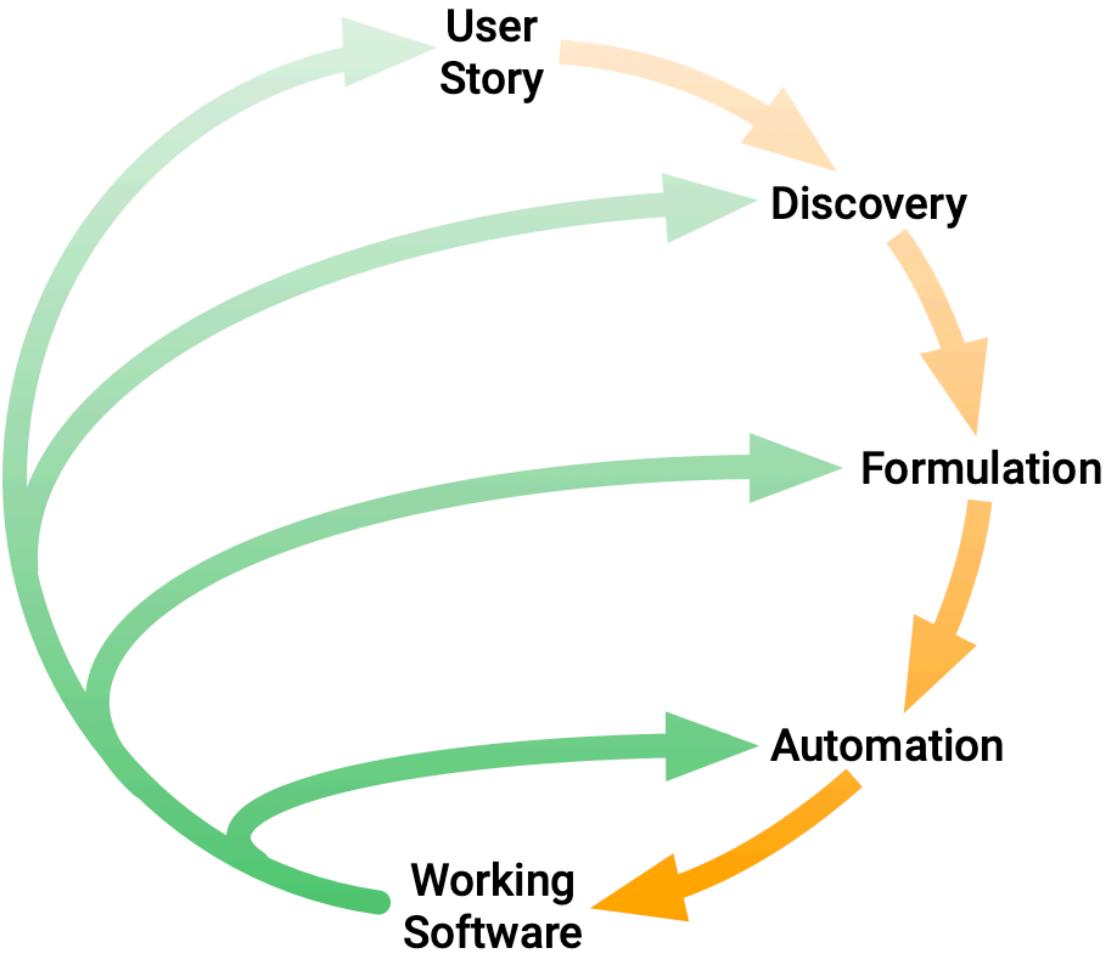
Komplexität

Modularity & Separation of Concerns

Testing

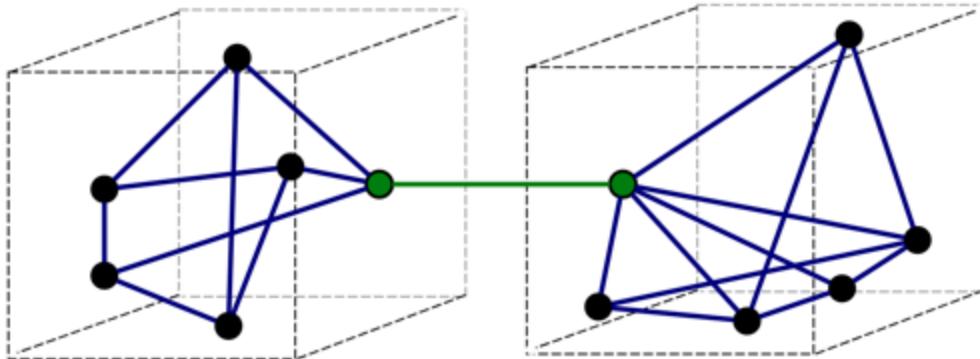
The hardest single part of building a software system is deciding precisely what to build.

– Fred Brooks, *The mythical man-month*

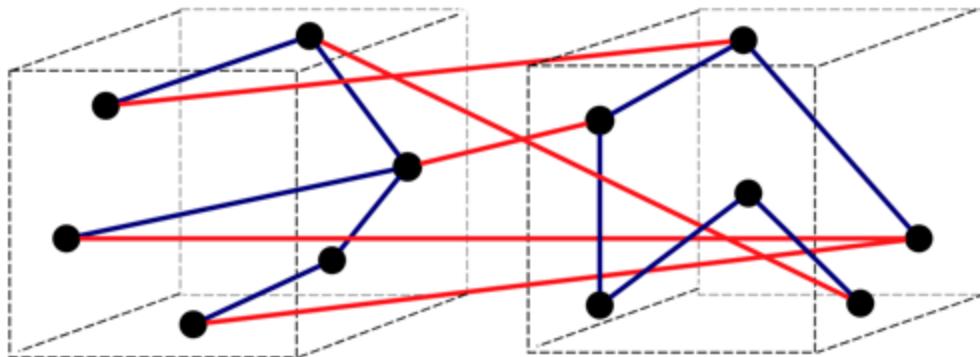


<https://cucumber.io/docs/bdd/>

Cohesion & Coupling



a) Good (loose coupling, high cohesion)

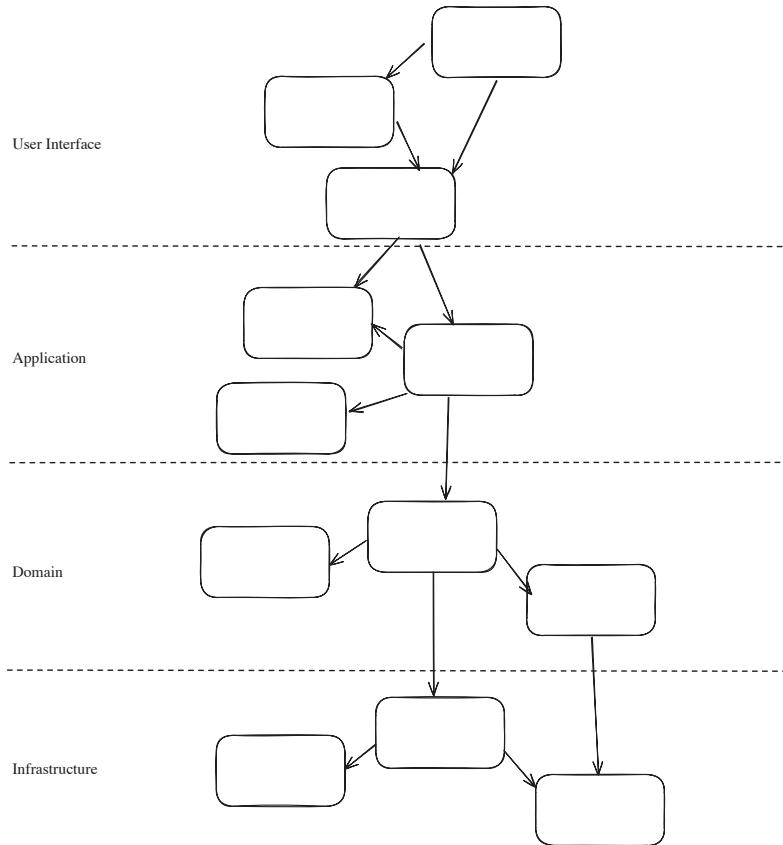


b) Bad (high coupling, low cohesion)

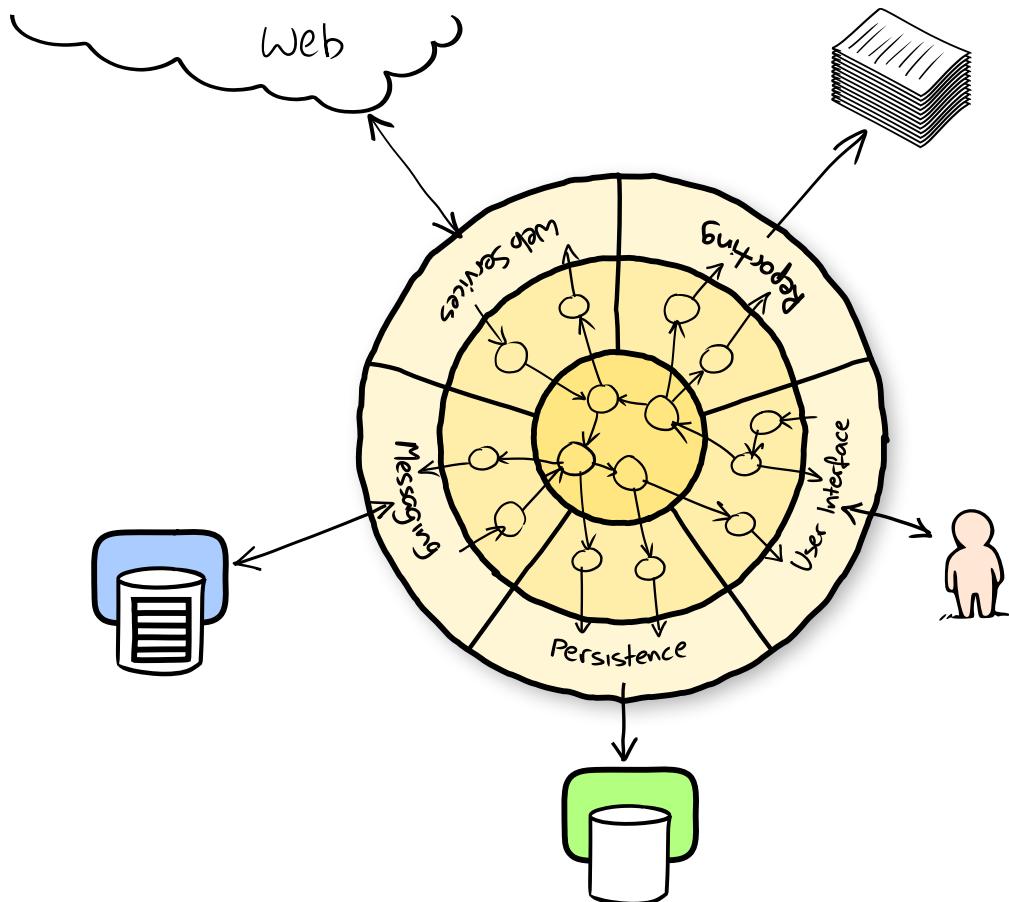
Abstraction

Architekturen

Schichtenarchitektur

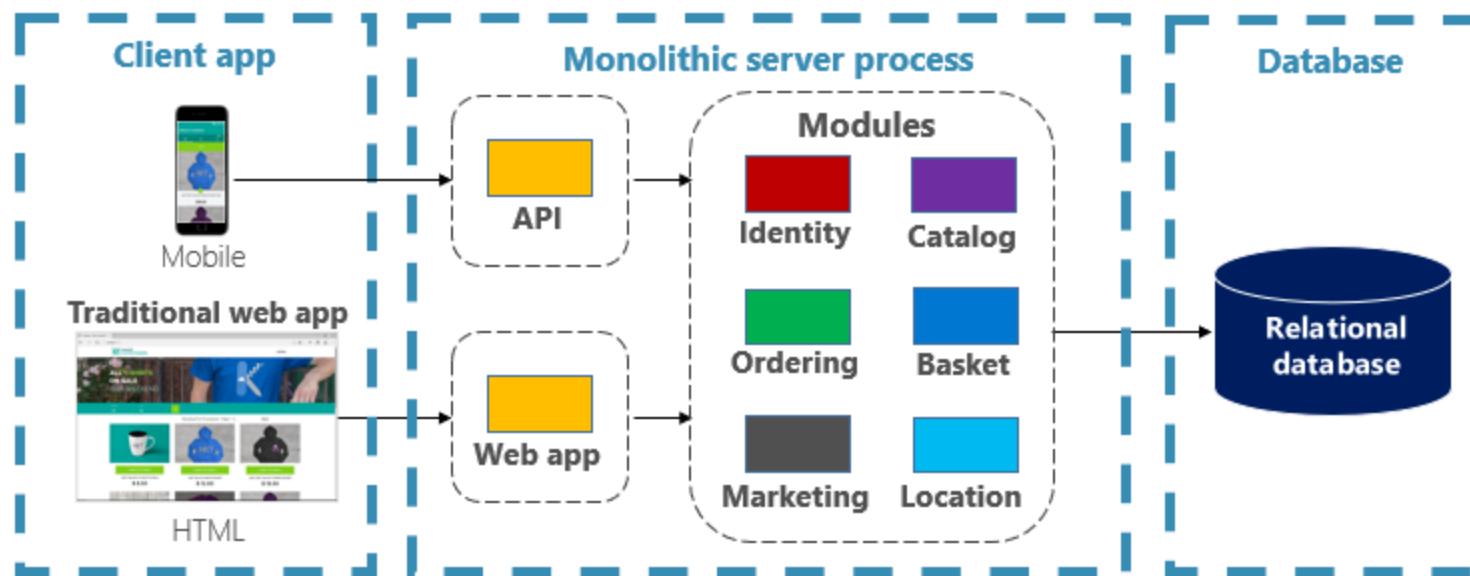


Ports and Adapters

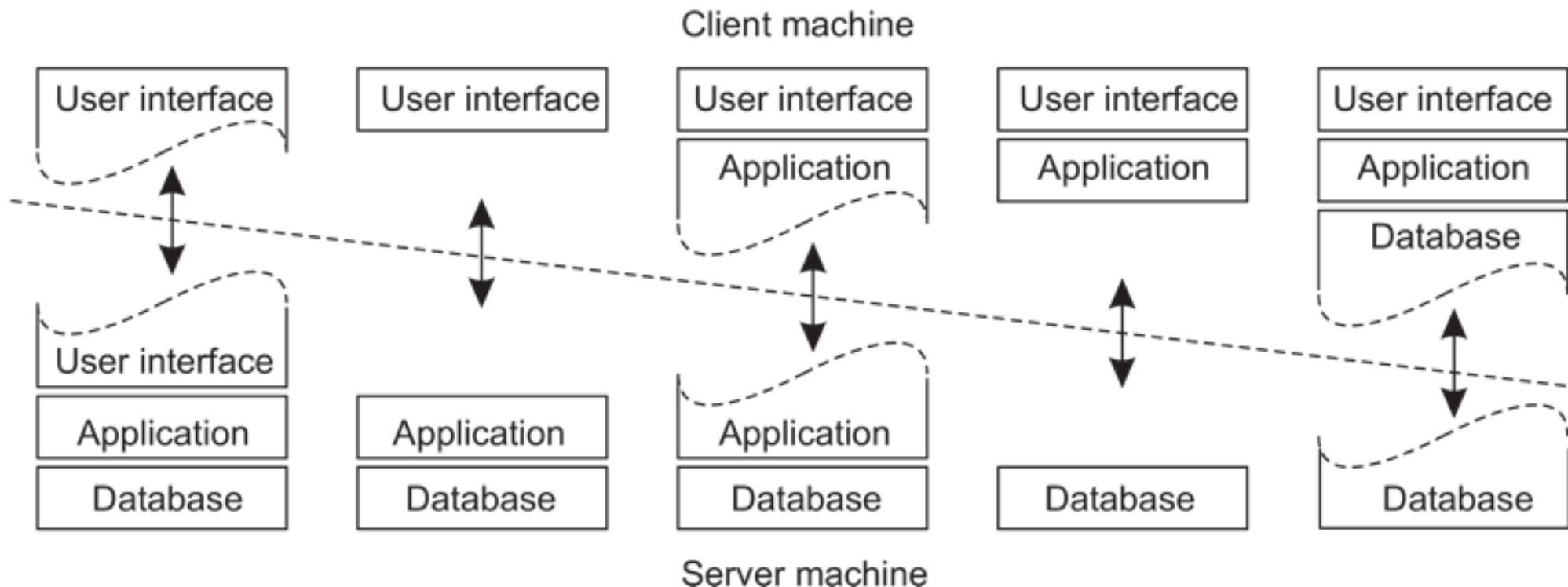


growing-object-oriented-software.com

Traditional Monolithic Design



Schichtenarchitektur im Client Server Modell



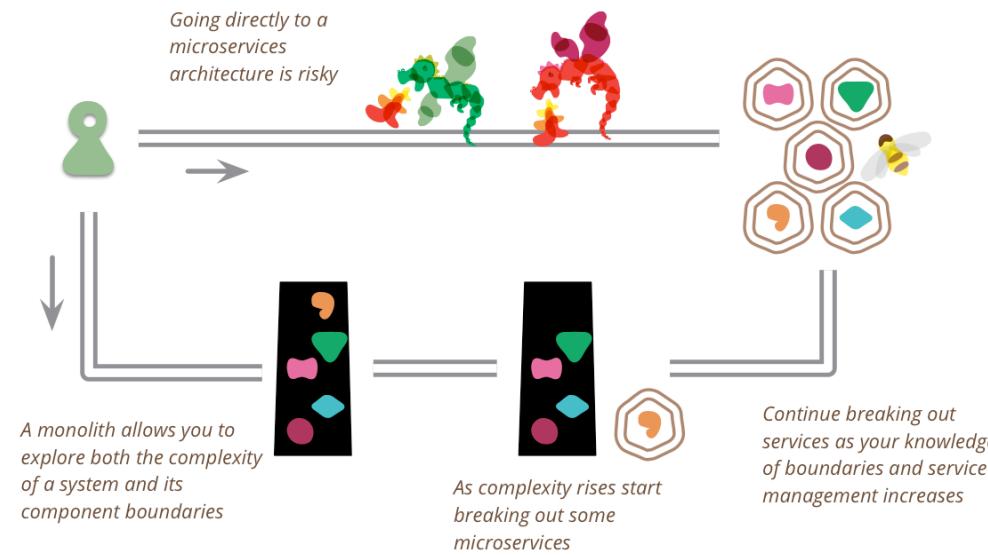
Microservices

- Maximale Skalierbarkeit
- Einzelne Services können von **kleinen**[^1] Teams **unabhängig entwickelt und deployed** werden
- Bessere Wart- und Erweiterbarkeit
- Unterschiedliche Technologien können eingesetzt werden
- Kommunikation nicht trivial
- Höhere Wahrscheinlichkeit eines Ausfalls
- **Hohe Komplexität**

martinfowler.com/articles/microservices.html [^1]: "We try to create teams that are no larger than can be fed by two pizzas"

Monolith First

- Vorsicht vor **Cargo-Kult**: Amazon, Google, Meta etc. haben heute andere Herausforderungen als Startups
- Technologien oder Architekturen wählen, "weil Google macht das auch so" ist ein

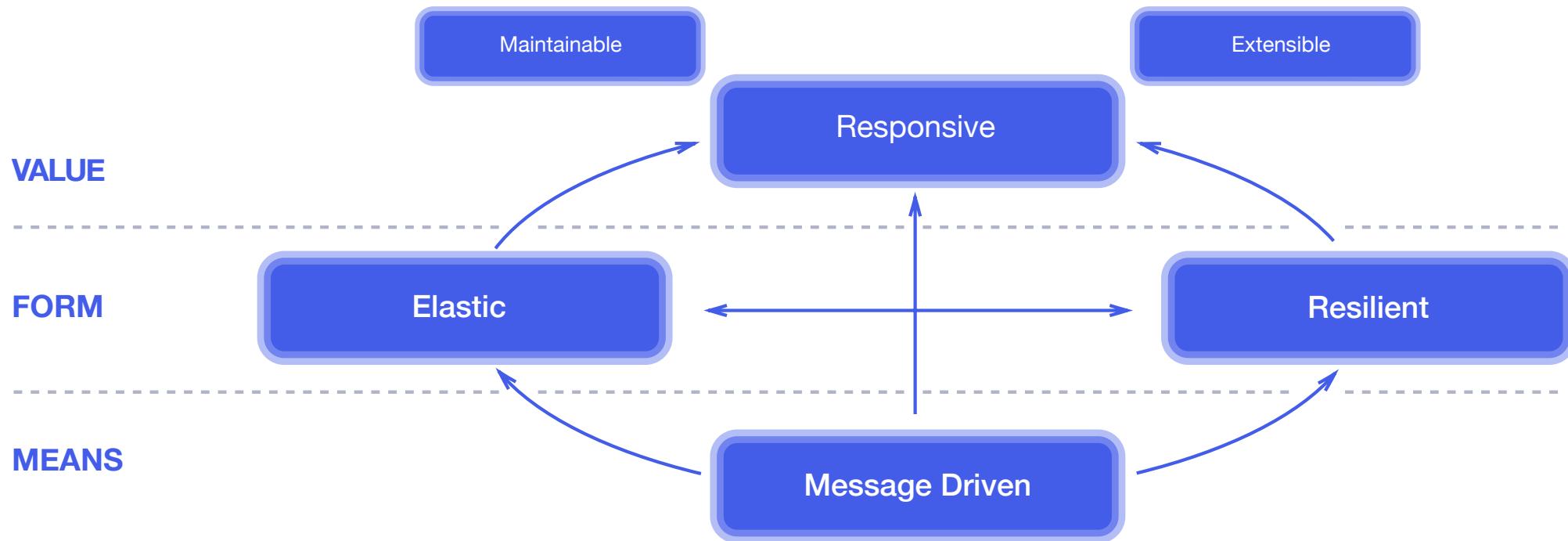


schlechter Grund

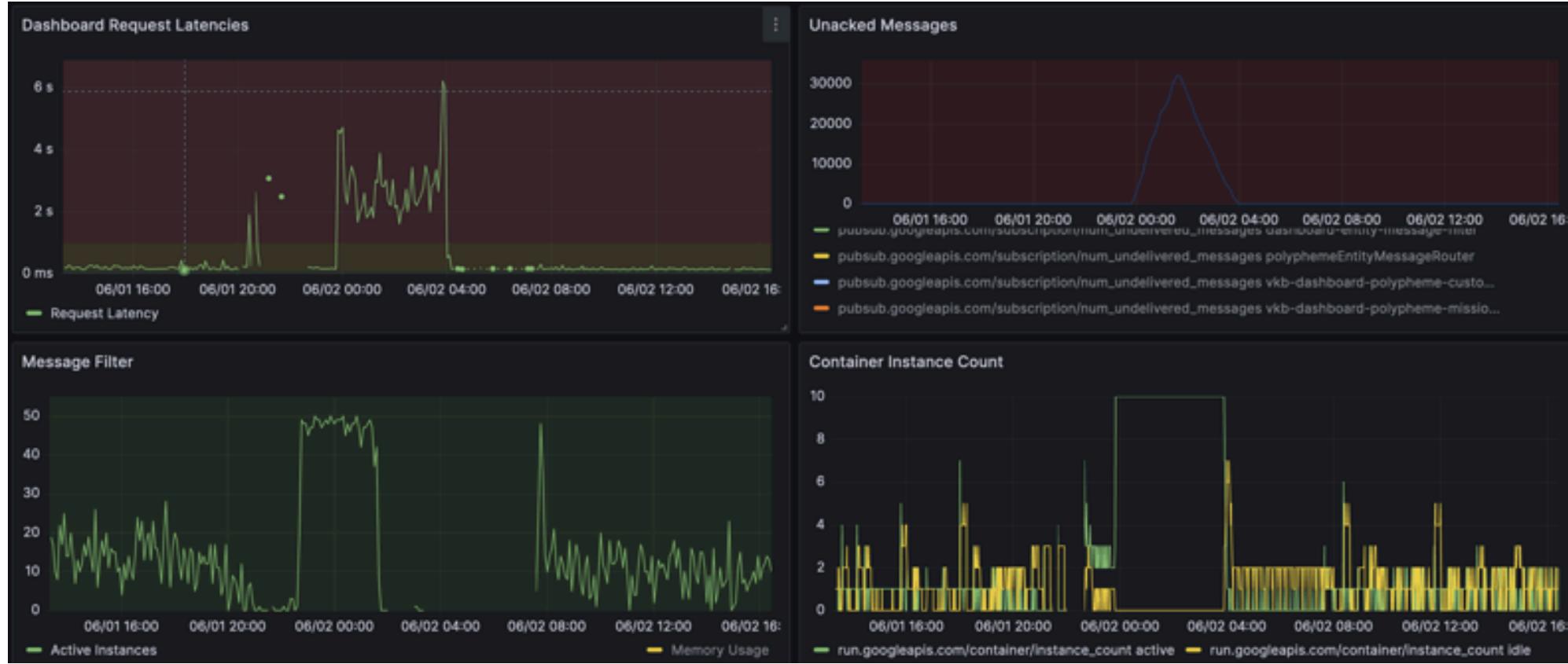
martinfowler.com/bliki/MonolithFirst.html

Event Driven Architecture

Reactive Systems



Fallstudie



Quellen

Farley, 2022 : David Farley (2022): Modern Software Engineering: Doing What Works to Build Better Software Faster, Addison-Wesley

Martin, 2018 : Robert C. Martin (2018): Clean Architecture: A Craftman's Guide to Software Structure and Design, Prentice Hall

Richards, 2021 : Mark Richards, Neal Ford (2021): Handbuch moderner Softwarearchitektur: Architekturstile, Patterns und Best Practices, O'Reilly, 978-3-96009-149-3