embarc Software Consulting GmbH
SOCREATORY

Architekturvision in Zeiten von Clean Architecture

FALK SIPPACH // EMBARC

betterCode Clean Architecture

Dienstag, 06.12.2022

betterCode (CA)

1

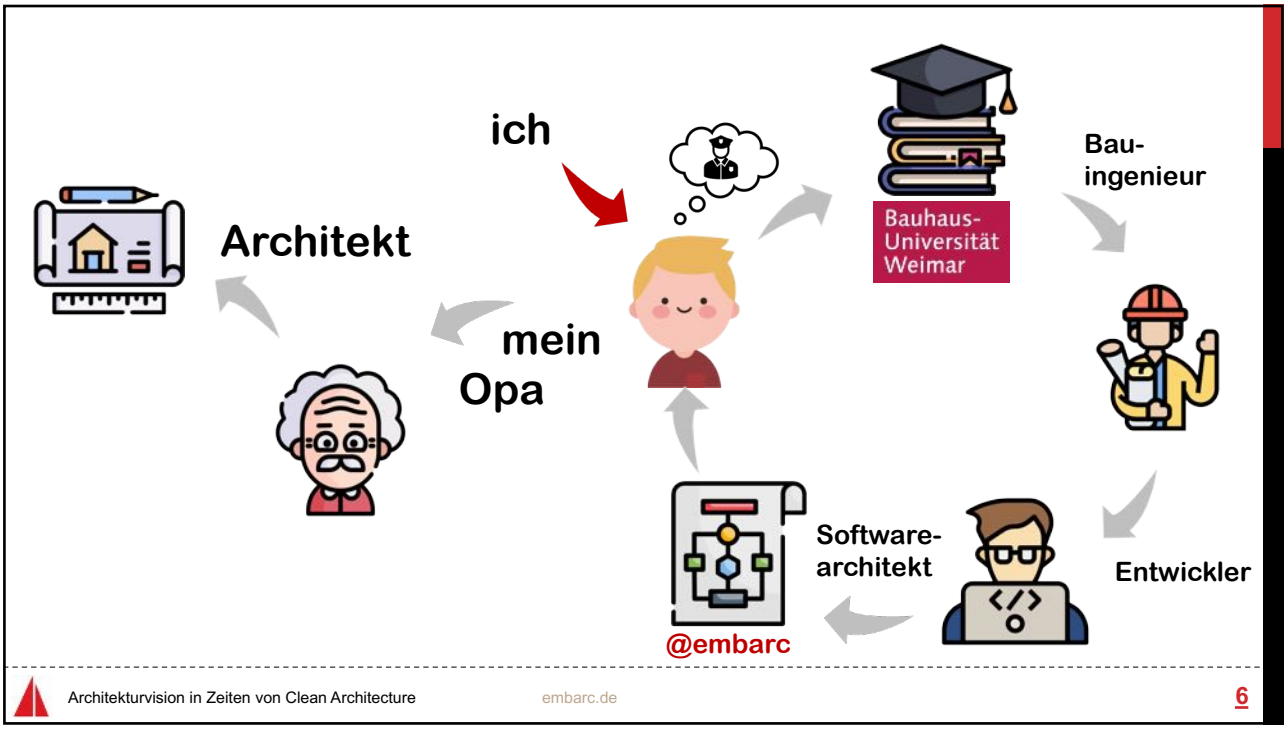
Architekturvision in Zeiten von Clean Architecture

Die von uns gebauten Anwendungen oder Systemlandschaften werden immer anspruchsvoller. Mögliche Antworten darauf sind der Einsatz von Microservices-Architekturen und Domain-Driven Design, Cloud-basierte Deployments oder verschiedene Varianten der hexagonalen bzw. Clean Architecture. Diese Ansätze helfen, aber sie werden auch zum Teil des Problems. Denn die Komplexität steigt durch ihren Einsatz nur noch weiter. Und gerade deshalb braucht es zunächst eine stabile und verlässliche, explizite Softwarearchitekturbasis.

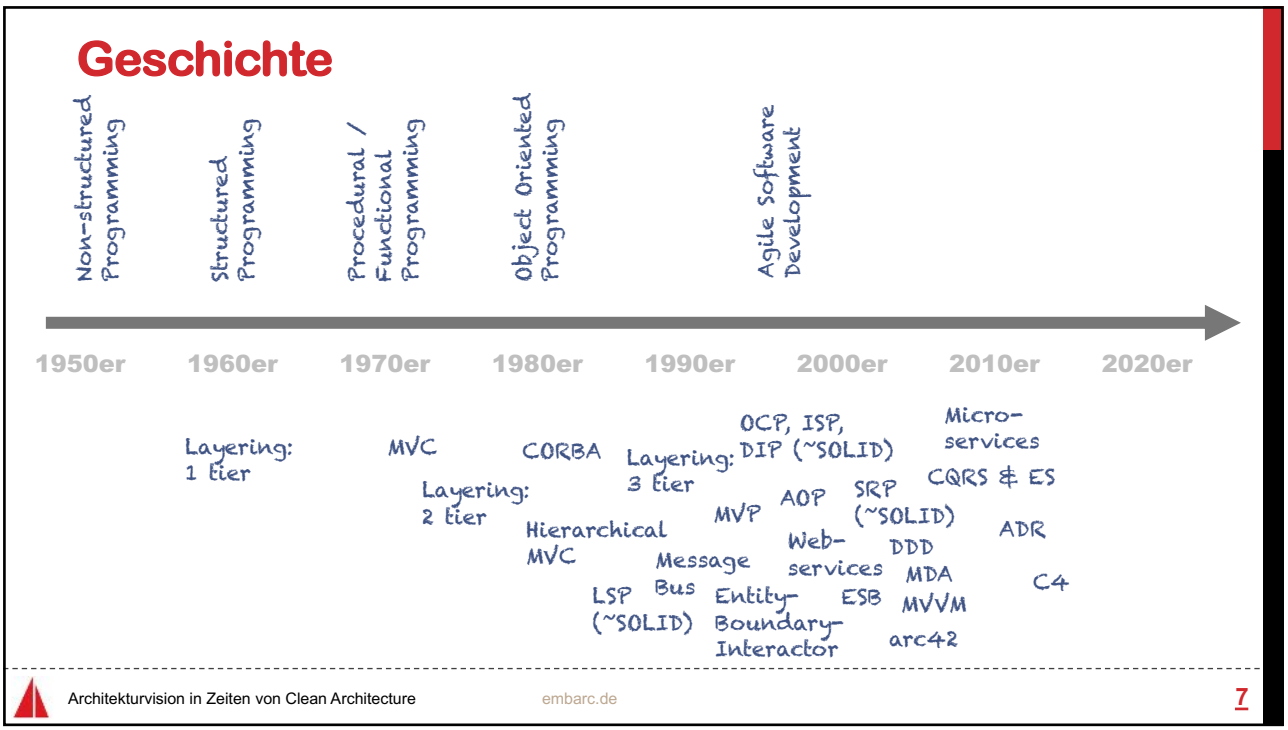
Denn die Architektur sollte der Fels in der Brandung des wogenden Projektgeschehens sein. In diesem Vortrag diskutieren wir, wie ihr mittels der Architekturvision zielgerichtet vorgehen und ohne Overengineering sowie bei bestmöglicher Risikoabschätzung Entscheidungen treffen und so eure Software möglichst flexibel entwickeln könnt.

Falk zeigt dir, wie du die technischen Anforderungen besser verstehst, daraus gezielt die richtigen Lösungsansätze ableitest und das Ganze transparent für alle Stakeholder festhalten kannst. Explizite Softwarearchitekturarbeit ist essenziell, muss aber nicht schwer sein.

2


ich
Architekt
mein Opa
Bauhaus-Universität Weimar
Bau-ingenieur
Software-architekt
Entwickler
@embarc
Architekturvision in Zeiten von Clean Architecture
embarc.de
6

6


Geschichte
Non-structured Programming
Structured Programming
Procedural / Functional Programming
Object Oriented Programming
Agile Software Development
1950er
1960er
1970er
1980er
1990er
2000er
2010er
2020er
Layering: 1 tier
MVC
CORBA
Layering: 3 tier
OCP, ISP, DIP (~SOLID)
Micro-services
Layering: 2 tier
AOP
SRP (~SOLID)
CQRS & ES
MVP
Hierarchical MVC
Web-services
DDD
ADR
Message Bus
MDA
C4
LSP (~SOLID)
Entity-Boundary-Interactor
ESB
MVVM
arc42
Architekturvision in Zeiten von Clean Architecture
embarc.de
7

7

Was erwartet Euch in diesem Vortrag

- Haben **alle Softwaresysteme** eigentlich auch eine **Architektur**?
- **Wieviel Architekturarbeit** braucht es aus meiner Sicht?
- **Tipps und Tricks** zur notwendigen Architekturarbeit

8

Agenda

1 Clean Architecture

2 Zufällige Architektur

3 Architekturvision

4 Ausblick

9

Agenda

1 **Clean Architecture**

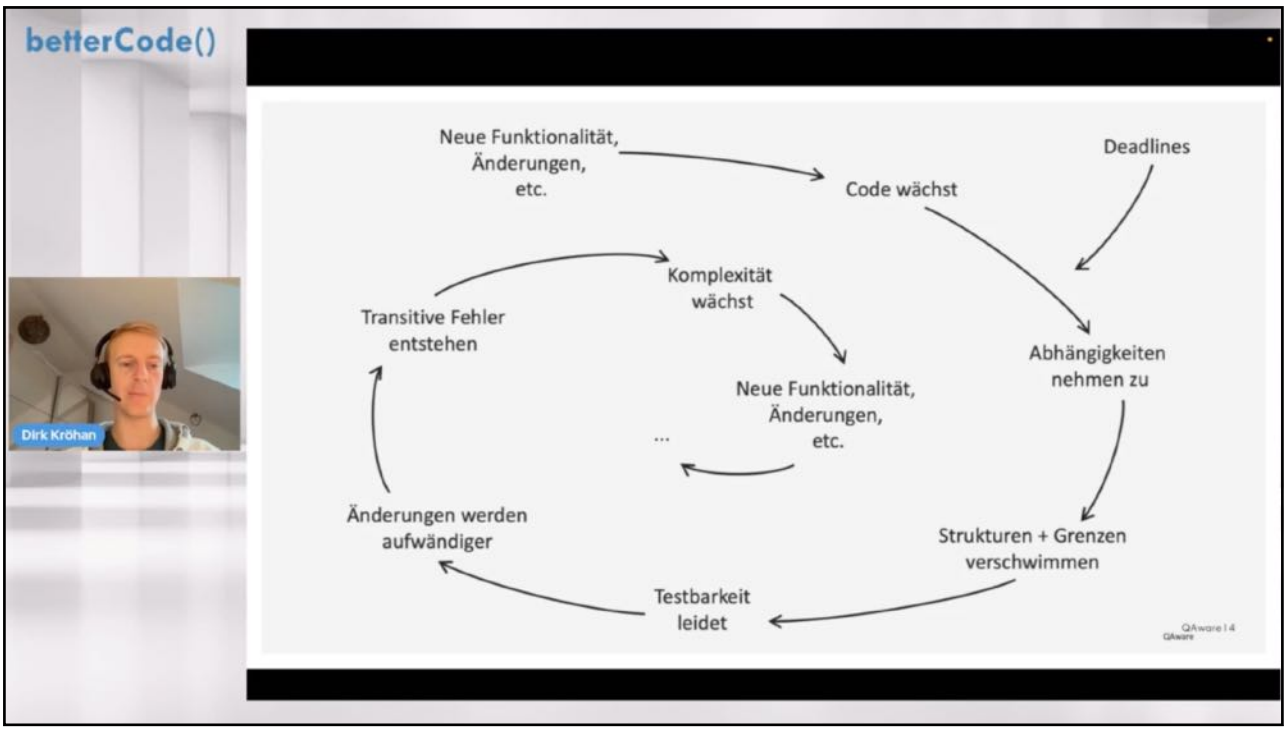
2 Zufällige Architektur

3 Architekturvision

4 Ausblick

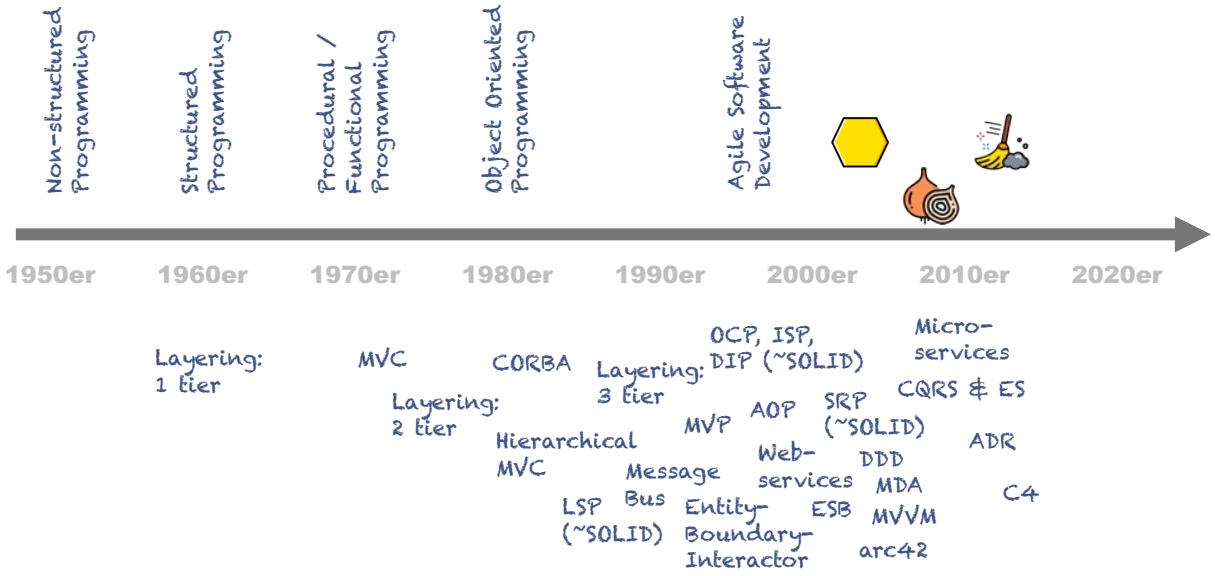
1

10



11

Geschichte



12

Von Ports and Adapters zu Clean Architecture

13

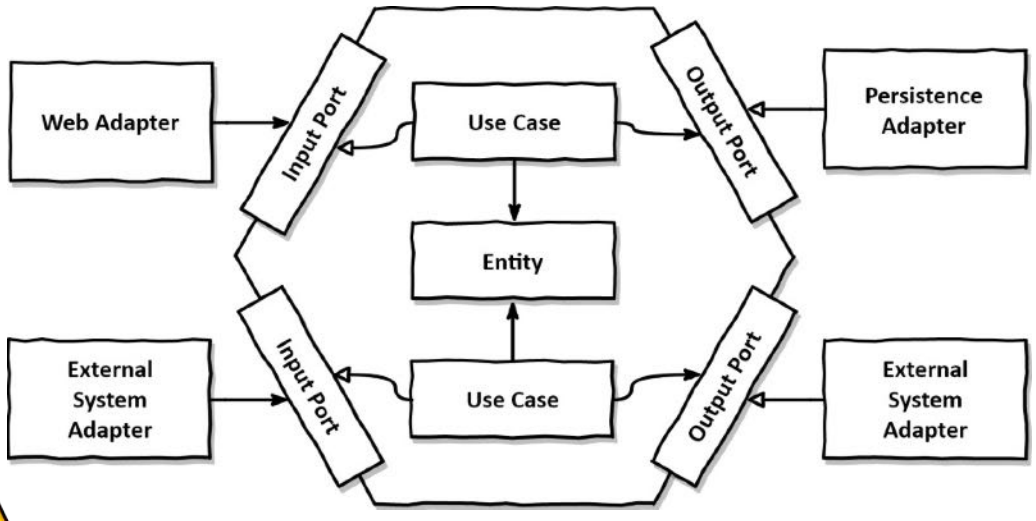
Von Ports and Adapters zu Clean Architecture

Alistair Cockburn

Jeffrey Palermo

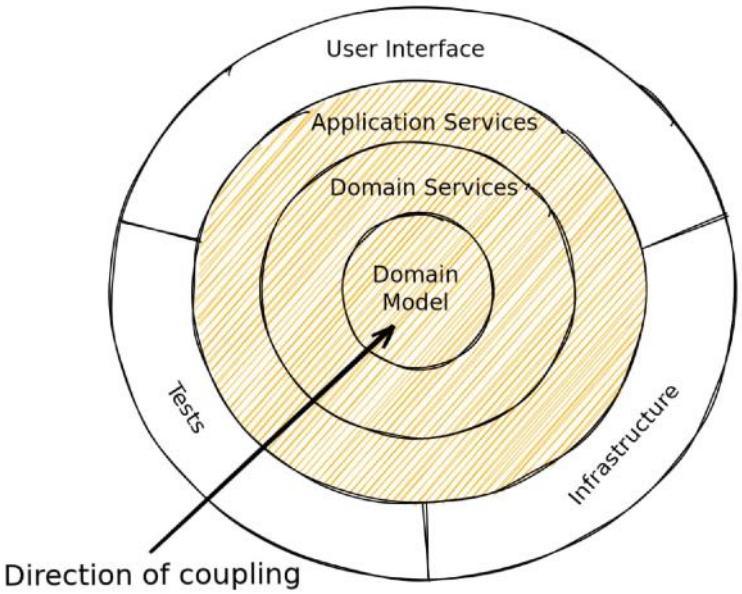
Robert C. Martin

14



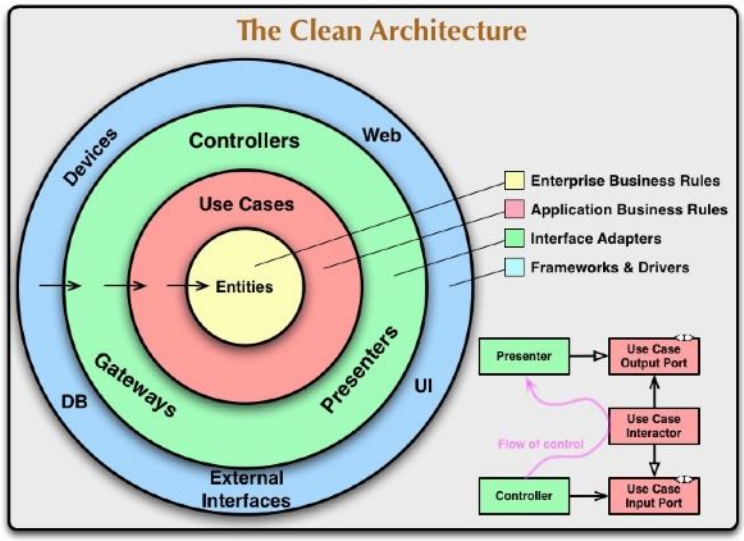
https://www.dineshonjava.com/a-practical-example-of-hexagonal-architecture-in-java/

15



https://dev.to/barrymcauley/onion-architecture-3fgl

16



https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

17

Agenda

1 Clean Architecture

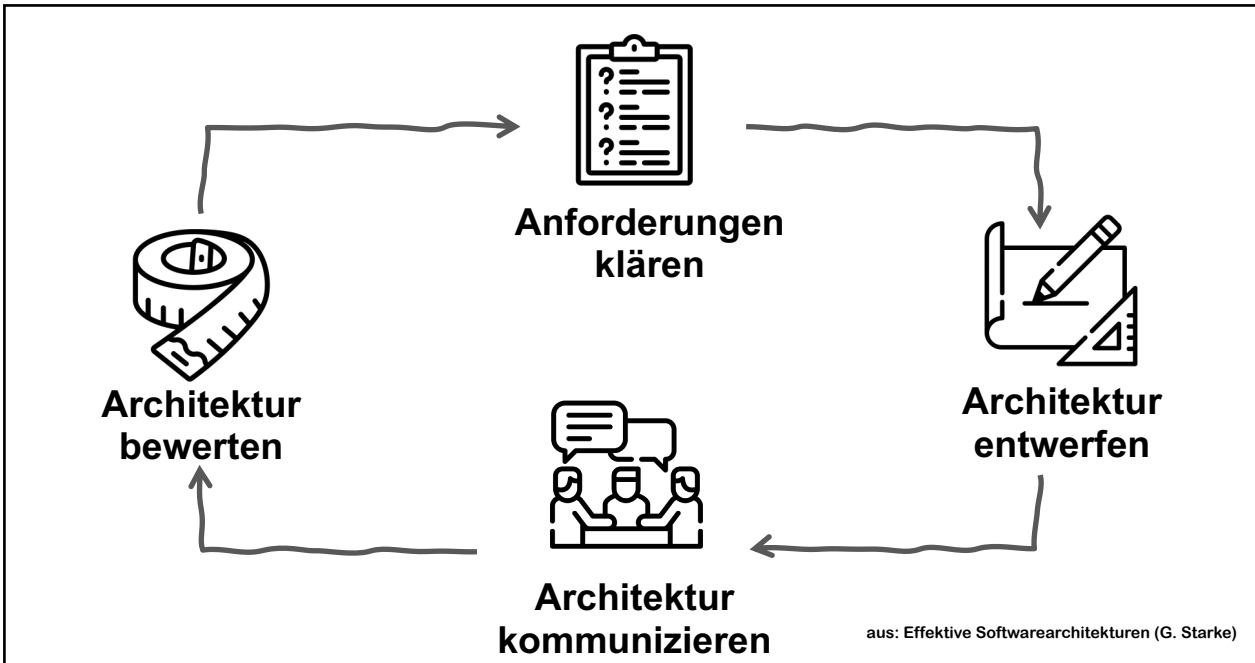
2 **Zufällige Architektur**

3 Architekturvision

4 Ausblick

2

19



aus: Effektive Softwarearchitekturen (G. Starke)

20

Definition

21

Was ist Softwarearchitektur?

Softwarearchitektur :=

$\sum$ wichtige Entscheidungen

wichtig :=

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

22

Softwarearchitektur

*Softwarearchitektur ist die Menge der **Entwurfsentscheidungen**, die, wenn falsch getroffen, Dein Projekt zum **Scheitern bringen** kann.**

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

Eoin Woods

23

Softwarearchitekt:in

*... das Leben eines **Softwarearchitekten** ist eine lange und schnelle Abfolge von **suboptimalen Designentscheidungen**, die teilweise im **Dunkeln getroffen** werden.*

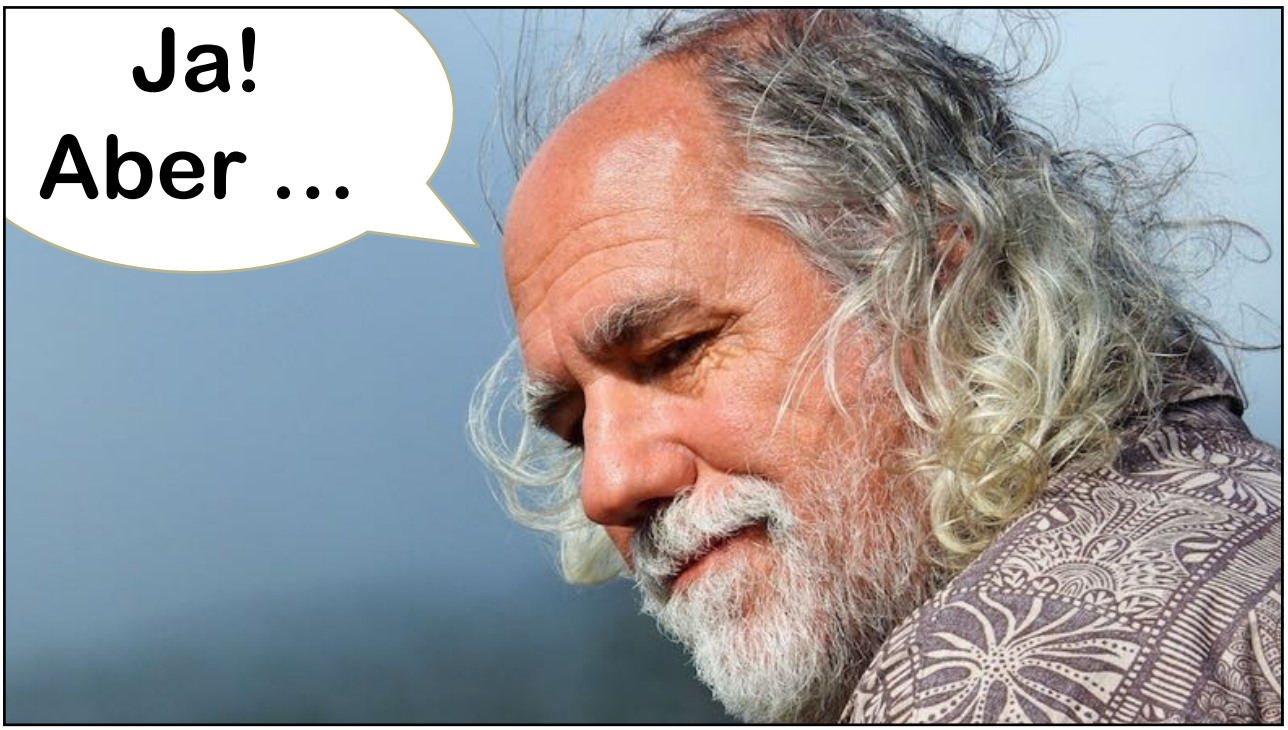
* Wörtlich: "… the life of a software architect is a long and rapid succession of suboptimal design decisions taken partly in the dark."

Philippe Kruchten

25

Gibt es eigentlich immer eine Architektur?

28



29

Accidental Architecture

Grady Booch:
„The Accidental Architecture“
IEEE Software 2006

vs.

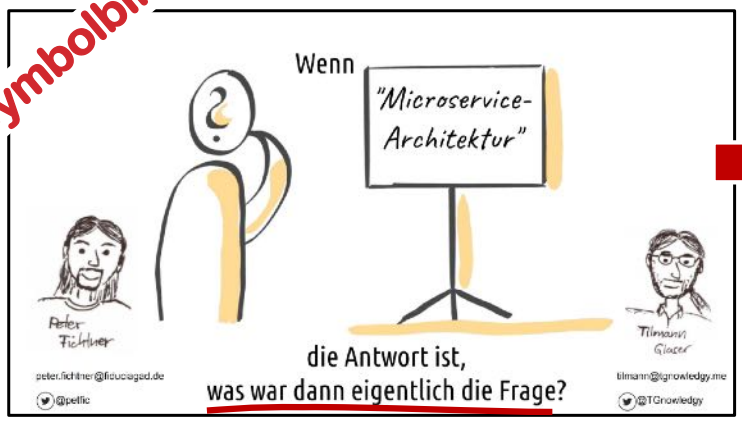
zufällig gewollt

https://www.inf.ed.ac.uk/teaching/courses/seoc/2006_2007/resources/Arc_Accidental.pdf

30

Bewusst oder unbewusst?

Symbolbild



Anforderungen kennen!!!

https://entwicklertag.de/karlsruhe/2021/sites/entwicklertag.de.karlsruhe.2021/files/folien/Wenn_Microservices_die_Anwort_sind.pdf

31

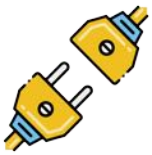
Typische Probleme bei zufälliger Architektur

Geringe Performance

Eingeschränkte Skalierbarkeit

Niedrige Zuverlässigkeit

Schlechte Wartbarkeit



Fehlende Kompatibilität

Nicht ausreichende Sicherheit

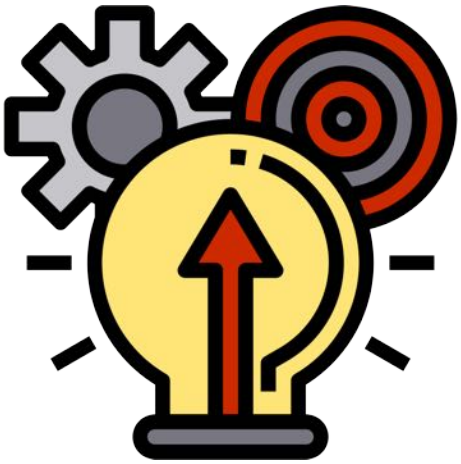
Schlechte Benutzbarkeit

32

Vor der Wand …

34

Systematisches Vorgehen hilft …



35

Agenda

1 Clean Architecture

2 Zufällige Architektur

3 **Architekturvision**

4 Ausblick

3

36

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

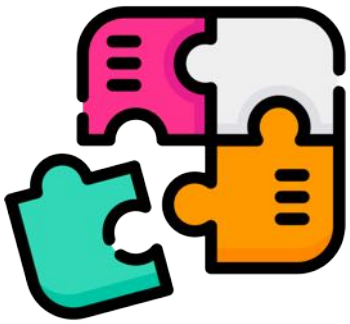
Stefan Toth

38

Architekturvision statt BUFD

(Top down ~~vs.~~ & Bottom Up)

Anforderungen/
Architekturziele



Lösungsansätze/
Entscheidungen

39

Zutaten


Systemkontext

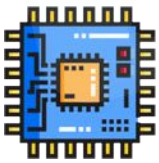

Metapher
“Produktkarton“


Entscheidungen


Architektur-Stil/-
Muster/…


Vorgaben


Qualitätsziele


Technologien


Informeller
Überblick

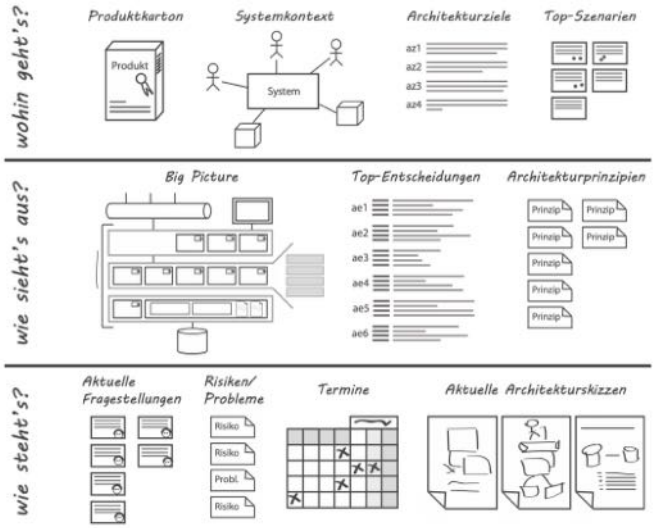
41

arc42 vs. Architekturüberblick

vs.

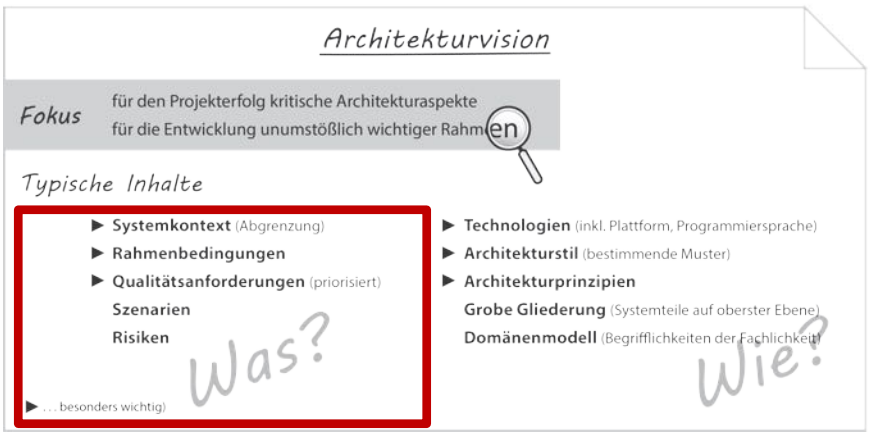
42

Inhalte einer "Architekturwand"



43

Architekturvision statt BUFD



Als Idee, nicht als irreversible Entscheidung!

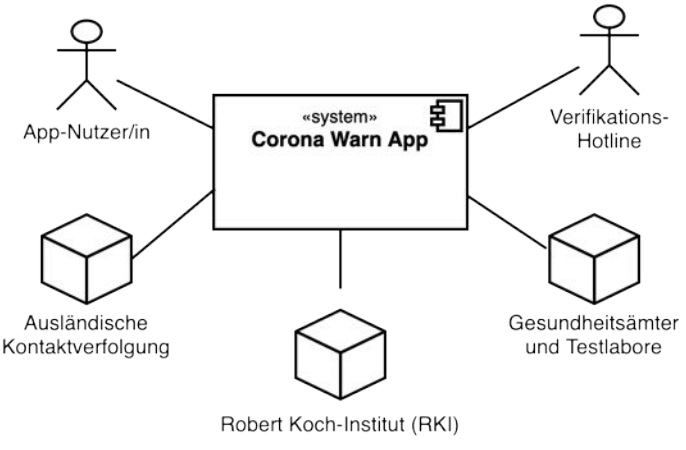
44

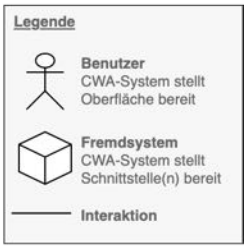
Systemkontext/Kontextabgrenzung

45

Kontextabgrenzung

Dieser fachliche Systemkontext zeigt das Corona-Warn-App-System im Zusammenspiel mit den wichtigsten Benutzern und Fremdsystemen.





46

Fachlicher Systemkontext, Akteure

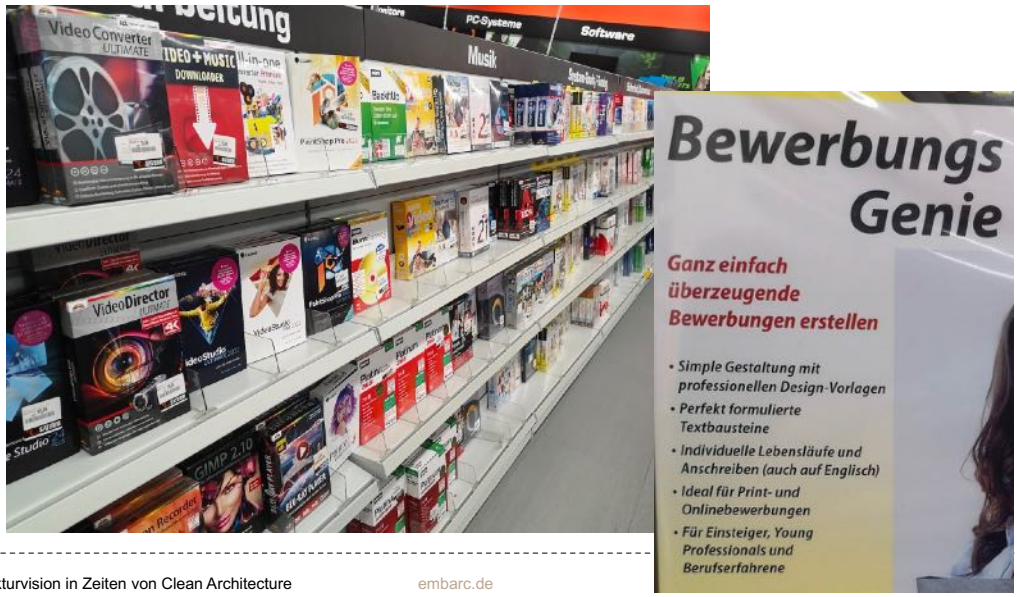
Kurze Erläuterungen zu den Benutzern und Fremdsystemen

Akteur	Beschreibung
App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse (“teleTAN").
Gesundheitsämter und Testlabore	Liefern anonymisierte Testergebnisse an das System.
Robert Koch-Institut (RKI)	Stellt Inhalte ("Content") für die App zur Verfügung und bestimmt Parameter für die Messung der Kontakte ("Risiko-Ermittlung"). Empfängt Auswertungen, etwa aus der Datenspende.
Ausländische Kontaktverfolgungen	Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.

Systemkontext – Gefahren

- **Doppelimplementierung**
- **wichtige Akteure ignoriert**
- **Schnittstellen vergessen**
- **Einstieg neuer Mitarbeiter erschwert**
- **fehlende Transparenz bei allen Projektbeteiligten**

Mission Statement aka Produktkarton



Mission Statement

CORONA
WARN-APP

Die **Corona-Warn-App** ist eine App, die hilft, **Infektionsketten** des SARS-CoV-2 (COVID-19-Auslöser) in Deutschland **nachzuverfolgen** und zu **unterbrechen**.

Die App basiert auf Technologien mit einem **dezentralisierten Ansatz** und informiert Personen, wenn sie mit einer infizierten Person in Kontakt standen.

Transparenz ist von entscheidender **Bedeutung**, um die Bevölkerung zu schützen und die **Akzeptanz zu erhöhen**.

Quelle: https://www.coronawarn.app/de/

Architekturziele

51

CWA – Konkrete Entscheidungen

Zerlegung

Client/Server mit Apps und Backend-Server als verteilte Menge einzeln deploybarer Services, **fachlich zerlegt** (Test-ergebnisse, Verifikation ...)

Zielumgebung

Clients: **Smartphones** der Endnutzer, Backend: **Kubernetes in der Telekom-Cloud** …

Technologie-Stack

Native Apps in Swift und Kotlin auf iOS und Android.
Backend in **Java mit Spring Boot**, Postgres, …

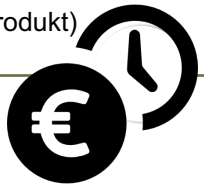
Vorgehen

Entwicklung als **Open Source**, Quelltexte in **GitHub**, **Dokumentation** in Markdown, automatisierte **Tests**
...

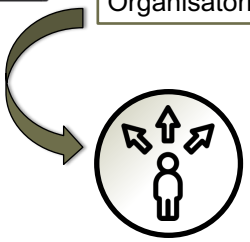
52

Einflüsse auf Entscheidungen

Rahmenbedingungen
Technisch (z.B. Datenbankprodukt)
Organisatorisch (z.B. Team)

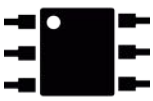


- schränken die Lösung ein
- schließen Optionen aus



53

Zentrale Rahmenbedingungen (CWA)



Technisch

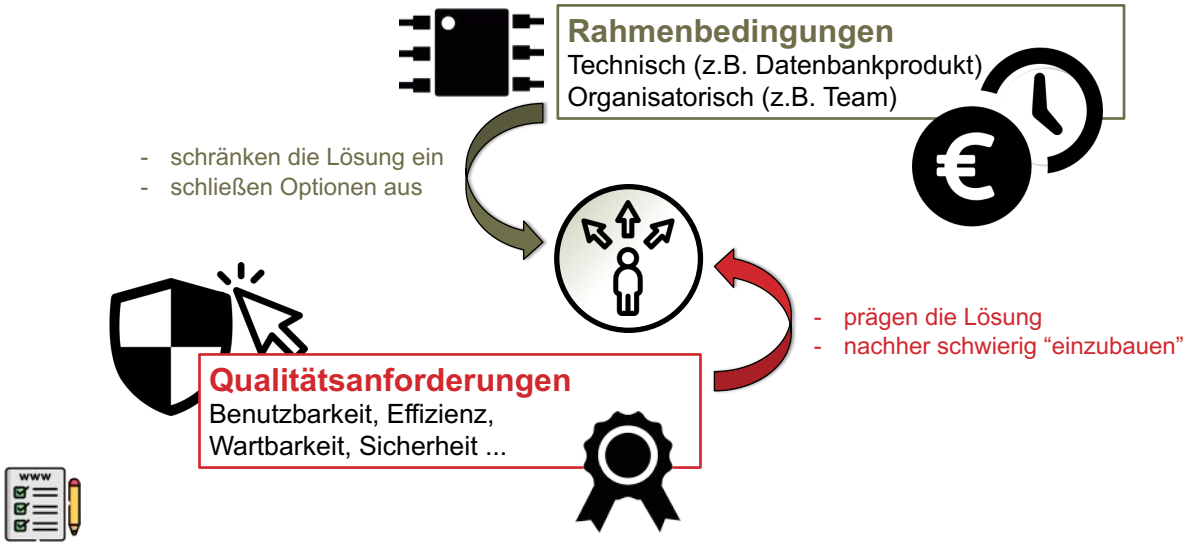
- Betrieb in der **Cloud**
- **Native mobile** Clients
- Einsatz des **Exposure Notification Framework**

Organisatorisch

- **Große Medienaufmerksamkeit**, gewisse **Skepsis** am Mehrwert innerhalb der Bevölkerung
- Konsortium aus **zwei Auftragnehmern** (SAP und Deutsche Telekom)
- **Enger Zeitrahmen**
- Hoher **politischer Druck**, viele Parteien involviert (Ministerien, Behörden, RKI)
- **Hohe Datenschutzanforderungen**

54

Einflüsse auf Entscheidungen



55

Qualitätsmerkmale

Begriffe nach ISO 25010

Benutzbarkeit (Usability)
Ist die Software intuitiv zu bedienen, leicht zu erlernen, attraktiv?

Portabilität (Portability)
Ist die Software leicht auf andere Zielumgebungen (z.B. anderes OS) übertragbar?

Funktionale Eignung (Functional Suitability)
Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen? ...

Effizienz (Performance)
Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch? ...

Kompatibilität (Compatibility)
Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen?

Zuverlässigkeit (Reliability)
Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt? ...

Sicherheit (Security)
Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...

Wartbarkeit (Maintainability)
Ist die Software leicht zu ändern, erweitern, testen, verstehen? Lassen sich Teile wiederverwenden? ...

56

Top-Qualitätsziele Corona-Warn-App

	Ziel	Beschreibung
	Höchster Datenschutz	Der Schutz der personenbezogenen Daten hat oberste Priorität. *(Sicherheit)*
	Effektive Warnfunktionalität	Die App ist ein effektiver Baustein bei der Pandemie-Bekämpfung. *(Funktionale Eignung)*
	Attraktive Lösung für App-Nutzer	Die App ist leicht zu installieren sowie intuitiv und effizient zu bedienen. *(Benutzbarkeit)*
	Hohe Zuverlässigkeit	Die Lösung geht mit Lastspitzen wegen hoher Nutzer- oder Infektionszahlen ebenso souverän um, wie mit böswilligen Angriffen. *(Zuverlässigkeit)*
	Gute Änderbarkeit	Die Software lässt sich leicht anpassen, wenn z. B. Nutzer/-innen oder neue Forschungsergebnisse es erfordern. *(Wartbarkeit/Erweiterbarkeit)*

Die Reihenfolge gibt Orientierung bezüglich der Wichtigkeit.

57

Qualitätsszenarien brainstormen

In Präsenz:
Post-its

Remote:
Digitale Whiteboards
(miro, Mural, …)

miro

MURAL

58

Was ist ein Szenario?

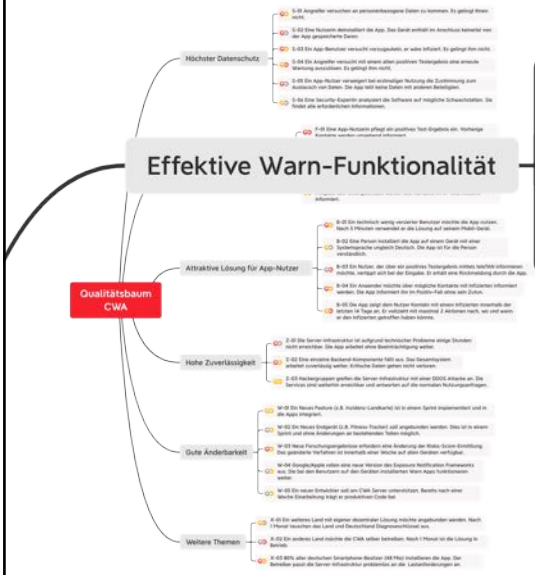
Ein Qualitätsszenario (auch: Bewertungsszenario) ...

- ... ist ein kurzer Text (1-3 Sätze).
- ... beschreibt **beispielhaft** die Verwendung des Systems, und zwar so dass ein **Qualitätsmerkmal** die Hauptrolle spielt.

Wie konkret?

Man muss sinnvoll drüber reden können.
Man muss es (theoretisch) überprüfen können.
(Kein Abnahmekriterium, kein Testfall!)

Qualitätsbaum CWA, priorisiert



F-01 Eine App-Nutzerin pflegt ein positives Test-Ergebnis ein. Vorherige Kontakte werden umgehend informiert.

F-02 Ein App-Nutzer hatte Kontakt mit einem später positiv Getesteten. Die App informiert ihn umgehend über das Risiko.

F-03 Ein App-Nutzerin trifft auf einen anderen App-Nutzer. Ein Gerät befindet sich im Idle-Modus. Die App speichert die Begegnung.

F-04 Ein App-Nutzer wurde im Ausland positiv getestet. Nach Einspielen und Freigabe des Testergebnisses werden alle Kontakte im In- und Ausland informiert.

Legende für Priorität

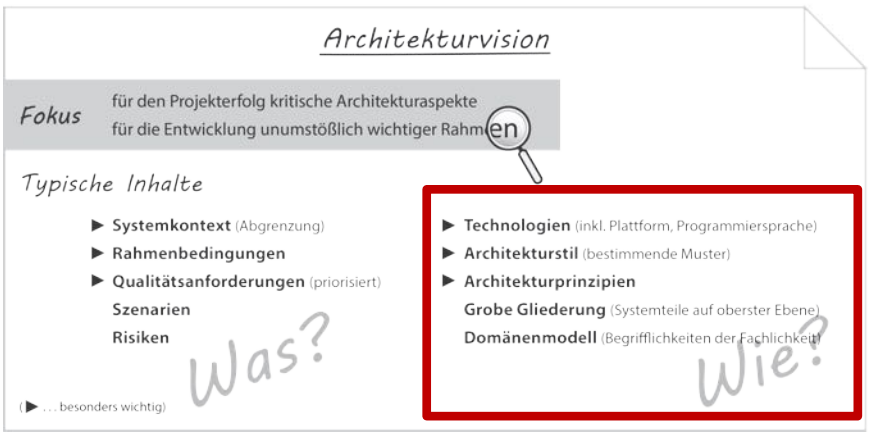
Fachliche Wichtigkeit
Technische Schwierigkeit

Rot: Hoch
Gelb: Mittel
Grau: Niedrig

Risiken, technische Schulden

61

Architekturvision statt BUFD



Als Idee, nicht als irreversible Entscheidung!

Toth, Stefan: "Vorgehensmuster für Softwarearchitektur" Carl Hanser Verlag, 2014

62

Lösungsstrategie Corona-Warn-App

Ziel	Passende Lösungsansätze (Auswahl)
Höchster Datenschutz	• Speicherung der **Daten lokal** • **Verschlüsselung** aller Bewegungsdaten • Senden der Daten nur **nach Aufforderung** • **Transparente** Entwicklung (Open Source)
Effektive Warnfunktionalität	• Verwendung **Exposure Notification Framework** • **Digitale Abläufe** bevorzugt • Optionales, manuelles **Kontakt-Tagebuch**
Attraktive Lösung für App-Nutzer	• **Native Clients** (L&F) • **Übersichtliche** Gestaltung und **simple** Bedienung
Hohe Zuverlässigkeit	• **Microservices,** Docker, Kubernetes, **Public Cloud** • hohe **Testabdeckung** und **automatisierte Builds** • Bereitstellung von zu lesenden Daten über **CDN**
Gute Änderbarkeit	• Hoher **Modularisierungsgrad** • **Open Source** Projekt**,** gute **Dokumentation** • Verwendung von **Standard** & **Open Source** Libraries • Konsortium von **mehreren Auftragnehmern** • **Code-Qualität** (SonarQube, SwiftLint, Checkstyle, …)

63

Technologie-Stack

- Native Clients in Swift bzw. Kotlin für iOS und Android
- SQLite

- Java 11
- Spring Boot/Cloud/Data
- Lombok, Guava, …
- REST, Protobuf
- OpenAPI, Micrometer
- Liquibase

- Maven, Gradle
- Docker, Kubernetes
- Open Telekom Cloud (OpenStack)
- PostgreSQL, S3, CDN
- Keycloak

64

Architekturstil, -muster

Microservices

Monolith

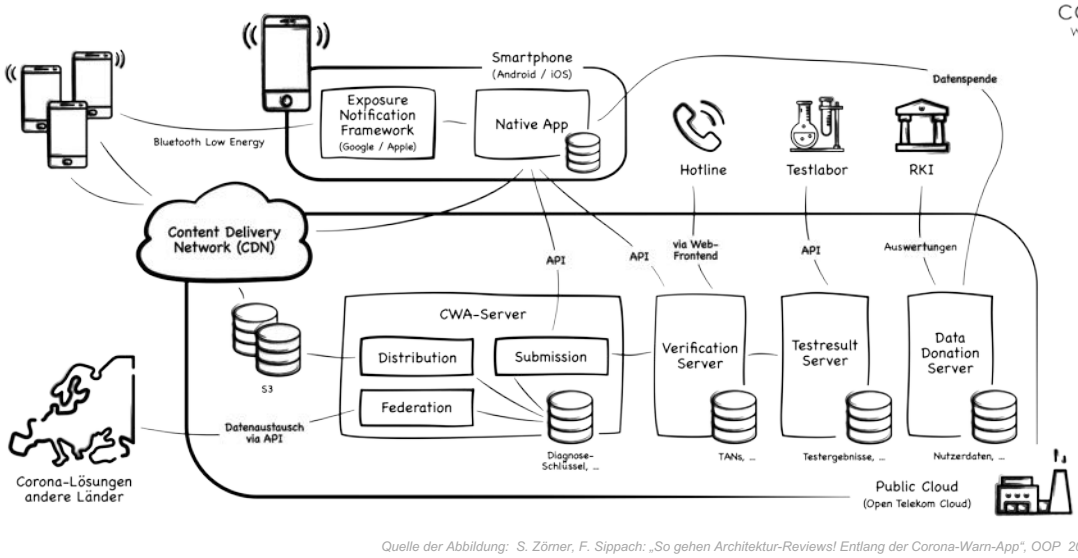
Tiers

Serverless

66

Informelles Überblicksbild

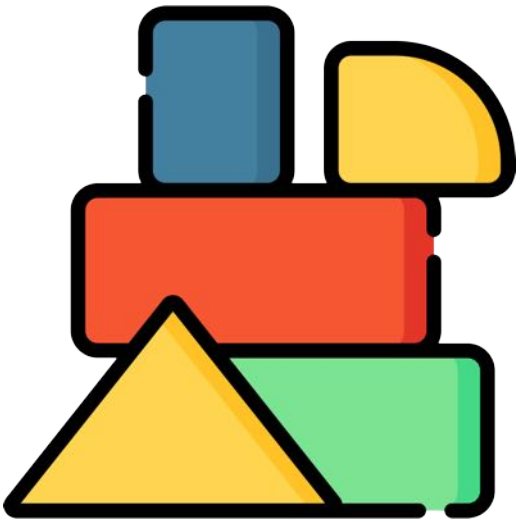
CORONA WARN·APP



Quelle der Abbildung: S. Zörner, F. Sippach: „So gehen Architektur-Reviews! Entlang der Corona-Warn-App", OOP 2021

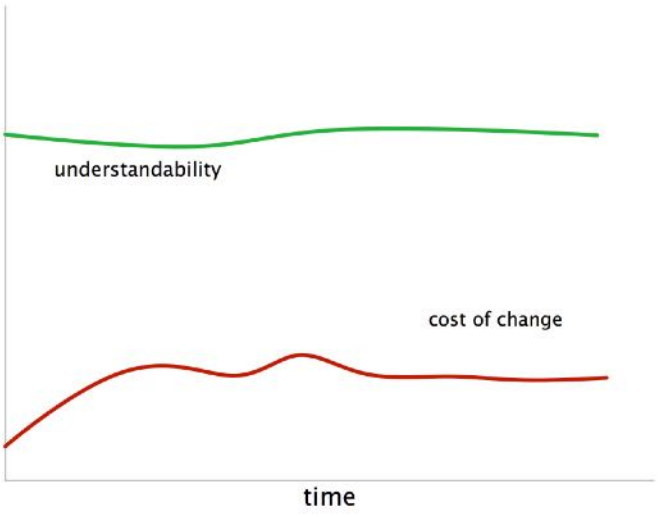
67

Modularisierung



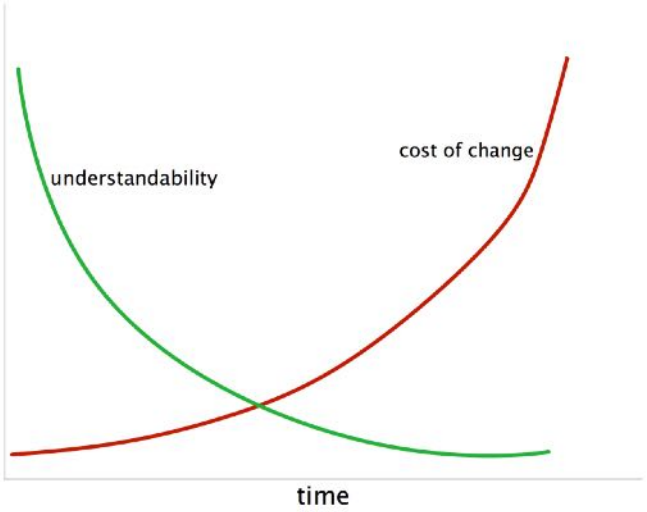
68

Wie oft gesehen?



69

Leider eher …



70

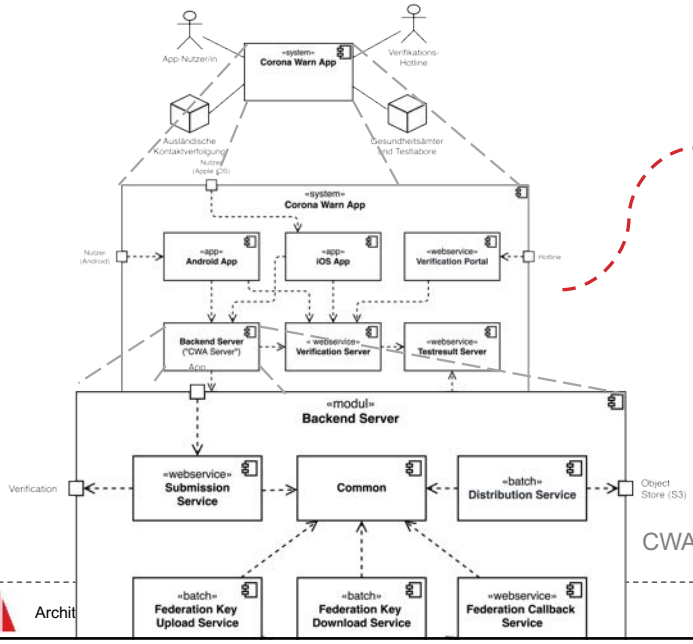
Warum modularisieren?

Vorteile „kleiner“ Software

- Verständlichkeit
- Wiederverwendung
- Lokale Änderungen und Änderbarkeit
- Kleinere „Units of Work“ (Workspace in der IDE, Compile, Test, Deploy)
- Lokale Risiken
- Kleine Teile sind schneller „fertig“ und „man kann was sehen“
- Schneller ersetzbar
- Individuell auszuprägen (technologisch)

71

Tatsächliche Zerlegung im Quelltext



„Bausteinsicht, Ebene 1“

GitHub-Repositories

https://github.com/corona-warn-app/...

- cwa-app-android
- cwa-app-ios
- cwa-server („Backend“)
- cwa-testresult-server
- cwa-verification-server
- cwa-verification-portal

CWA-Server (Backend), Bausteinsicht, Ebene 2

73

Werkzeuge

74

Unser täglich Brot

- Plain-Text
- Entwicklungsumgebung
- Kommandozeilen-werkzeuge
- Versionsverwaltung

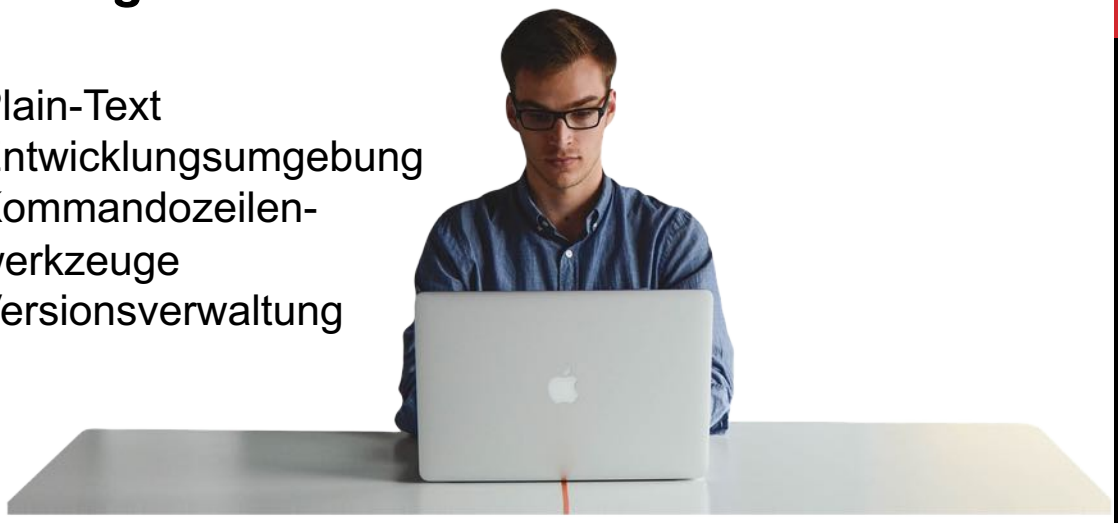
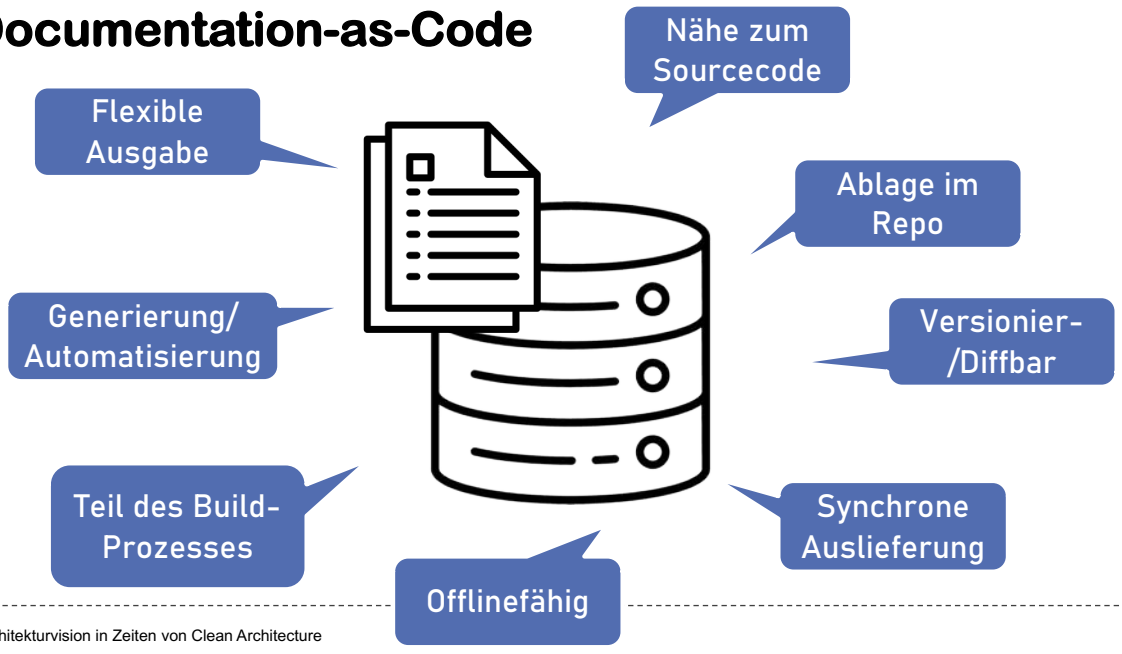


Foto von geralt: https://pixabay.com/de/unternehmer-start-start-up-karriere-696976/ (CC0 Public Domain Lizenz)

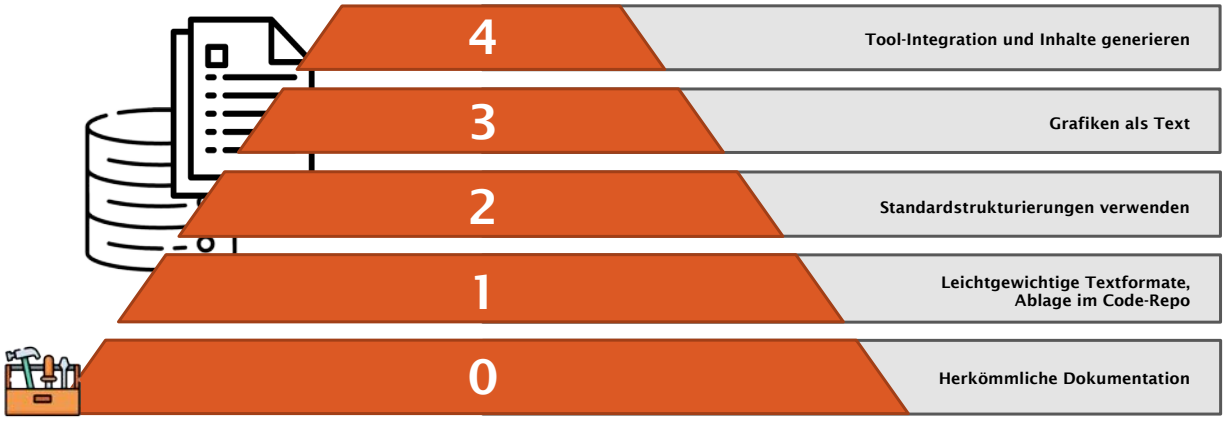
75

Documentation-as-Code



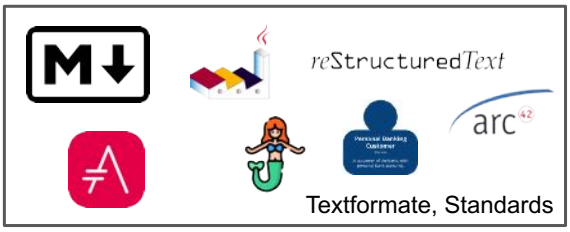
76

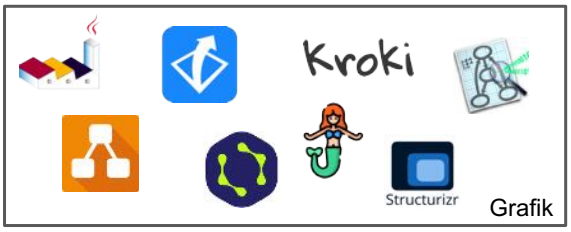
Docs-as-Code Reifegradmodell (Maturity Level)



77

Werkzeuge





78

arc42

1. Einführung und Ziele
1.1 Aufgabenstellung
1.2 Qualitätsziele
1.3 Stakeholder

2. Randbedingungen
2.1 Technische Randbedingungen
2.2 Organisatorische Randbedingungen
2.3 Konventionen

3. Kontextabgrenzung
3.1 Fachlicher Kontext
3.2 Technischer- oder Verteilungskontext

4. Lösungsstrategie

5. Bausteinsicht
5.1 Ebene 1
5.2 Ebene 2
...

6. Laufzeitsicht
6.1 Laufzeitszenario 1
6.2 Laufzeitszenario 2
...

7. Verteilungssicht
7.1 Infrastruktur Ebene 1
7.2 Infrastruktur Ebene 2
...

8. Konzepte
8.1 Fachliche Strukturen und Modelle
8.2 Typische Muster und Strukturen
8.3 Persistenz
8.4 Benutzungsoberfläche
...

9. Entwurfsentscheidungen
9.1 Entwurfsentscheidung 1
9.2 Entwurfsentscheidung 2
...

10. Qualitätsszenarien
10.1 Qualitätsbaum
10.2 Bewertungsszenarien

11. Risiken

12. Glossar


Dr. Peter Hruschka
http://www.peterhruschka.eu/


Dr. Gernot Starke
http://gernotstarke.de/

➔https://arc42.de/

79

Agenda

1 Clean Architecture
2 Zufällige Architektur
3 Architekturvision
4 **Ausblick**

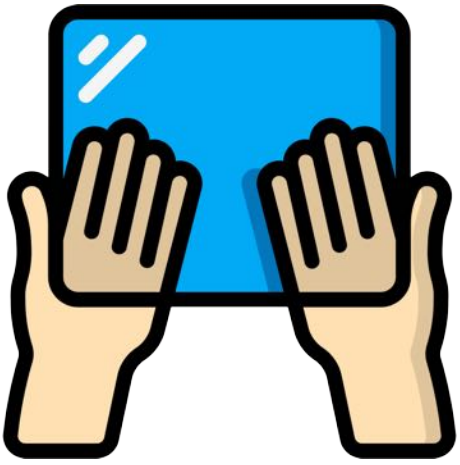
4

80

tl;dr

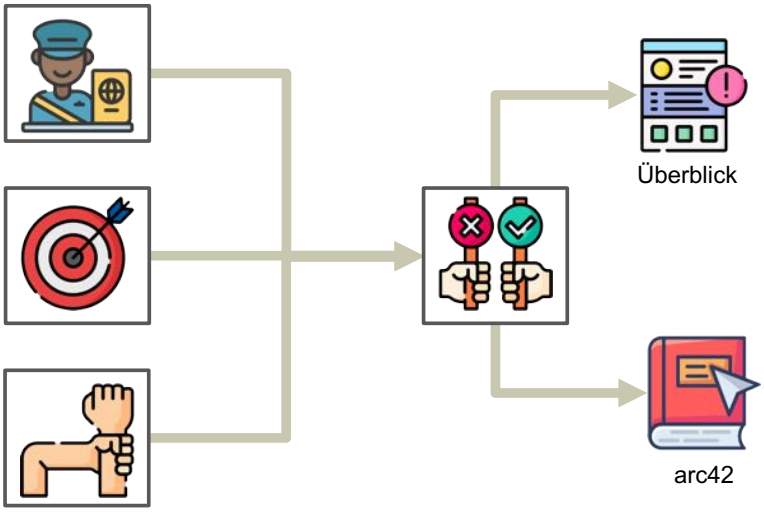
81

It's all about transparency …



82

Von den Anforderungen zu Umsetzung




Technologien

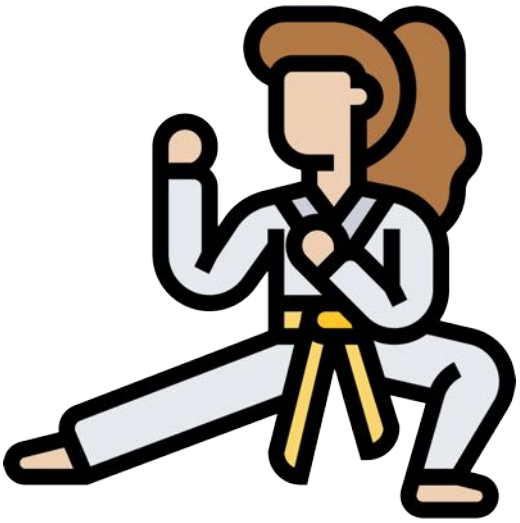

Architektur-Stil/-Muster/…


Informeller Überblick


Zerlegung

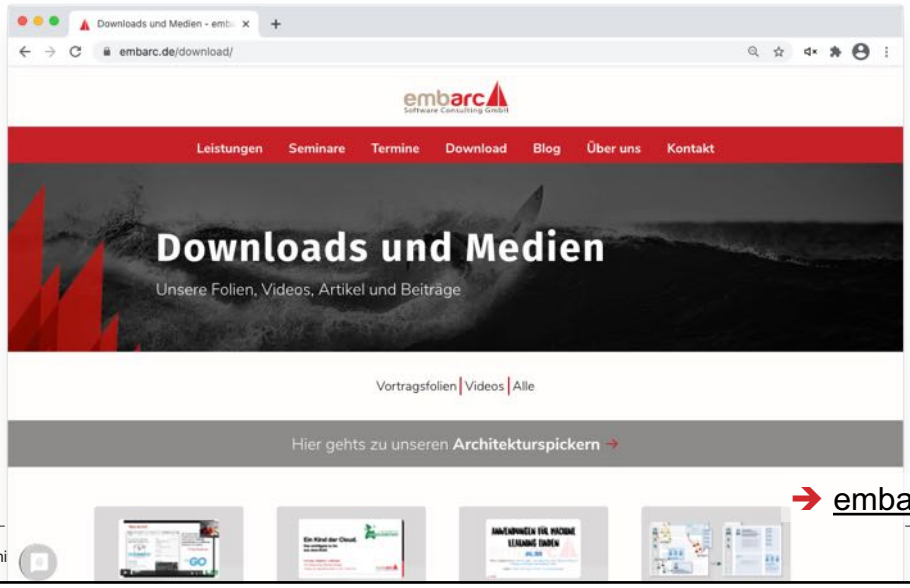
83

Üben mit Architektur-Katas



84

Folien von heute als PDF zum Download



➔ embarc.de/download/

87

Flyer, Folienvortrag unter anderem zur CWA

➔ https://www.embarc.de/architektur-ueberblicke/

88

Spicken erlaubt!

Unsere Architektur-Spicker beleuchten die konzeptionelle Seite der Softwareentwicklung.

➔ embarc.de/architektur-spicker/

89

Mini-Konferenz bei embarc (online)

Kostenlose Anmeldung (gerne eine Spende):

➔ embarc.de/ architektur-punsch-2022/

90

Vielen Dank.

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi

91

Falk Sippach

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

Schwerpunkte:

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

fs@embarc.de

@sippsack

➔ xing.to/fsi

embarc Software Consulting GmbH

92