Emergente Praktiken in der Softwarearchitektur

Falk Sippach

BERLIN EXPERT DAYS 2026

23.09.2026

0

Abstract

Emergente Praktiken in der Softwarearchitektur

In komplexen, agilen Projekten lassen sich Architekturentscheidungen nicht mehr vollständig im Voraus planen. Klassische Big-Design-Up-Front-Ansätze führen häufig zu Überdokumentation, veralteten Modellen und unflexiblen Strukturen. Gleichzeitig stehen viele Architekt:innen vor der Herausforderung, Architektur nicht mehr „von oben" vorzugeben, sondern sie gemeinsam mit Teams entstehen zu lassen – oft ohne klare Orientierung, welche Praktiken dabei helfen. Ohne geeignete Methoden droht entweder Chaos durch fehlende Leitplanken oder Stillstand durch zu starre Vorgaben.

Architecture Decision Records, evolutionäre Architektur, Fitness Functions, Walking Skeletons, letzter vernünftiger Moment oder kollaborative Modellierungssessions können bei einer kontinuierlichen, adaptiven Architekturarbeit unterstützen. Sie verbinden agile Prinzipien mit bewährten Vorgehensmustern und schaffen damit die Basis für eine lebendige, lernende und qualitativ hochwertige Architektur. Diese Ansätze fördern Emergenz, ohne die Architekturqualität dem Zufall zu überlassen – und helfen Teams, Architektur bewusst zu gestalten, anstatt sie einfach entstehen zu lassen. In dieser Session diskutieren wir, welche emergenten Praktiken für euch relevant sind und wie ihr sie in euren Projekten konkret umsetzen könnt.

1

Falk Sippach

- Softwarearchitekt, Berater, Trainer bei embarc
- früher bei Orientation in Objects (OIO), Trivadis

Schwerpunkte

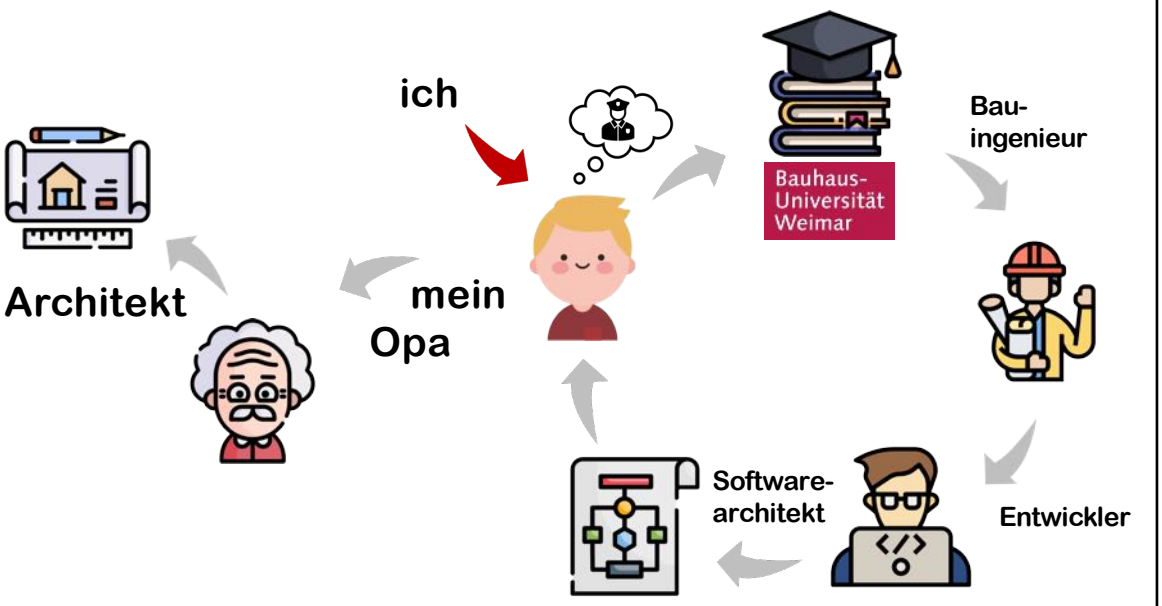
- Architekturberatung und -bewertung
- Cloud- und Java-Technologien

embarc

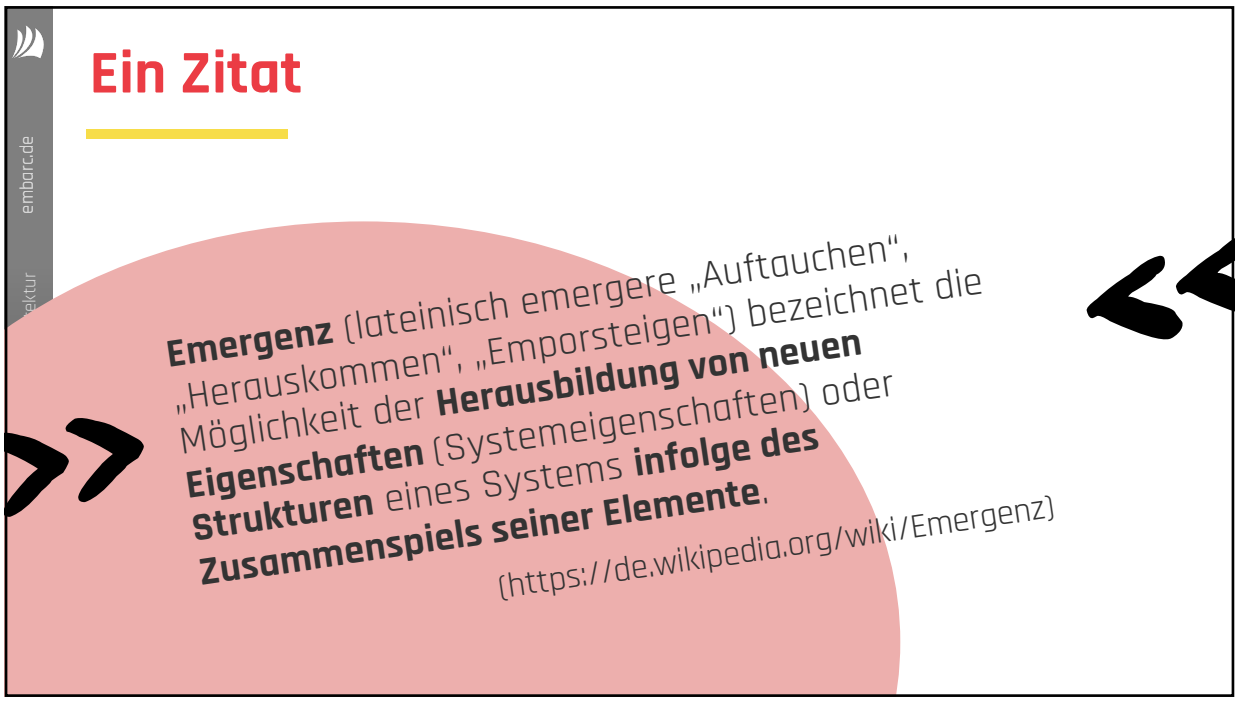
iSAQB

Java Champions

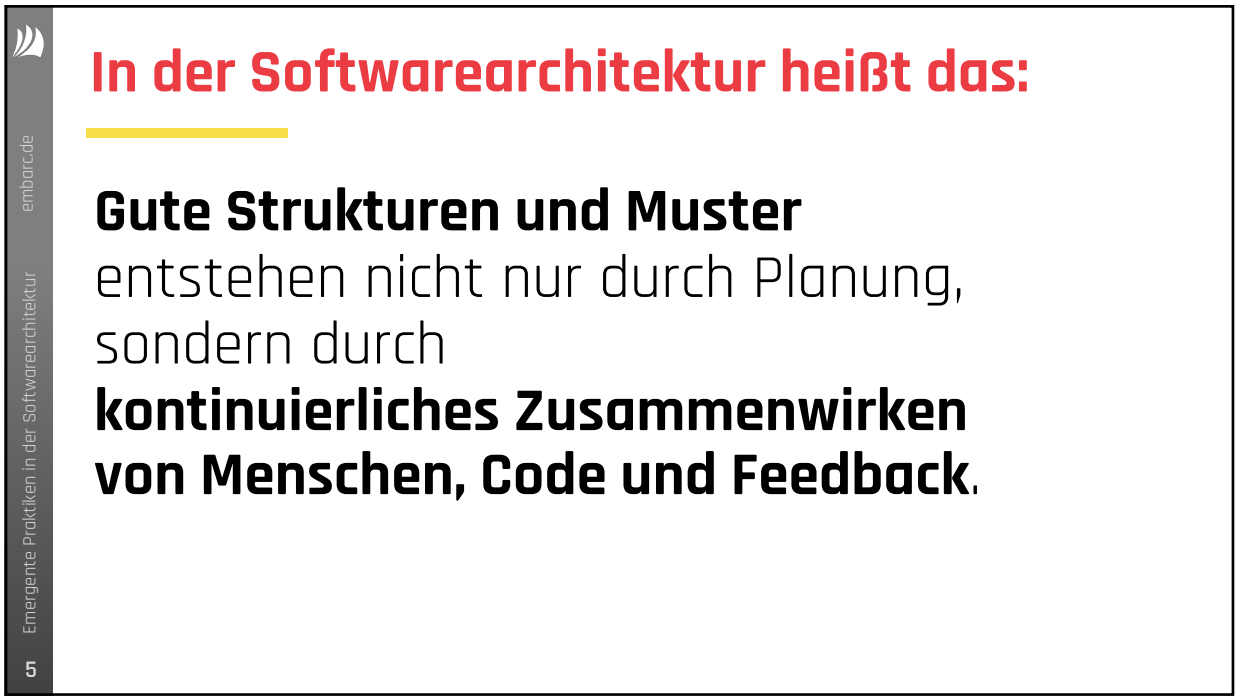
2



3


embarc.de
Ein Zitat
Emergenz (lateinisch emergere „Auftauchen“, „Herauskommen“, „Emporsteigen“) bezeichnet die Möglichkeit der Herausbildung von neuen Eigenschaften (Systemeigenschaften) oder Strukturen eines Systems infolge des Zusammenspiels seiner Elemente.
(https://de.wikipedia.org/wiki/Emergenz)

4


embarc.de
Emergente Praktiken in der Softwarearchitektur
5
In der Softwarearchitektur heißt das:
Gute Strukturen und Muster
entstehen nicht nur durch Planung,
sondern durch
kontinuierliches Zusammenwirken
von Menschen, Code und Feedback.

5

8

9

Noch ein Zitat

embarc.de

ektur

>>

Emergentes Verhalten ist das Phänomen unerwarteter und neuartiger Muster, die sich aus den **Wechselwirkungen einfacher Komponenten in einem komplexen System ergeben**. Zum Beispiel sind **Vogelschwärme, Ameisenkolonien, Staus und soziale Netzwerke** Beispiele für emergentes Verhalten.

<<

10

11

Softwarearchitektur

Eine etwas andere Definition

> *"Softwarearchitektur ist die Menge der Entwurfsentscheidungen, die, wenn falsch getroffen, Dein Vorhaben zum Scheitern bringen kann."**
>
> (Eoin Woods)

* Im Englischen eigentlich: "Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled."

Was ist Softwarearchitektur?

Softwarearchitektur :=

Σ wichtige Entscheidungen

wichtig =
- fundamental (betrifft viele)
- im weiteren Verlauf nur schwer zu ändern
- entscheidend für den Erfolg Eures Softwaresystems

Die große Herausforderung …

Wie können wir den **Widerspruch auflösen**, dass

a) Architekturentscheidungen **schwer zu ändern** sind
b) Software, die lange leben soll, **sich Veränderungen** im Umfeld **anpassen muss**

14

Was erwartet Euch in diesem Vortrag?

- Was ist **risikobasiertes Vorgehen** in der Softwarearchitektur?
- Wie können wir Softwarearchitektur **iterativ, kleinteilig und verzahnt** mit Entwicklung der **fachlichen Features** vorantreiben?
- Was sind die **Vor-/Nachteile** eines Walking Skeleton und wie unterscheidet es sich von einem **MVP** (Minimum Viable Product), **Prototyp** sowie **Durchstich?**

15

00.

Agenda

Was euch erwartet

01 Risikoorientiert

02 Agile Architektur

03 Praktiken

04 Fazit & Ausblick

16

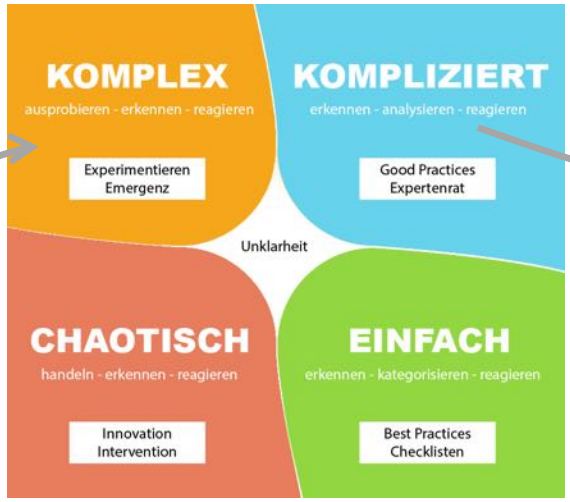
01.

Risikoorientiert

17

Cynefin-Framework

Softwareprojekte sind **heutzutage** eher **komplex**!



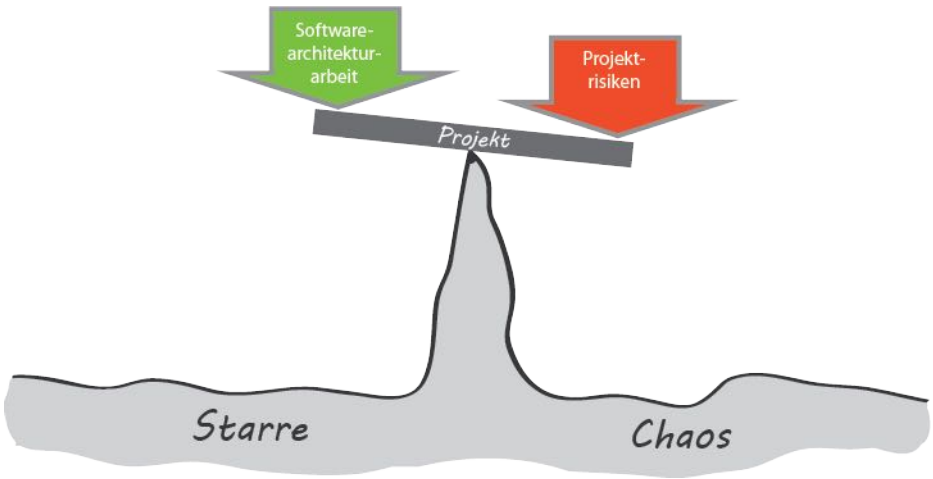
Aber wir versuchen nach **diesen Spielregeln** zu arbeiten!

Komplexitätstreiber: Unsicherheit

Risiken für Projekte aus Architektursicht:

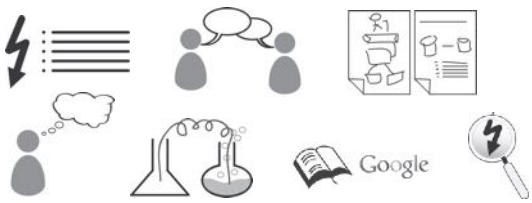
- Hohe Qualitätsanforderungen
- Enger Projektrahmen (Zeit, Budget)
- Viele Projektmitglieder
- Hoher räumlicher Verteilungsgrad
- Neue Technologien
- Wenig Erfahrung im Lösungsspektrum
- Dünner technischer Rahmen
- Viele Abhängigkeiten zu (externen) Projekten
- Vorhandene Zielkonflikte
- ...

Generell...



20

Architekturarbeit ist ...



- Aufwand
- Kein Code
- Kein direkter Kundennutzen

Risikominderungs-maßnahme

für nicht triviale Problemstellungen

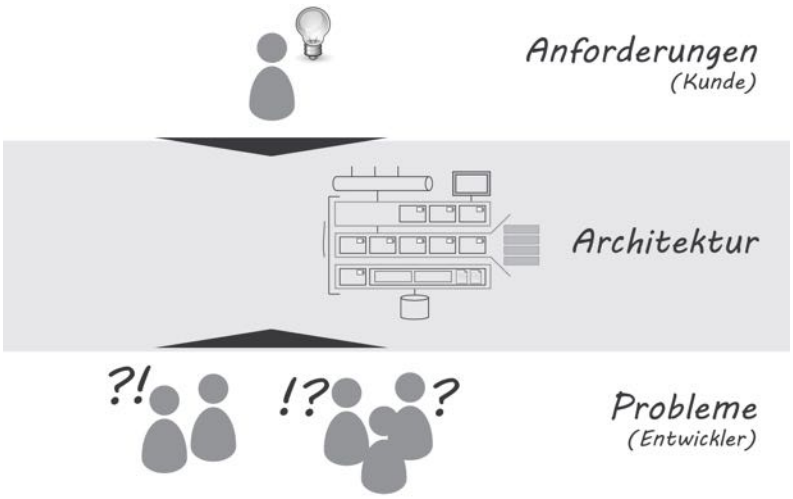
21

Gute Prinzipien zum Umgang mit Risiken

- Fragestellungen **früh erkennen** und analysieren
- Möglichst **spät entscheiden** (bei mehr als einer sinnvollen Alternative)

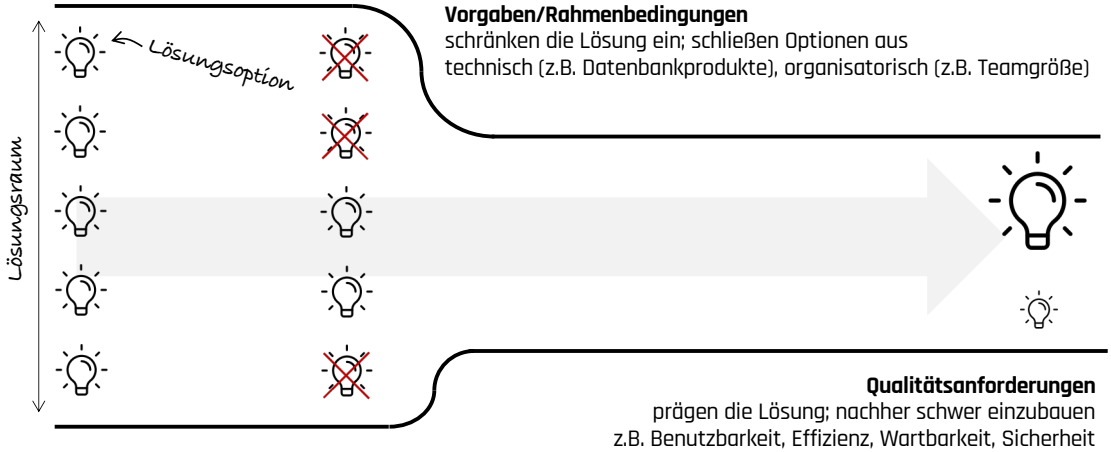
Das schafft ein großes "Lernfenster"
(bzw. Zeit zur Risikominderung)

Architektonische Fragestellungen



Einflüsse auf Entscheidungen

Wir stehen vor einer Architekturentscheidung ...

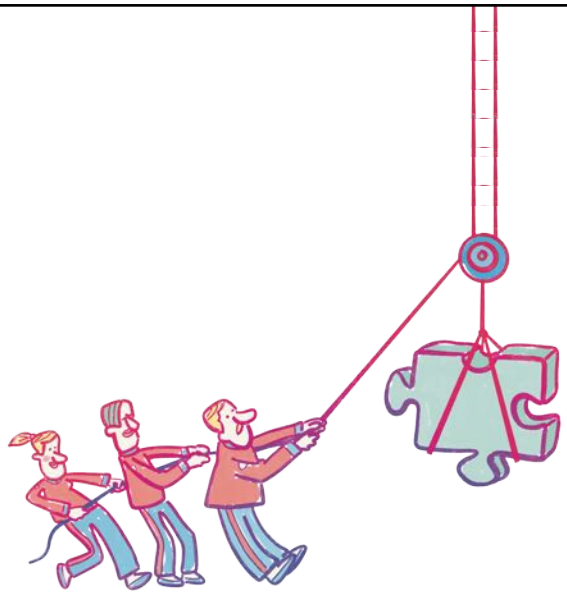


Vorgaben/Rahmenbedingungen
schränken die Lösung ein; schließen Optionen aus
technisch (z.B. Datenbankprodukte), organisatorisch (z.B. Teamgröße)

Qualitätsanforderungen
prägen die Lösung; nachher schwer einzubauen
z.B. Benutzbarkeit, Effizienz, Wartbarkeit, Sicherheit

24

02. Agile Architektur



25

Was erwartet man sich von „Agil“?

Flexibilität

Schnellere T2M

Nachvollziehbareren Fortschritt

Bessere Termintreue (nicht unbedingt inhaltstreue)

Bessere Produkte (es wird gebaut, was gewollt wird)

Weniger “Waste”

...

Das agile Manifest

Individuen und Interaktion	vor	Prozessen und Werkzeugen
Funktionierende Software	vor	Umfassender Dokumentation
Zusammenarbeit mit dem Kunden	vor	Vertragsverhandlungen
Reagieren auf Veränderung	vor	Befolgen eines Plans

Das agile Manifest umgedreht

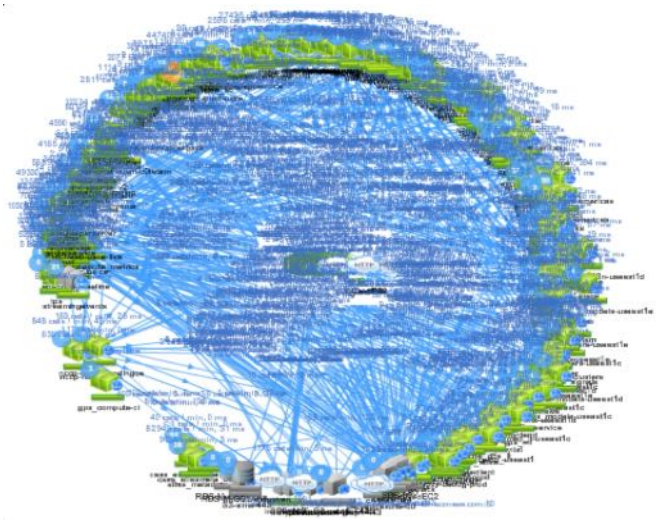
Prozesse und Werkzeug	für	**Individuen und Interaktion**
Nützliche Dokumentation	für	**Funktionierende Software**
Vertragsverhandlungen	für	**Zusammenarbeit mit dem Kunden**
Ausreichend Planung	für	**Reagieren auf Veränderung**

28

Arbeit in komplexen Umfeldern

29

Netflix Architekturüberblick (?)



30

Einige Netflix Ingenieure zitiert...

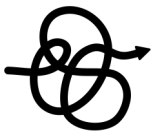
"[...] we quickly see that a distributed system of any meaningful size becomes too complex for a human [to understand]. There are simply too many parts, changing and innovating too quickly, interacting in too many unplanned and uncoordinated ways for a human to hold those patterns in their head."

"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

31

Agile Weltsicht / Erkenntnis

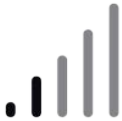


Softwareentwicklung ist **komplexes** Problem

Interaktion, **Feedback** & **Zusammenarbeit**

Werkzeuge sind **kein Selbstzweck**

Probleme sind **Quelle** für **Wettbewerbsvorteil**



Akzeptable statt die beste **Lösung**

32

Agile Prinzipien

Inkrementelle Entwicklung zur **Verifikation** von **Annahmen**

Risiken früh bearbeiten

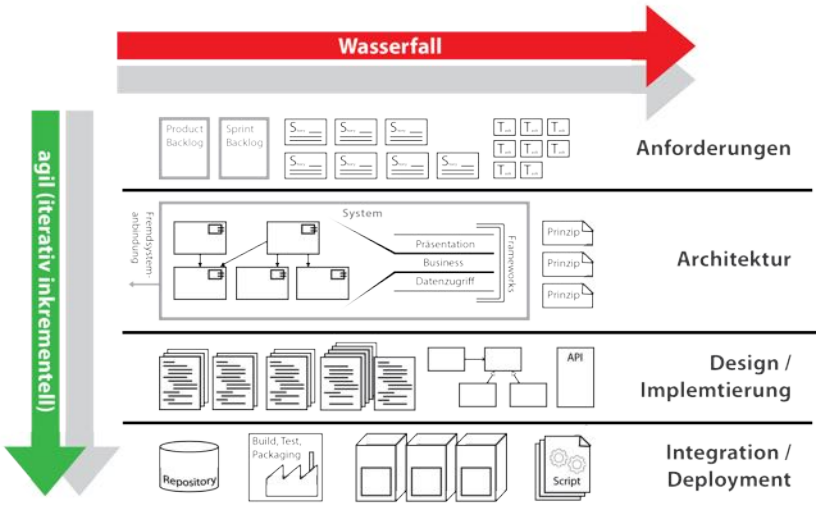
Kleine, **fokussierte** Entwicklungs**schritte**

Entscheidungen zum **LVM** treffen

Risikobearbeitung, **Wissens-aufbau** und **Wertschöpfung**

33

Was macht Agile Architektur anders?



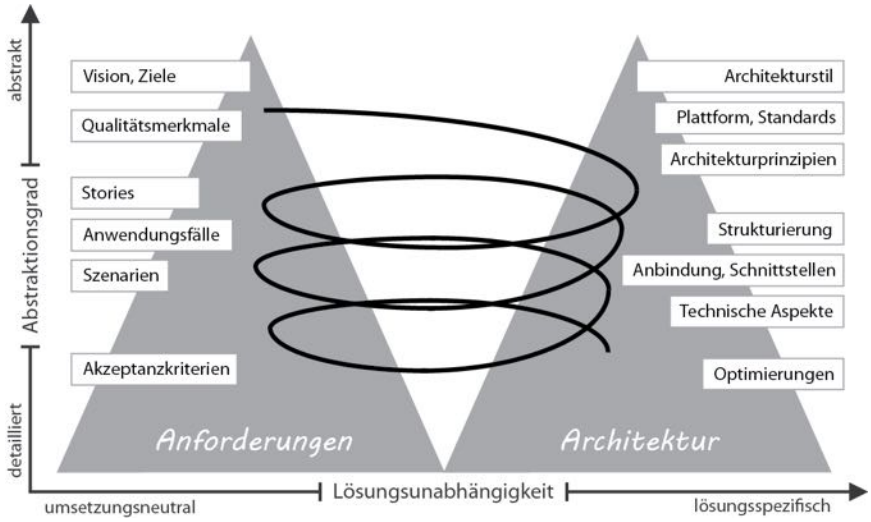
Agile Architektur

Architekturaufwand dem Problem **angemessen**

Architektur entsteht **iterativ** (kein **BUFD**)

Crossfunktionale Teams (transparent/reflektiert)

Iterative Architekturarbeit



36

03.

Praktiken

37

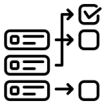
Architektur-vision

Walking Skeleton

Architektur-überblick

Umgang mit Risiken

Letzter vernünftiger Moment



Architekturarbeit sichtbar machen

Technische Schulden

Auf den Prüfstand stellen

39

statt **Big Design Up Front**

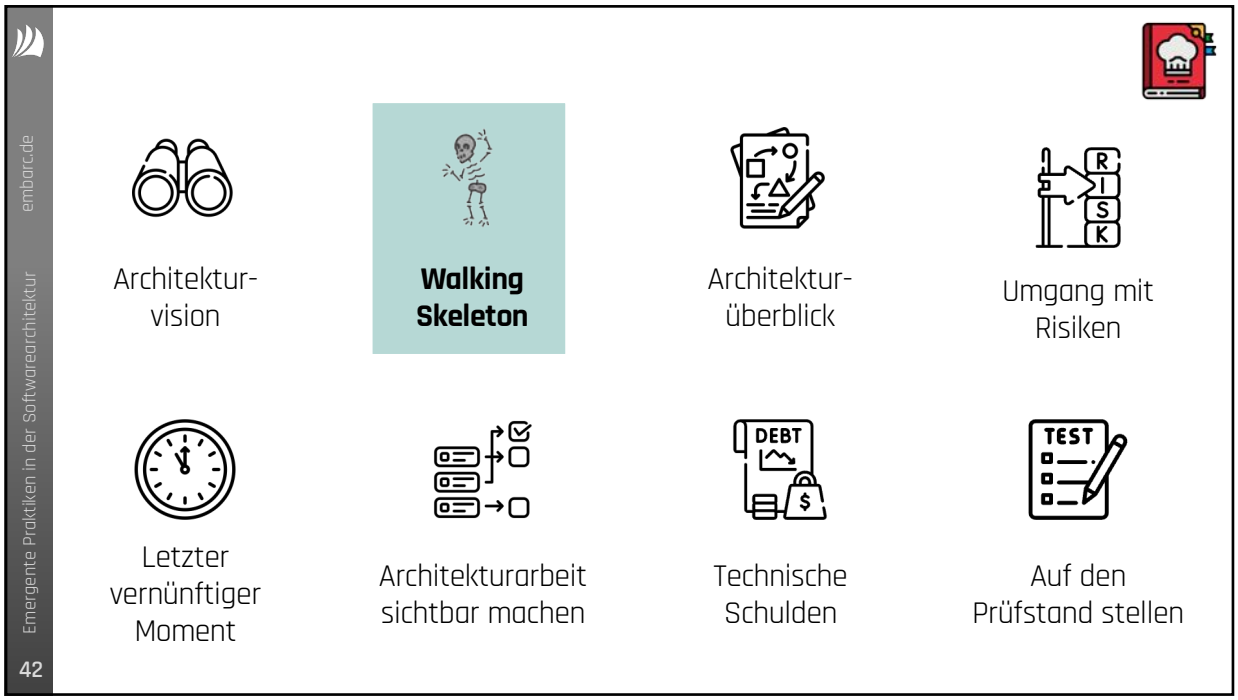
Eine Architekturvision ist eine schlanke, sich stetig weiterentwickelnde Übersicht zur aktuellen Architekturidee und deren Motivation.

Stefan Toth

40

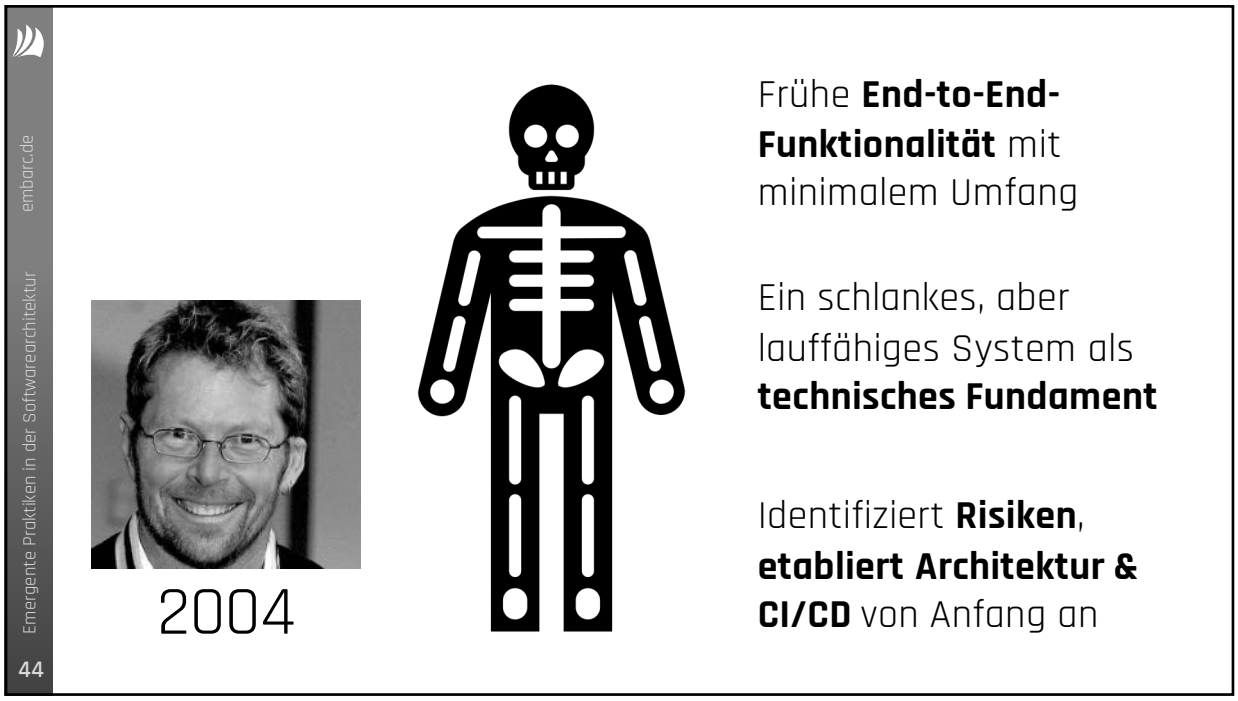

embarc.de
Emergente Praktiken in der Softwarearchitektur
41
Architekturvision statt BUFD
Big Design Up Front
Architekturvision
Fokus
für den Projekterfolg kritische Architekturaspekte für die Entwicklung unumstößlich wichtiger Rahmen
Als Idee, nicht als irreversible Entscheidung!
Typische Inhalte
Systemkontext (Abgrenzung)
Rahmenbedingungen
Qualitätsanforderungen (priorisiert)
Technologien (inkl. Plattform, Programmiersprache)
Architekturstil (bestimmende Muster)
Architekturprinzipien
Grobe Gliederung (Systemteile auf oberster Ebene)
Was?
Wie?

41


embarc.de
Emergente Praktiken in der Softwarearchitektur
42
Architektur-vision
Walking Skeleton
Architektur-überblick
Umgang mit Risiken
Letzter vernünftiger Moment
Architekturarbeit sichtbar machen
Technische Schulden
Auf den Prüfstand stellen
DEBT
TEST
RISK

42

43



2004

Frühe **End-to-End-Funktionalität** mit minimalem Umfang

Ein schlankes, aber lauffähiges System als **technisches Fundament**

Identifiziert **Risiken**, **etabliert Architektur & CI/CD** von Anfang an

44

Walking Skeleton

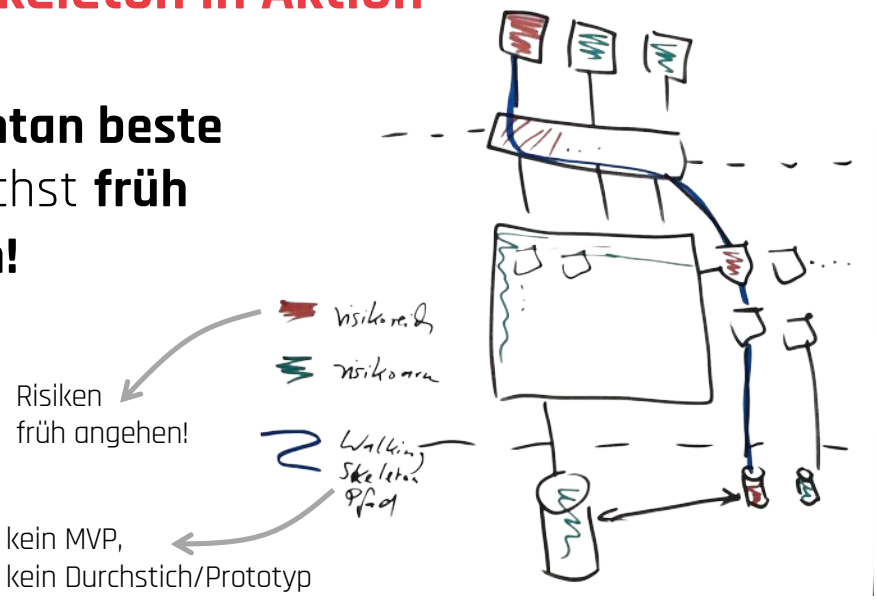
„**A Walking Skeleton** is a tiny implementation of the system that performs a small end-to-end function. It need **not** use the **final** architecture, but it should **link together** the **main architectural components**. The architecture and the functionality can then evolve in parallel."

- Alistair Cockburn

45

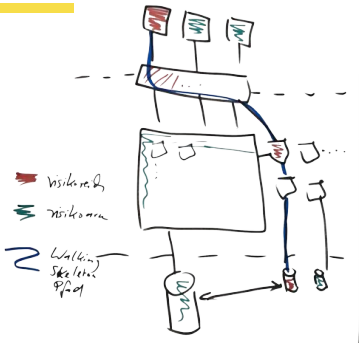
Walking Skeleton in Aktion

Die **momentan beste Idee** möglichst **früh widerlegen!**



46

Möglichst effektiv beweisen dass es Müll ist.



Möglichst schnell **widerlegen**. Und zwar richtig.

- In Risiken stechen
- Unsicherheiten ausnutzen
- zentrale Ziele in Gefahr bringen

} Entsprechende Anforderungen suchen

47

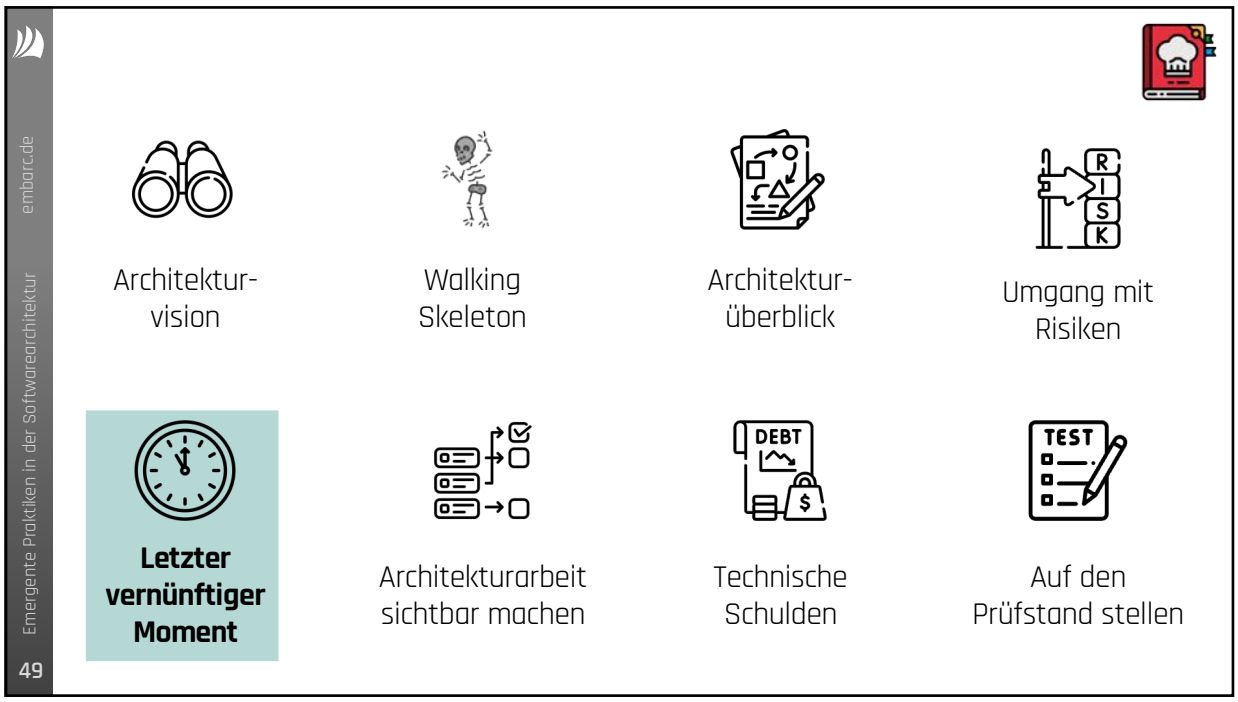
Abgrenzung zu ...

MVP

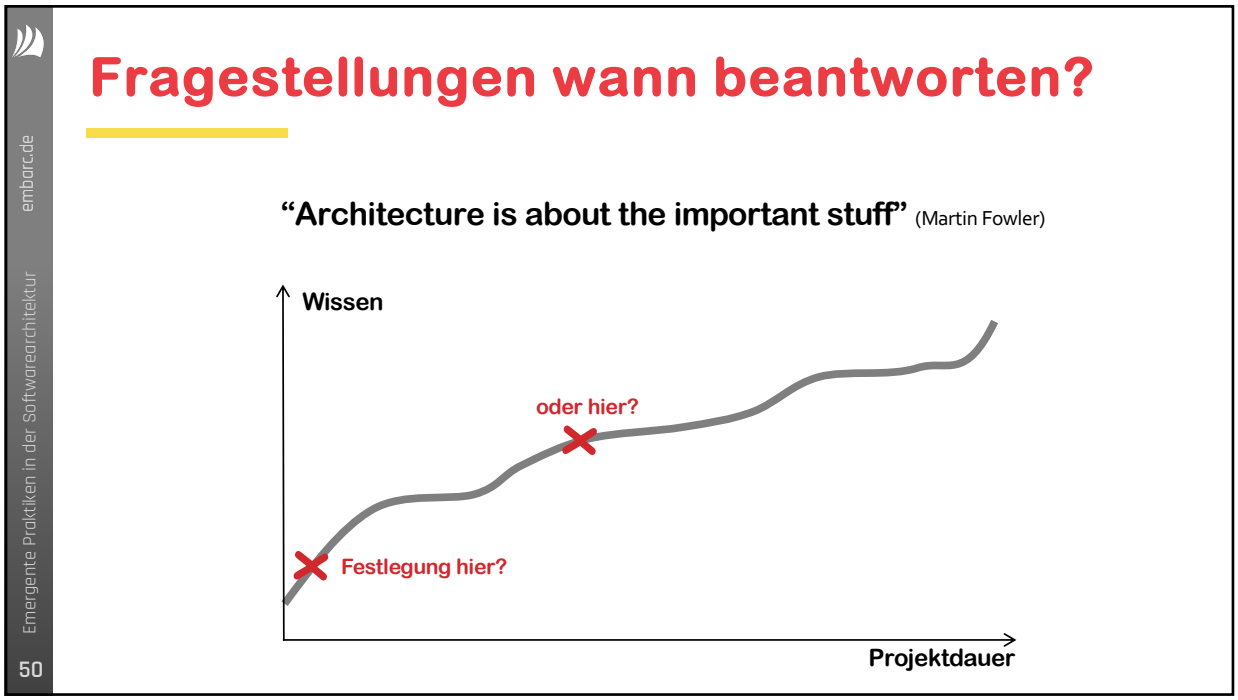
Prototyp

Durchstich

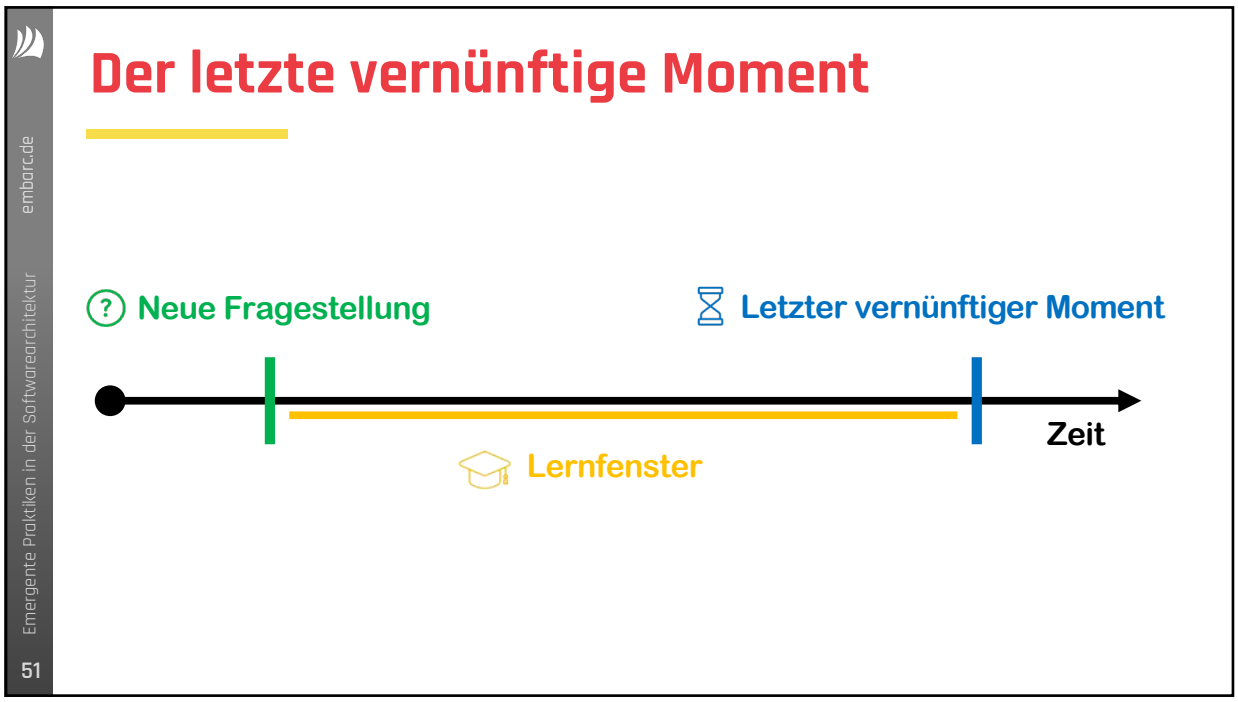
48


embarc.de
Emergente Praktiken in der Softwarearchitektur
Architektur-vision
Walking Skeleton
Architektur-überblick
Umgang mit Risiken
RISK
Letzter vernünftiger Moment
Architekturarbeit sichtbar machen
DEBT
Technische Schulden
TEST
Auf den Prüfstand stellen
49

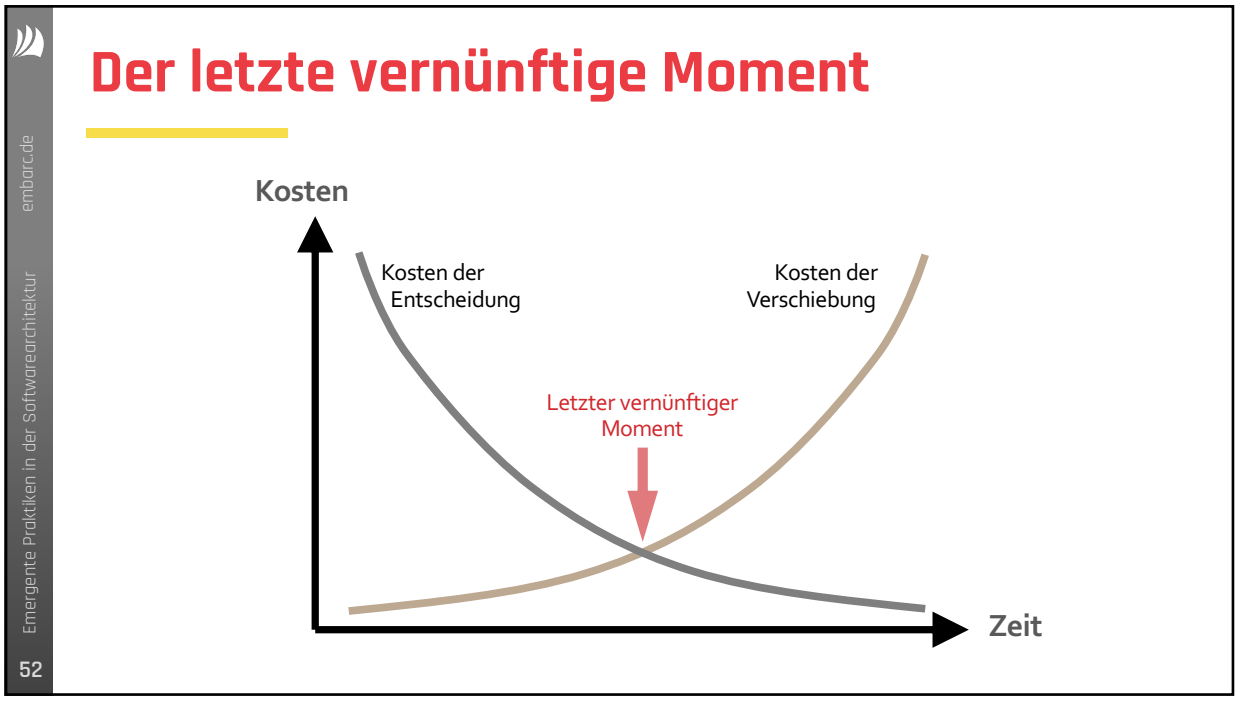
49


embarc.de
Emergente Praktiken in der Softwarearchitektur
Fragestellungen wann beantworten?
“Architecture is about the important stuff” (Martin Fowler)
Wissen
oder hier?
Festlegung hier?
Projektdauer
50

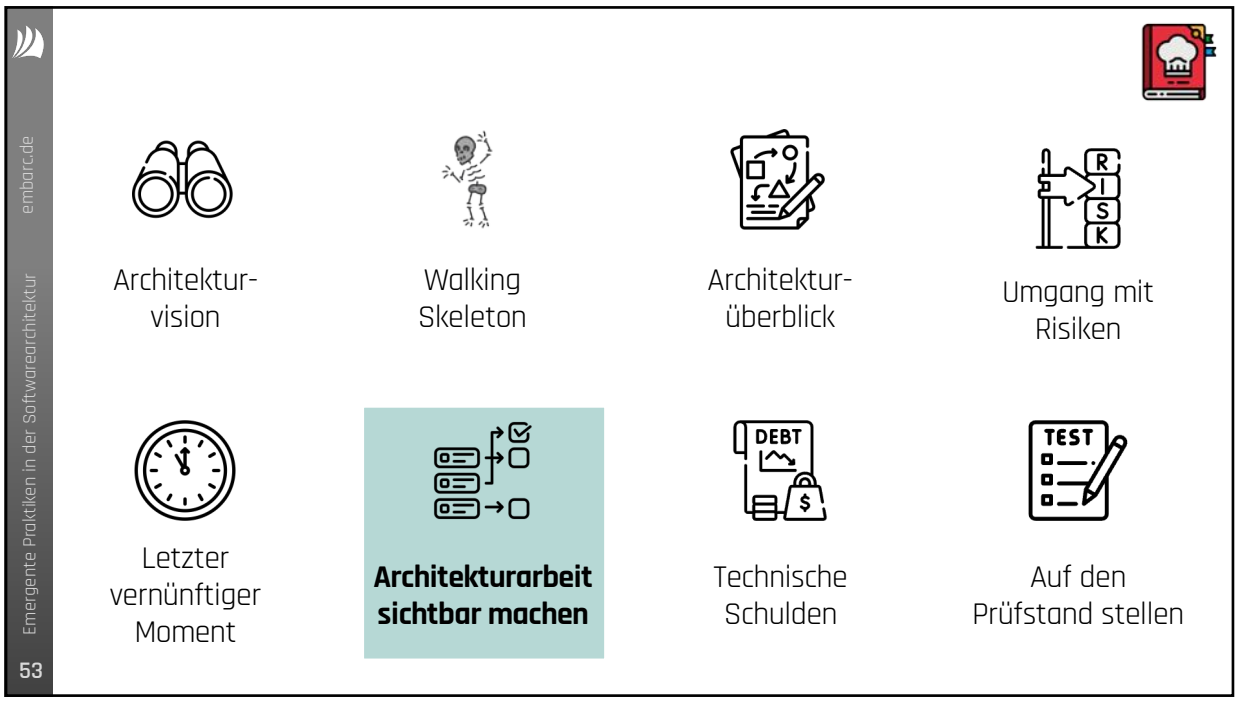
50


Der letzte vernünftige Moment
embarc.de
Emergente Praktiken in der Softwarearchitektur
51
Neue Fragestellung
Letzter vernünftiger Moment
Zeit
Lernfenster

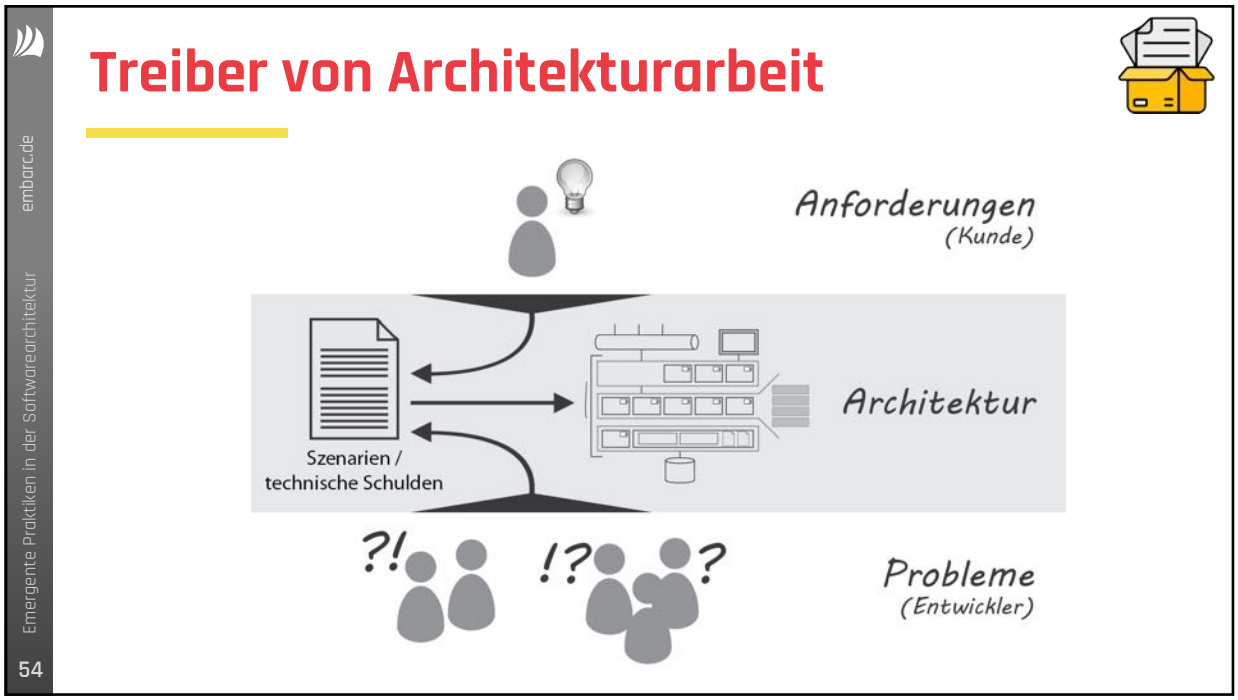
51


Der letzte vernünftige Moment
embarc.de
Emergente Praktiken in der Softwarearchitektur
52
Kosten
Kosten der Entscheidung
Kosten der Verschiebung
Letzter vernünftiger Moment
Zeit

52

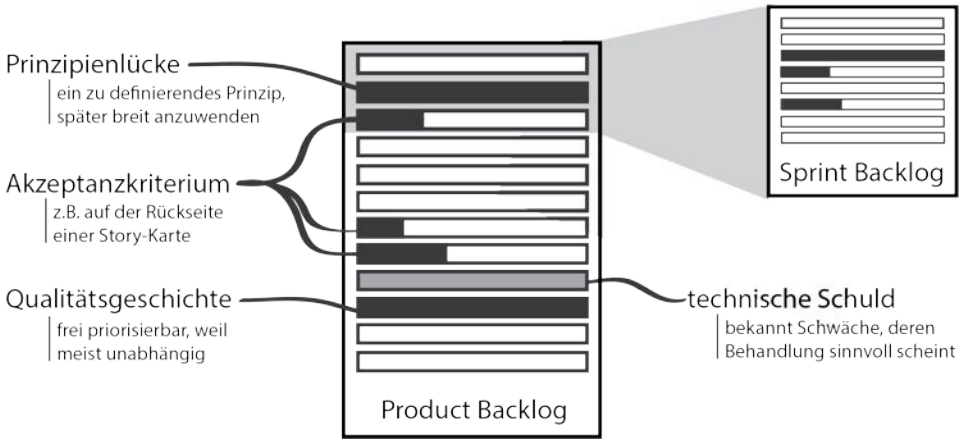

embarc.de
Emergente Praktiken in der Softwarearchitektur
Architektur-
vision
Walking
Skeleton
Architektur-
überblick
Umgang mit
Risiken
Letzter
vernünftiger
Moment
Architekturarbeit
sichtbar machen
Technische
Schulden
Auf den
Prüfstand stellen
DEBT
TEST
R
I
S
K
53

53

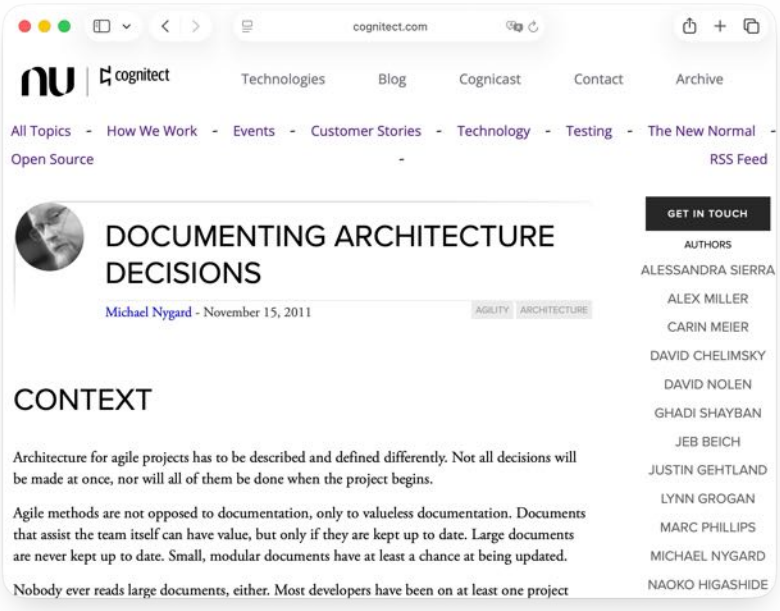

embarc.de
Emergente Praktiken in der Softwarearchitektur
Treiber von Architekturarbeit
Anforderungen
(Kunde)
Architektur
Szenarien /
technische Schulden
Probleme
(Entwickler)
54

54

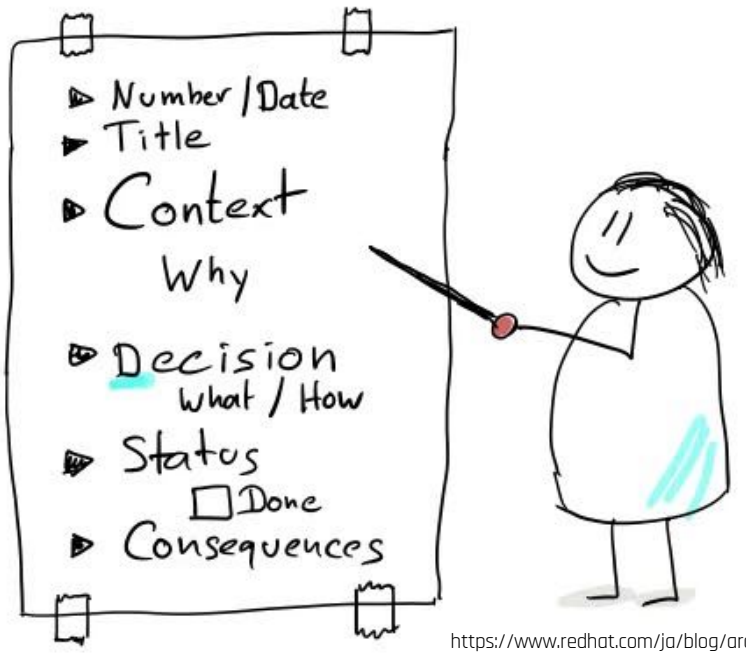
Architekturarbeit sichtbar machen



55



56



https://www.redhat.com/ja/blog/architecture-decision-records

57

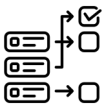
Architektur-vision

Walking Skeleton

Architektur-überblick

Umgang mit Risiken

Letzter vernünftiger Moment



Architekturarbeit sichtbar machen

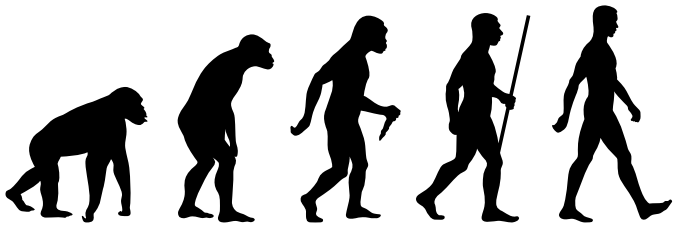
Technische Schulden

Auf den Prüfstand stellen

58

Die Natur hat Lösungen …

… für komplexe Probleme …



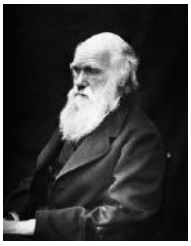
… durch kleinteilige, stetige Anpassung

59

Charles Darwin: Natürliche Auslese

Diese Nachkommen sind nach dem Zufallsprinzip alle mit etwas unterschiedlichen Merkmalen ausgestattet. Einige Individuen sind dadurch besser an ihre Umwelt angepasst als andere, sie überleben und können sich vermehren.

So setzen sich ganz automatisch die vorteilhaften Merkmale durch. Darwin nennt dies "**Survival of the Fittest**". Damit meint er "**der am besten Angepasste überlebt**" und nicht etwa "der Stärkste".



https://www.planet-wissen.de/natur/forschung/evolutionsforschung/pwiesurvivalofthefittestdiehauptthesenderevolutionstheorie100.html

60

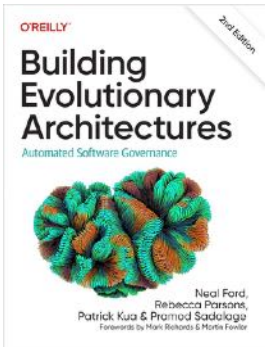
Evolutionäre Algorithmen

- **Evolutionäre Algorithmen** sind Optimierungsverfahren, deren Funktionsweise von der Evolution natürlicher Lebewesen inspiriert ist.
- In Anlehnung an die Natur werden Lösungskandidaten für ein bestimmtes Problem künstlich evolviert.
- Evolutionäre Algorithmen finden meist **nicht die beste Lösung** für ein Problem, aber bei Erfolg eine **hinreichend gute** - in der Praxis bereits wünschenswert.

Natürliches Vorbild	Evolutionärer Algorithmus	Beispiel
Organismus	**Lösungskandidat**	Flugzeugflügel
Fortpflanzungserfolg	**Wert der Fitnessfunktion**	Strömungswiderstand
Natürliche Mutation	**Mutation**	Änderung der Form

61

Übertragung auf Softwarearchitektur



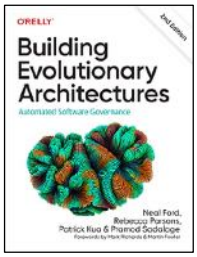
*„Eine **evolutionäre Architektur** unterstützt **geleitete, inkrementelle Veränderungen** über **mehrere Dimensionen** hinweg.“**

N. Ford, R. Parsons, P. Kua, P. Sadalage:
Building Evolutionary Architectures: Automated Software Governance
O'Reilly 2022

* Engl.: "An evolutionary architecture supports guided, incremental change across multiple dimensions."

62

> Eine **Fitness Function** misst objektiv, wie gut eine Lösung die an sie gesetzten Ziele erreicht.
>
> - Building Evolutionary Architectures – Ford, Parsons, Kua, Sadalage



63

Gutes Feedback …

- wird **möglichst nah am realen Einsatz** gesammelt, z.B. nah an Nutzer*innen oder am physischen Endgerät
- erfolgt **regelmäßig** und **zeitnah**, idealerweise bei jedem Release
- ist **aussagekräftig**, entweder durch das (Nicht-)Erreichen eines Zielwerts oder durch Trends
- ist für alle relevanten Zielgruppen **zugreifbar**
- ist **von zentralen Zielen abgeleitet**

64

Einteilung / Kategorisierung

Art der Ausführung
- angestoßen
- kontinuierlich

Qualitätsmerkmal
- Wartbarkeit
- Zuverlässigkeit
- Effizienz
- ...

Art der Ergebniskontrolle
- o/1
- Wert
- Tendenz

Breite der Ausführung
- atomar
- holistisch

Ort und Zeitpunkt der Ausführung
- CI/CD
- Testumgebung
- Produktion

Gültigkeit der Tests
- Temporär
- Permanent

Kategorisierung = Dimensionen des "Lösungsraums"

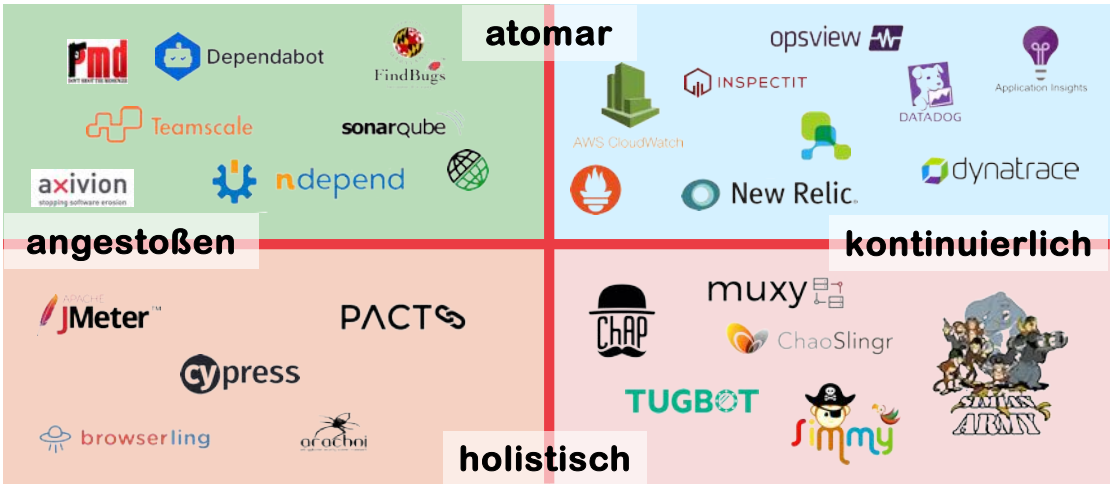
65

Beispiele für Fitness Functions

	atomar	kontinuierlich
	Code-Metriken	Monitoring & Logging
angestoßen	Software-Reviews	
holistisch		Chaos Engineering + Messung der Zuverlässigkeit

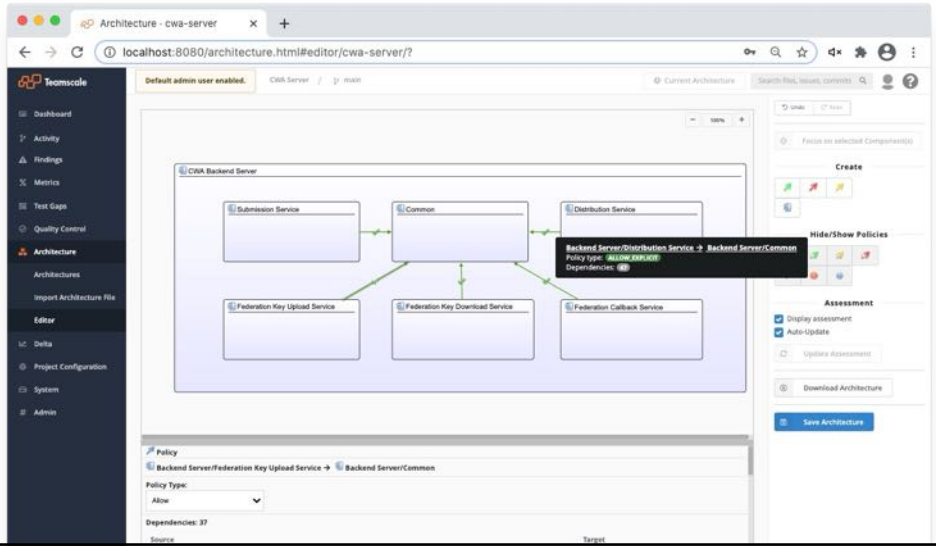
66

Fitness Function Quadrant - Tooling



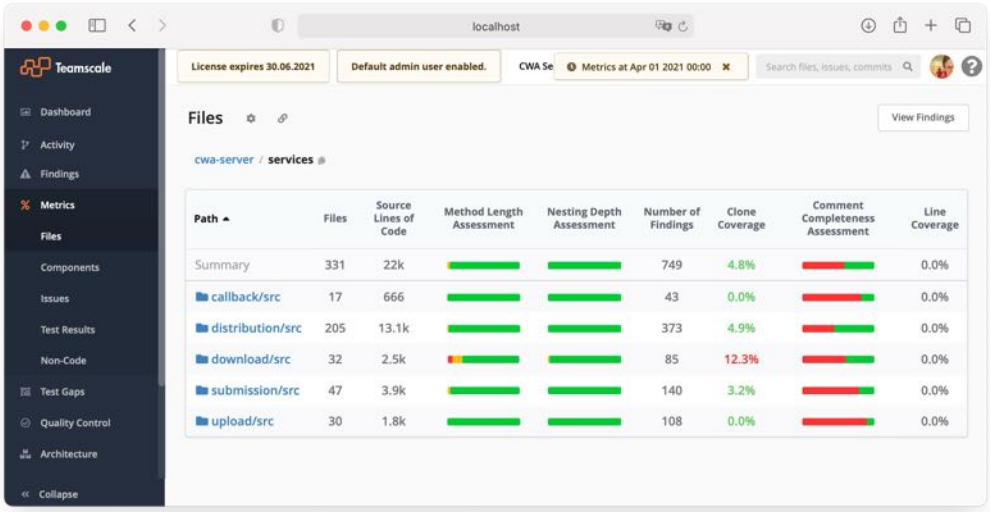
67

Architekturregeln



68

Metriken



69

Beispiele für Fitness Functions

Fragestellung – Wartbarkeit (Testbarkeit)
Ist unsere Testabdeckung ausreichend?

Zielgruppe:
Entwickler

Umsetzungsidee:
Regelmäßige Überprüfung Test-Coverage

Zeitpunkt:
Während jedem CI-Build
Während Nightly Integration-Test Build

Fitness Function(s):

- Testabdeckung (Unit) > 0.9
- Testabdeckung (Integration) > 0.6

70

Beispiele für Fitness Functions

Fragestellung – Zuverlässigkeit (Verfügbarkeit)
Funktioniert unsere Service auch bei höherer Latenz zuverlässig?

Zielgruppe:
Entwickler / Betrieb

Umsetzungsidee:
Regelmäßige Tests mit erhöhter Latenz

Zeitpunkt:
- Während Nightly Integration-Test Build

Fitness Function(s):
- Integration-Test
  Fehler = 0%
  (Netzwerklatenz für 3rd Party Service = 5s)

71

Ein Steckbrief für Fitness-Functions

Fitness-Function # ___
Vorhaben: Schadensgutachten
Adressiertes Architekturziel: Kompatibilität
Beispielszenario: Unsere Software lässt sich ohne Änderung mit der neuen Version der Fremdbibliothek bauen und funktioniert einwandfrei

Art der Ausführung: ☒ angestoßen ☐ kontinuierlich
Gültigkeit des Tests: ☒ permanent ☐ temporär
Zeitpunkt (und Ort) der Ausführung: ☐ beim Entwickeln (Entwicklerarbeitsplatz, IDE) ☒ während des Builds, CI/CD (Build-Server) ☐ im nachgelagerten Test (Test-System) ☐ in Produktion (Produktionssystem)
Breite der Ausführung: ☒ atomar ☐ holistisch
Ergebniskontrolle: ☒ 0 / 1 ☐ Wertüberprüfung ☐ Tendenz eines Wertes

Test-Idee: Läuft meine Software mit den neuesten Abhängigkeiten?
Zielgruppe: Entwickler

Umsetzungsansatz für die Test-Idee
-> Repositories nach neuesten Versionen scannen
-> Source Code mit neuesten Versionen im separaten Branch testen

Leitfragen zur Umsetzung
Wann, wie und wo führen wir die Testfunktion aus?
Wie kommen wir an die benötigten Daten?
Wie werten wir die Daten aus?
Was passiert, wenn die Ergebniskontrolle einen Fehler anzeigt?
Was passiert, wenn der Test selbst fehlschlägt?
Wie kommunizieren wir das Ergebnis? Wie oft? Und an wen? ...

72

04.

Fazit und Ausblick

73

Agile Architektur

Frühes Feedback & Risikominimierung

Iterative Entwicklung & Evolutionäre Architektur

Fitness Functions & **Automatisierung**

Klare Kommunikation & **Transparenz**

74

75

Anderes Beispiel

Blueprint „Neuer Service“

Name

Service-Art

Serverless (FaaS) Container

Serverless (FaaS) | Container

Serverless Function

REST-Service

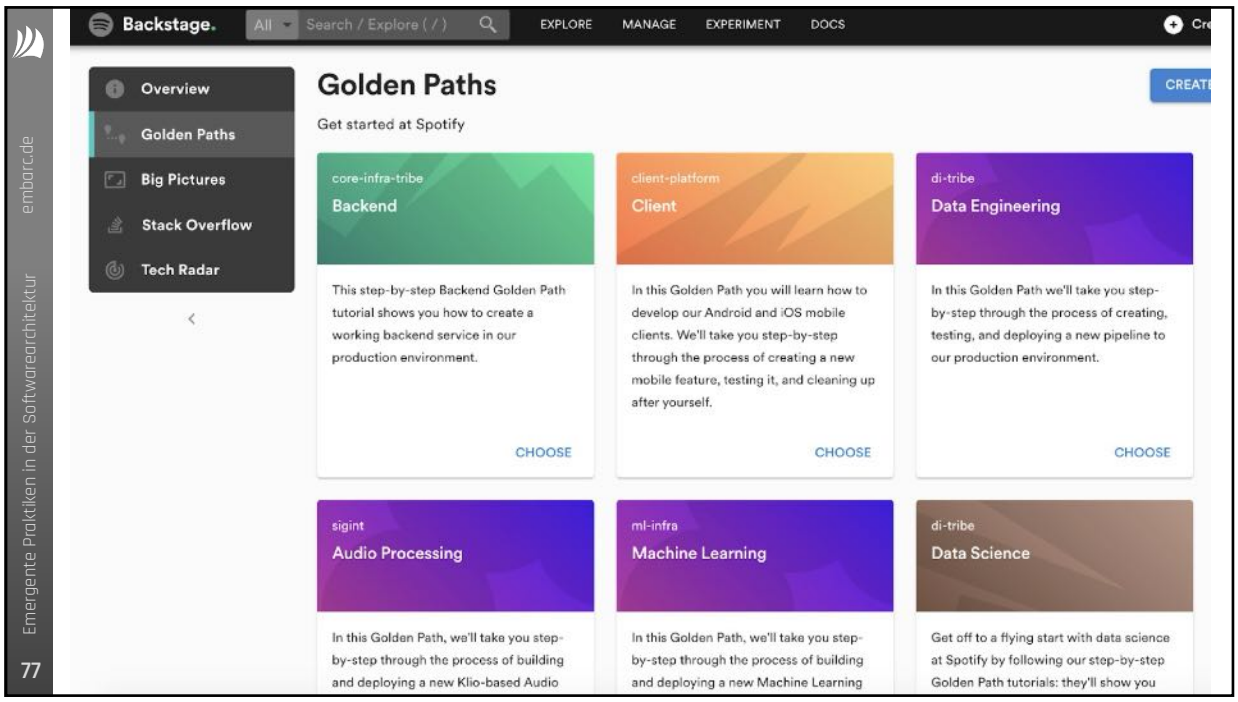
amazon DynamoDB

PostgreSQL

spring

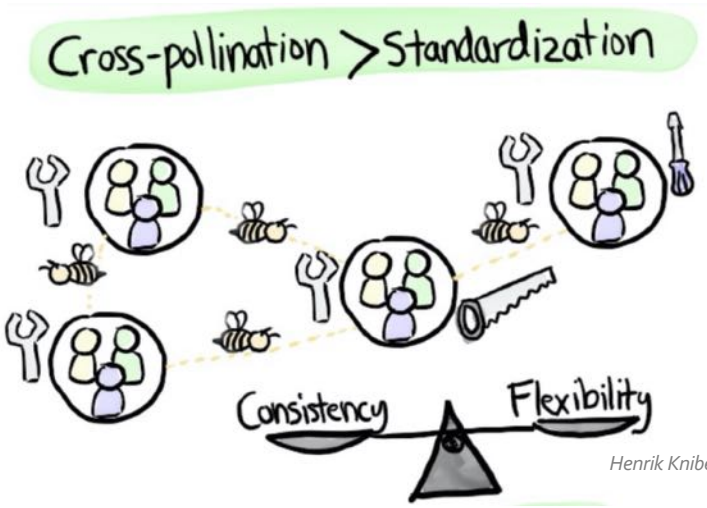
AWS Fargate

76



77

Gute Ideen setzen sich durch...



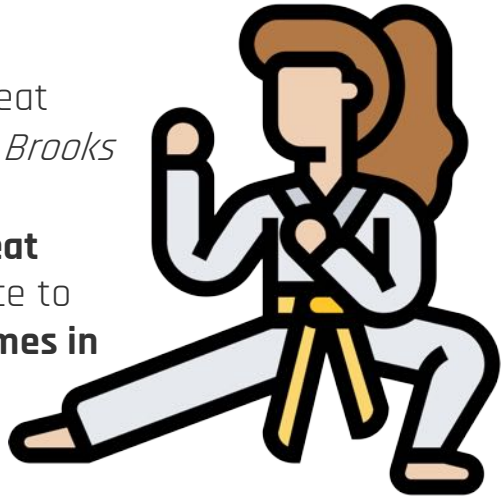
Henrik Kniberg (Spotify Labs Blog)

78

Üben mit Architektur-Katas

"How do we get great designers? Great designers **design**, of course." -- *Fred Brooks*

"So how are we supposed **to get great architects**, if they only get the chance to architect fewer than **a half-dozen times in their career**?" -- *Ted Neward*



https://www.architecturalkatas.com/

iSAQB Advanced Prüfung "als Übung"

Voraussetzungen:

Foundation Zertifikat

70 Creditpoints über Seminare

Durchführung:

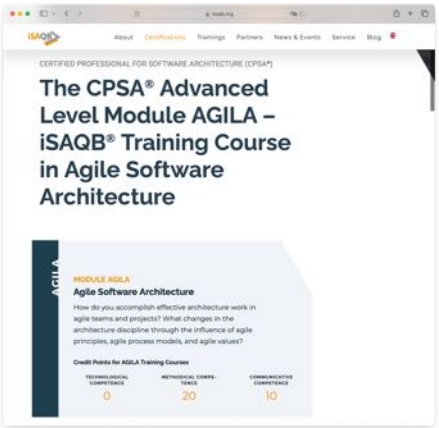
Hausaufgabe und Interview

Buch zum Thema

https://www.swamuster.de/musteruberblick/richtig-entscheiden/

81

iSAQB Advanced Modul AGILA



https://www.socreatory.com/en/trainings/agila

https://www.isaqb.org/certifications/cpsa-certifications/cpsa-advanced-level/agila-agile-software-architecture/

82

Architektur-Spicker

„Mit unseren Architektur-Spickern beleuchten wir die konzeptionelle Seite der Softwareentwicklung."

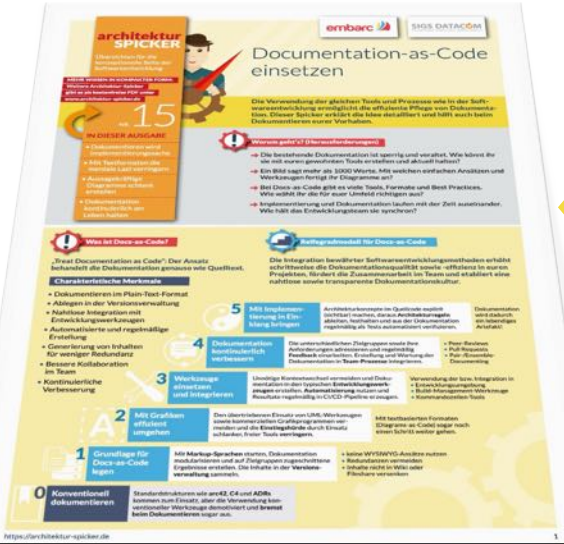
Architektur-Spicker #6
Agile Architektur

PDF, 4 Seiten
Kostenloser Download.

➔ embarc.de/architektur-spicker

83

Architektur-Spicker



„Mit unseren Architektur-Spickern beleuchten wir die konzeptionelle Seite der Softwareentwicklung."

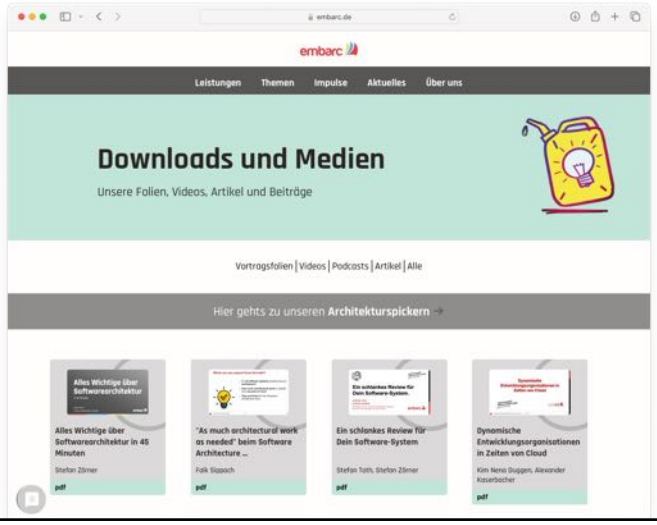
Architektur-Spicker #15
Documentation-as-Code einsetzen

PDF, 4 Seiten
Kostenloser Download.

➔ architektur-spicker.de/

84

Folien als PDF zum Download



➔ embarc.de/download/

85

86

Feedback & Fragen?

Ich freue mich auf Fragen, Diskussionen, Anregungen!

87

Vielen Dank.

Ich freue mich auf Eure Fragen!

falk.sippach@embarc.de
linkedin.com/in/falk-sippach
@sippsack
@sippsack@ijug.social

88

88