Die neue Schule der Softwarearchitektur

Stefan Toth | st@embarc.de | st_toth (twitter)

online
05. Mai 2021

embarc
Software Consulting GmbH

Stefan Toth

st@embarc.de

@st_toth

xing.to/sto

www.embarc.de
www.swamuster.de


Ideal für das ISAQB® Advanced Modul AGILA
stefan TOTH
VORGEHENSMUSTER FÜR
SOFTWARE-ARCHITEKTUR
3. Auflage
KOMBINIERBARE PRAKTIKEN IN ZEITEN VON AGILE UND LEAN
E-Book
EXTRA: E-Book inside
www
im Internet: www.swamuster.de
HANSER
stefan TOTH
HANSER
HANSER


shortcuts
Resilience
Wie Netflix sein System schützt
Uwe Friedrichsen, Stefan Toth, Eberhard Wolff


embarc
Software Consulting GmbH


NOSERENGINEERING
WE KNOW HOW


JUNGHEINRICH
Machines. Ideas. Solutions.


Lufthansa Systems


bon
prix
it's me!


BISON


BOSCH
Technik fürs Leben


CLAAS


ING DiBa
Die Bank und Du


Soptim
Zukunft gestalten


MHP
A PORSCHE COMPANY


holisticon


Raiffeisen Bank
International


EOS UPTRADE
KOMPLEXES EINFACH MACHEN.


HAMBURG SÜD

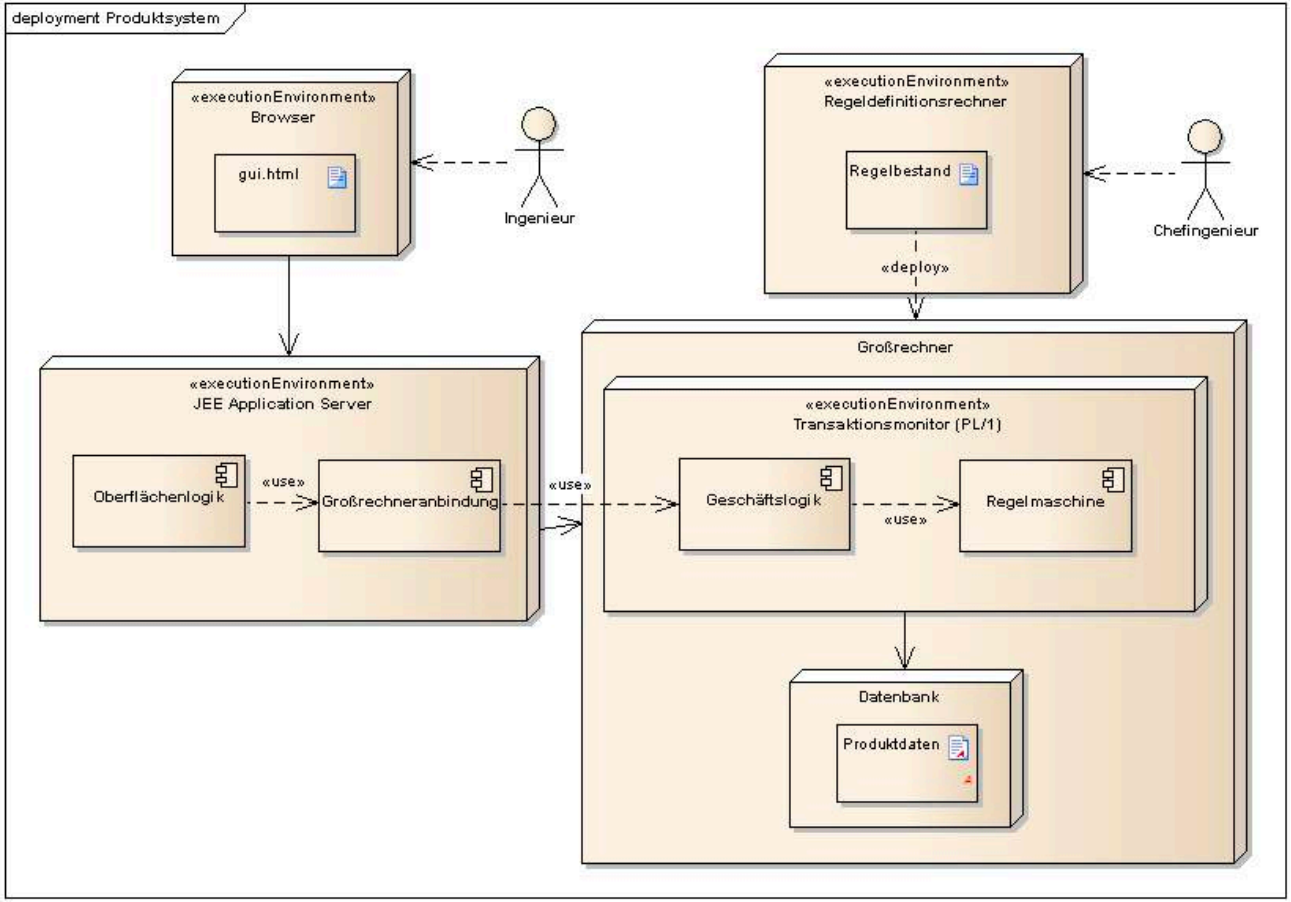

AIRBUS

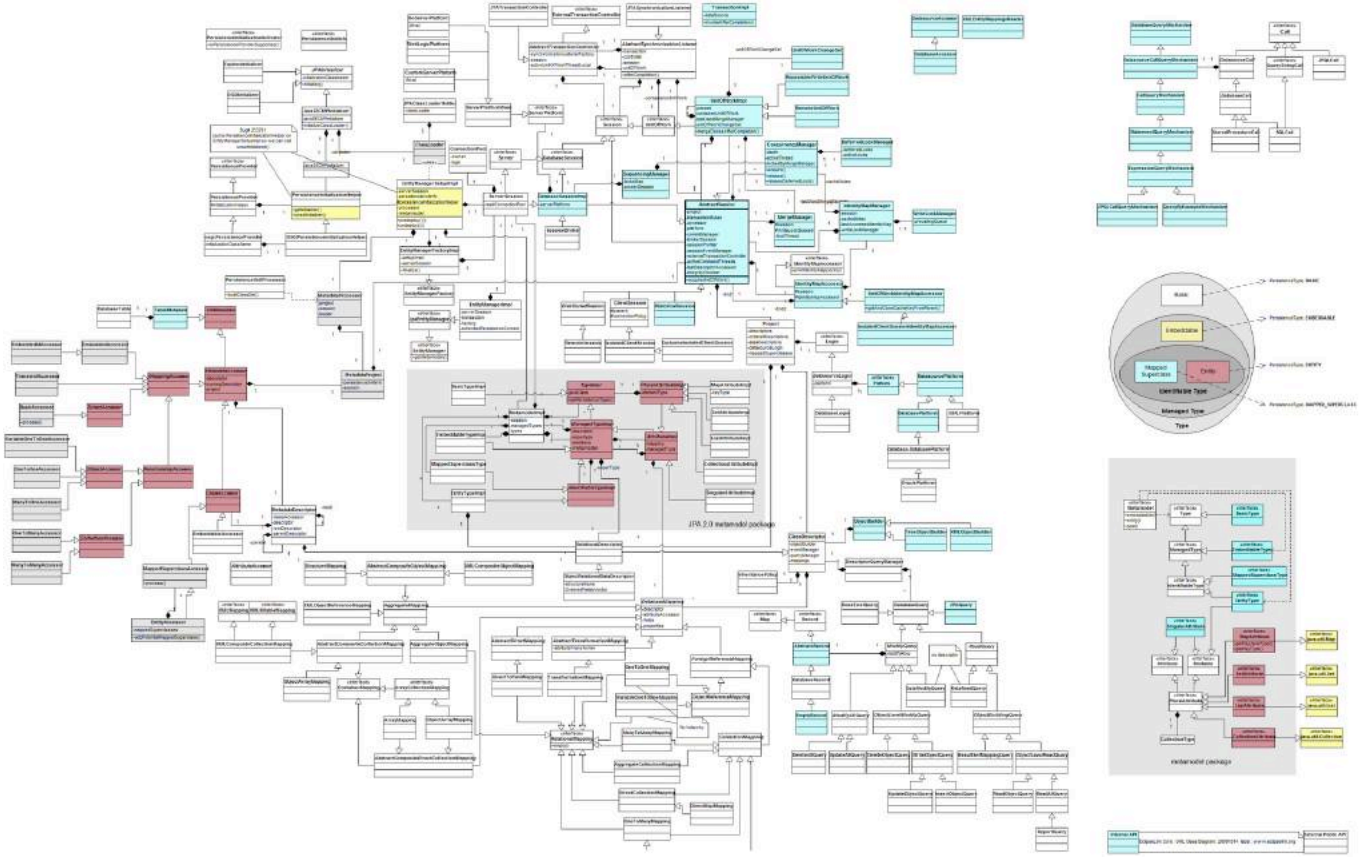
Worüber ich heute spreche...

Architektur-Arbeit

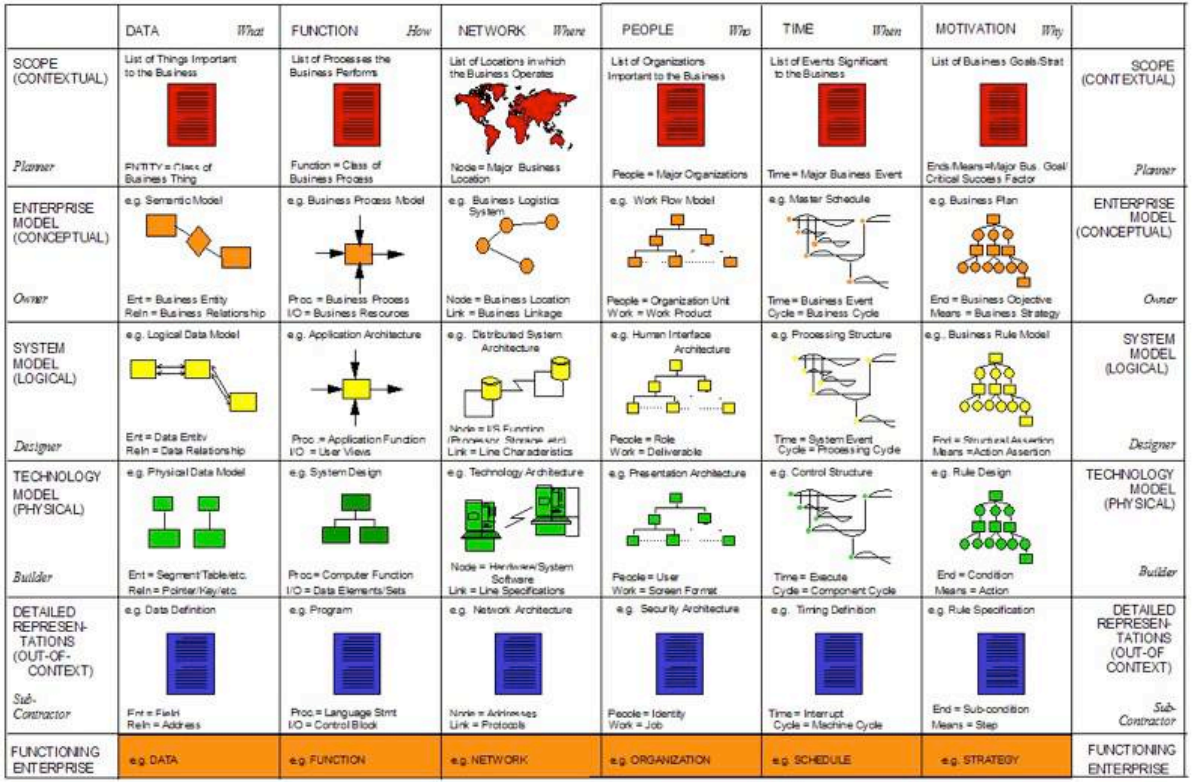
Architekturen entwerfen und weiterentwickeln

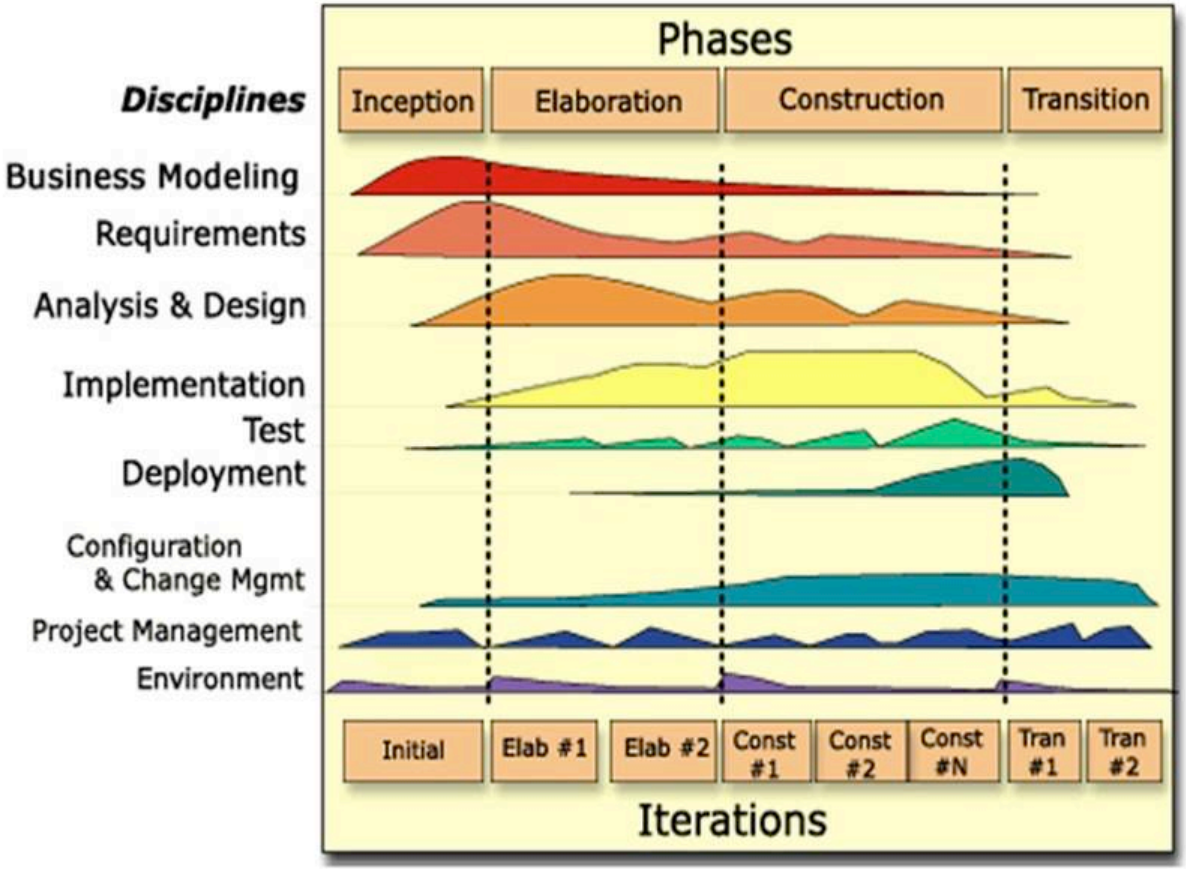
Alte Schule der Software Architktur


deployment Produktsystem
«executionEnvironment»
Browser
gui.html
Ingenieur
«executionEnvironment»
Regeldefinitionsrechner
Regelbestand
Chefingenieur
«deploy»
Großrechner
«executionEnvironment»
JEE Application Server
Oberflächenlogik
«use»
Großrechneranbindung
«use»
«executionEnvironment»
Transaktionsmonitor (PL/1)
Geschäftslogik
«use»
Regelmaschine
Datenbank
Produktdaten

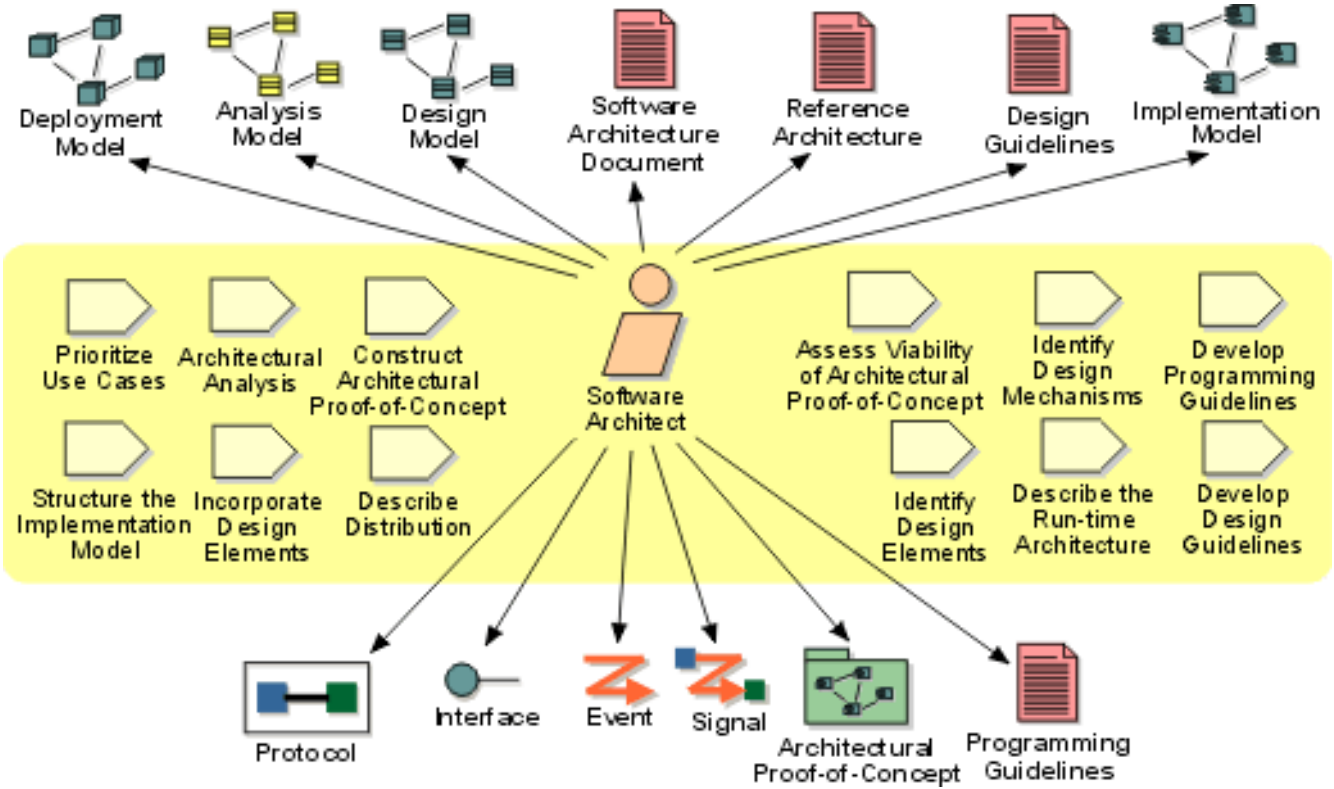


Zachman Framework

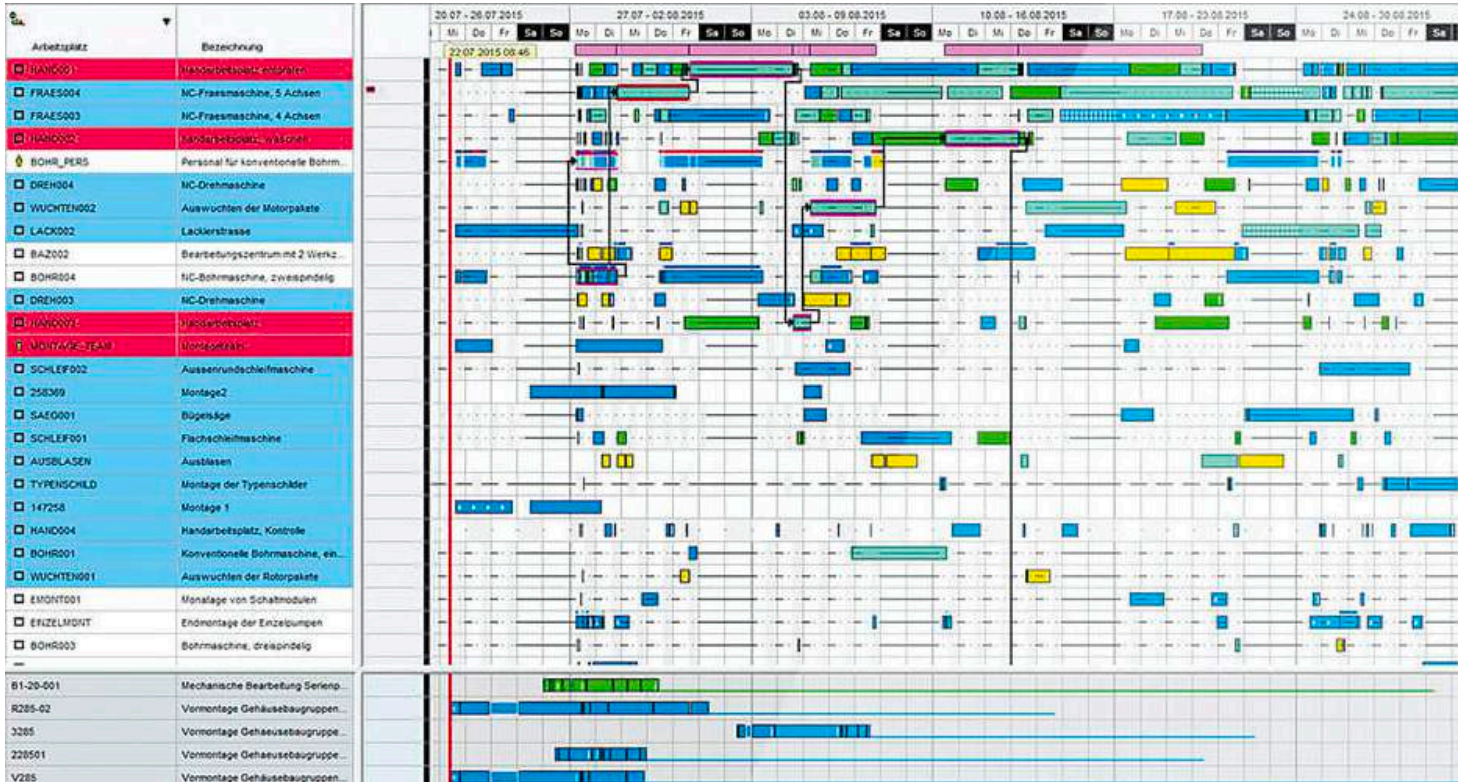



Phases
Disciplines
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

Der Architekt (90er)



Master Plan


Arbeitsplatz
Bezeichnung
HAND001
Handarbeitsplatz entgraten
FRAES004
NC-Fraesmaschine, 5 Achsen
FRAES003
NC-Fraesmaschine, 4 Achsen
HAND002
Handarbeitsplatz, waschen
BOHR_PERS
Personal für konventionelle Bohrm...
DREH004
NC-Drehmaschine
WUCHTEN002
Auswuchten der Motorpakete
LACK002
Lackierstrasse
BAZ002
Bearbeitungszentrum mit 2 Werkz...
BOHR004
NC-Bohrmaschine, zweispindelig
DREH003
NC-Drehmaschine
HAND003
Handarbeitsplatz
MONTAGE_TEAM
Montageteam
SCHLEF002
Aussenrundschleifmaschine
258369
Montage2
SAEG001
Bügelsäge
SCHLEF001
Flachschleifmaschine
AUSBLASEN
Ausblasen
TYPENSCHILD
Montage der Typenschilder
147258
Montage 1
HAND004
Handarbeitsplatz, Kontrolle
BOHR001
Konventionelle Bohrmaschine, ein...
WUCHTEN001
Auswuchten der Rotorpakete
EMONT001
Monatage von Schaltmodulen
EINZELMONT
Endmontage der Einzelpumpen
BOHR003
Bohrmaschine, dreispindelig
20.07 - 26.07.2015
27.07 - 02.08.2015
03.08 - 09.08.2015
10.08 - 16.08.2015
17.08 - 23.08.2015
24.08 - 30.08.2015
22.07.2015 08:46
61-20-001
Mechanische Bearbeitung Serienp...
R285-02
Vormontage Gehäusebaugruppen...
3285
Vormontage Gehaeusebaugruppe...
228501
Vormontage Gehaeusebaugruppe...
V285
Vormontage Gehäusebaugruppen...

Was ist Softwarearchitektur?

(Historische) Definitionen fokussieren oft auf die Strukturierung

“Eine Softwarearchitektur ist die Summe verschiedener wichtiger Entscheidungen über

- *die Organisation eines Softwaresystems,*
- *die Auswahl von Strukturelementen und deren Schnittstellen aus denen das System zusammengesetzt ist*
- *das Verhalten und Zusammenspiel dieser Elemente*
- *den hierarchischen Aufbau von Subsystemen*
- *den zugrunde liegenden Architekturstil“*

(Rational Unified Process)

Klassisch gewünschter Architekturprozess

Experten machen das möglichst ohne Probleme...

Architektur schafft Ordnung,

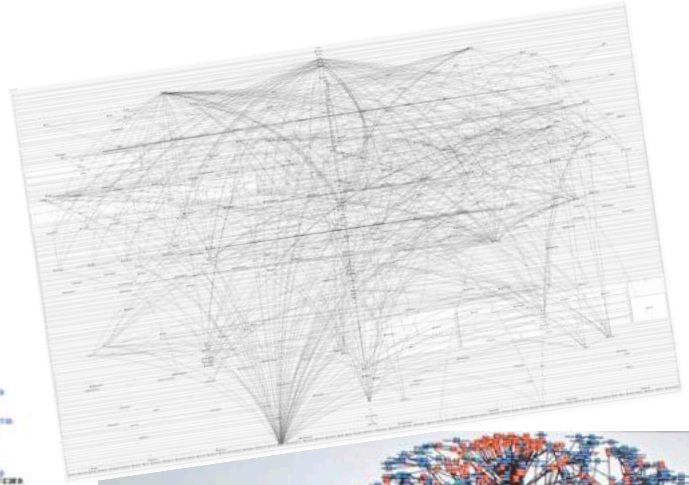
schafft Sicherheit im Vorgehen,

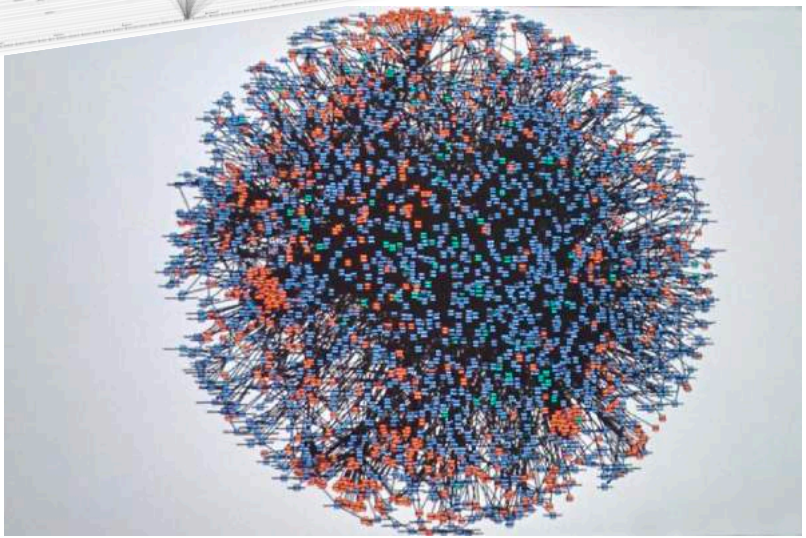
verhindert teure Fehler.

Was wir heute bauen

Ist komplex, verwoben,
dynamisch, …

Unsere Systeme...





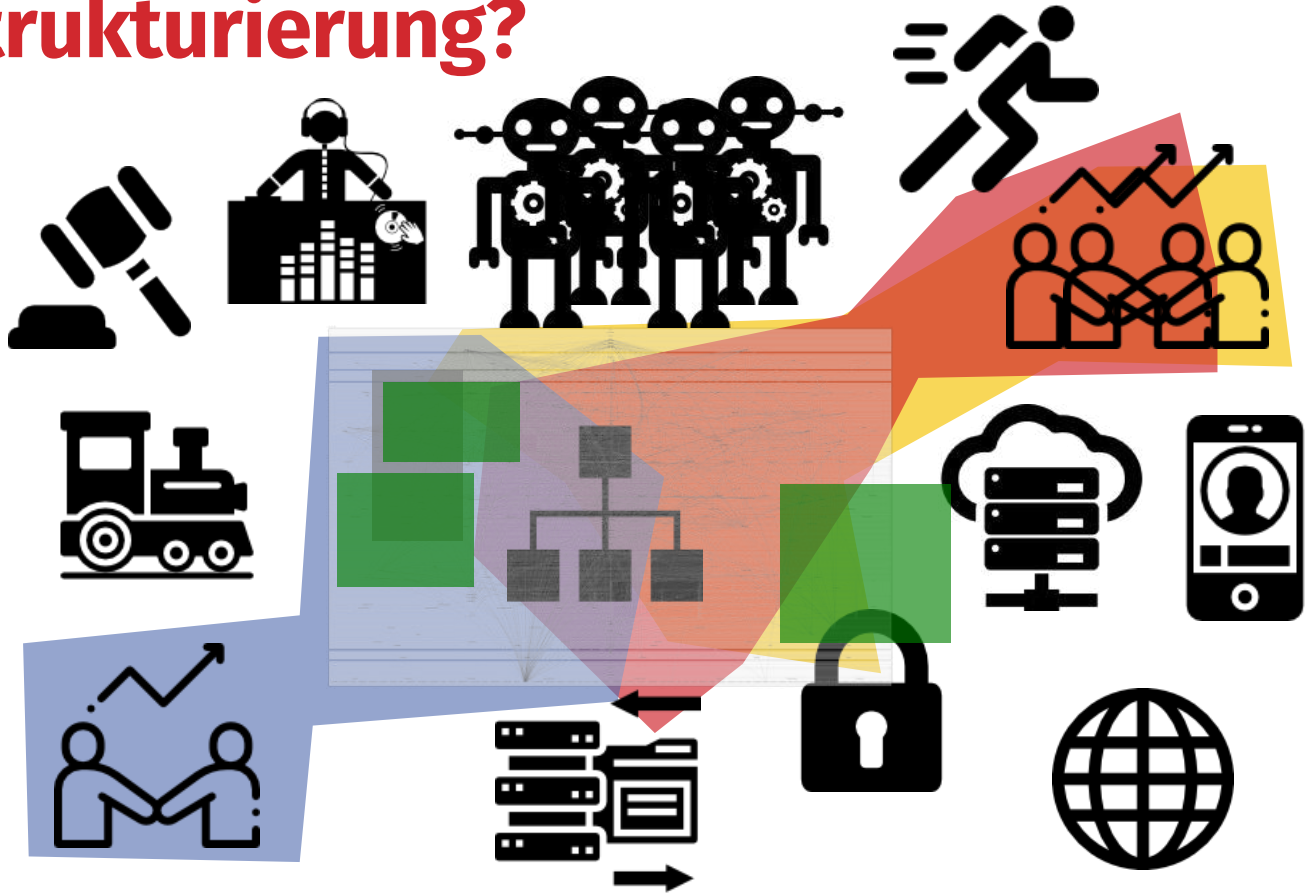
Einige Netflix Ingineure zitiert...

"[…] we quickly see that a distributed system of any meaningful size becomes too complex for a human [to understand]. There are simply too many parts, changing and innovating too quickly, interacting in too many unplanned and uncoordinated ways for a human to hold those patterns in their head."

"The same is true in other complex systems, including monoliths (usually with many, often unknown, downstream dependencies) that become so large that no single architect can understand the implications of a new feature on the entire application."

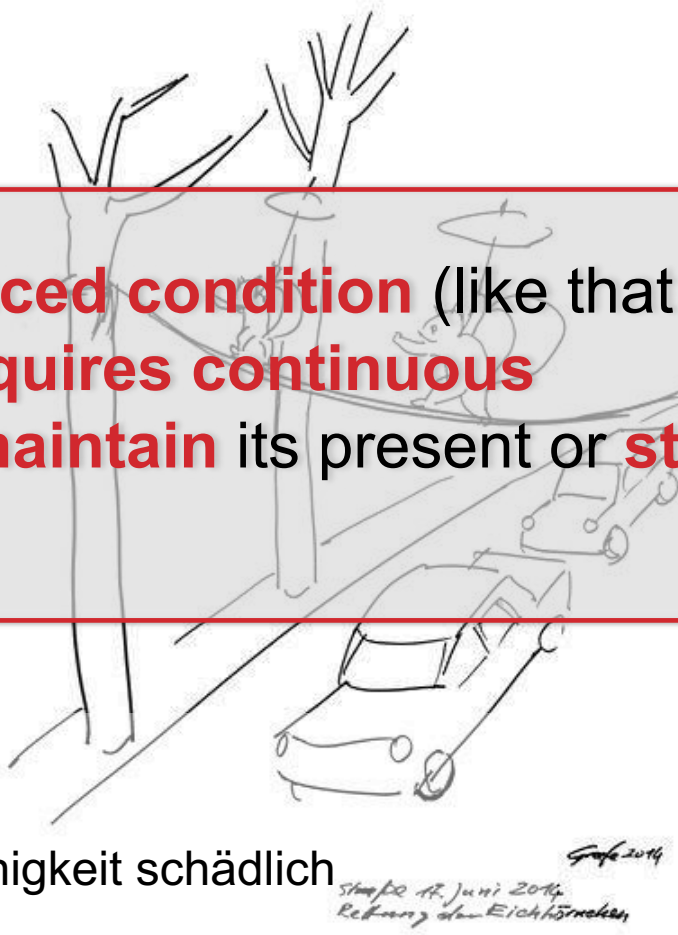
Aus "Chaos Engineering" von Casey Rosenthal, Lorin Hochstein, Aaron Blohowiak, Nora Jones, Ali Basiri

Nur Strukturierung?



Dynamic Equilibrium

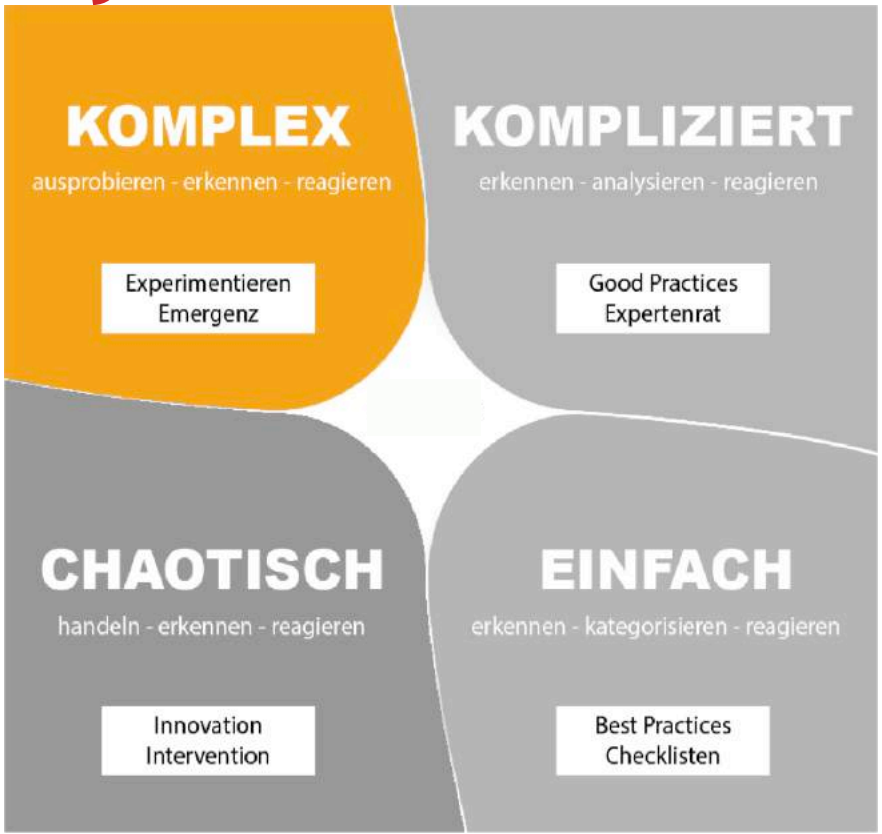
Changing yet **finely-balanced condition** (like that of a tightrope walker) which **requires continuous adjustments** in order **to maintain** its present or **stable state**.



Für eine konstant gute Lösung:

- Ist stetige Anpassung notwendig
- Ist Fixierung oder Änderungs-Zähigkeit schädlich

„Komplex“ - Cynefin



Ist Einheitlichkeit immer ein Feature?
oder macht es langsam und uninnovtiv?

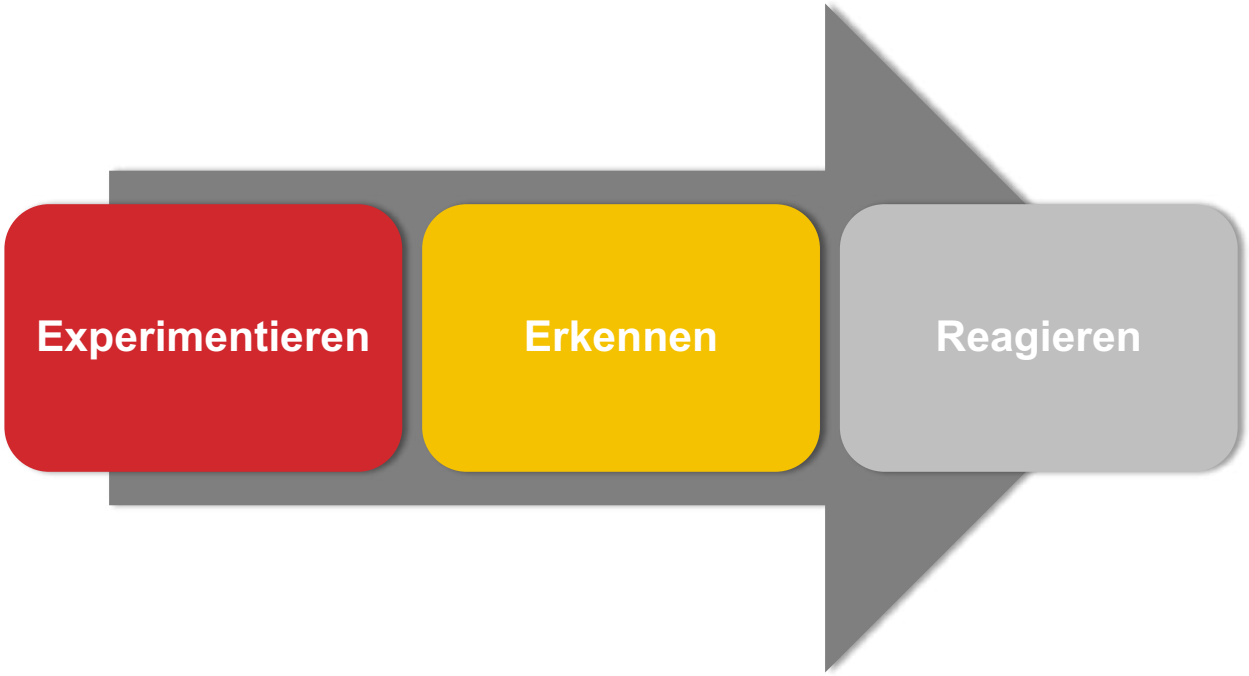
Sind Entwickler ohne Freiheiten dauerhaft motiviert?
oder leidet das Lernen, Mitdenken, Mithelfen, …?

Ist Fehlervermeidung ein gutes Ziel?
oder vermeidet man Lernchancen und Weiterentwicklung?

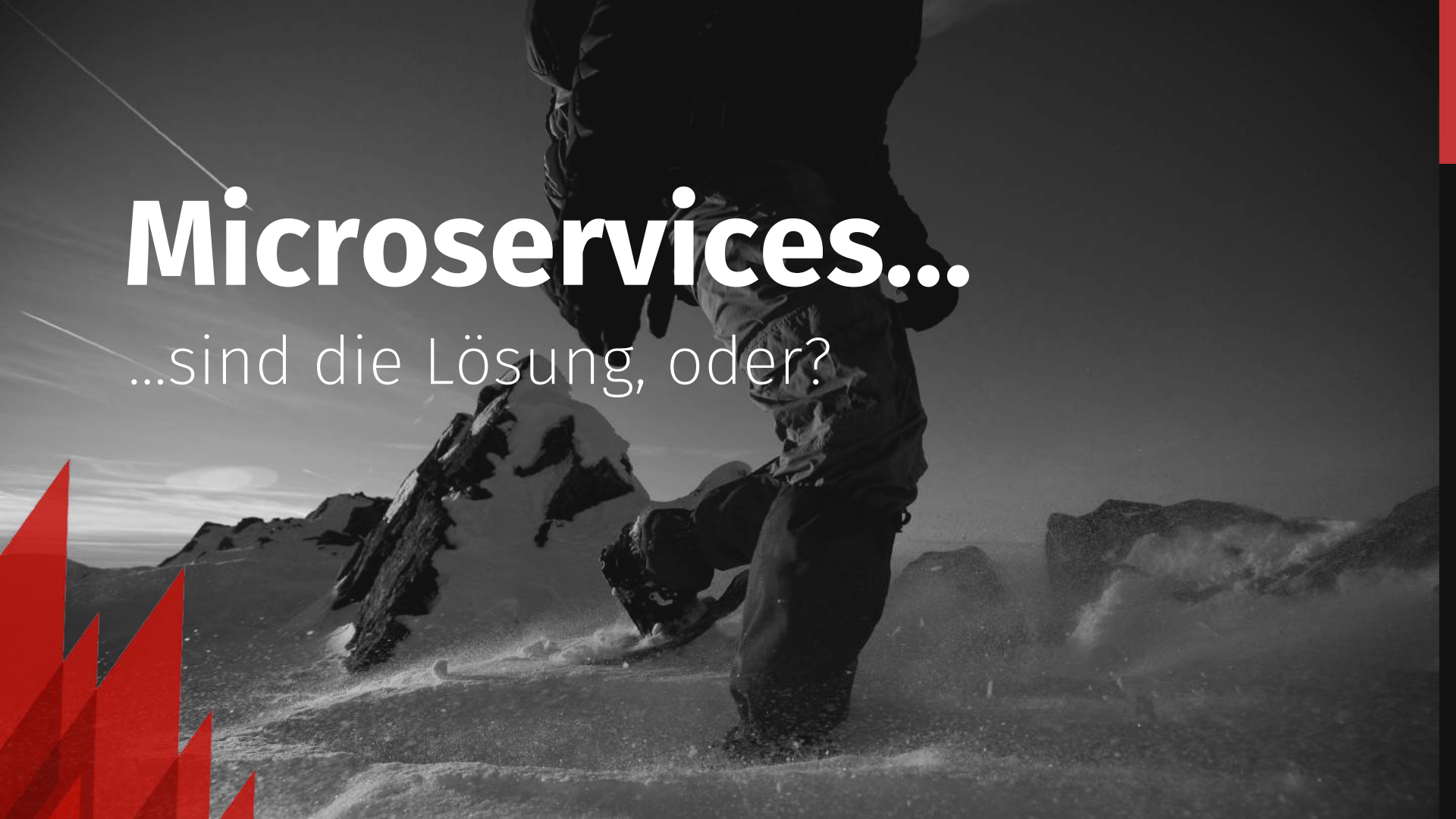
Haben Overall-Architekten tatsächlich Überblick UND Tiefe?
oder ist das unrealistisch wenn es nicht nur um Strukturierung geht?

Sollten Entscheidungsprozesse zentralisiert sein?
oder ist Wissen/Kontext nicht unbedingt dort?

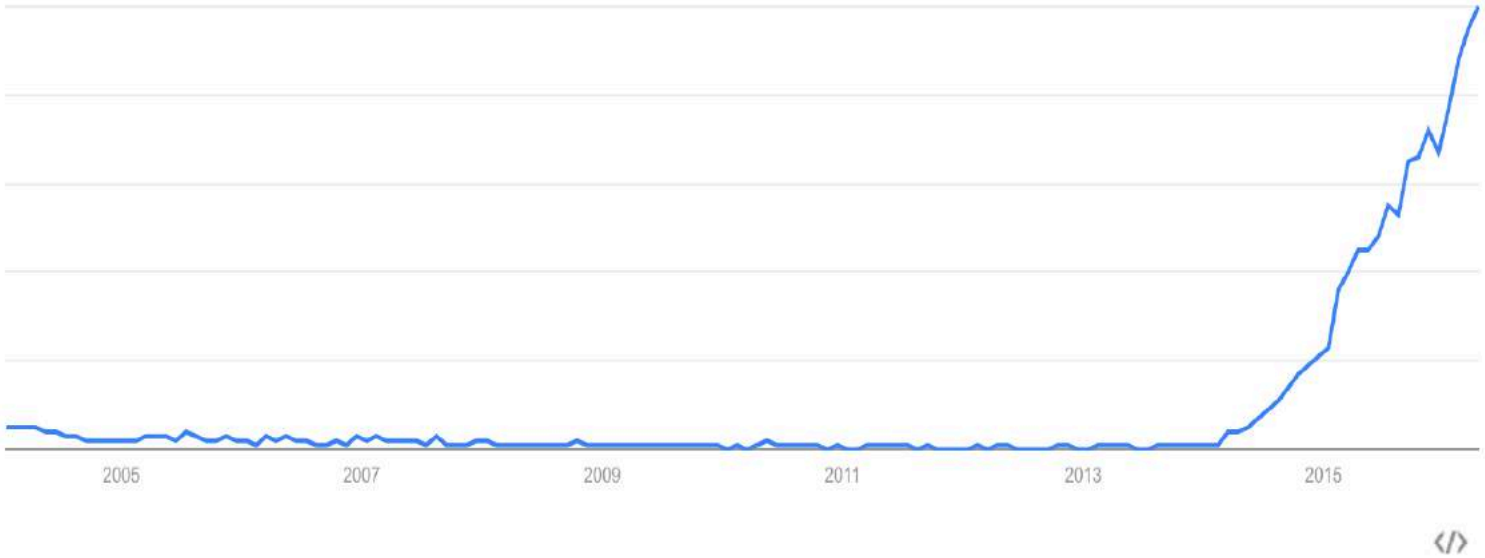
Komplexer Architekturprozess



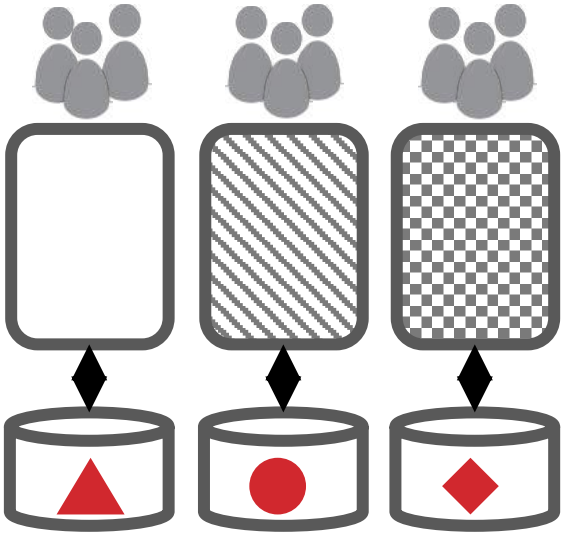
Probleme sind der einzige Weg zu einer gültigen Lösung…


Microservices...
...sind die Lösung, oder?

Microservices: Yeah!



Unabhängigkeit, Freiheit, Variation



Echte **Vertikalen**

Geringe Standardisierungstiefe

Ggf. **unterschiedliche Datenrepräsentationen** von “überlappender” Fachlichkeit

Ggf. **unterschiedliche Technologische Ausrichtung** (optimiert auf Service-Zweck)

Microservices als Herausforderung

Relevante, (übergreifende) Technologiethemen:

- Continuous Integration/Continuous Delivery
- Container/Virtualisierung/Container-Plattform
- Logging
- Service Registration & Location
- Kommuikations-Mechanismen (incl. Protokolle, Gateways, APIs, ...)
- Load Balancing & Resiliency
- Persistence: Database, NoSQL, etc.
- Security (IP Netzwerk, User-Level & Application Level Auth.)
- Größe von Microservices
- COTS Infra: Spring Cloud, Kubernetes, Service Mesh (Linkerd, Conduit, Istio, ...)
- Monitoring
- UI-Integration
- ...

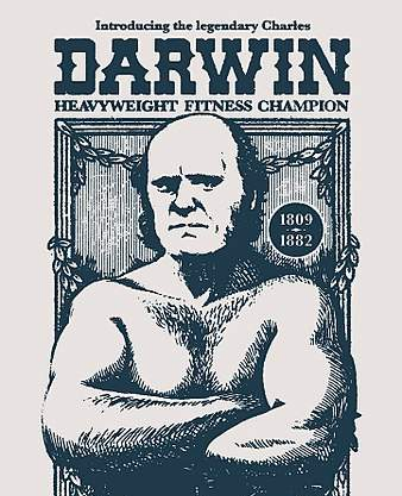
Netflix OSS: Alle Projekte

Stand März 2018, 135 Projekte.
Font-Größe: Anzahl ★
Farbe: Programmiersprache
Java JavaScript Python
Go Ruby Sonstige

Evolutionäre Architektur

Freiräume, Experimente, Zielorientierung, Anti-Zähigkeit und Transparenz/ Kommunikation sorgen für Architekturleben

Der Status Quo der momentan besten Architekturideen entwickelt sich weiter.
Dezentral und durch die Ideen vieler…

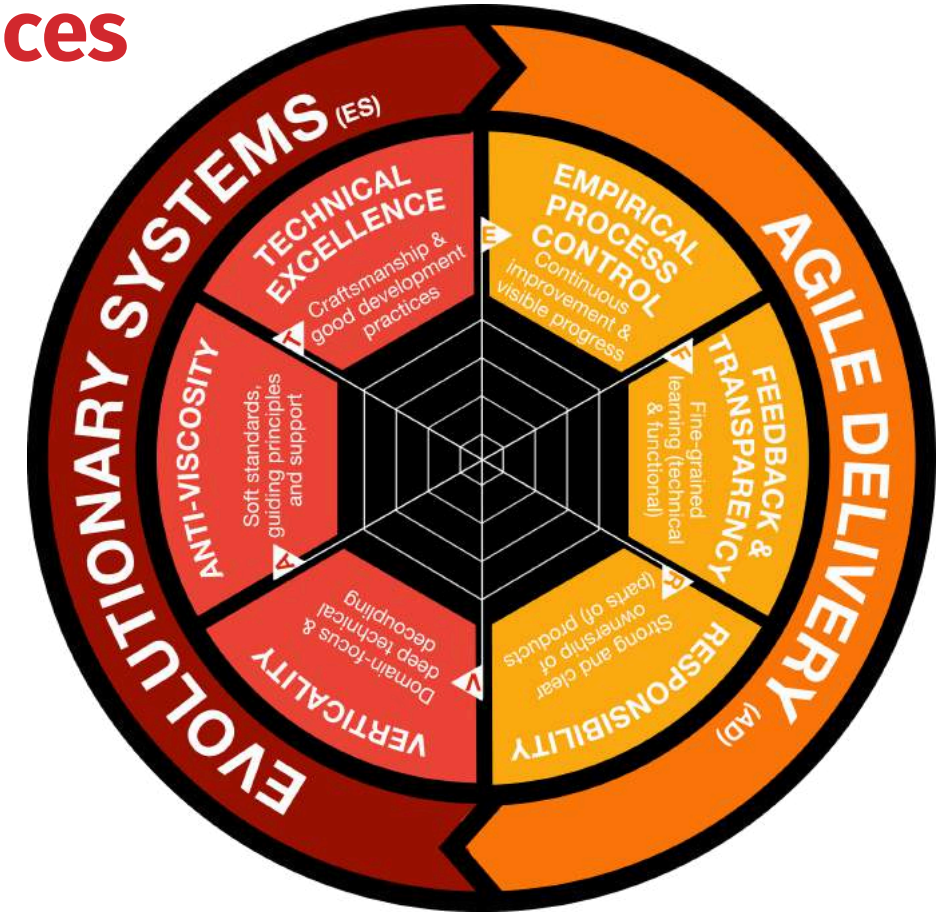


Evolution > Microservices

- **Microservices** unterstützen Vertikalität (Isolierung, schrittweise Migration)

Für wirkliche Evolution benötigt man auch:

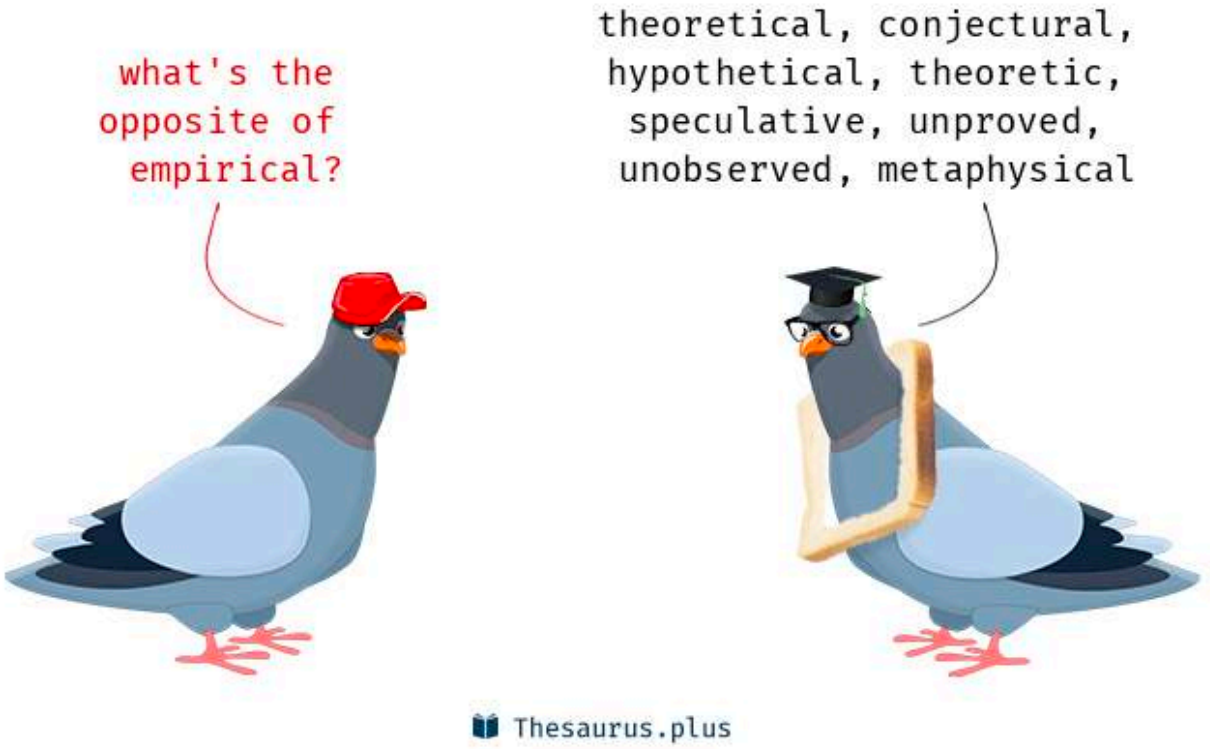
- **Experimentierkultur**: Lösung dezentral und oft challengen. Zielgerichtet!
- **Feedback**: Automatisiert, häufig und kleinteilig auf Produktziele achten.
- **Verantwortung**: Starke Eigentümerschaft von Systemteilen.
- **Soft Standards**: Unterstützung zu Good Practices, kein Durchsetzen von Regeln
- **Transparenz**: Sichtbarkeit von Weiterentwicklungen und Innovation
- ....



Neue Ziele

Für die neue Schule der Softwarearchitektur

Empirisches Vorgehen:

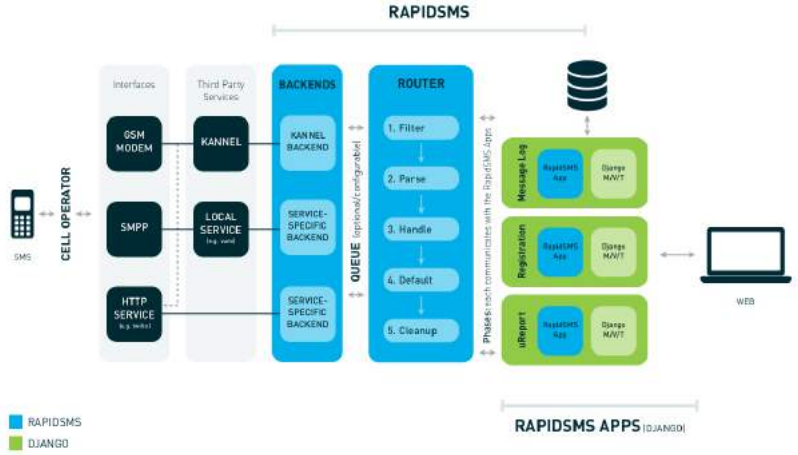


1

Empirisches Vorgehen – Fragen:

- Wie kommen wir zur **raschen Erprobung** von Ideen?
- Wie können wir (andere) Umsetzungsteams **für Ideen begeistern**?
- Sind **Technologiestandards hilfreich** und ordnend oder störend?
- Wo leben wir Standards-First statt **Eventual Consistency**?
- Wie **nutzen wir Erfolge** von Entwicklern/Teams und deren Lösungen?

Ideen und Visionen effektiv wiederlegen!

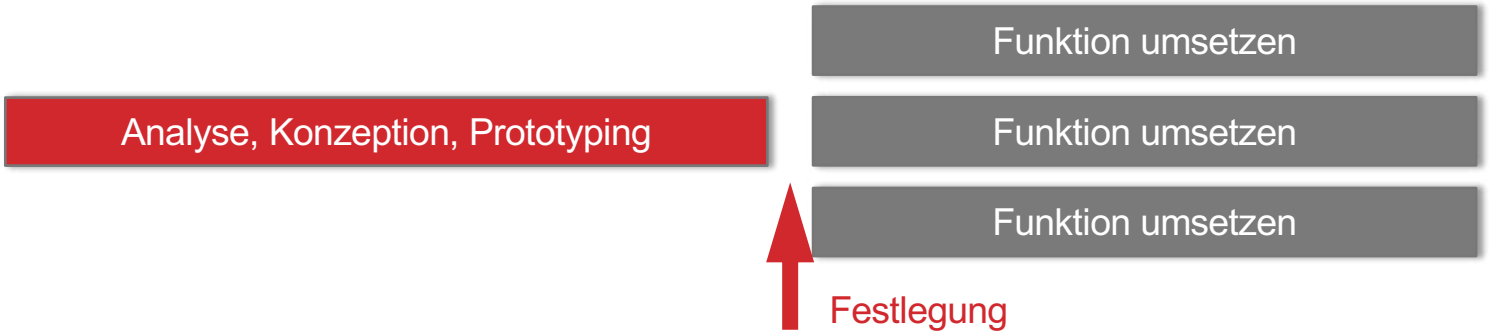


- **Beste Idee definieren** die man momentan hat
- Möglichst schnell **widerlegen**. Und zwar richtig.
  - In Risiken stechen
  - Unsicherheiten ausnutzen
  - zentrale Ziele in Gefahr bringen

} Ensprechende Anforderungen suchen

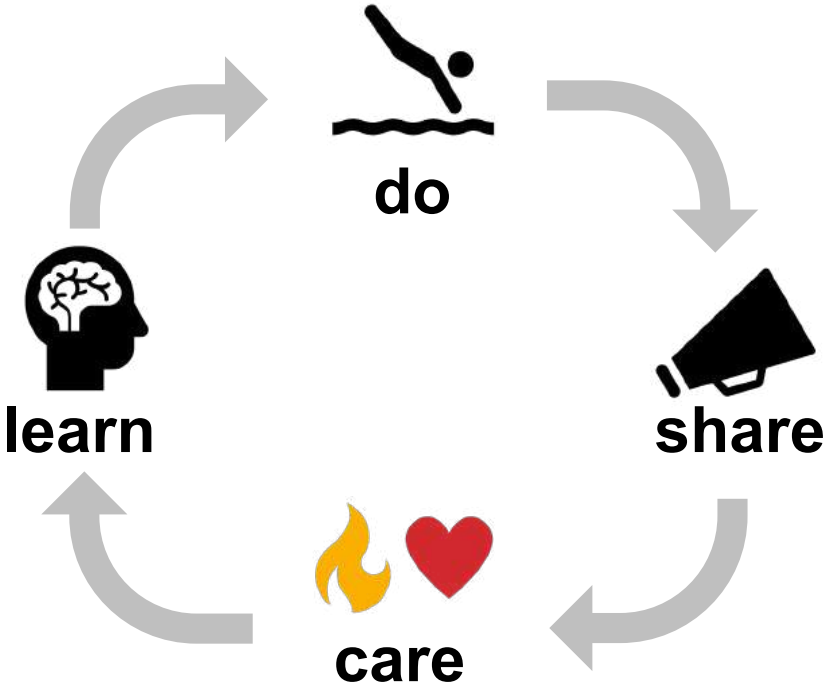
Fail safe, fail in reality

Klassische Idee von Architektur als Vorabarbeit zur „Risikominderung“



Agile Idee der frühen Konfrontation mit der Realität

Feedback und Transparenz

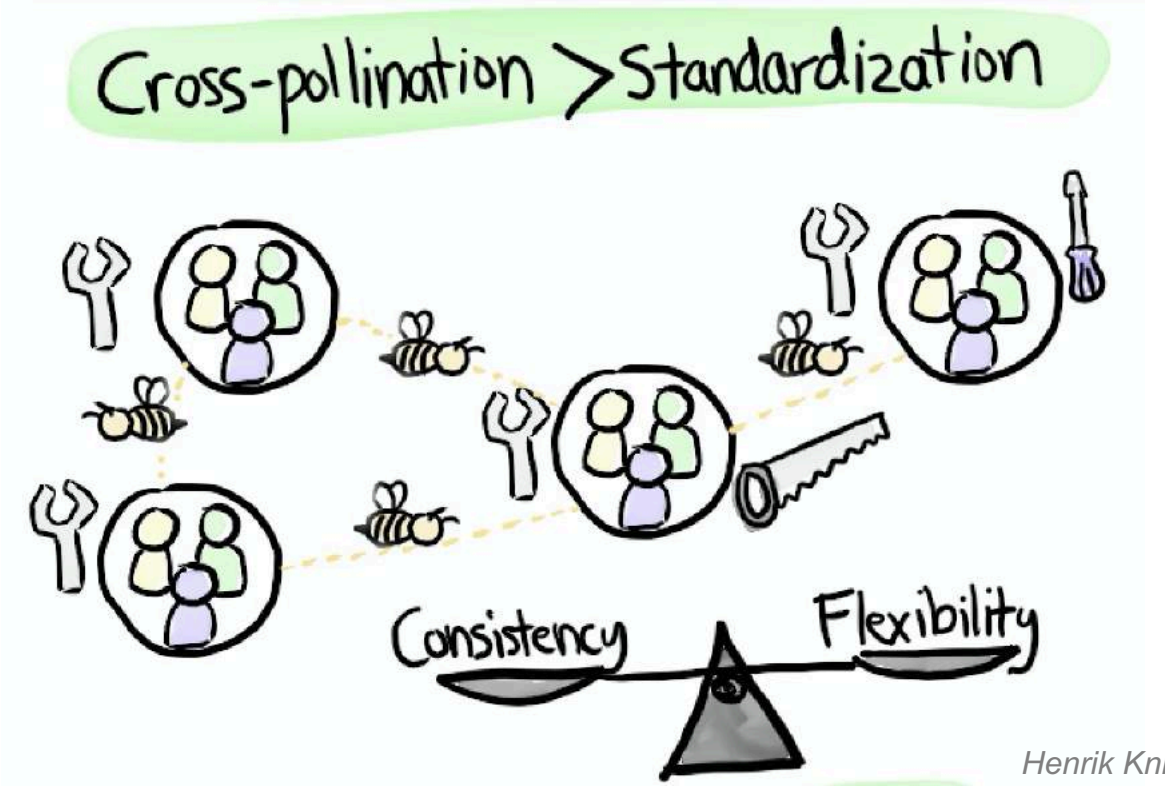


- Information Radiators
- Communities
- Techtalks
- Fachforen
- Tech-Radar
- Qualitätsziele
- Fitness Functions
- Liberating Structures

Feedback und Transparenz - Fragen

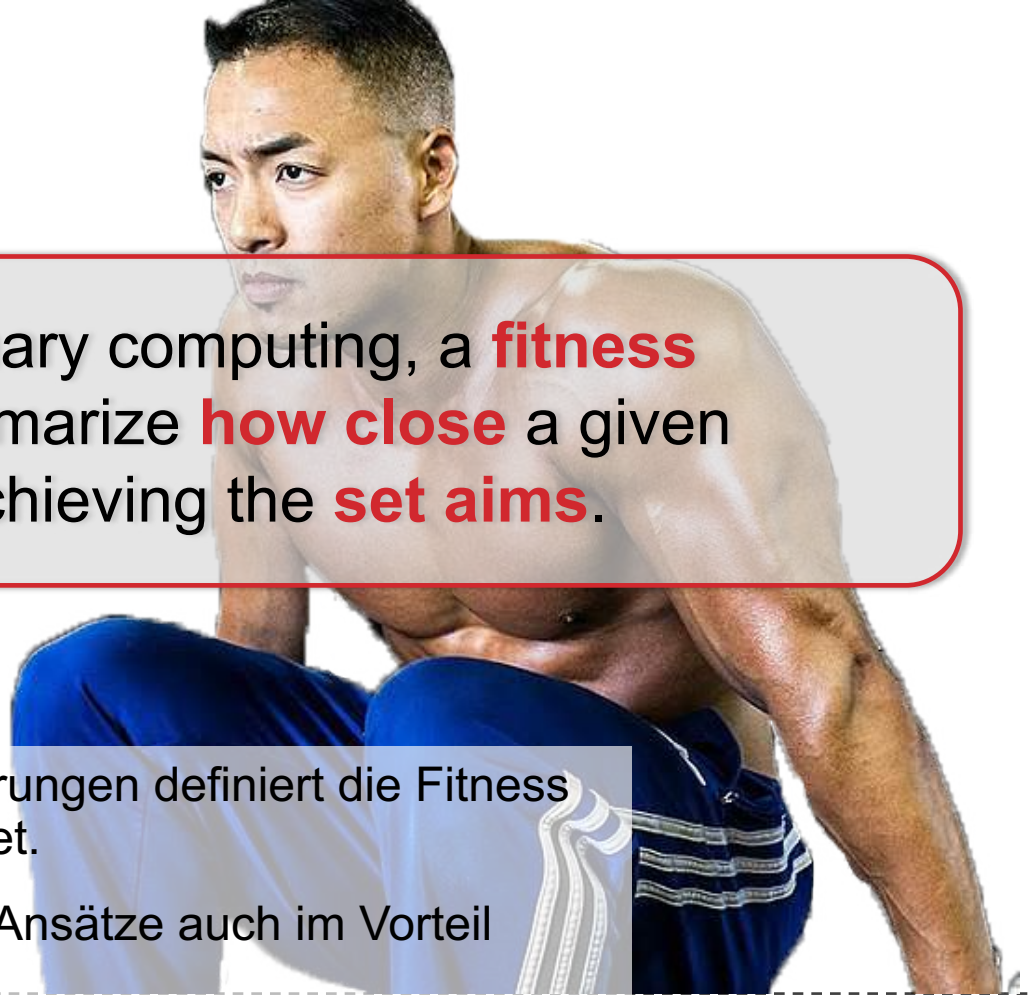
- Sind aktuelle Architekturideen und deren **Evolution sichtbar**?
- Wie **schlank** sind indirekte Kommunikation und **Dokumentation**?
- Gibt es **aktive Communituies** zu technologisch, strategischen Themen?
- Hat automatisiertes und **zeitnahes Feedback** für Entwickler hohe Priorität?
- Sind **Peer-Reviews** auf allen Lösungsebenen üblich?
- Gibt es eine **Kultur** für direktes und ehrliches **Feedback**?

Gute Ideen setzen sich durch...



Henrik Kniberg (Spotify Labs Blog)

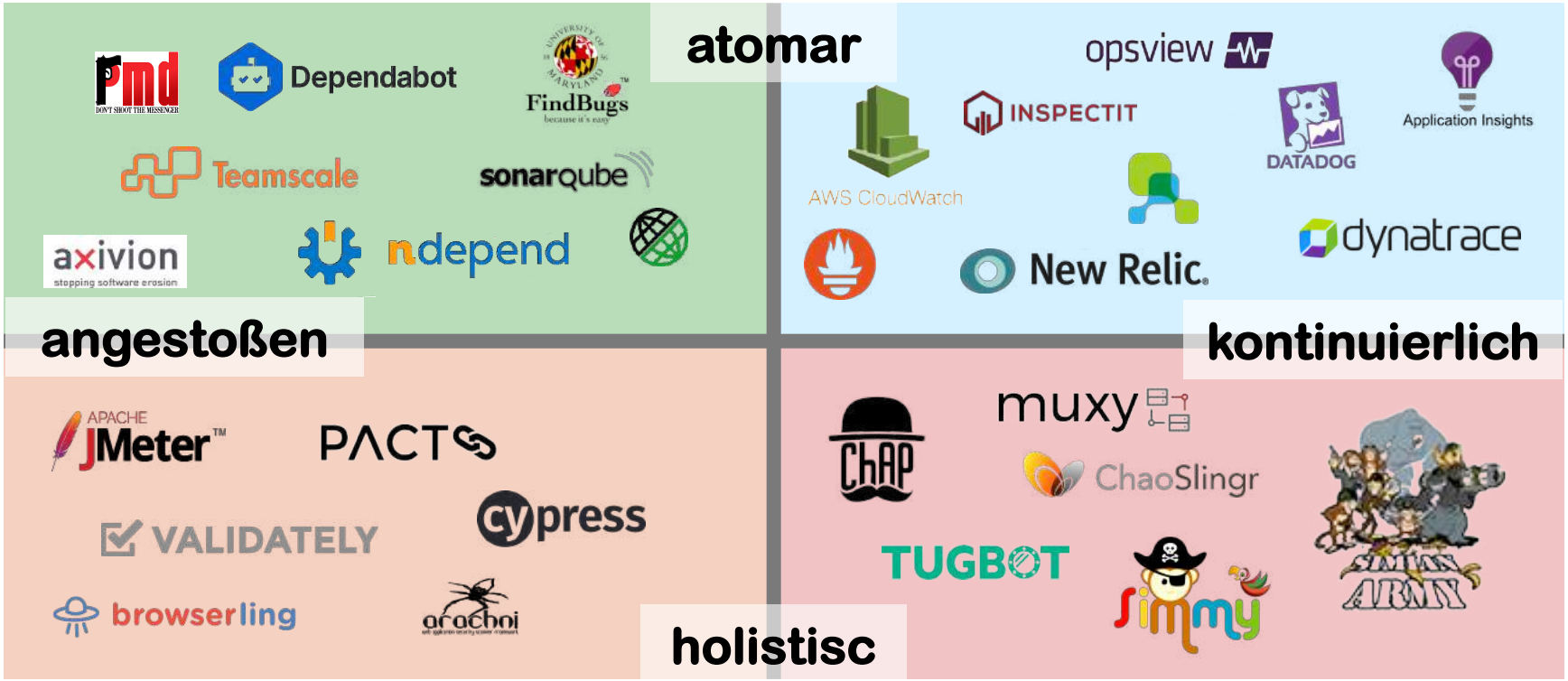
Fitness Functions



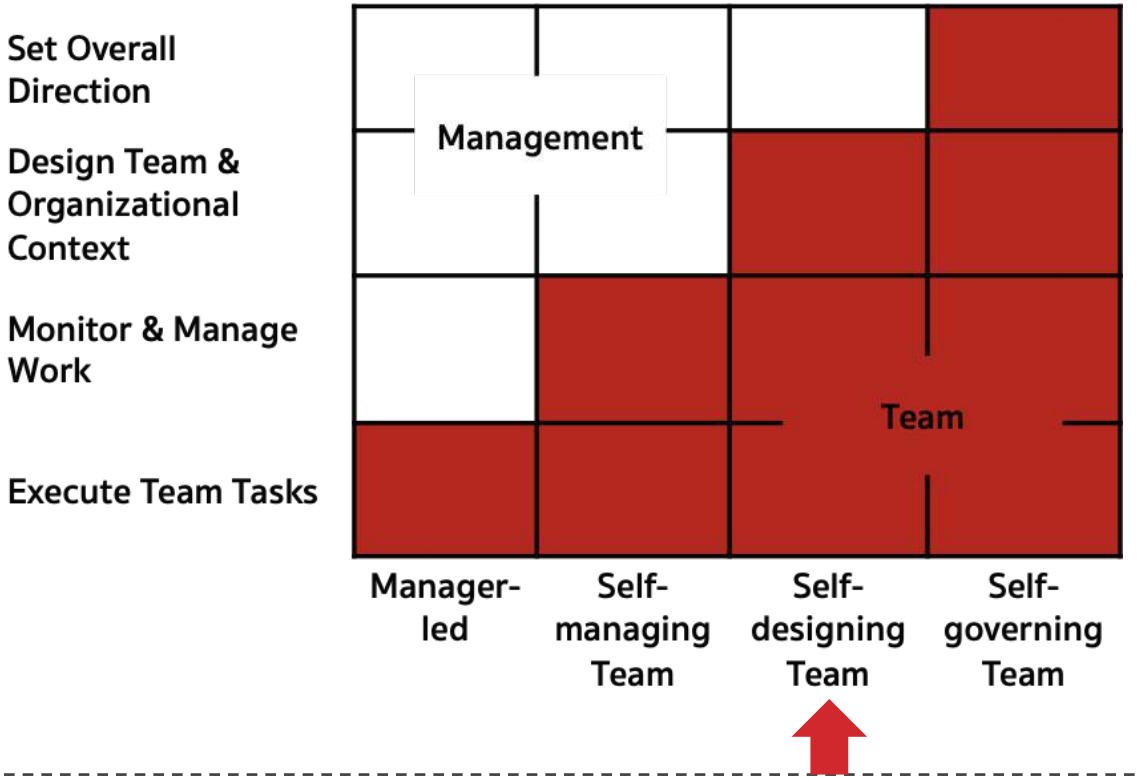
Borrowed from evolutionary computing, a **fitness function** is used to summarize **how close** a given **design solution** is **to** achieving the **set aims**.

- Auf der Suche nach Verbesserungen definiert die Fitness Function was „besser“ bedeutet.
- Gleich „gute“ aber einfachere Ansätze auch im Vorteil

Fitness Function Quadrant - Tooling



Verantwortung in Umsetzungsteams



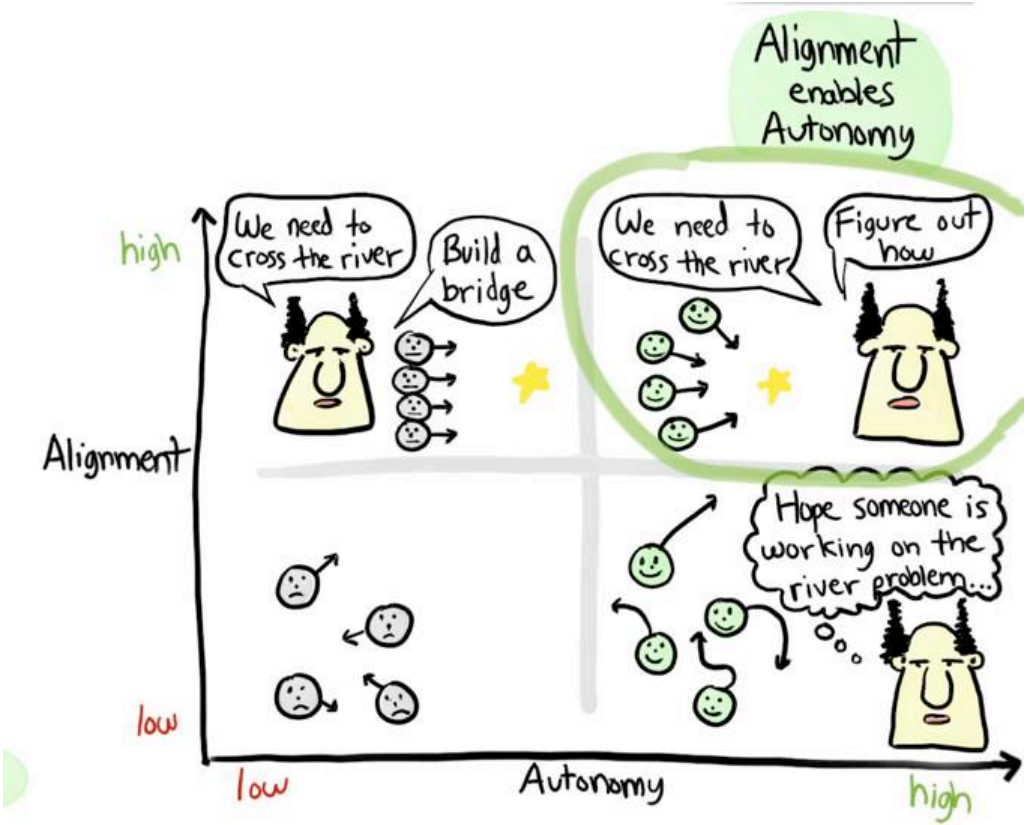
Richard Hackman's Authority Model

3

Verantwortung in Umsetzungsteams - Fragen

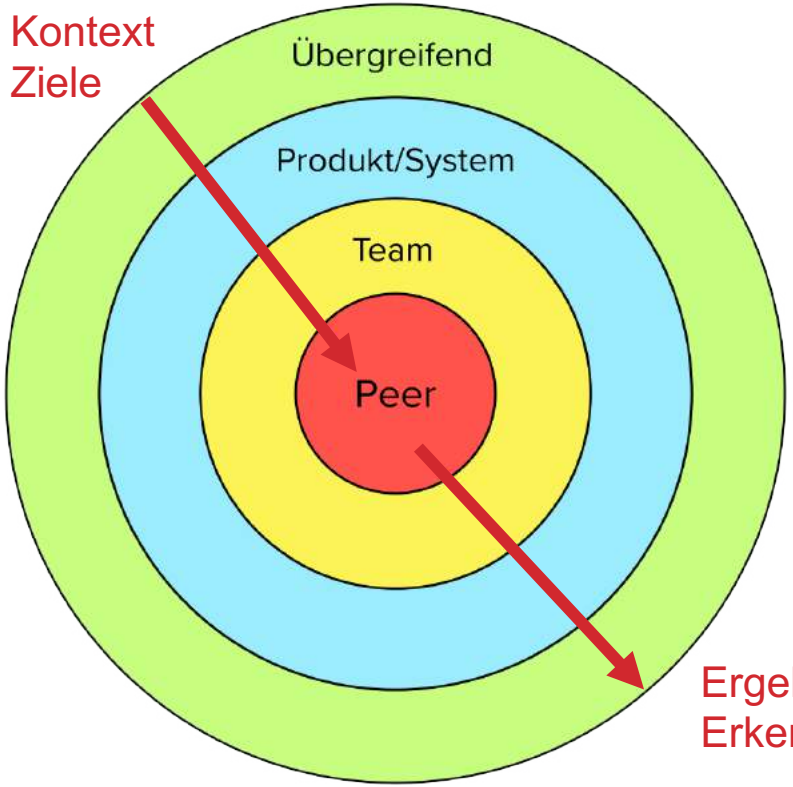
- Sind **wenige Teams** am **Lebenszyklus** von Komponenten/Services beteiligt?
- Ist das **Softwareprodukt** der wichtigste **Maßstab** für den „Erfolg“ von Teams?
- Sind **strategische** und qualitative **Ziele** auf Entwicklungssebene präsent?
- Sind Entwickler **motiviert** und **zielstrebig** wenn es um Probleme geht?
- Wie **klar** ist der **Verantwortungsbereich** der Teams?

Spotify – Autonomie von Teams



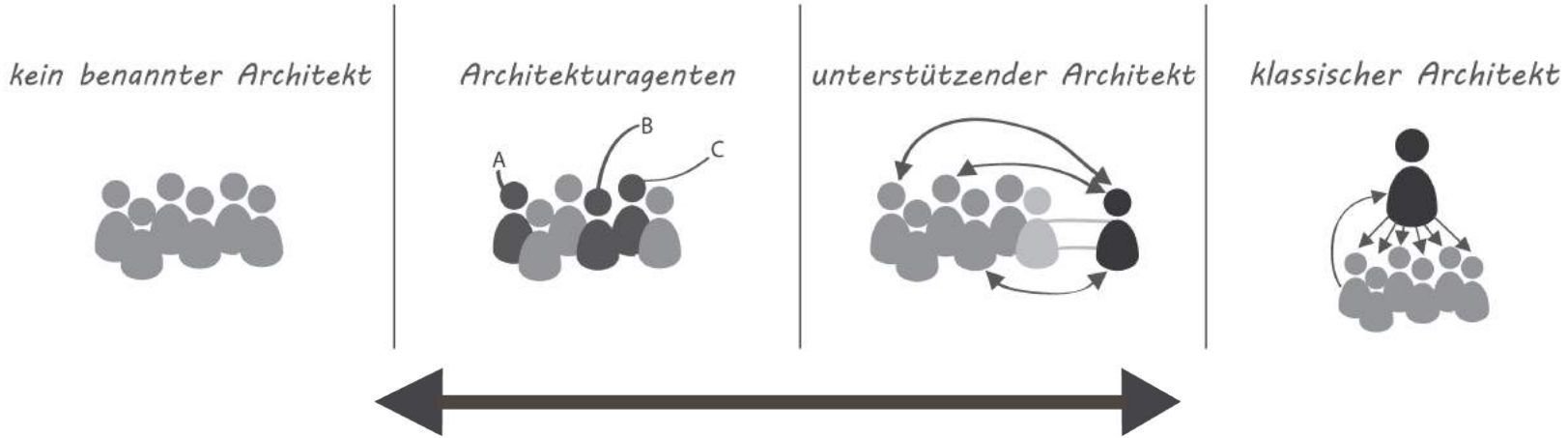
Henrik Kniberg
(Spotify Labs Blog)

Transparenz und Steuerung…

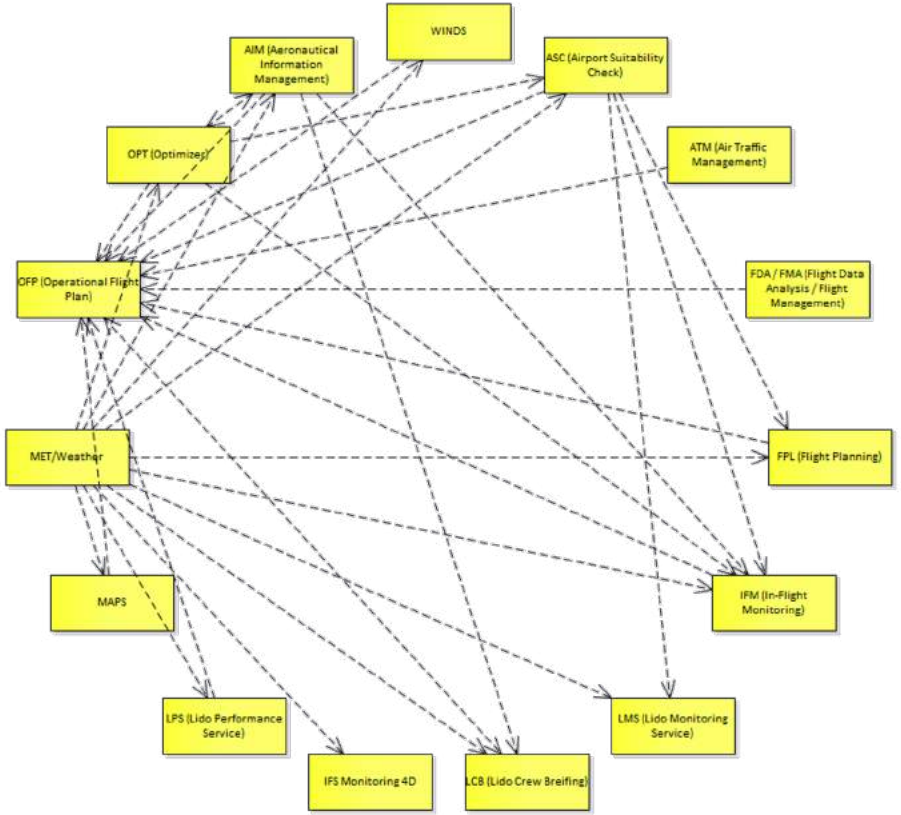


- **Transparenz** auf allen Ebenen
- Zwiebelschalen **filtern**
- Zwiebelschalen sind **keine Barrieren**
- **Keine Exklusivinformation** für äußere Bereiche
- Steuerung über **Kontext** und **Ziele**

Rollenmodell für Architekten kann variieren



Team-Abhängigkeiten

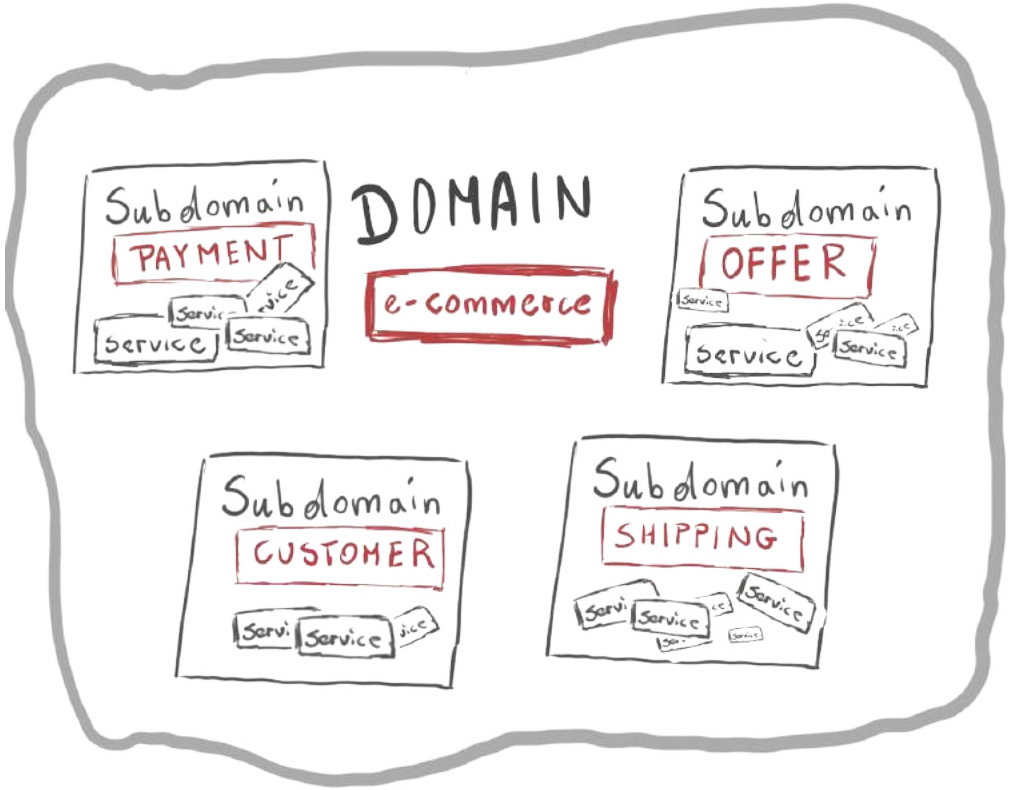


Schnittstellen!?
Daten!?
Releases!?
Arbeitsweise!?

Team Dependency Tracking

Team name	Depends on Team	Type (blocking/slowing/ok)	Cross-streams (Yes/No)	Short description of dependency (artifacts, approvals, other)
Analytics	DBA	slowing	NA	sometimes we need data exports that can take several days because DBA team is busy
Self Heal	Mobile	slowing	NA	mainly synchronize changes to data being used for recommendations, but even small changes take time beause Mobile team is constantly overloaded
Self Heal	Ops	blocking	NA	for deploying changes
Self Heal	DBA	blocking	NA	for reviewing schema changes
Self Heal	InfoSec	blocking	NA	for changing firewall settings and other configurations
QA	InfoSec	blocking	NA	when we suspect there might a security violation but it takes very long to get an answer and be able to create the final test report
DBA	Ops	ok	NA	provisioning more storage
IoT	Networking	slowing	NA	changes in networking rules can take a few days to become active
IoT	InfoSec	blocking	NA	sometimes new features are ready but takes weeks to validate security
Sensors	Networking	ok	NA	
Sensors	Mobile	slowing	NA	can't update firmware until mobile apps use the new version
Support	Ops	blocking	NA	we need to wait for Ops team for infra related tickets
Support	Mobile	blocking	NA	we need to wait for Mbile team for mobile related tickets
Support	Self Heal	blocking	NA	we need to wait for Self Heal team for recommendations related tickets

Vertikalität



- Domänenorientierung
- Klare Produktgrenzen
- Tiefe Entkopplung
- Private Datenmodelle
- Interaktionsregeln
- ...

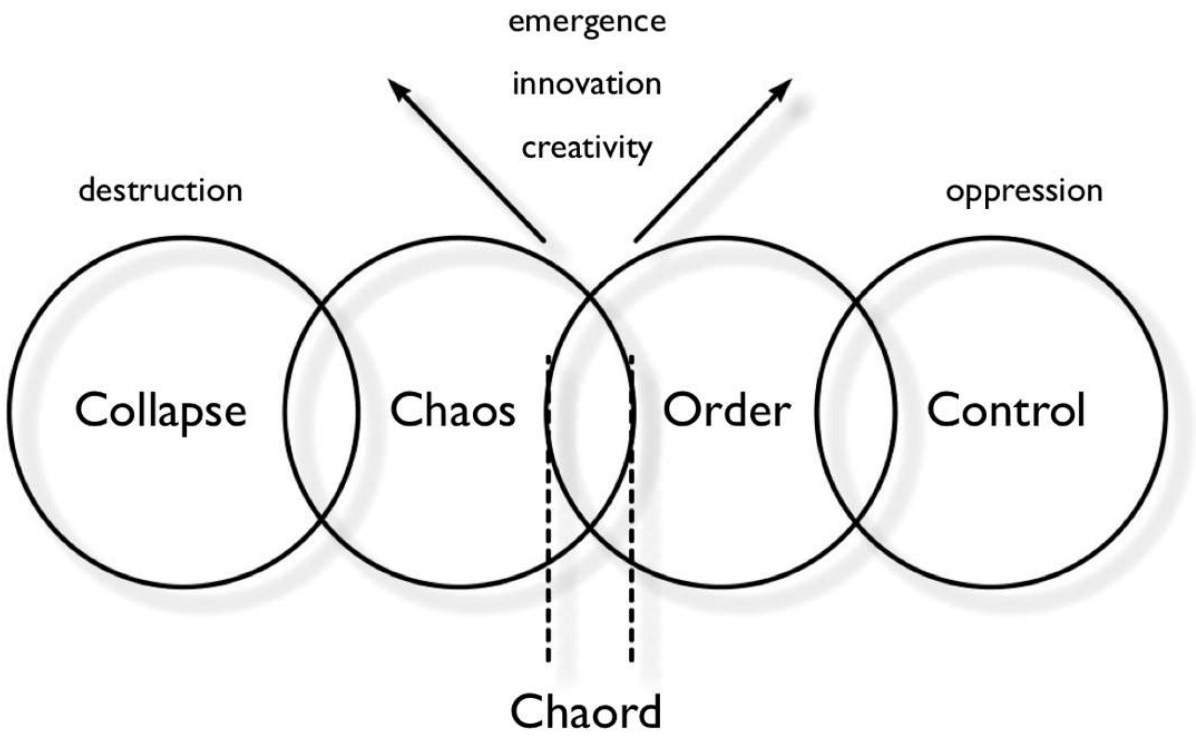
4

Vertikalität - Fragen

- Wie gut mappt die **Domäne** auf einzelne Applikationen/**Services**?
- Sind **Abhängigkeiten** zwischen Services **gering**?
- Sind übergreifende **technologische Aspekte wenig einschränkend**?
- Sind die **Interaktionsprinzipien** zwischen Systemteilen **klar** definiert?
- Ist die **Ursache** für **Verzug oder techn. Probleme** immer in **einem Umsetzungsteam** selbst zu finden?

4

Architektonische Emergenz



© Simon Robinson From *Holonomics: Business Where People and Planet Matter*, Floris Books, 2014

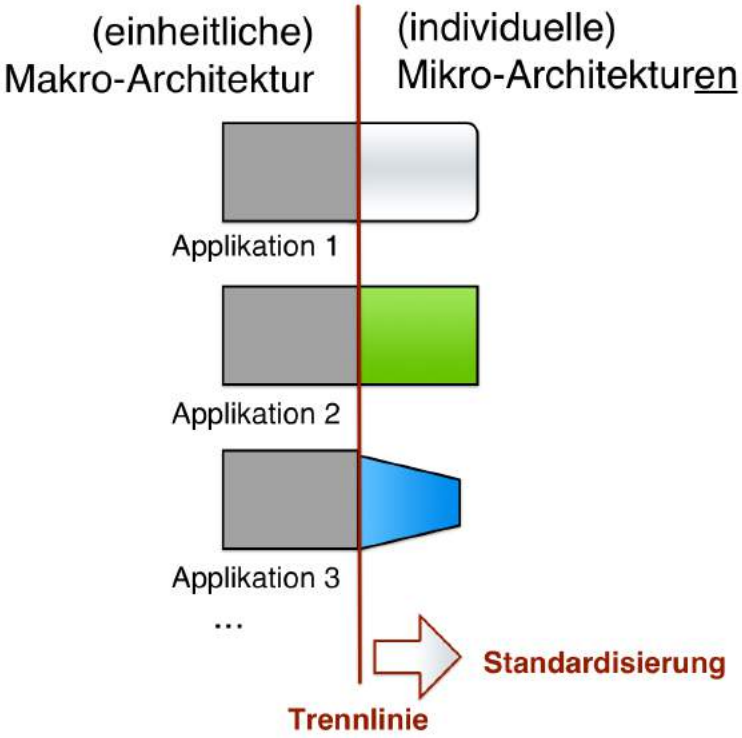
5

Architektonische Emergenz - Fragen

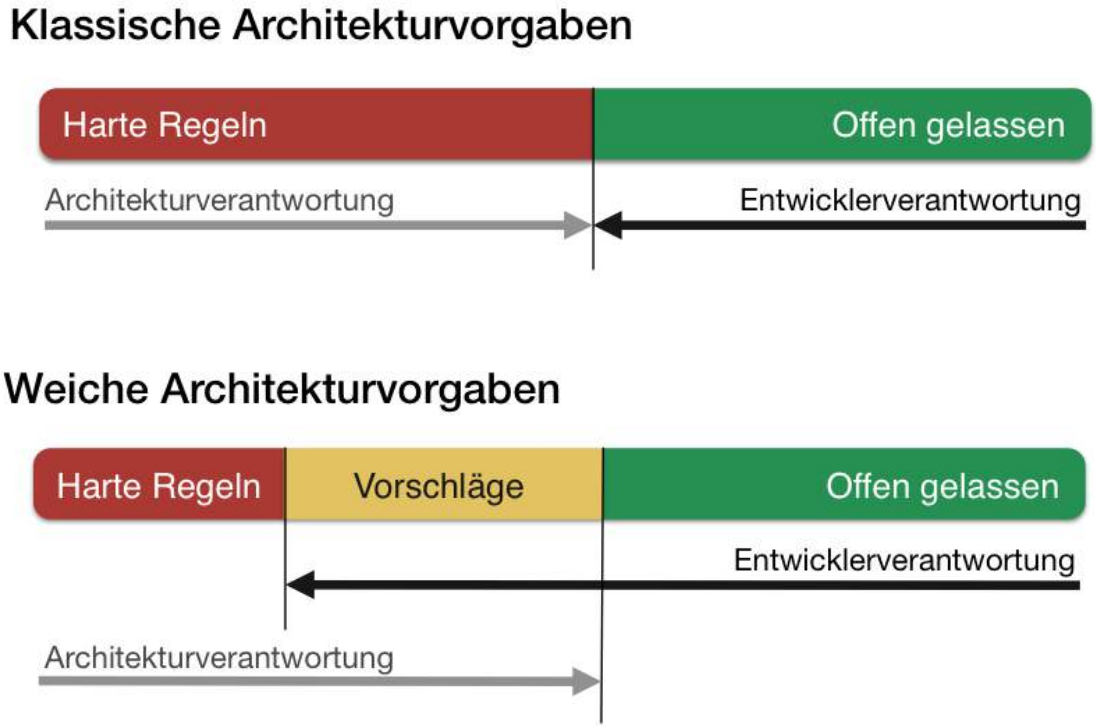
- Gibt es mehr **Vorschläge und Defaults** als harte Regeln?
- Würden faule Leute den **Architekturideen folgen** oder sie umgehen?
- Gibt es verstandene, und breit akzeptierte **Architekturprinzipien**?
- Sind **Architekturrollen** mehr „**Kurator**“ als „Terminator“?
- Werden **Innovationspotentiale** auch **bottom-up** getrieben?
- Werden **Fehlschläge** offen und gern **kommuniziert**?

5

Standardisierungstiefe...



Vorschläge statt Vorschriften



Transparenz & Richtung

Zalando Tech Radar — 2018.01

Frameworks

ADOPT
1. Akka (Scala)
2. Node.js
3. OpenAPI (Swagger)
4. Play (Scala)
5. ReactJS
6. RxJava (Android)
7. scikit-learn
8. Spring

ASSESS
15. Akka-Http
16. Aurelia
17. Ember.js
18. gRPC
19. Http4s
20. Redux
21. Vert.x
22. Vue.js

TRIAL
9. Angular
10. AspectJ
11. Camel
12. Camunda
13. OpenNLP
14. Thymeleaf

HOLD
23. Activiti
24. AngularJS 1.x
25. BackboneJS
26. Drools
27. Spray

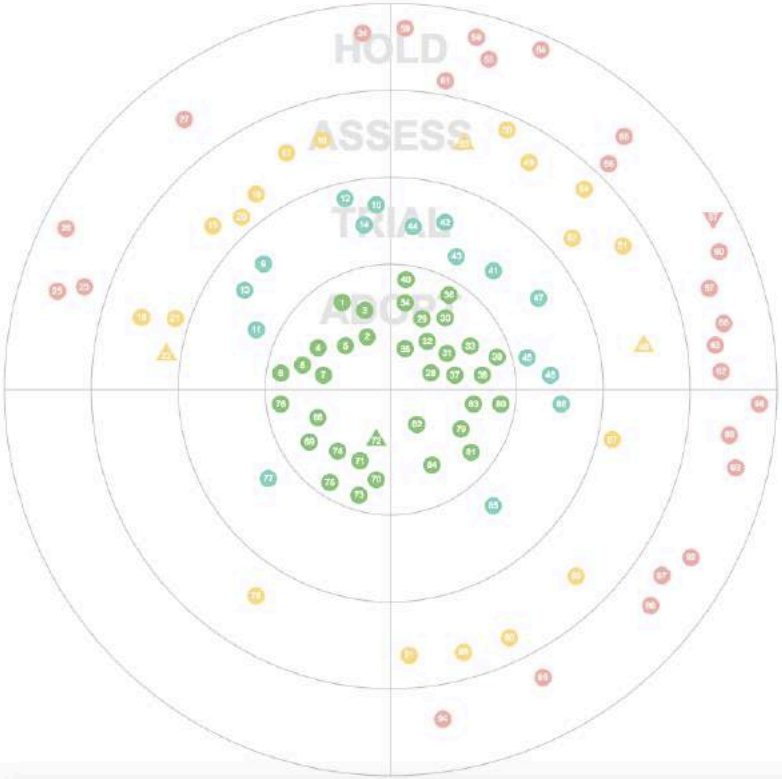
Infrastructure

ADOPT
68. Docker
69. HAProxy
70. Hystrix
71. Jetty
72. Kubernetes
73. Nginx
74. STUPS
75. Tomcat
76. ZMON

ASSESS
78. AWS Lambda

HOLD

TRIAL
77. Undertow



Data Management

ADOPT
28. AWS EMR
29. AWS S3
30. AWS SNS
31. AWS SQS
32. Cassandra
33. Elasticsearch
34. etcd
35. Kafka
36. Nakadi
37. PostgreSQL
38. Redis
39. Solr
40. Spark

ASSESS
48. Airflow
49. AWS Kinesis
50. Consul
51. Google Bigtable
52. Hadoop
53. RocksDB
54. YARN

HOLD
55. ActiveMQ
56. Aerospike
57. CouchBase
58. Esper
59. HBase
60. HornetQ
61. Memcached
62. MongoDB
63. MySQL
64. Oracle DB
65. RabbitMQ
66. Riak
67. ZooKeeper

TRIAL
41. AWS Data Pipeline
42. AWS DynamoDB
43. Flink
44. Google BigQuery
45. HDFS
46. KairosDB
47. Presto

Languages

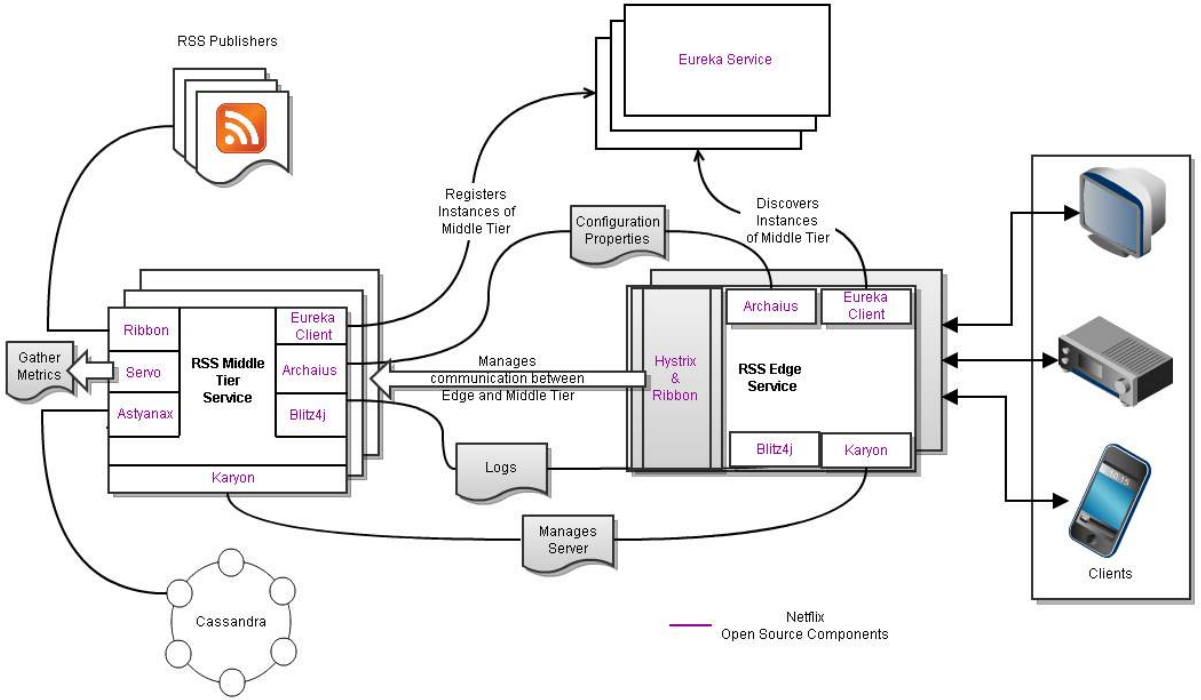
ADOPT
79. Go
80. Java
81. JavaScript
82. Python
83. Scala
84. Swift

ASSESS
87. Elm
88. Haskell
89. Kotlin
90. R
91. Rust

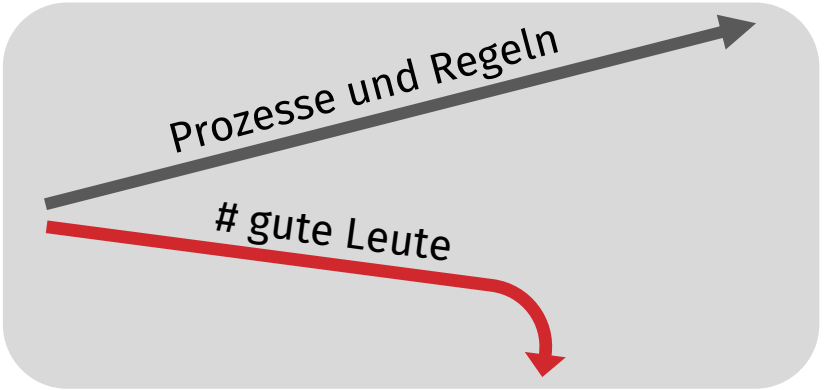
HOLD
92. .NET languages
93. C languages
94. CoffeeScript
95. Erlang
96. Groovy
97. Perl
98. PHP
99. Ruby

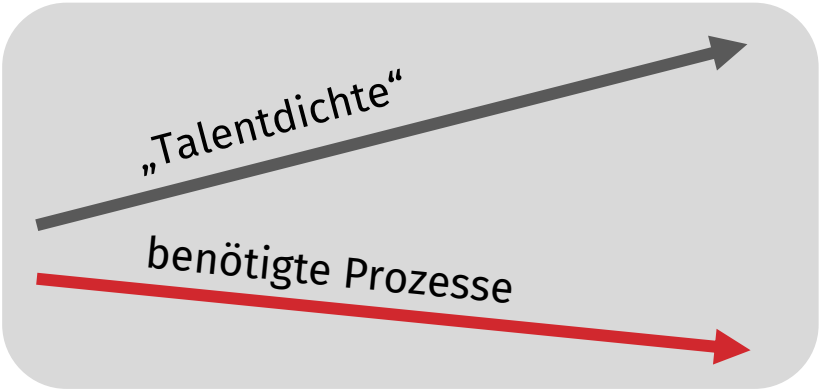
TRIAL
85. Clojure
86. TypeScript

Standard-Blueprint von Netflix



Technische Exzellenz





Quelle: Reed Hastings (Netflix CEO)

6

Technische Exzellenz - Fragen

- Sind **persönliche Weiterentwicklung** und entsprechende Freiräume normal?
- Ist **Wissen und Können wichtiger** als Position und Politik?
- Sind **Qualitätsbewusstsein** und Produktfokus als Haltung **breit** gestreut?
- Haben **Trends und Erkenntnisse** eine große Bühne?
- Gibt es wenige „**undone-departments**“?

6

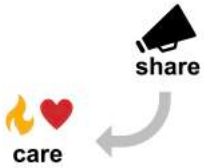
Zusammenfassung

Was ist die neue Schule?

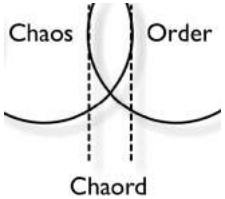
Die wichtigsten Aspekte

Fokus auf Empirie

Vertikalität & Unabhängigkeit



Transparenz & Feedback



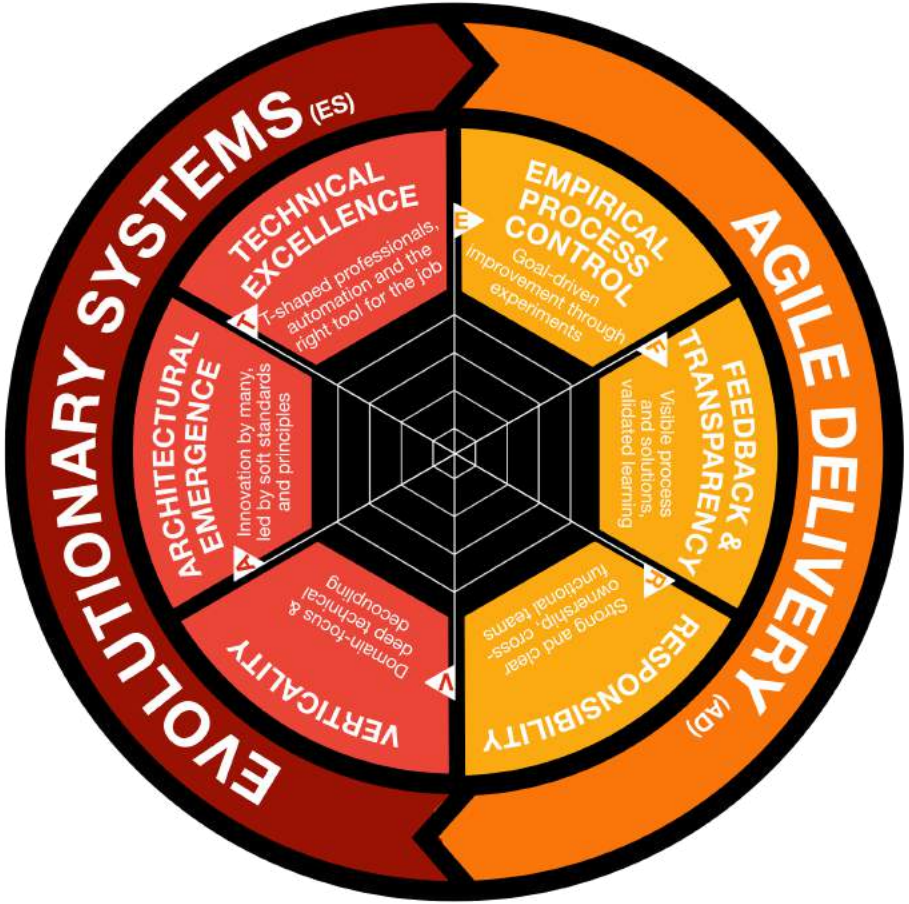
Emergenz durch „Chaord“

Verantwortung bei Umsetzung

Technische Exzellenz

Das ADES-Framework

- **Wichtige Eigenschaften** von agilen Organisationen
- **6 Lerngebiete**. Enthalten Beispiel-Techniken für Experimente
- **Iterative Optimierung** der Lerngebiete
- Wiederholte **Selbsteinschätzung** (Spinnennetz)
- **KEIN** Maturity Modell
- **KEIN** Skalierungs-Framework



http://ades-framework.org/

Experten & Best-Practices

Bei gut abgehangenen komplizierten oder einfachen Problemen ok
Ansonsten: Verantwortung und Innovation zentralisiert, In Komplexität ineffizient

Standardisierung

Bei stabilen, erprobten Lösungen, Rahmenbedingungen, Schnittstellen (APIs)
Ansonsten: Kann Innovation, Autonomie und Verantowrtungsübernahme behindern

Wiederverwendung

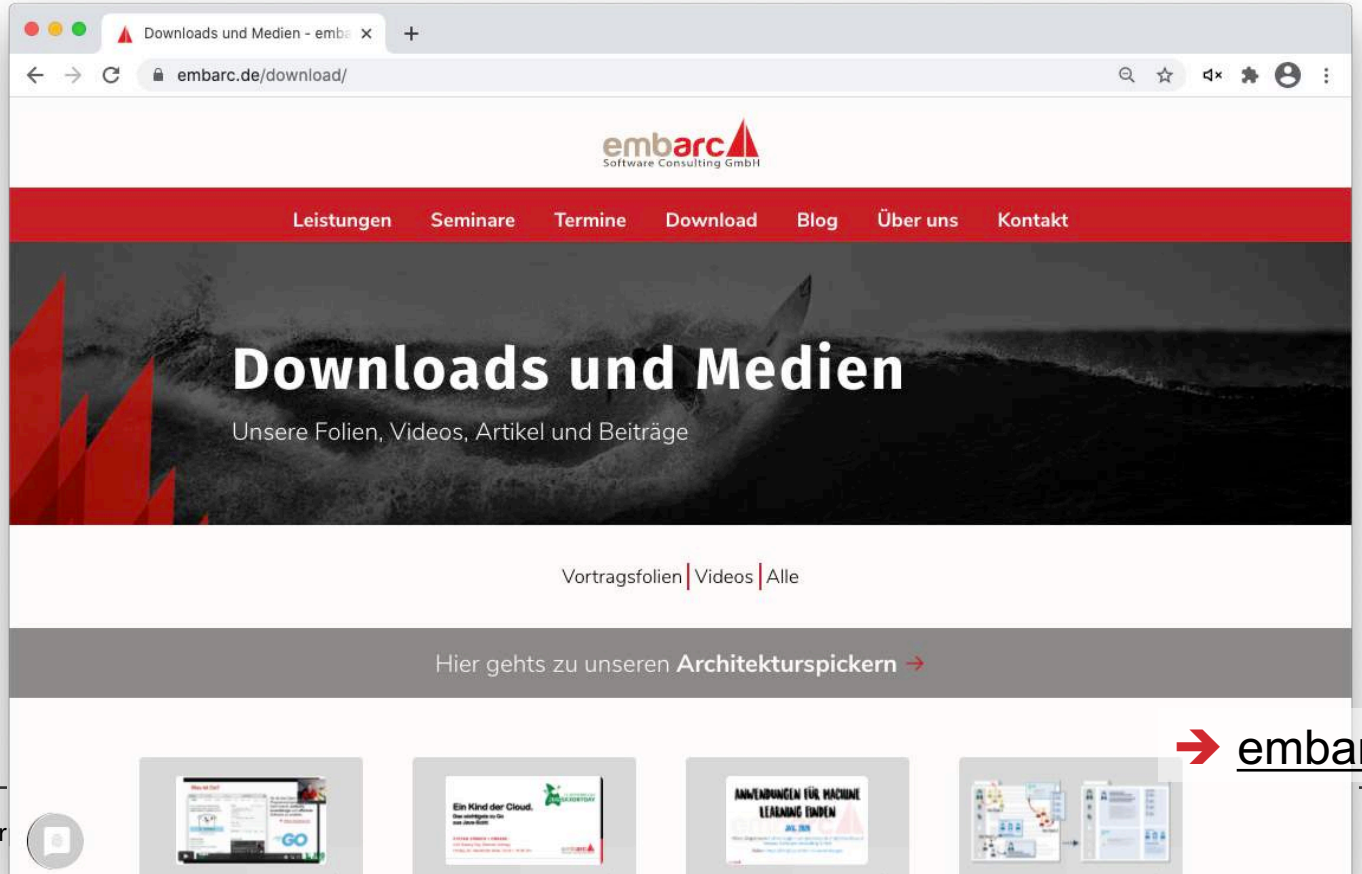
Evtl. bei zentraler Anforderungsstellung
Ansonsten: höhere Varianz, Komplexität, Abstimmungsaufwand

Governance Boards o.Ä.

Evtl. bei nicht testbaren regulatorischen Themen als Kompromiss
Ansonsten: Flaschenhals, Negativkommunikation, Verantwortungsstreuung

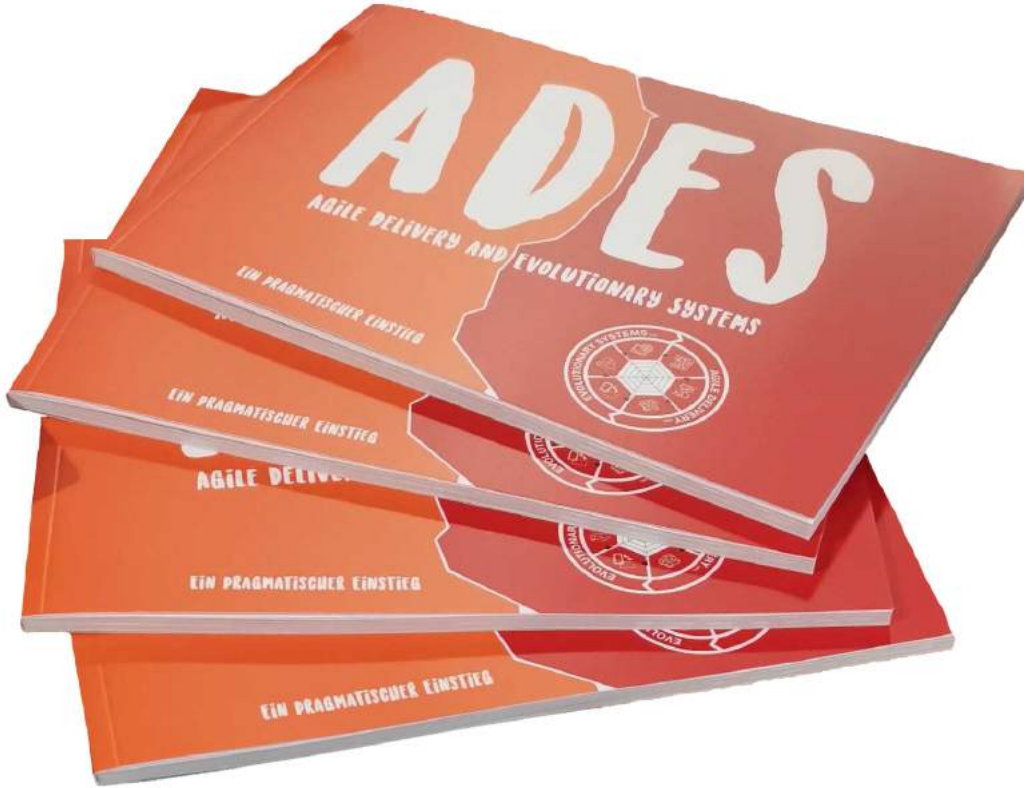

Danke.
Jegliche Fragen sind willkommen!
stefan.toth@embarc.de
@st_toth
xing.to/sto
embarc
Software Consulting GmbH

Folien von heute als PDF zum Download



➔ embarc.de/download/

Agile Organisationsarchitektur



ADES Framework

HOME WHY ADES? ADES FRAMEWORK FAQ CONTACT

http://ades-framework.org/