



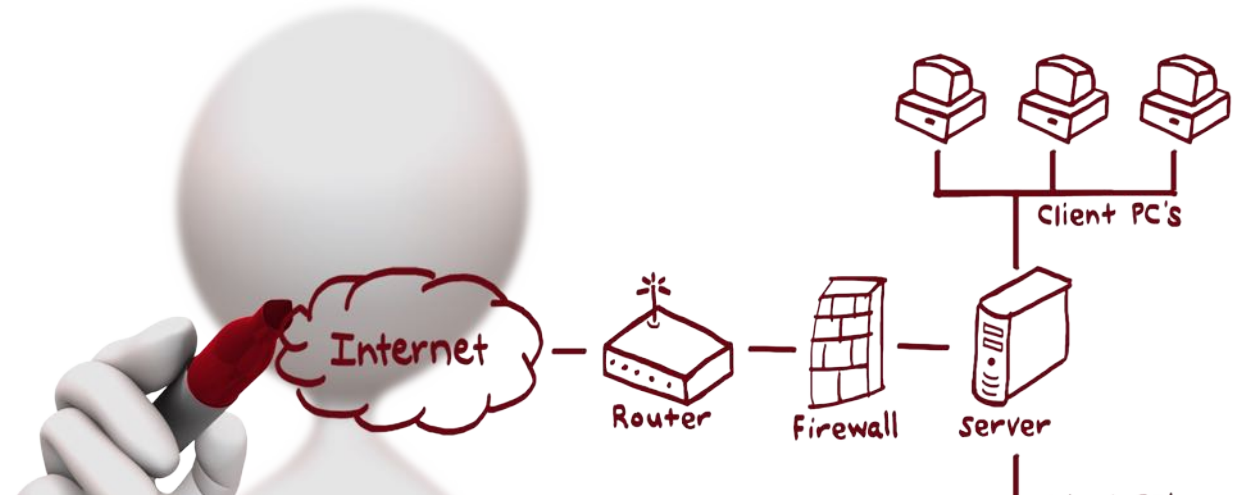
Phantastische Diagramme

...und wie Du sie selbst erstellst

Falk Sippach
embarc

Ralf D. Müller
DB Systel GmbH






The diagram is a hand-drawn network architecture. On the left, a hand holds a red marker, drawing a cloud labeled 'Internet'. A line connects the cloud to a 'Router' (a box with an antenna). Another line connects the Router to a 'Firewall' (a rack-mounted server). A third line connects the Firewall to a 'Server' (a server rack). Finally, a line connects the Server to three 'Client PC's' (desktop computers) arranged in a row.

1

Phantastische Diagramme

...und wie Du sie selbst erstellst



Zusammenfassung

Die wichtigste Aufgabe eines Software-Architekten besteht darin, die **Architektur zu kommunizieren**.

Neben den textuellen Inhalten gilt es auch, Grafiken zu erstellen. Getreu dem Motto "**ein Bild sagt mehr als tausend Worte**" helfen Diagramme bei einer effektiven und pragmatischen Dokumentation. Damit sie wirken, müssen sie leicht erfassbar, stets aktuell und korrekt sein..

In diesem Talk spüren wir zuerst die Schwachstellen vieler Diagramme auf und überlegen uns anschließend, wie wir sie umgehen können. Das Ergebnis ist eine **Checkliste für richtig gute, fast schon phantastische Architektur-Diagramme**.

Als Bonus werden wir aufzeigen, wie **Diagramme wartbar mit dem Docs-as-Code Ansatz umgesetzt werden**, um sie jederzeit aktuell und im Einklang mit der Architektur halten zu können.

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

2

2




Falk Sippach

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



 fs@embarc.de

 [@sipsack](https://twitter.com/sipsack)



 → [xing.to/fsi](https://www.xing.to/fsi)






Ralf D. Müller | @ralfdmueller | DB Systel
Falk Sippach | @sipsack | embarc
4


4






Ralf D. Müller







- was mit Dokumentation, Security und Architektur by der DB Systel
- Maintainer von docToolchain, Contributor arc42
- co-Autor



 ralf.d.mueller@deutschebahn.com

 [@ralfdmueller](https://twitter.com/ralfdmueller)

 → [xing.to/rdmuller](https://www.xing.to/rdmuller)

Ralf D. Müller | @ralfdmueller | DB Systel
Falk Sippach | @sipsack | embarc
5

5

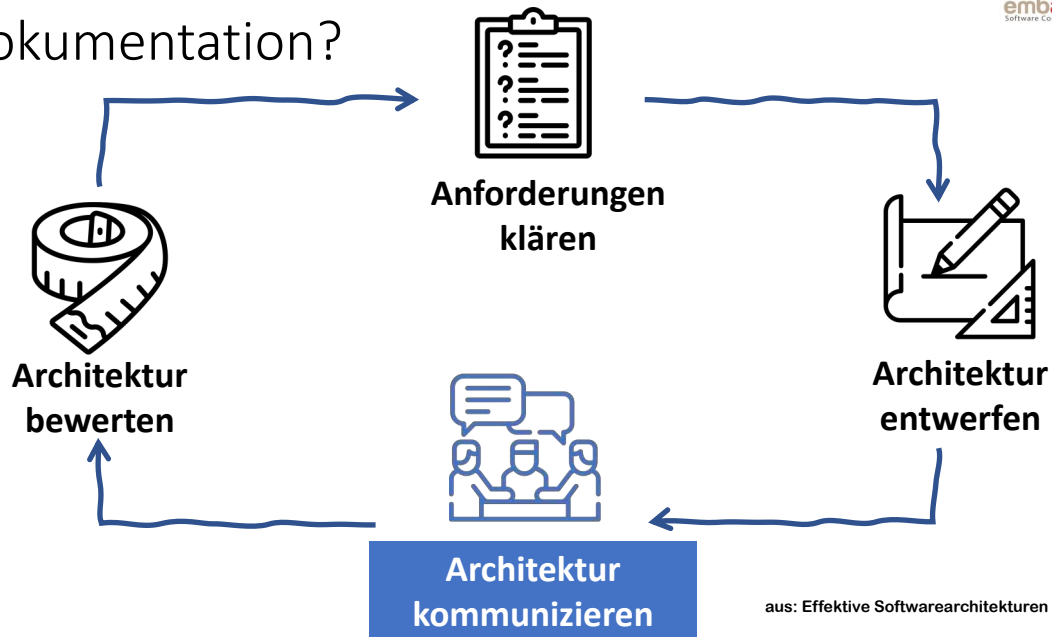
Mission Statement



- **Schwachstellen** vieler Diagramme aufzeigen.
- Checkliste für **richtig gute Architekturdiagramme** diskutieren.
- Für Architekturdokumentation **relevante Diagrammart** vorstellen.
- Diagramme **wartbar mit Docs-as-Code** Ansatz umsetzen.
- **Leichtgewichtige Werkzeuge** zur Diagrammerstellung zeigen.

7

Dokumentation?



8

Gute Gründe ...

embarc Software Consulting GmbH DB

Where's the fun in just knowing what the code is supposed to do?



Essential
Excuses for Not Writing Documentation
RLY? @ThePracticalDev

Grafik von [The Practical Dev](#) (CC0 Public Domain Lizenz)

Ralf D. Müller | @ralfdmueller | DB Systel Falk Sippach | @sippack | embarc

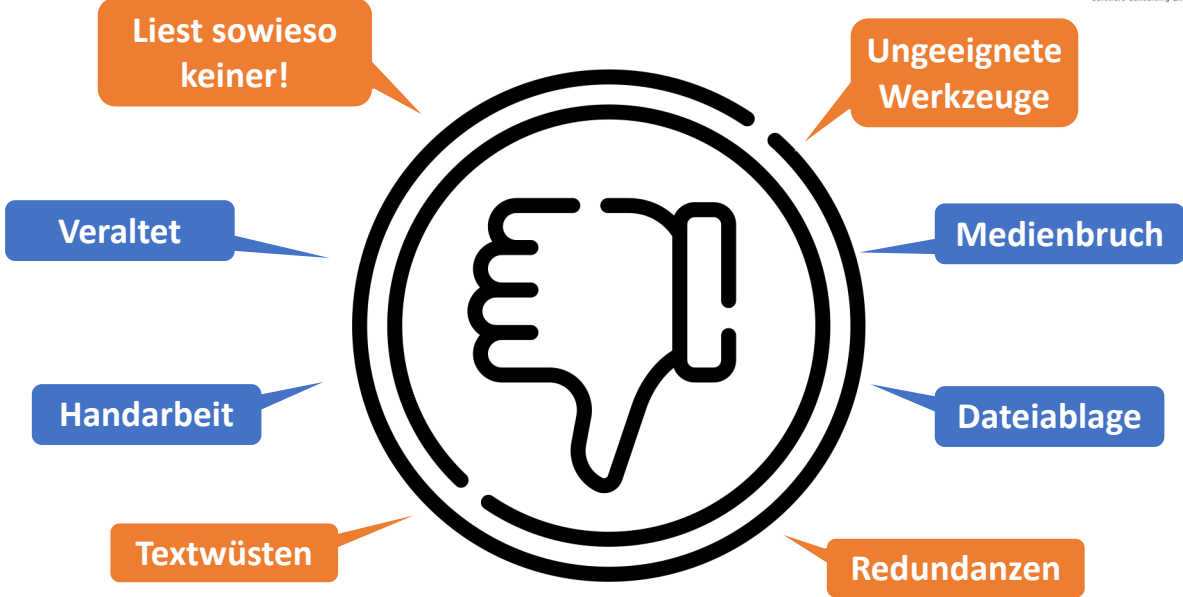
9

- Entwurfsunterstützung (Icon: drawing tool)
- Frage nach dem Warum (Icon: signpost)
- Bewertbarkeit (Icon: measuring tape)
- Kommunikation (Icon: group of people)
- Neue Mitarbeiter (Icon: person with code symbol)

9

Helfen uns hier Diagramme?

embarc Software Consulting GmbH DB

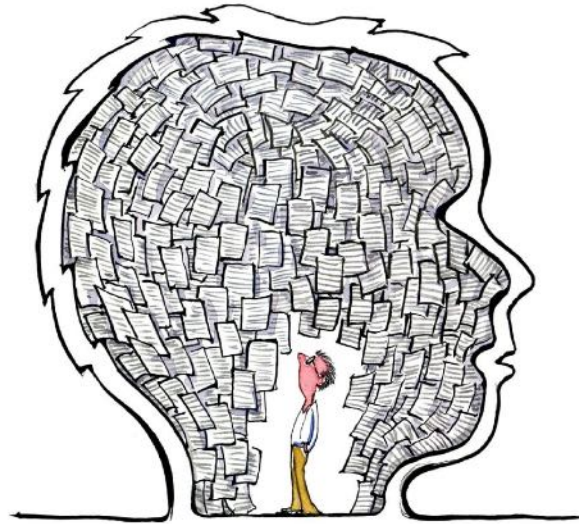


- Liest sowieso keiner!
- Ungeeignete Werkzeuge
- Medienbruch
- Dateiablage
- Redundanzen
- Textwüsten
- Handarbeit
- Veraltet

Ralf D. Müller | @ralfdmueller | DB Systel Helfen uns hier Diagramme? 10

10

Ein Bild sagt mehr als 1000 Worte ...



11

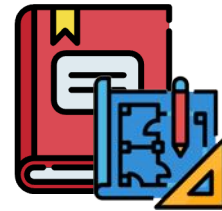
Agenda



Gute Diagramme



Werkzeuge



**Einbettung in
Dokumentation**

12

Agenda



Gute Diagramme



Werkzeuge



Einbettung in Dokumentation

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

13

13

Einleitung



Qualitätskriterien

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

14

14

Tipp 1: Qualitätskriterien für Diagramme

Ein Diagramm dient der **Kommunikation**

Darum muss es **lesbar und verständlich** sein

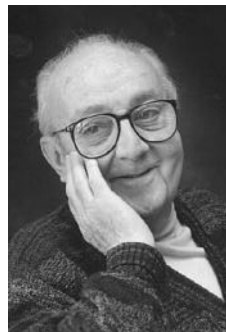
Es muss die gewünschte **Information transportieren**

Denke an einen **Rückkanal**, sonst handelt es sich um einseitige Kommunikation

Denke an die **Wart- und Modifizierbarkeit**



„All Models are
Wrong but some
are Useful“



George Box

Was ist ein Modell?



Bild von [Hans Benn](#) auf [Pixabay](#)

Was ist ein Modell?



Bild von [Hans Benn](#) auf [Pixabay](#)



Tipp 2: Zweck des Diagramms beachten

Ein Diagramm / eine View erfüllt einen **einzigen Zweck**

Widerstehe der Versuchung **zu viele Arten von Informationen** in einem Diagramm unterzubringen

Ja, betrachtet man das Diagramm aus der **falschen Sicht, wird es falsch sein**

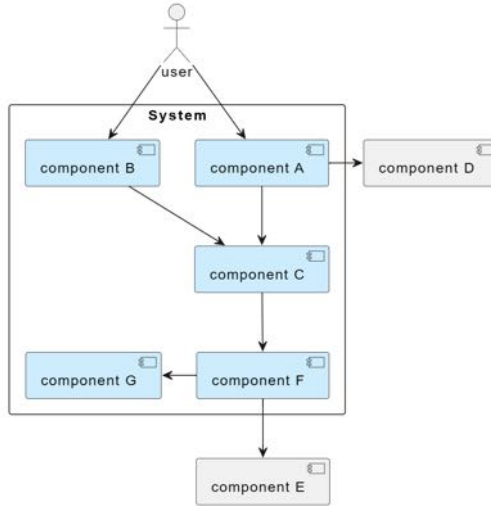
Darum **beschrifte das Diagramm**, um den Zweck klarzustellen



Visual Style

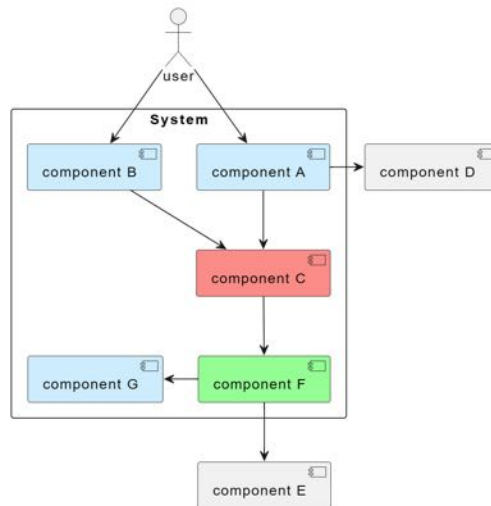


Ein Beispiel

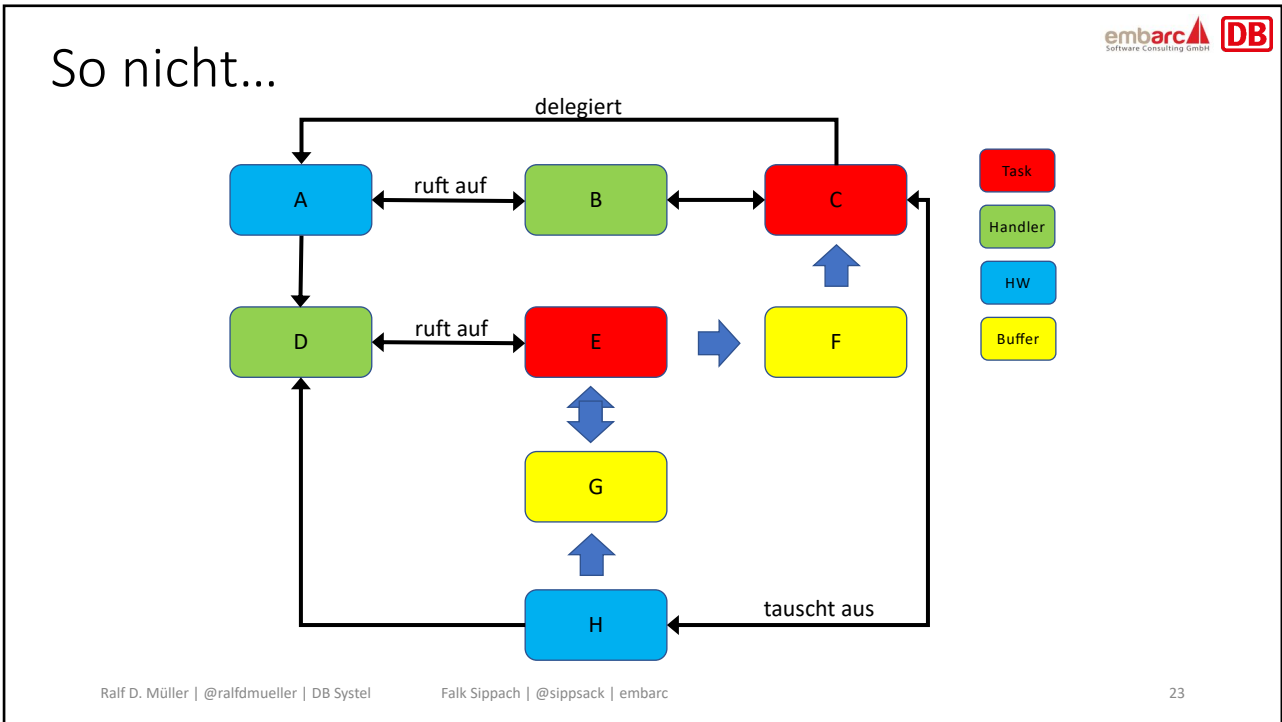


21

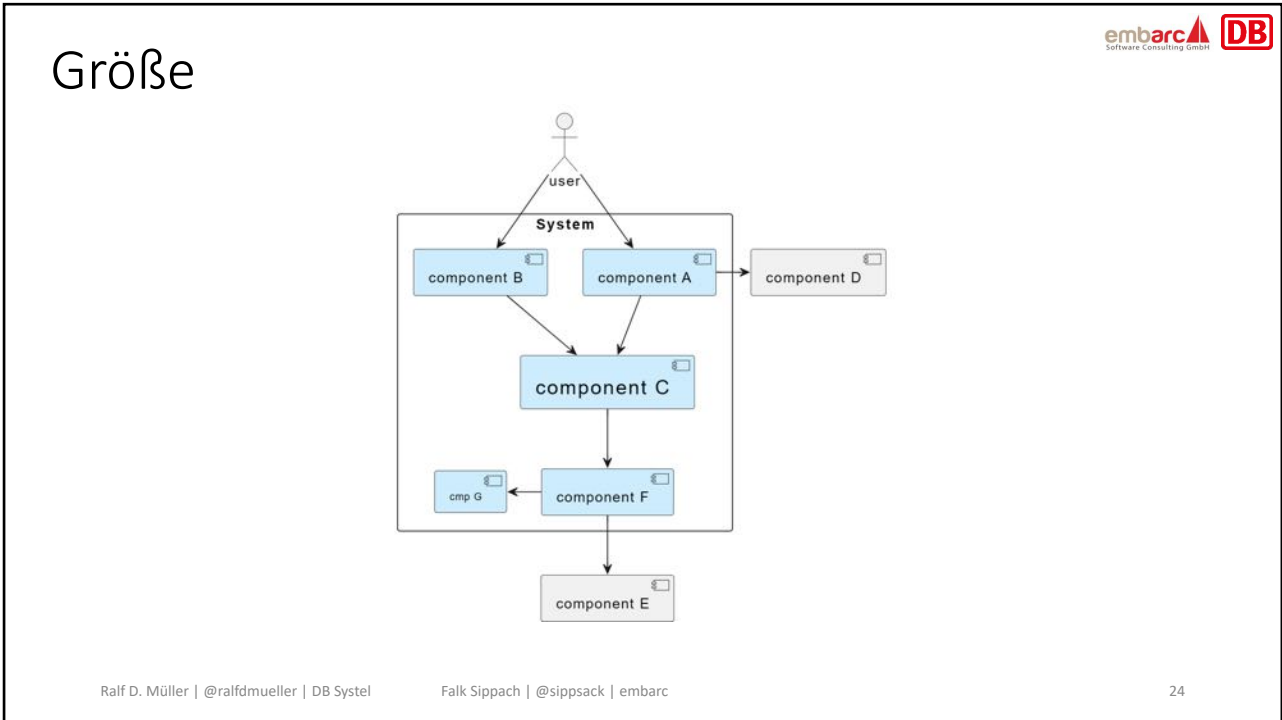
Farben



22

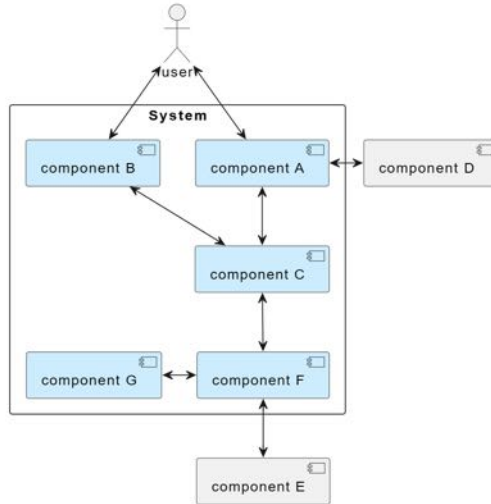


23



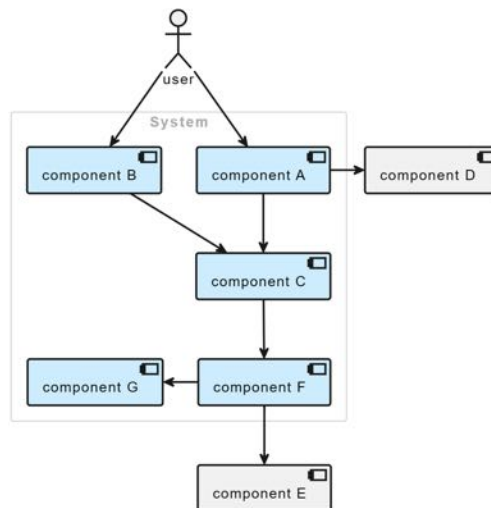
24

Pfeilrichtung



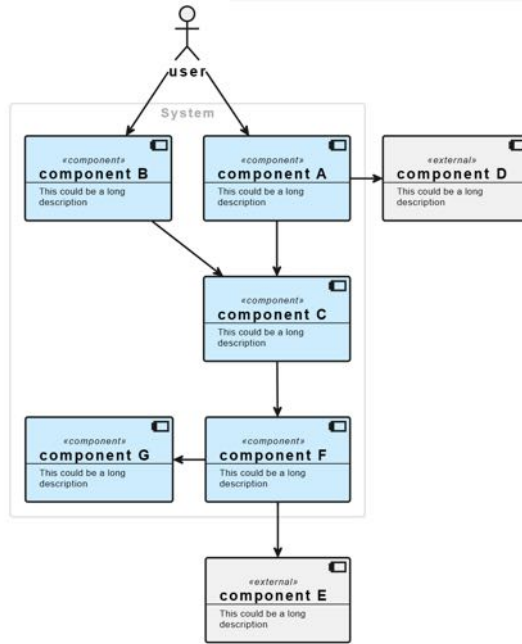
25

Linienstärke



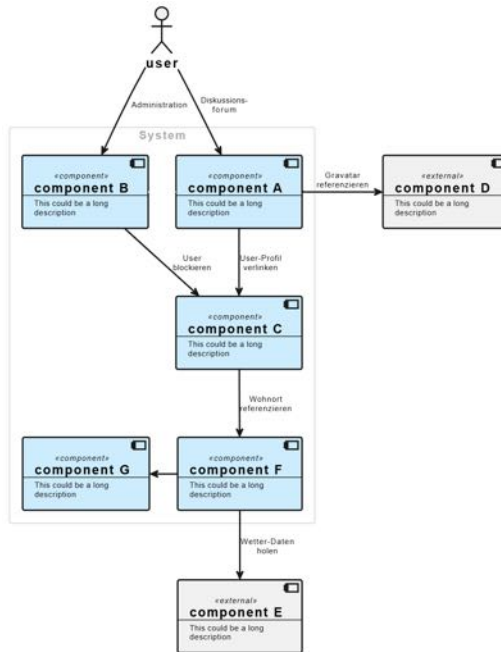
26

Text-Stile



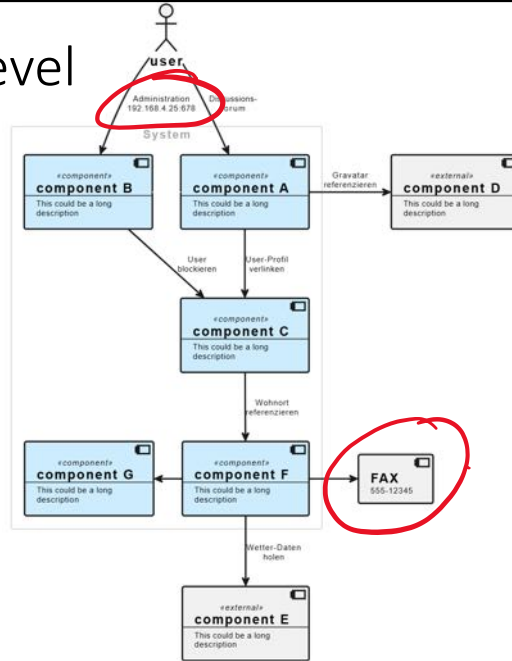
27

Beschriftung



28

Abstraktionslevel



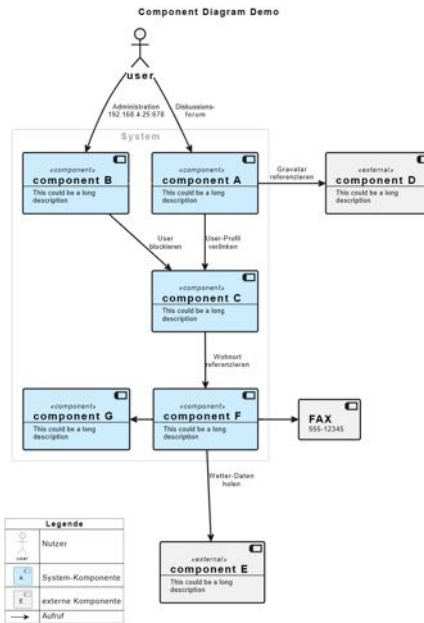
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

29

29

Legende



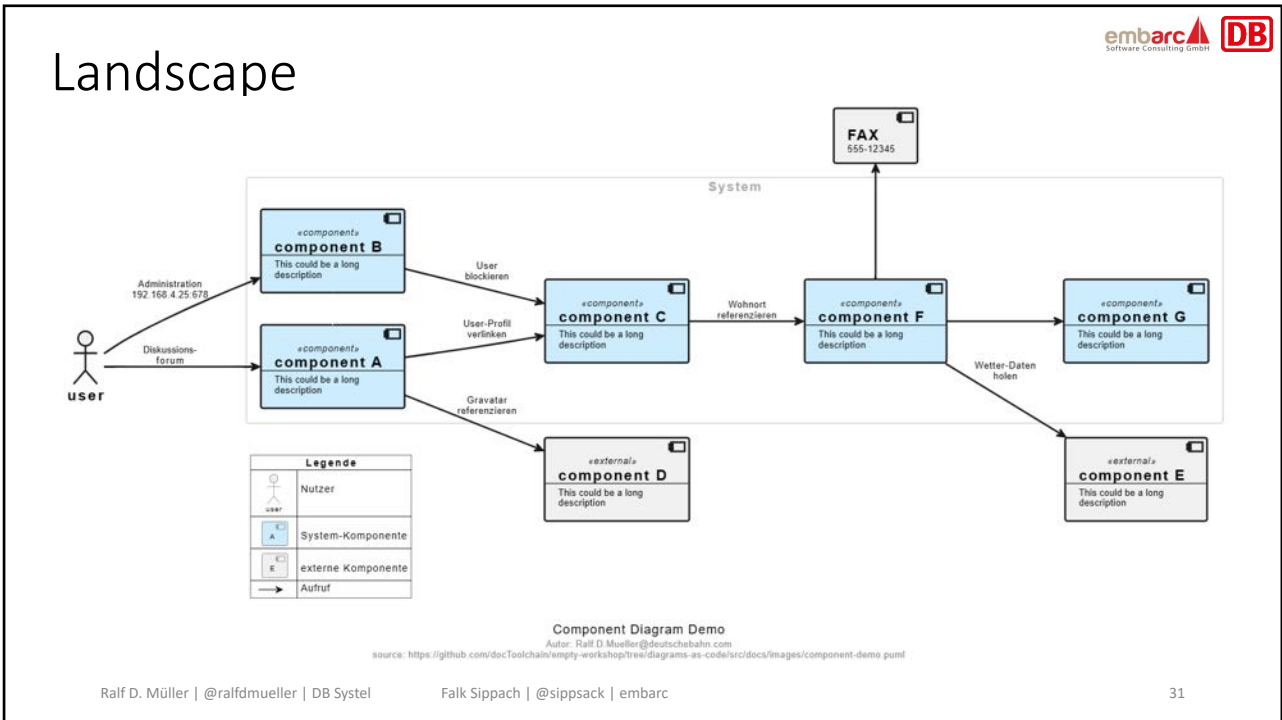
Author: Ralf D. Müller | @ralfdmueller | DB Systel
source: <https://github.com/DB-Systel/techtalks/tree/master/workshops/uml/component-diagram-01-coder/notebooks/legende/component-demo.puml>

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

30


30



31

Tipp 3: Visuelle Stile

- Farben sparsam und entsprechend ihrer natürlichen Bedeutung einsetzen
- Decorator oder Formen zur Unterscheidung der Elemente einsetzen
- A A** Schriftgrößen und Stile bewusst einsetzen
- Verbindungen und Elemente beschriften
- Unterschiedliche Größen bewusst einsetzen
- Verbindungen nicht überlappen und nur in eine Richtung



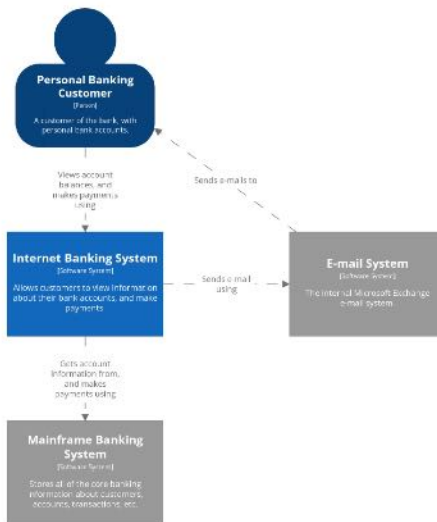
Ralf D. Müller | @ralfdmueller | DB System
Falk Sippach | @sipsack | embarc
33

33

c4model.com

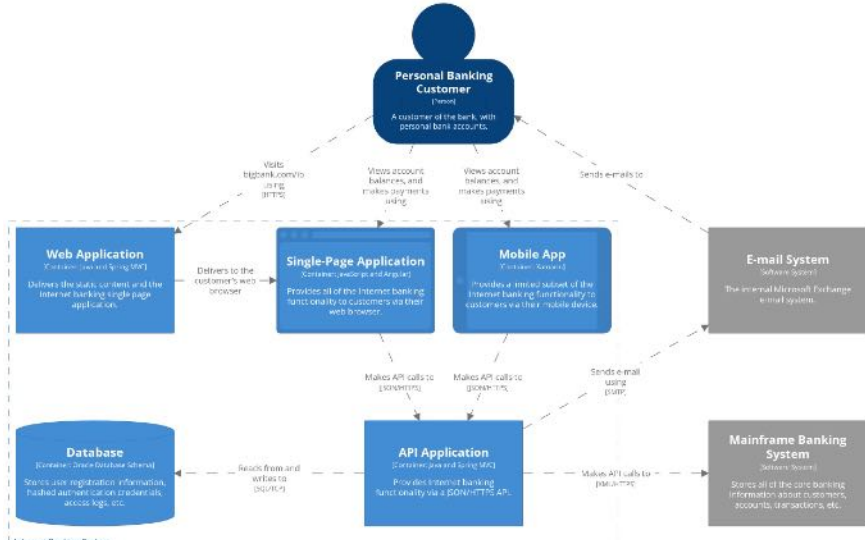
34

c4model.com: Context



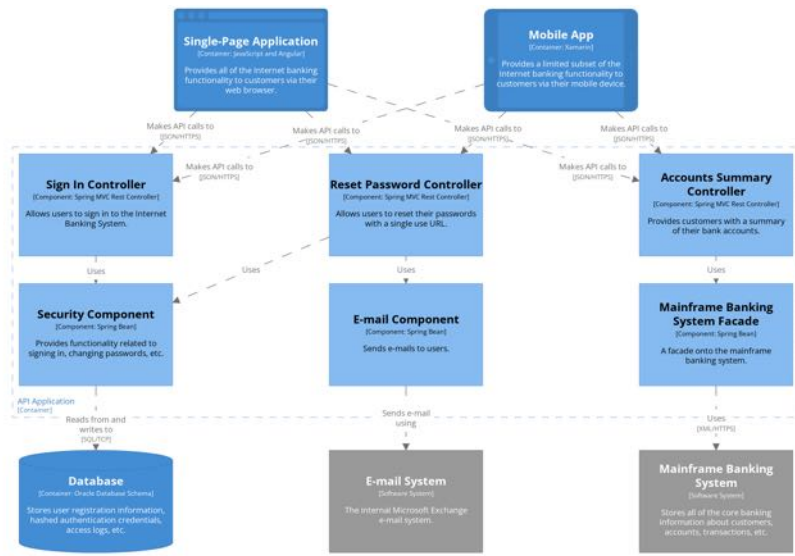
35

c4model.com: Containers



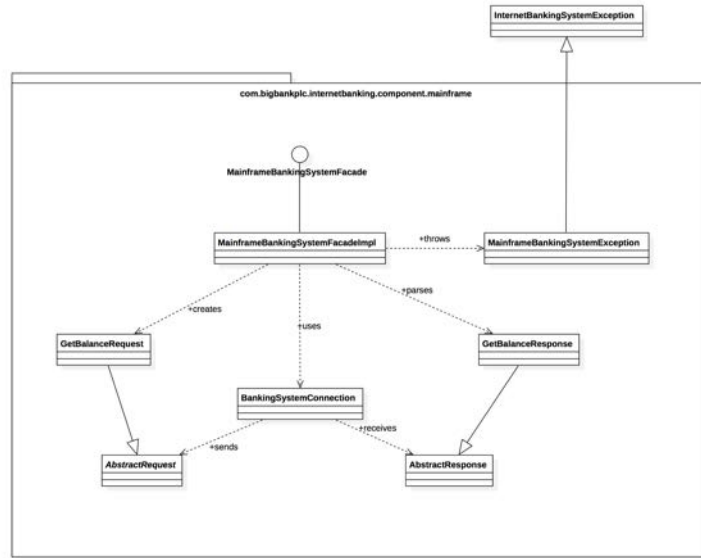
[Container] Internet Banking System
 The container diagram for the Internet Banking System.
 Monday, February 27, 2023 at 5:23 PM Coordinated Universal Time

c4model.com: Components



[Component] Internet Banking System - API Application
 The component diagram for the API Application.
 Monday, February 27, 2023 at 5:24 PM Coordinated Universal Time

c4model.com: Classes



Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

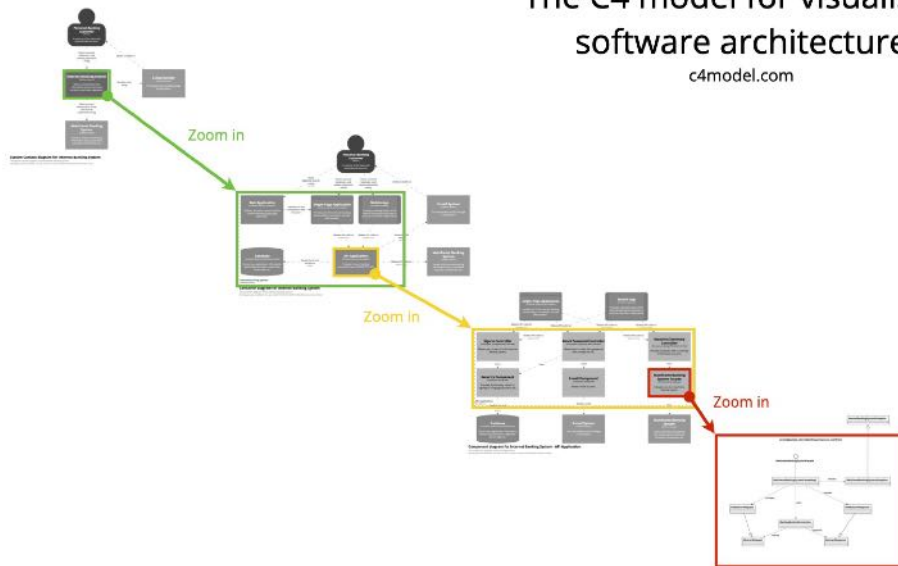
38

Quelle der C4-Diagramme c4model.com

38

The C4 model for visualising software architecture

c4model.com



Level 1
Context

Level 2
Containers

Level 3
Components

Level 4
Code

39

Agenda



Gute Diagramme

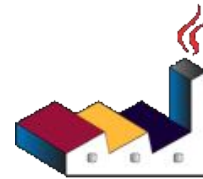


Werkzeuge



**Einbettung in
Dokumentation**

PlantUML



```

1 @startuml
2
3 User -> Server: via Browser
4   Server -> Database: fetch profile
5   Server <-- Database
6 User <-- Server
7
8 @enduml

```

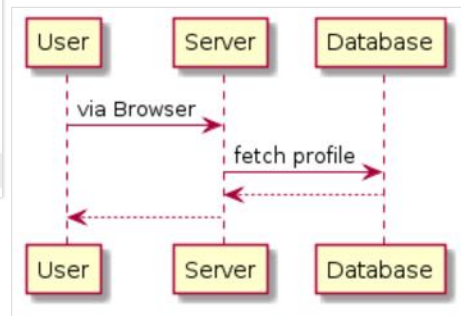
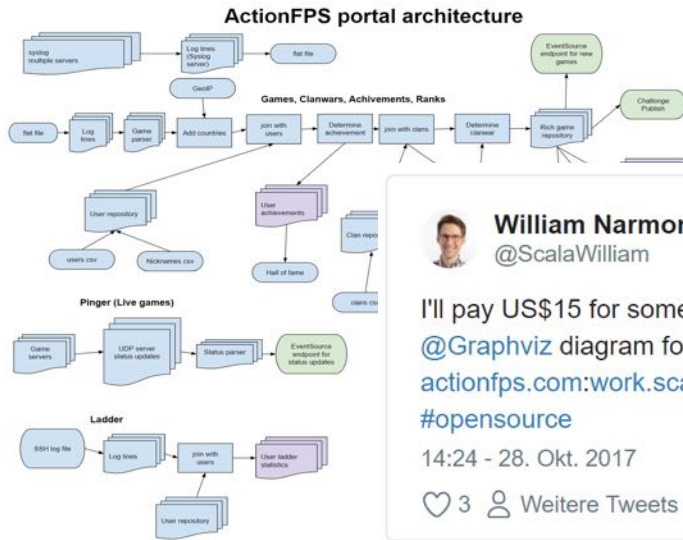


Diagramme: PlantUML



William Narmontas
@ScalaWilliam

I'll pay US\$15 for someone who can do this @PlantUML or @Graphviz diagram for [actionfps.com:work.scalawilliam.com/graphviz-plant...](https://actionfps.com/work.scalawilliam.com/graphviz-plant...) #opensource

14:24 - 28. Okt. 2017

3 likes · Weitere Tweets von William Narmontas ansehen

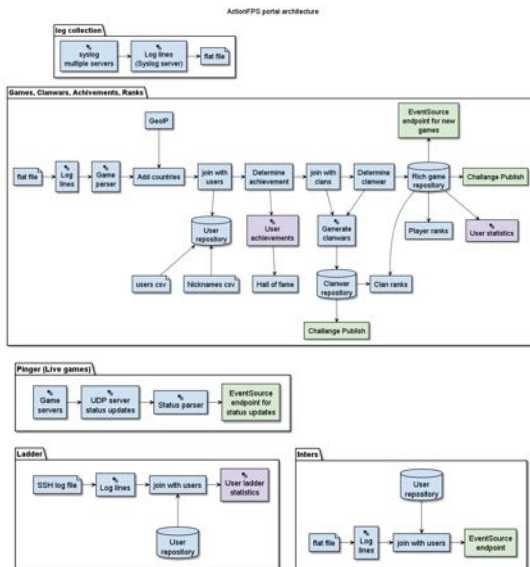
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

42

42

Diagramme: PlantUML



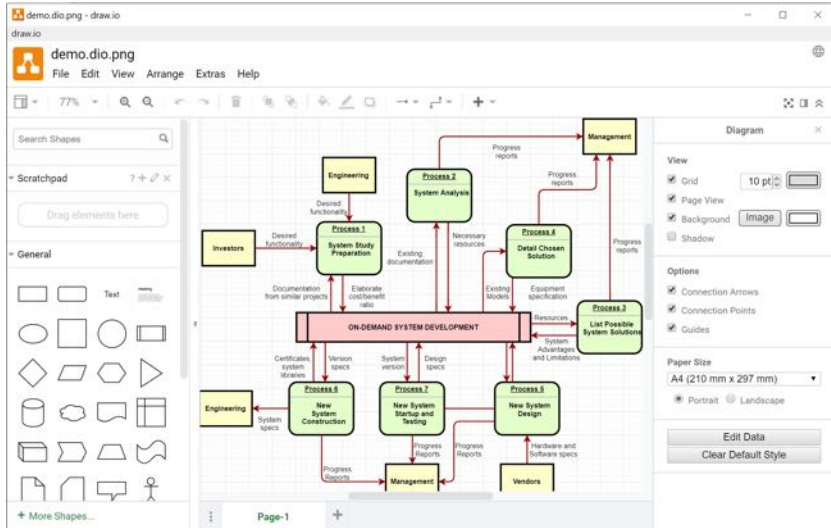
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

43

43

draw.io/Diagrams.net



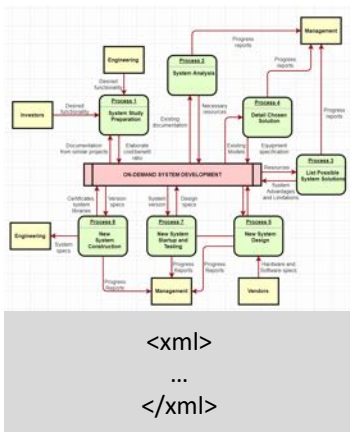
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

44

44

draw.io/Diagrams.net



PNG-Diagramm

Meta-Daten

```
<xml>
...
</xml>
```

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

45

45

Kroki

Creates diagrams from textual descriptions!

Kroki provides a unified API with support for BlockDiag (BlockDiag, SeqDiag, ActDiag, NwDiag, PreKuDiag, RackDiag), BPMN, Bytefield, C4 (with PlantUML), Ditaai, Erd, Excalidraw, GraphViz, Mermaid, Nomnoml, PlantUML, SvgBob, UMLet, Vega, Vega-Lite, WaveDrom... and more to come!

#Features

Ready to use

Diagrams libraries are written in a variety of languages: Haskell, Python, JavaScript, Go, PHP, Java... some also have C bindings. Trust us, you have better things to do than install all the requirements to use them. Get started in no time!

Simple

Kroki provides a unified API for all the diagram libraries. Learn once use diagrams anywhere!

Free & Open source

All the code is available on GitHub and our goal is to provide Kroki as a free service.

Fast

Built using a modern architecture, Kroki offers great performance.

<https://kroki.io/>

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

46

46

Vega – A Visualization Grammar



<https://vega.github.io/vega/>

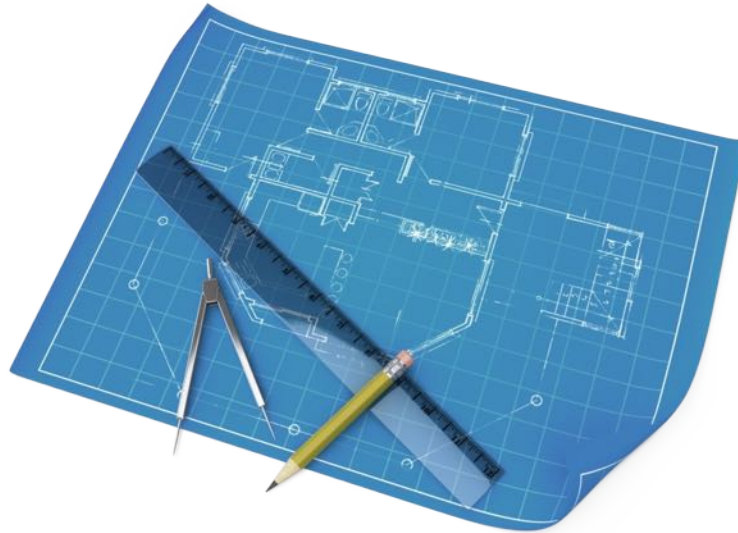
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

47

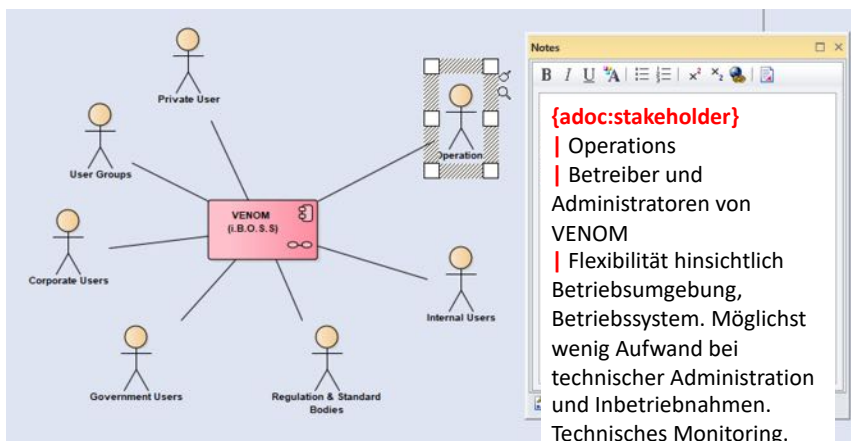
47

Diagramme: Nicht malen, modellieren!



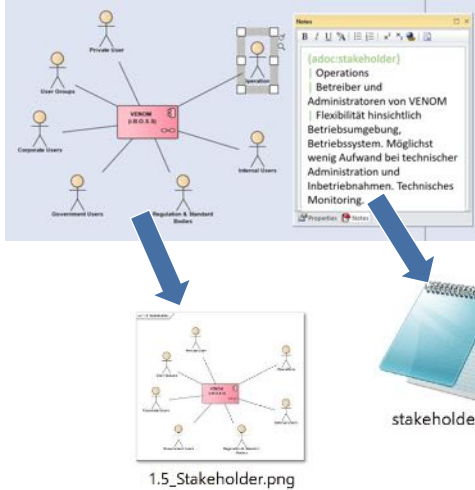
48

treat Docs-as-Code: automate!



49

treat Docs-as-Code: automate!



```

=== Stakeholder

==== Users and Groups of Users

image::ea/1.5_Stakeholder.png[]

[cols="2,3,3,2" options="header"]
.Users and Groups of Users
|===
| Role | Description | Goal |
Comment
include::../../ea/stakeholder.ad[]
|===
    
```

Ralf D. Müller | @ralfdmueller | DB Systemel

Falk Sippach | @sipsack | embarc

50

50

treat Docs-as-Code: automate!



VENOM: Architecture Documentation

1.5. Stakeholder

1.5.1. Benutzer und Benutzergruppen

Abbildung 2: Benutzer und Benutzergruppen von VENOM

Tabelle 2: Benutzer und Benutzergruppen

Rolle	Beschreibung	Ziel	Bemerkungen
Cooperate User	(Clia) Unternehmen oder kommerzielle Organisationen	Möchten komplexe Produkte entwickeln, konfigurieren und mit Aufbau und Inbetriebnahme wenig Aufwand haben.	

Ralf D. Müller | @ralfdmueller | DB Systemel

Falk Sippach | @sipsack | embarc

51

51

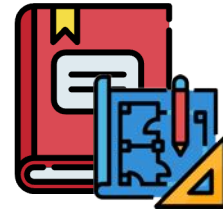
Agenda



Gute Diagramme



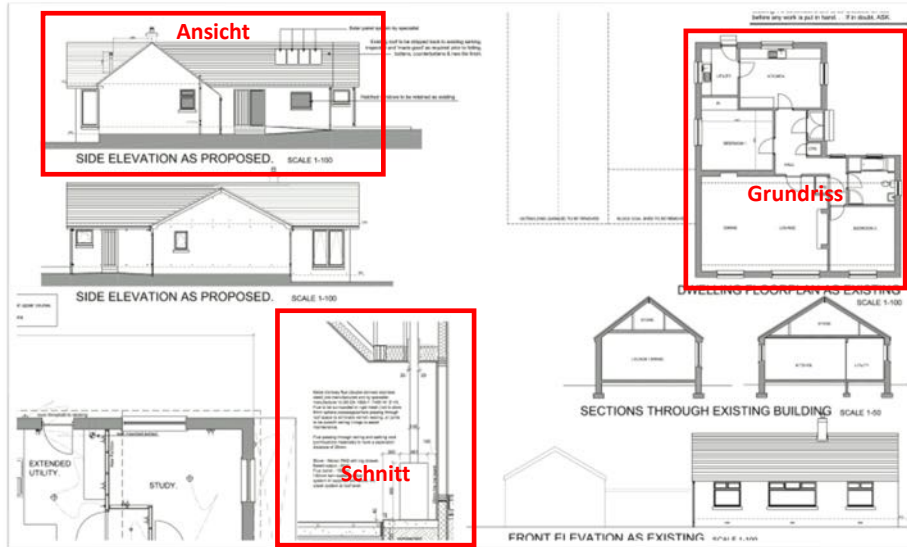
Werkzeuge



Einbettung in Dokumentation




52

Analogie Bauwesen




53


Sichten in arc42


1. Einführung und Ziele 1.1 Aufgabenstellung 1.2 Qualitätsziele 1.3 Stakeholder	7. Verteilungssicht 7.1 Infrastruktur Ebene 1 7.2 Infrastruktur Ebene 2 ...
2. Randbedingungen 2.1 Technische Randbedingungen 2.2 Organisatorische Randbedingungen 2.3 Konventionen	8. Konzepte 8.1 Fachliche Strukturen und Modelle 8.2 Typische Muster und Strukturen 8.3 Persistenz 8.4 Benutzungsoberfläche ...
3. Kontextabgrenzung 3.1 Fachlicher Kontext 3.2 Technischer- oder Verteilungskontext	9. Entwurfsentscheidungen 9.1 Entwurfsentscheidung 1 9.2 Entwurfsentscheidung 2 ...
4. Lösungsstrategie	10. Qualitätsszenarien 10.1 Qualitätsbaum 10.2 Bewertungsszenarien
5. Bausteinsicht 5.1 Ebene 1 5.2 Ebene 2 ...	11. Risiken
6. Laufzeitsicht 6.1 Laufzeitszenario 1 6.2 Laufzeitszenario 2 ...	12. Glossar



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



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





→ <https://arc42.de/>


Ralf D. Müller | @ralfdmueller | DB Systel Falk Sippach | @sipsack | embarc


54


54


Sichten in arc42


- 

Kontextabgrenzung (System Context)
 Zeigt die beteiligten Fremdsysteme und Benutzer, mit denen das zu beschreibende System interagiert.
- 

Bausteinsicht (Building Block View)
 Zeigt die Struktur des Softwaresystems in Form seiner Bausteine, und wie diese zusammenhängen. Bündelt viele Entscheidungen rund um Verantwortlichkeiten.
- 

Laufzeitsicht (Runtime View)
 Zeigt Elemente der Baustein- und/oder Kontextsicht in Aktion, veranschaulicht dynamische Strukturen und Verhalten des beschriebenen Systems.
- 

Verteilungssicht (Deployment View)
 Zeigt wie die Software in Betrieb genommen (verteilt) wird, und wie die Zielumgebung aussieht.



Ralf D. Müller | @ralfdmueller | DB Systel Falk Sippach | @sipsack | embarc

55

55

Ursprung: 4+1 Sichten

Philippe Kruchten

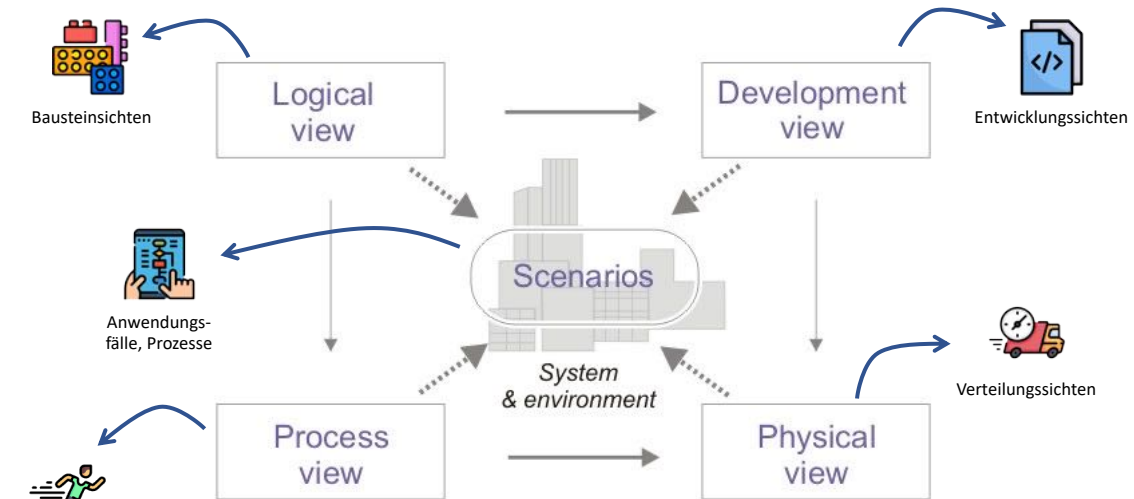


Das **4+1-Sichten-Softwarearchitekturmodell** ist ein weit verbreitetes Modell der Sichten auf ein Softwaresystem, das von Philippe Kruchten zur „**Beschreibung der Architektur eines Software-intensiven Systems auf der Grundlage von mehreren konkurrierenden Sichten**“ entwickelt wurde.

https://de.wikipedia.org/wiki/4%2B1_Sichtenmodell

56

Elemente des 4+1



https://de.wikipedia.org/wiki/4%2B1_Sichtenmodell#/media/Datei:4+1_Architectural_View_Model.jpg

57

To UML or to not UML?



“Wir benutzen für unsere Architekturdiagramme UML. Ist standardisiert. Kann jeder lesen.”

- Modeling Languages
- UML 2.0 Diagrams
- Custom Diagrams
- UML Behavioral Diagrams
 - Activity Diagrams
 - Communication Diagrams
 - Interaction Overview Diagrams
 - Sequence Diagrams
 - State Machine Diagrams
 - Timing Diagrams
 - Use Case Diagrams
- Interactions
- Timing
- Use Case Model
- UML Structural Diagrams
 - Class diagrams
 - Component Diagrams
 - Package Diagrams
 - Composite Structure
 - Composite Structure

-- unbekannte Softwarearchitekt:in



Mögliche UML-Diagrammarten

Klassendiagramm
Kommunikationsdiagramm
Sequenzdiagramm

Komponentendiagramm
Paketdiagramm

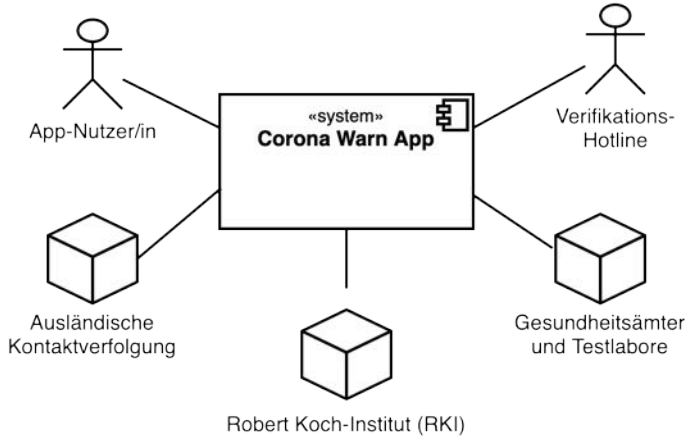
Anwendungsfalldiagramm

Sequenzdiagramm
Aktivitätsdiagramm

Verteilungsdiagramm

Kontextabgrenzung

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



Legende

- Benutzer
CWA-System stellt Oberfläche bereit
- Fremdsystem
CWA-System stellt Schnittstelle(n) bereit
- Interaktion



Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

60

60

Tatsächliche Zerlegung im Quelltext

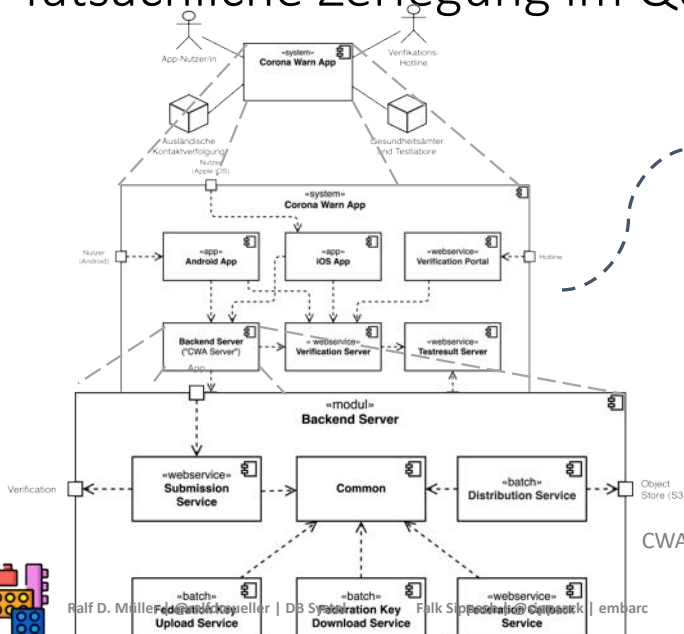


„Bausteinsicht, Ebene 1“

GitHub-Repositories

[https://github.com/corona-warn-app/...](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

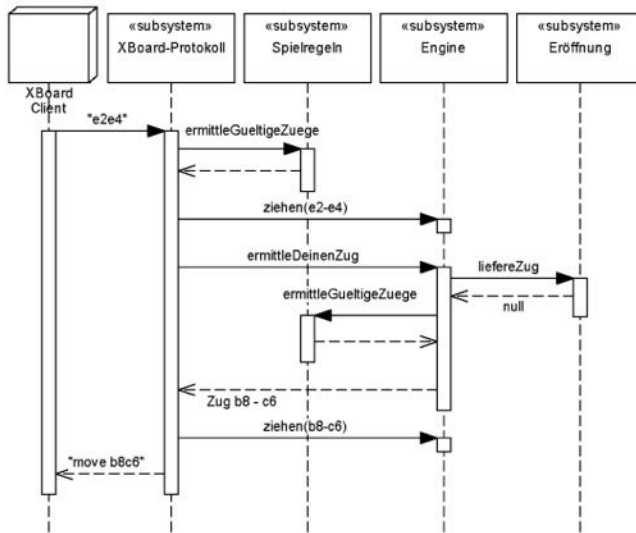
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

61

61

Beispiel eines Ablaufes



<https://www.dokchess.de/>



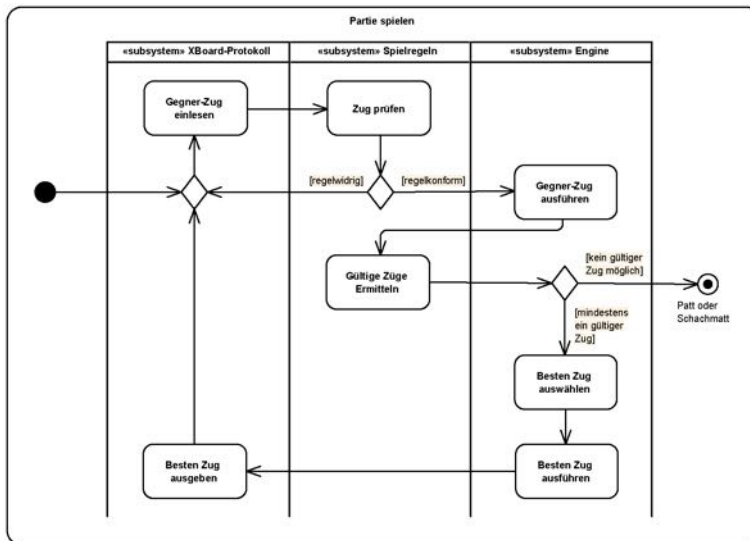
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

62

62

Beispiel eines Ablaufes



<https://www.dokchess.de/>



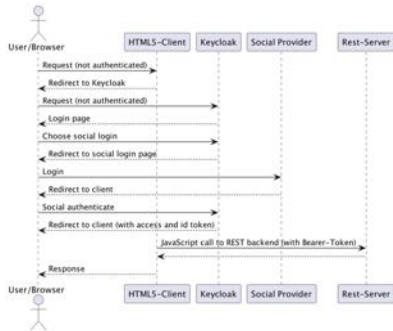
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

63

63

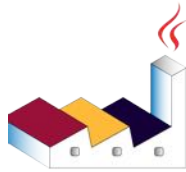
Sequenzdiagramm mit PlantUML



```

@startuml
actor browser as "User/Browser"
participant html5 as "HTML5-Client"
participant Keycloak
participant socialprovider as "Social Provider"
participant server as "Rest-Server"

browser->>html5: Request (not authenticated)
html5->>Keycloak: Redirect to Keycloak
Keycloak->>browser: Request (not authenticated)
Keycloak->>browser: Login page
browser->>Keycloak: Choose social login
Keycloak->>socialprovider: Redirect to social login page
socialprovider->>Keycloak: Login
Keycloak->>html5: Redirect to client
Keycloak->>browser: Social authenticate
Keycloak->>html5: Redirect to client (with access and id token)
html5->>server: JavaScript call to REST backend (with Bearer-Token)
server-->>html5: Response
html5-->>browser: Response
    
```



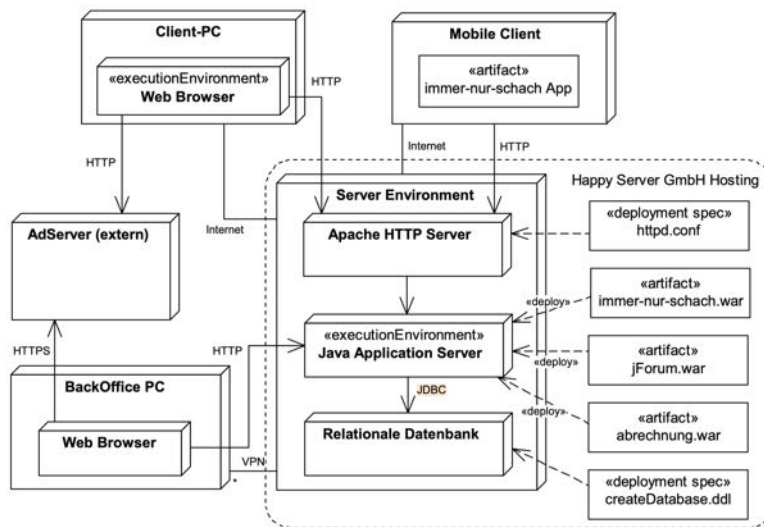
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

64

64

Verteilungssicht "immer-nur-schach"



aus: "Software-Architekturen dokumentieren und kommunizieren" von Stefan Zörner



Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sippack | embarc

65

65

Waren das schon alle Sichten?

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



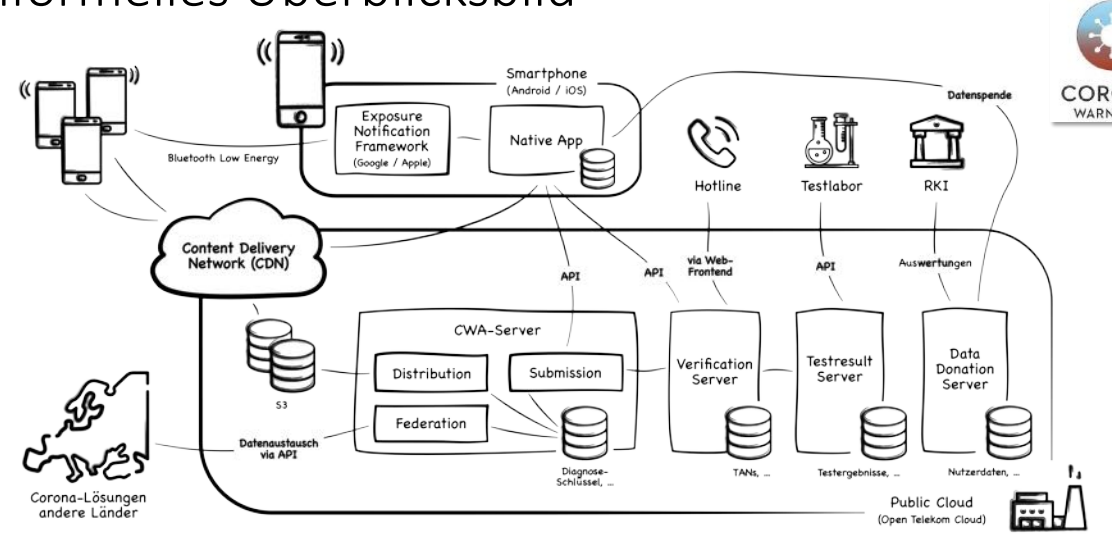



Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

66

Informelles Überblicksbild





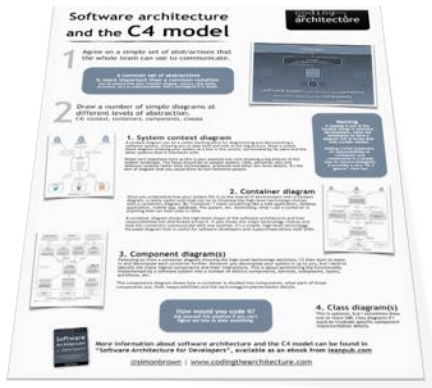
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

67

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

Alternative: C4 Model (Simon Brown)



- Agree on a simple set of abstractions that the whole team can use to communicate.
- Draw a number of simple diagrams at different levels of abstraction.
- C4: context, containers, components, classes (code)

→ <http://static.codingthearchitecture.com/c4.pdf>



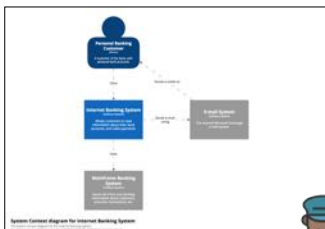
Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

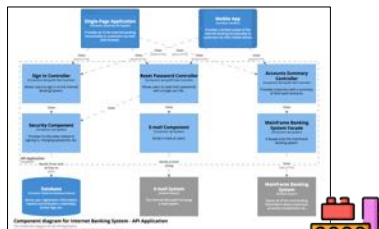
68

68

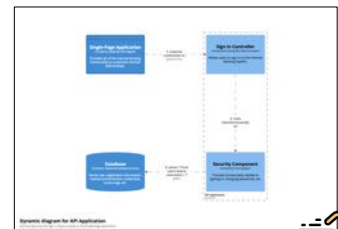
Alternative C4: Inhalte



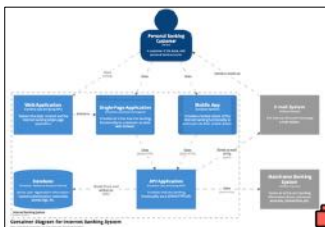
Level 1: System Context



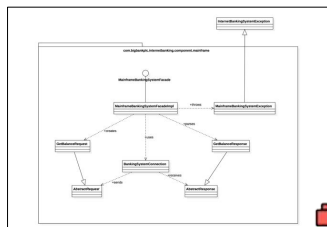
Level 3: Component Diagram



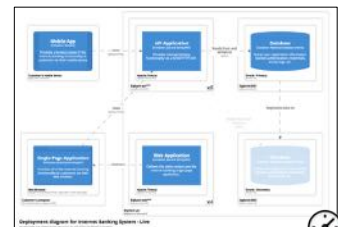
Dynamic Diagram



Level 2: Container Diagram



Level 4: Code



Deployment Diagram



Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

69

69

embarc  DB
Software Consulting GmbH

Alle Sichten auf einen Blick



Kontextsichten



Bausteinsichten

4 + 1 (Kruchten)



Entwicklungssichten



Informelles
Überblicksbild



Laufzeitsichten



Anwendungsfälle/
Prozesse



Verteilungssichten

Ralf D. Müller | @ralfdmueller | DB Systel Falk Sippach | @sippack | embarc

70

70


embarc  DB
Software Consulting GmbH

Diagramm-Tools- Empfehlungen

draw.io/diagrams.net
PlantUML (Komponenten-
diagramm)

draw.io/diagrams.net

draw.io/diagrams.net

PlantUML (Anwendungsfall-
diagramm)

draw.io/diagrams.net

PlantUML
(Sequenzdiagramm,
Aktivitätsdiagramme)

PlantUML
(Sequenzdiagramm,
Aktivitätsdiagramme)

Ralf D. Müller | @ralfdmueller | DB Systel Falk Sippach | @sippack | embarc

71

71

Beispiele Umsetzung in diagrams.net



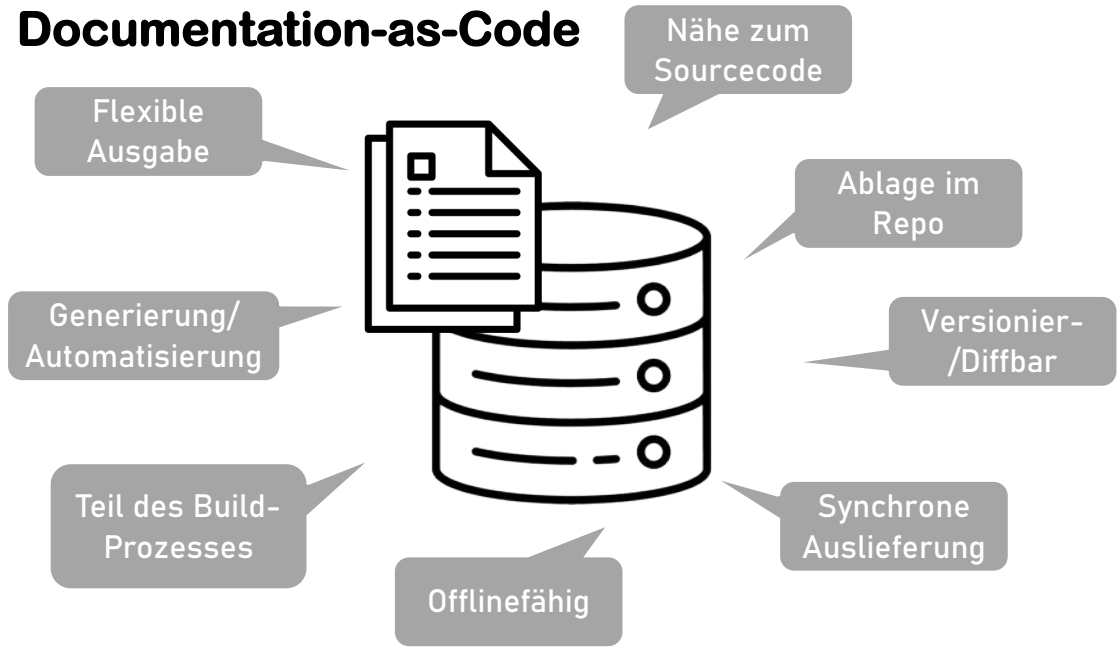
The image displays three screenshots of the diagrams.net application. The first screenshot, titled 'CWA Systemkontext (fachlich).drawio', shows a high-level system context diagram with actors like 'App-NutzerIn' and 'Verifikations-Hotline', and components like 'Corona Warn App' and 'Gesundheitsämter und Testlabore'. The second screenshot, 'CWA_Informeller_Ueberblick.drawio', provides a more detailed overview of the system's components and their interactions. The third screenshot, 'Bausteinsicht Ebene 2, Backend.drawio', details the backend architecture, including the 'CWA Backend Server' and various services like 'Authentifizierung' and 'Datenbank'. A small icon of a person with a magnifying glass is positioned between the first and second screenshots. In the bottom right corner, there is a logo for 'CORONA WARN-APP'.

Ralf D. Müller | @ralfdmueller | DB System Fallbach | @sipsack | embarc

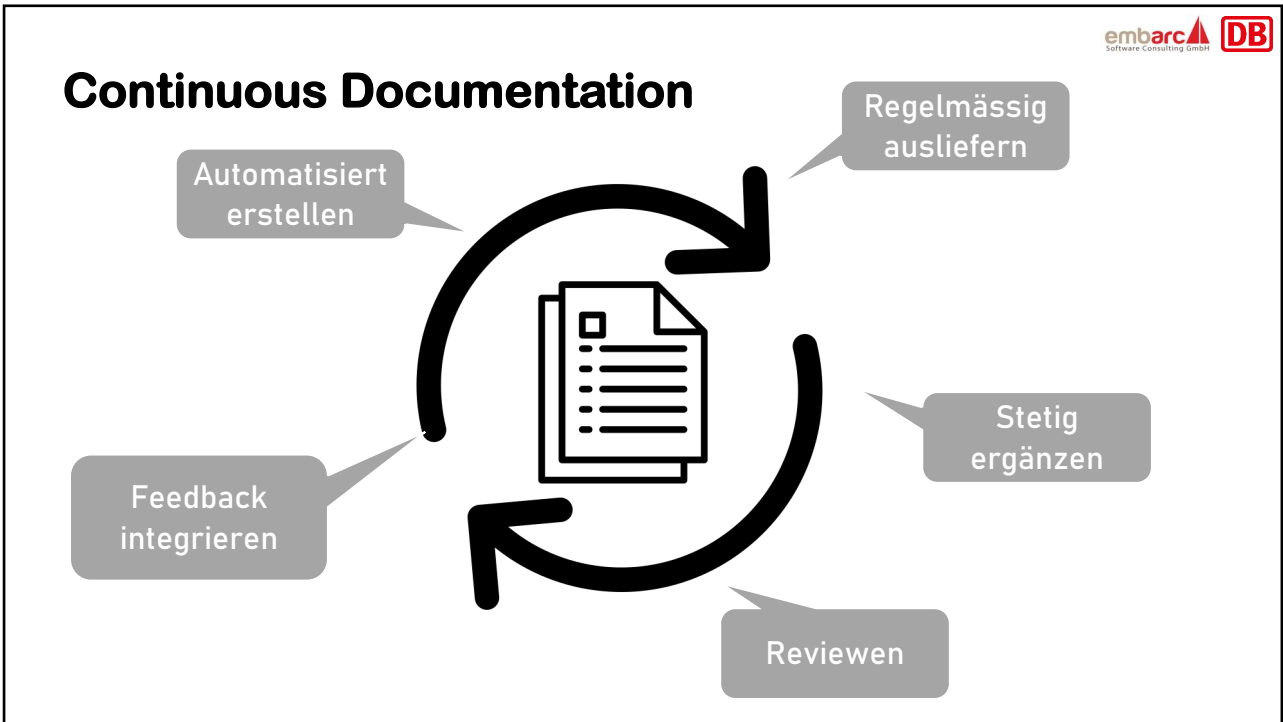
72

72

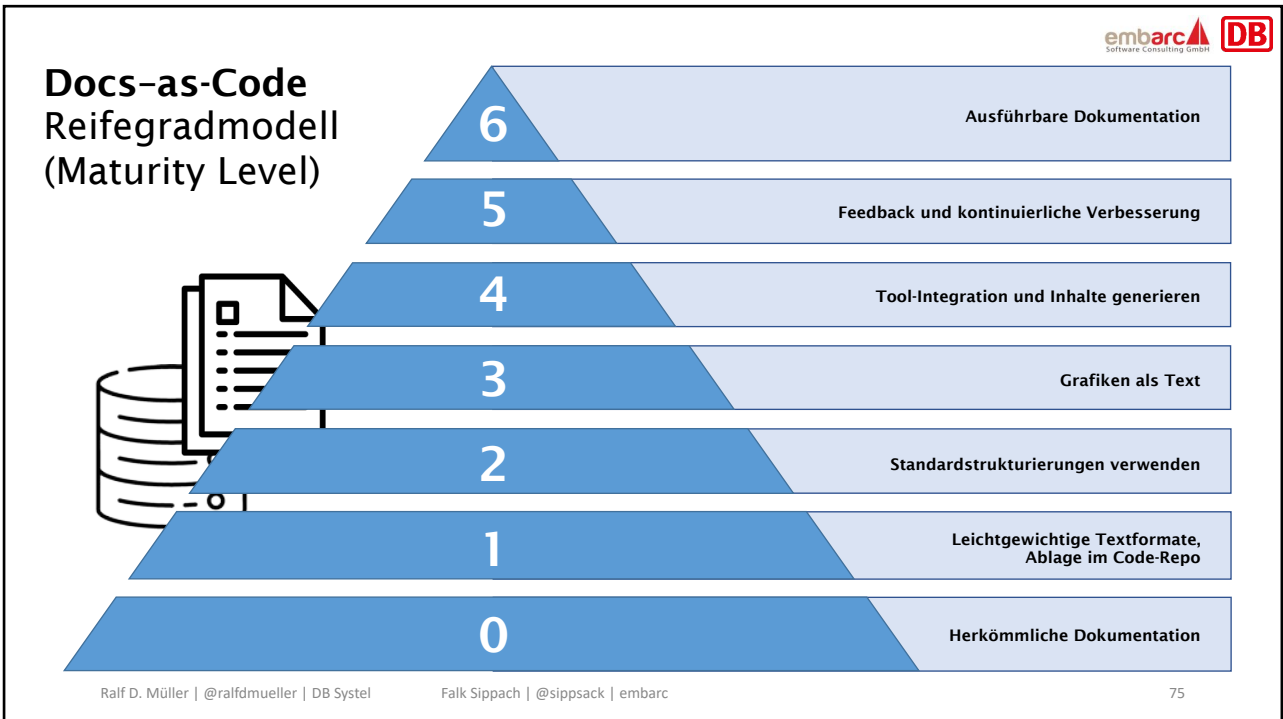
Documentation-as-Code



73



74



75

Werkzeuge

IDEs, Texteditoren

Build-Management

Textformate, Standards

Grafiken

embarc Software Consulting GmbH DB

76

Basic Docs-as-Code with AsciiDoc

```
graph LR; subgraph "your solution architecture documentation"; direction TB; doc[your solution architecture documentation]; end; subgraph "asciidoc"; direction TB; plantUML[plantUML]; pdfPlugin[pdf plugin]; end; doc -- read --> plantUML; doc -- read --> pdfPlugin; plantUML --> arc42[arc42 asciidoc template]; pdfPlugin -- generate PDF --> pdf[PDF]; pdfPlugin -- generate HTML --> html5[HTML5]; arc42 -- is derived from --> doc;
```

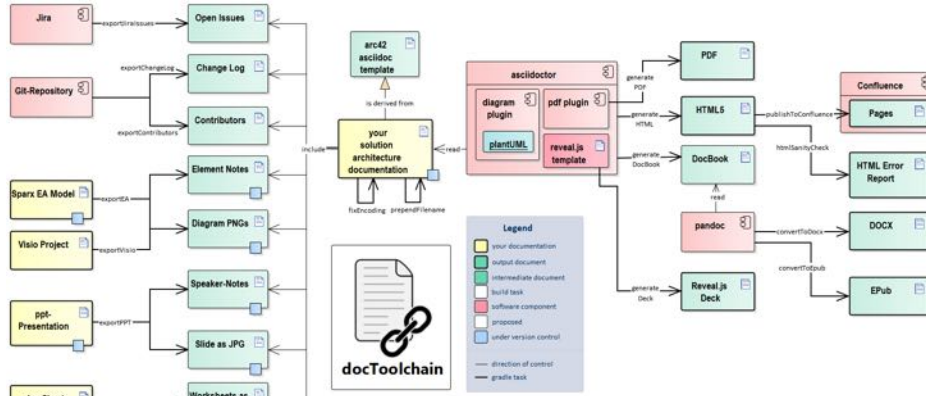
Ralf D. Müller | @ralfdmueller | DB System | Falk Sippach | @sippack | embarc

77

embarc Software Consulting GmbH DB

77

Basic Docs-as-Code with AsciiDoc & docToolchain



78

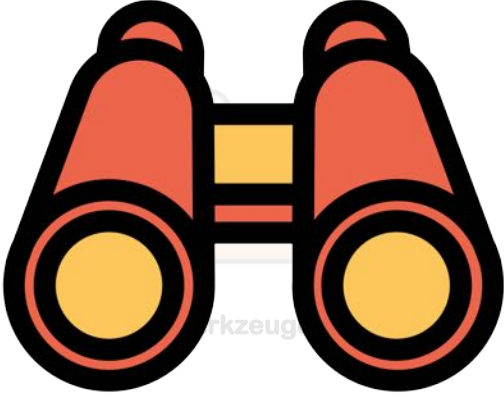
78

Docs-as-Code in der IDE – eine besondere Liebe 🍷💖



79

Ausblick



embarc Software Consulting GmbH **DB**

Gute Diagramme


Einbettung in Dokumentation

Ralf D. Müller | @ralfdmueller | DB System Falk Sippach | @sipsack | embarc

80

80

Checkliste für Architekturdiagramme



embarc Software Consulting GmbH **DB**

Gibt es eine **Überschrift** und eine **Legende**?

Autor, Erstellungs- und Änderungs**datum** vorhanden?

Per **URL verlinkbar** statt Copy-by-Value?

Ist die **Quelle bekannt** und ist es leicht **editierbar**?

Ist der **Diagrammtyp** klar erkennbar und wurde der **passende Typ** gewählt?

Wurde ein **passendes Werkzeug** verwendet?

Diagrammzweck ersichtlich?

Ist jedes **Element benannt**, die **Bedeutung ersichtlich** und die **Funktion verständlich**?

Sind die verwendeten **Abkürzungen ersichtlich**?

Bedeutung aller Farben und Formen ersichtlich?

Bedeutung **unterschiedlicher Größe** und aller **Rahmenstile** verständlich?

Sind **alle Verbindungen** beschriftet und die **Bedeutung der Linien** klar?

Ist die Bedeutung der Pfeilspitzen und Linienstile (fett, gestrichelt, ...) ersichtlich?

Ralf D. Müller | @ralfdmueller | DB System Falk Sippach | @sipsack | embarc

81

81

Vorteile Diagrams-as-Code (und leichtgewichtige Diagramme)

Jederzeit und von jedem editierbar.

Leicht zu erstellen.

Entwicklertools verwendbar.

Automatisches Layouting und Rendering.



Versionier- und historisierbar.

Direkte Unterstützung der Notation.

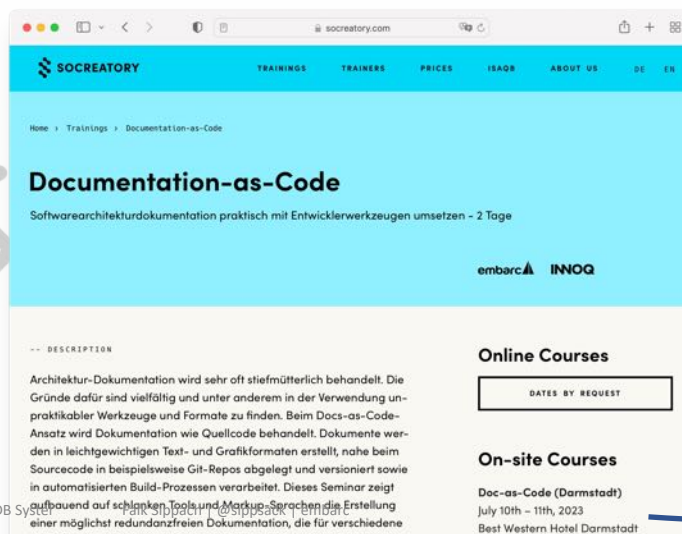
Leicht zu aktualisieren

einheitlicher Stil

Training zu Docs-as-Code

Softwarearchitekturdokumentation praktisch mit Entwicklerwerkzeugen umsetzen

→ <https://www.socreatory.com/en/trainings/docascode>



Wir kommen auch zu Euch in die Firma!

embarc  **DB**
Software Consulting GmbH

Weiterführende Blog-Posts


Was ist eigentlich Docs-as-Code? & Eine Einführung in docToolchain

→ <https://www.embarc.de/was-ist-docs-as-code/>
 → <https://www.embarc.de/doctoolchain/>



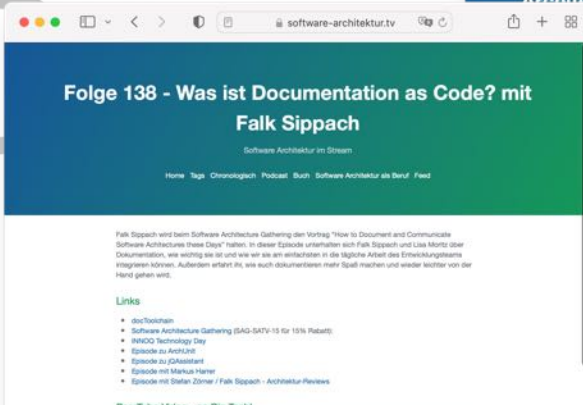
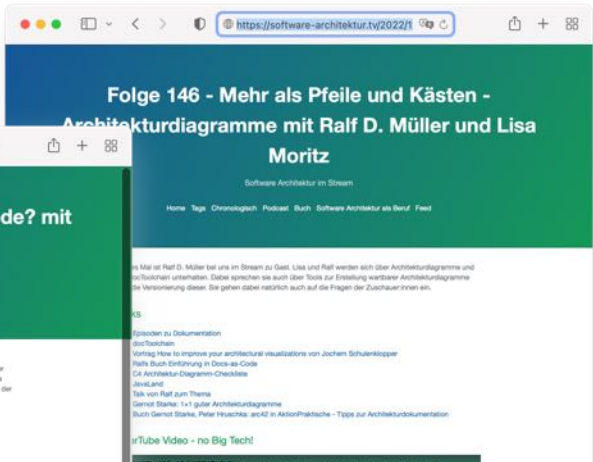

Ralf D. Müller | @ralfdmueller | DB Systel
 Falk Sippach | @sipsack | embarc

84

embarc  **DB**
Software Consulting GmbH

Weiterführende Videos/Podcasts

→ <https://software-architektur.tv/2022/10/14/folge138.html/>
 → <https://software-architektur.tv/2022/12/16/folge146.html/>

Ralf D. Müller | @ralfdmueller | DB Systel | Falk Sippach | @sipsack | embarc

85

Einführung in Docs-as-Code



<https://leanpub.com/praxisbuchdocs-as-code/c/SAS-Munich>

Ralf D. Müller | @ralfdmueller | DB System

Falk Sippach | @sipsack | embarc

86

86

Spicken erlaubt!



Unsere Architektur-Spicker beleuchten die konzeptionelle Seite der Softwareentwicklung.



Spicker #1:
„Der Architekturüberblick“

- Welche Zutaten gehören in einen Architekturüberblick?
- Welche Formen bewähren sich in welchen Situationen?
- Wie fertigen Sie einen Architekturüberblick an?

→ embarc.de/architektur-spicker/

PDF, 4 Seiten
Kostenloser Download.

Ralf D. Müller | @ralfdmueller | DB System

Falk Sippach | @sipsack | embarc

88

88

Architekturüberblicke als Vorlage



[E: Bewertung & Ausblick] [D: Eintauchen in die Corona-Warn-App]

WICHTIGE KOMPROMISSE
Diese Entscheidungen sind kritisch und negativ einflussreich auf die Qualität der Softwareentwicklung.

ABLAUF: EINE PARTIE SPIELN
Dieses Ablaufdiagramm zeigt den Ablauf einer Partie aus Sicht von DOKCHES. Die Subsysteme verantworten jeweils die in diesen platzierten Schritte (z.B. "Zug prüfen").

SELBER STARTEN!
Mit DOKCHES lässt sich eine eigene Schachprogrammierung ("Engine") zu bauen. Hier ein Beispiel in 4 Schritten.

WEITERE INFORMATIONEN
zum DOKCHES:
- Java-Codegenese von Architekturdarstellungen
- Java-Codegenese von Architekturdarstellungen
- Java-Codegenese von Architekturdarstellungen

DOKCHES
Kompaktentwurf
Stand: Januar 2022

Softwarearchitektur am Beispiel
DOKCHES ist ein Schachprogramm, das die Aufgabe hat, den besten Zug zu finden. Die Aufgabe ist, den besten Zug zu finden. Die Aufgabe ist, den besten Zug zu finden.

INHALT
- AUFRECHTLUNG
- ARCHITEKTUR
- UML-DIAGRAMME
- DOKCHES IM DETAIL
- WICHTIGE KOMPROMISSE
- WEITERE INFORMATIONEN

Zerlegung auf oberster Ebene
Die Diagramme zeigen die wichtigsten Bausteine der CWK gemäß der Struktur des Quellcodes. Die Pläne betonen Abhängigkeiten (A - B basiert "A verwendet B").

Die deutsche Corona Warn-App
Ein prägnanter Überblick über die Softwarearchitektur der Corona-Warn-App & Backend

Architekturüberblick
1. Aufgabenstellung
2. Systeme
3. Lebenszyklus
4. Technologie-Stack
5. Eintauchen in die Corona-Warn-App
6. Bewertung & Ausblick

embarc.de/architektur-ueberblicke/

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

89

89

Vielen Dank. Wir freuen uns auf Eure Fragen!



✉ falk.sippach@embarc.de
 🐦 @sipsack



✉ ralf.d.mueller@deutschebahn.com
 🐦 @RalfDMueller

Ralf D. Müller | @ralfdmueller | DB Systel

Falk Sippach | @sipsack | embarc

90

90

Links/Referenzen (1)

- 1x1 guter Architekturdiagramme:
<https://www.innoq.com/de/articles/2022/09/better-architecture-diagrams/>
- Checkliste Softwarearchitekturdiagramme:
<https://www.feststelltaste.de/checkliste-softwarearchitekturdiagrammen/>
- Praxisbuch Docs-as-Code (vorab) mit erweiterter Checkliste:
<https://leanpub.com/praxisbuchdocs-as-code/c/SAS-Munich>
- Jochem Schulenklopper „How to improve your architectural visualizations“
<https://conferences.oreilly.com/software-architecture/sa-eu-2018/public/schedule/detail/68915.html>

Links/Referenzen (2)

- Architecture diagrams should be code:
<https://news.ycombinator.com/item?id=34322130>
- render diagrams from the Structurizr DSL:
https://twitter.com/simonbrown/status/1586339632828284928?t=C7vjEZ5ZFckviUWBbTk_w&s=03
- The Ultimate Guide To Software Architecture Documentation:
<https://www.workingsoftware.dev/software-architecture-documentation-the-ultimate-guide/?s=03>
- Diagrams as code != automatic layout:
<https://twitter.com/simonbrown/status/1613824847812886528?t=2DA-1BXt0iRbqCl8OyOh1g&s=03>

Links/Referenzen (3)

- UML-Diagramme mit PlantUML:
https://www.youtube.com/@Randomcode_0/search?query=plantuml
- Alexander Schwartz: Technical Writing with AsciiDoc & IntelliJ
<https://www.udemy.com/course/technical-writing-with-asciidoc-and-intellij-idea/?couponCode=10E18DB7D7561FBBAC25>
- Alexander Schwartz: AsciiDoctor Deep Dive
<https://www.ahus1.de/post/asciidoc-intro-and-deep-dive>