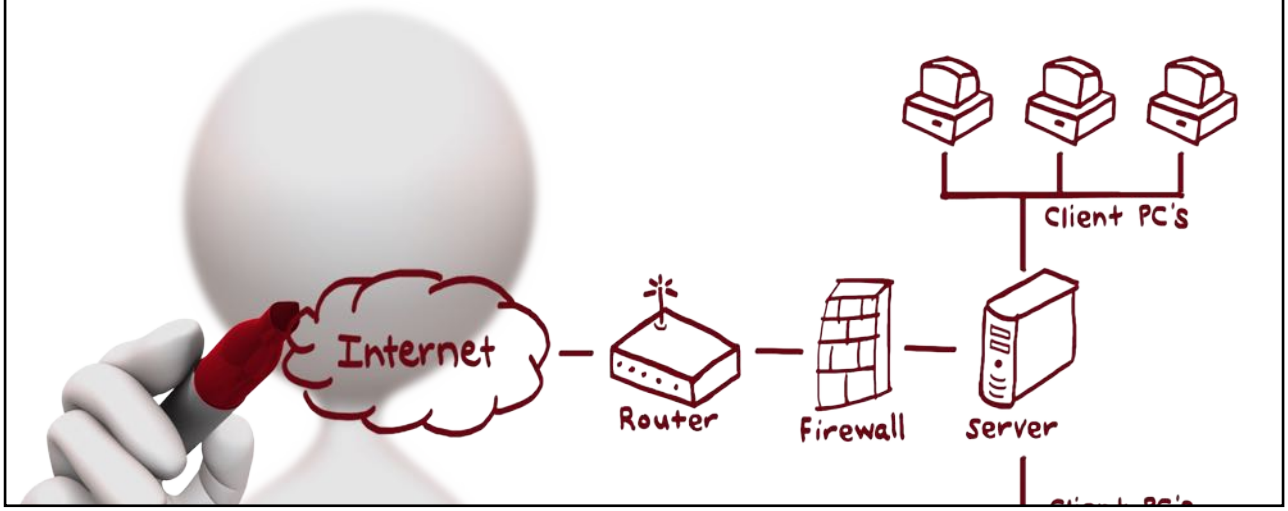
Workshop:

Phantastische Diagramme

…und wie Du sie selbst erstellst

Falk Sippach
embarc

Ralf D. Müller
DB Systel GmbH



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.

2

Falk Sippach

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

fs@embarc.de

@sippsack

➔ xing.to/fsi

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

➔ xing.to/rdmuller

5

Wer seid Ihr?

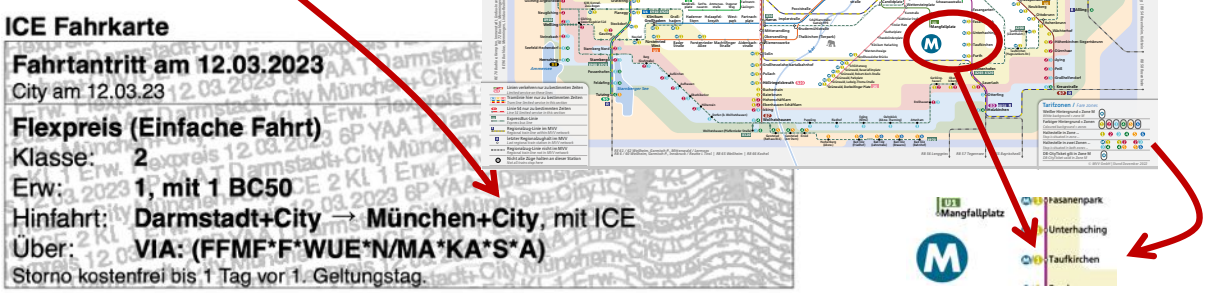
Architecture Summit München 2023: Phantastische Diagramme und wie Du sie selbst erstellst

https://forms.gle/J8oGsFse1Sx5fG1K7

6

Gestern Abend nach der Ankunft am Hauptbahnhof:

"Wie weit darf man in München mit **'+City'** fahren?"



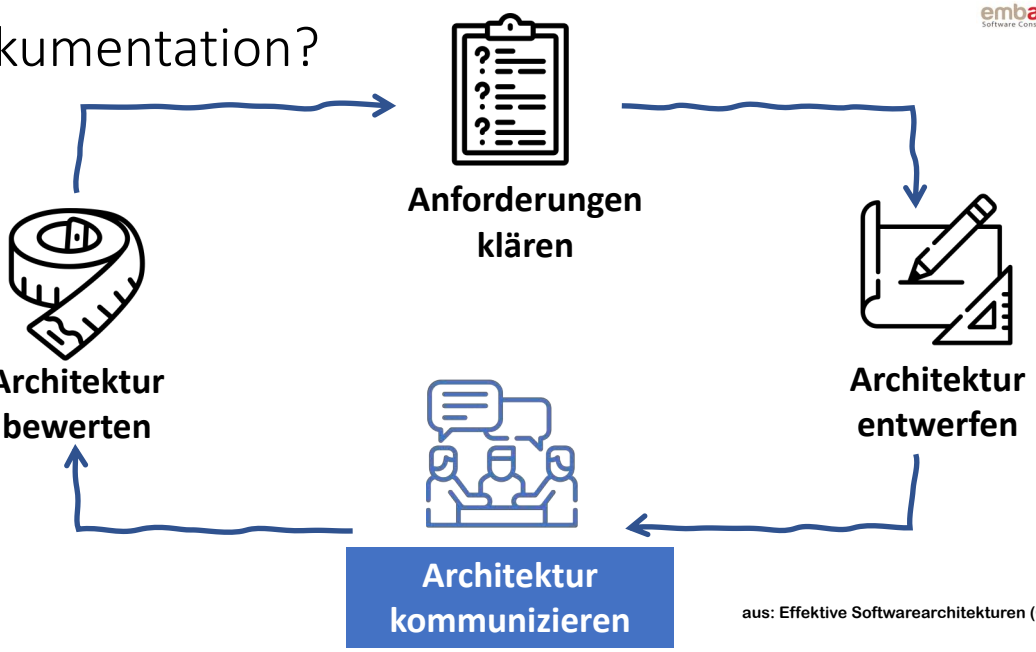
7

Mission Statement (für diesen Workshop)

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

8

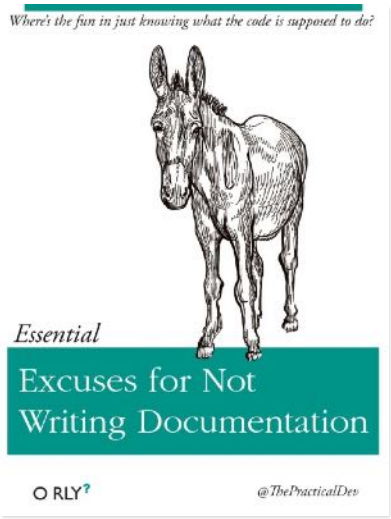
Dokumentation?



aus: Effektive Softwarearchitekturen (G. Starke)

9

Gute Gründe …



Grafik von The Practical Dev (CC0 Public Domain Lizenz)

Entwurfsunterstützung

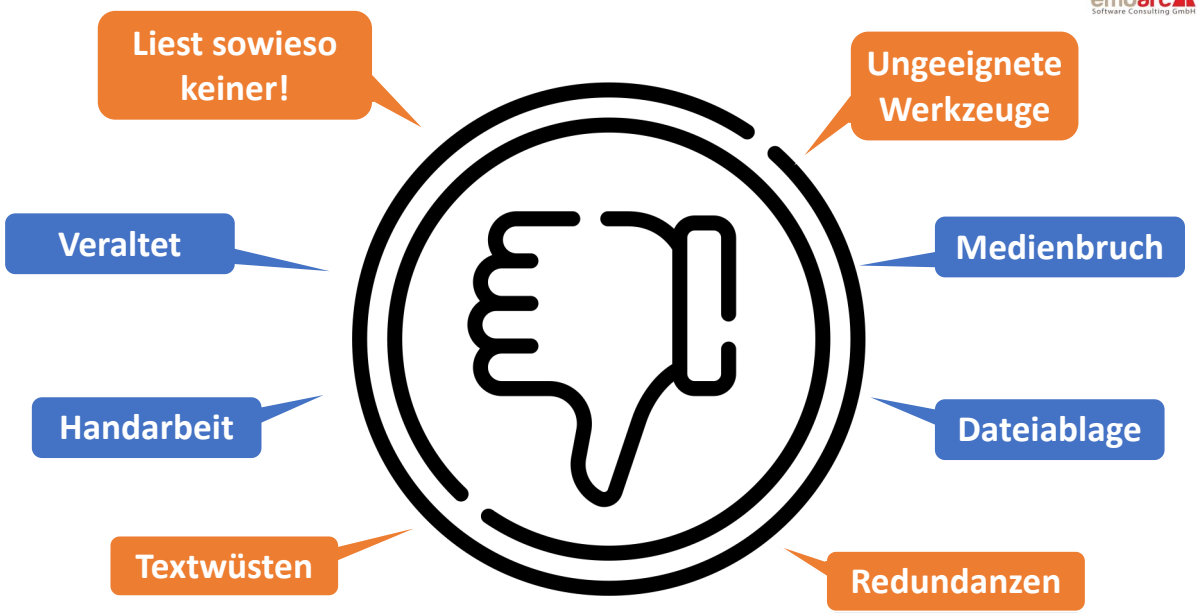
Frage nach dem Warum

Bewertbarkeit

Kommunikation

Neue Mitarbeiter

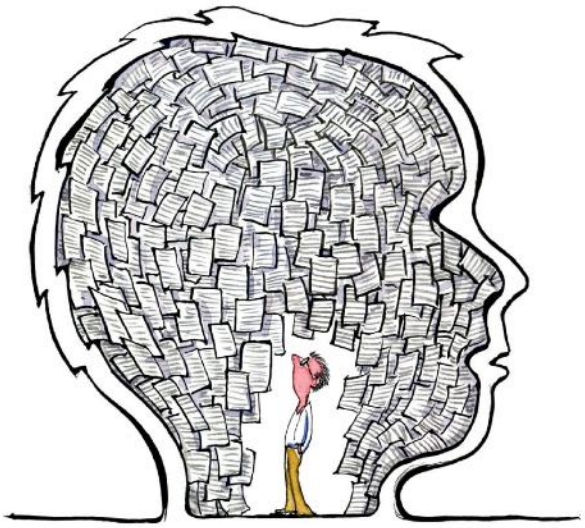
10



Helfen uns hier Diagramme?

11

Ein Bild sagt mehr als 1000 Worte …



12

14:00 - 14:45 **Teil 1: Schwächen von Diagrammen**

Pause!

14:45 - 15:30 **Teil 2: Diagrams-as-Code**

30 min

16:00 - 16:40 **Teil 3: Sichten**

Fragen?

16:40 - 17:20 **Teil 4: Einbinden in Dokumentation**

10 min Fragen

Theorie

Übungszeit

13

Agenda

Probleme/Schwächen von Diagrammen

Diagrams-as-Code

Sichten der Softwarearchitektur

Einbettung in Dokumentation

14

Agenda

Probleme/Schwächen von Diagrammen

Diagrams-as-Code

Sichten der Softwarearchitektur

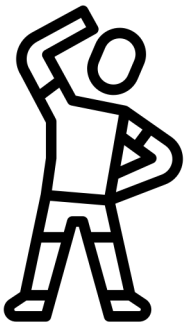
Einbettung in Dokumentation

15

Schlechte Diagramme?

16

Übung: Schlechte Diagramme für Euch!



Bitte folgt dem Link für eine kleine Übung (01)

Mural

https://tinyurl.com/sas-muenchen-2023

17

Schreckliche Diagramme

Wilde Farbkombinationen

Unklare Symbole

UML-Notation halbherzig angewendet

Verschiedene Pfeile/Linien ohne Erklärung

Vermischung von UML-Diagrammarten

Viele Pfeile, wenig Platz

Pfeile/Linien überkreuzen sich

18

Einleitung

Qualitätskriterien

?

19

Tipp 1: Qualitätskriterien für Diagramme

Ein Diagramm dient der Kommunikation

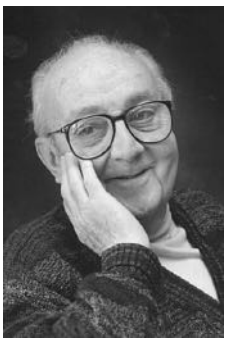
Darum muss es lesbar und verständlich sein

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

22

Was ist ein Modell?

Bild von Hans Benn auf Pixabay

23

Was ist ein Modell?

Bild von Hans Benn auf Pixabay

https://www.wolfsburgwest.com/cart/DetailsList.cfm?ID=211971013B

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

26

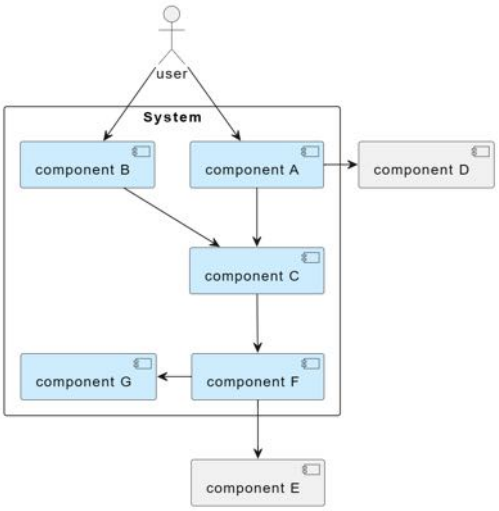
Visuelle Eigenschaften

?

Element

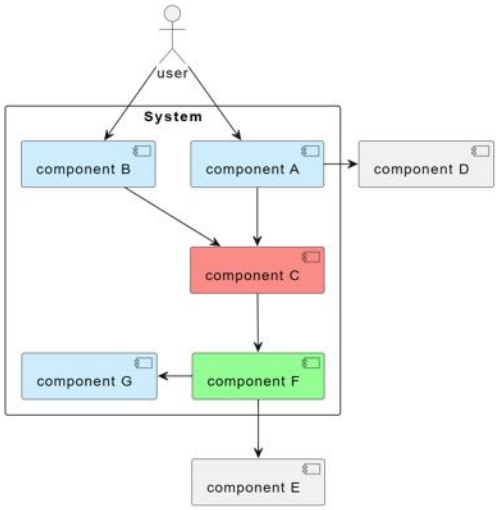
27

Ein Beispiel



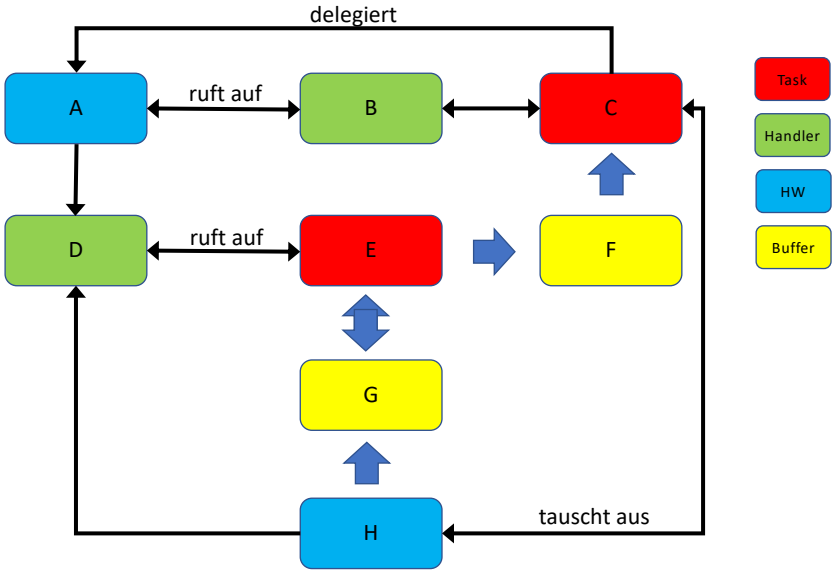
28

Farben



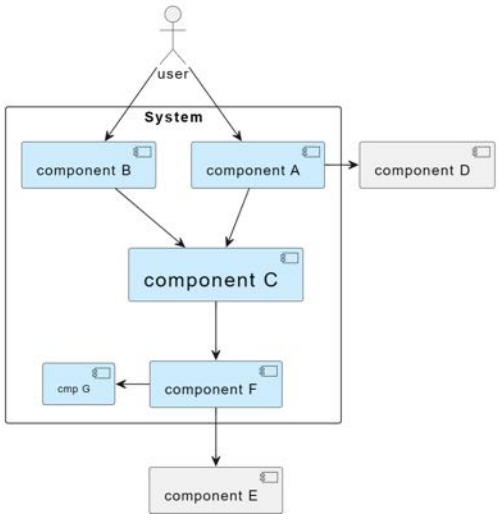
29

So nicht…



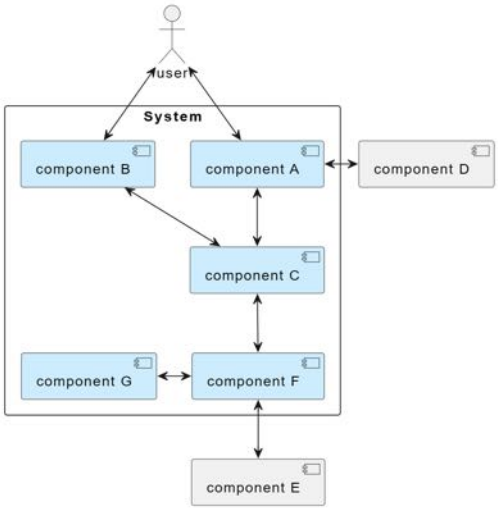
30

Größe



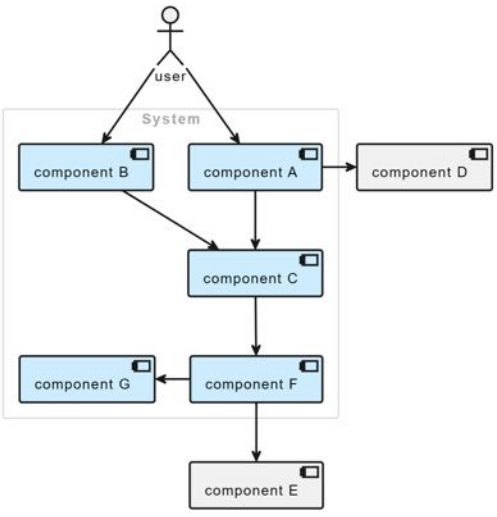
31

Pfeilrichtung



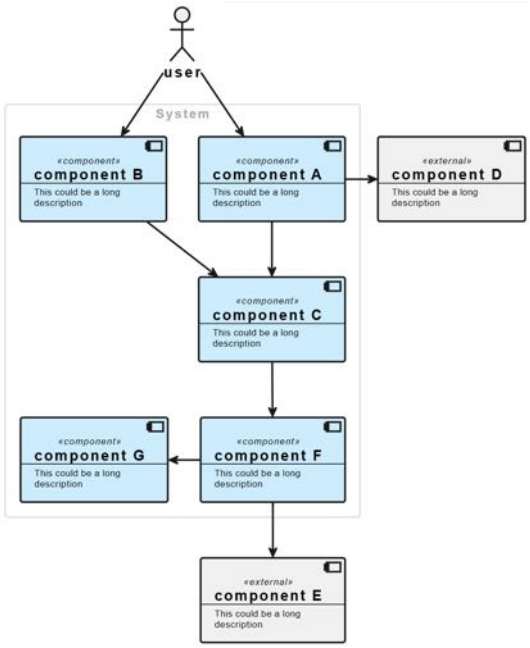
32

Linienstärke

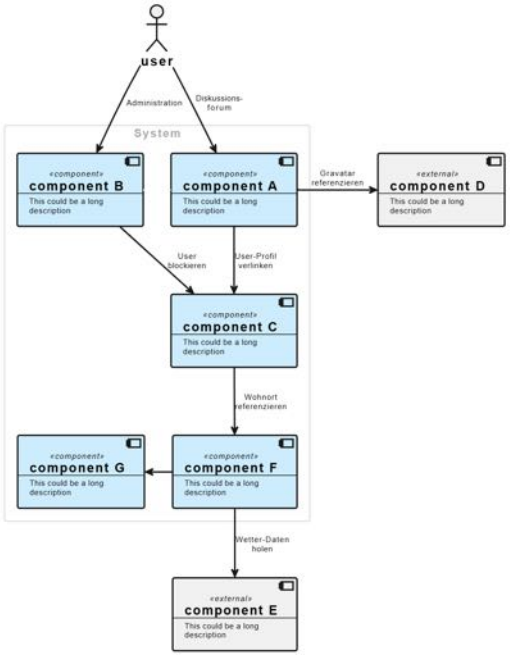


33

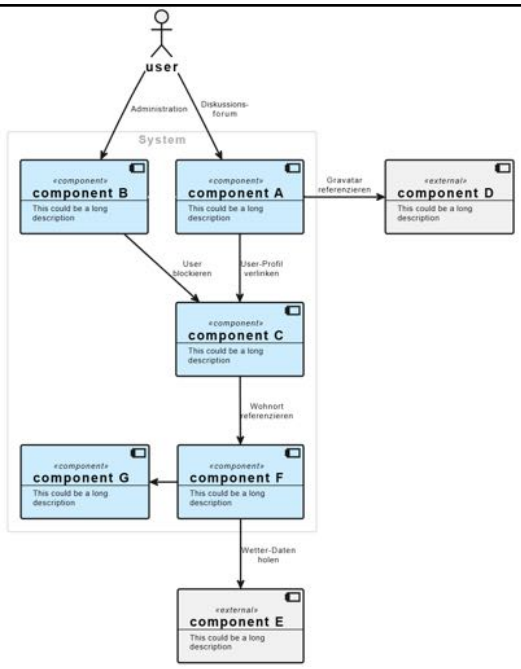
Text-Stile



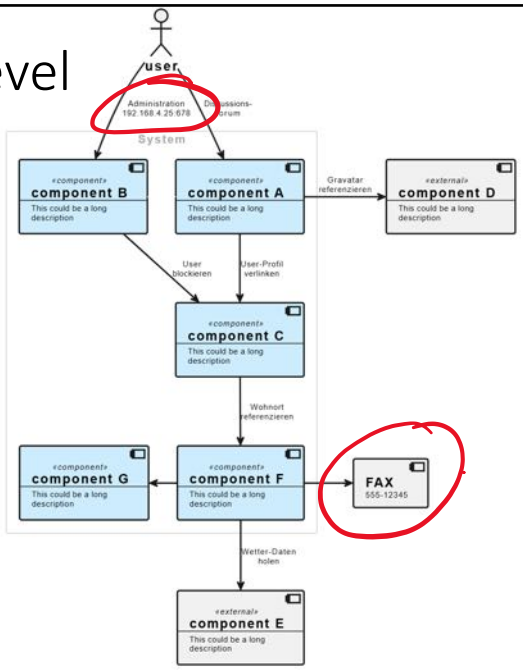
Beschriftung



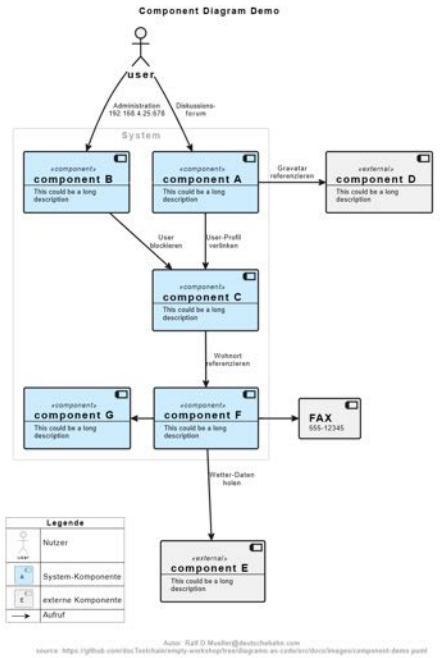
Beschriftung



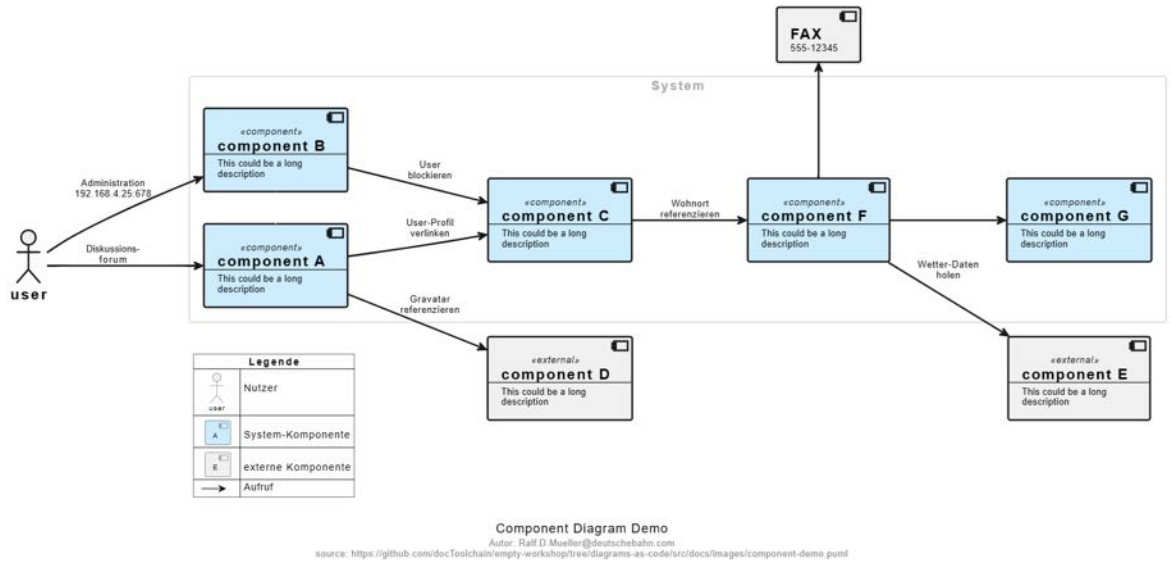
Abstraktionslevel



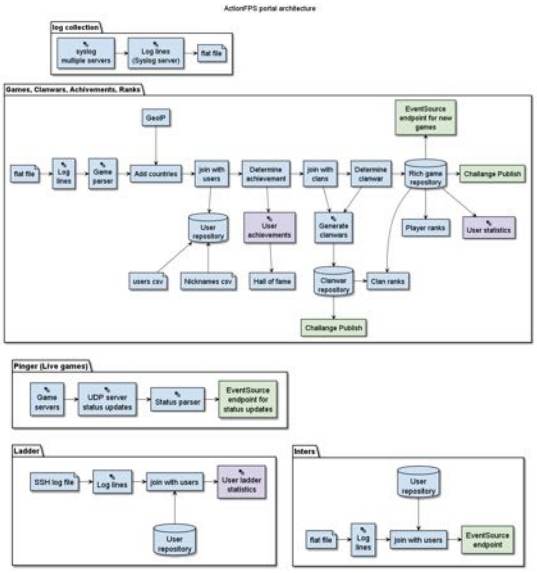
Legende



Landscape



Diagrammgröße



Tipp 3: Visuelle Stile

Farben sparsam und entsprechend ihrer natürlichen Bedeutung einsetzen

Decorator oder Formen zur Unterscheidung der Elemente einsetzen

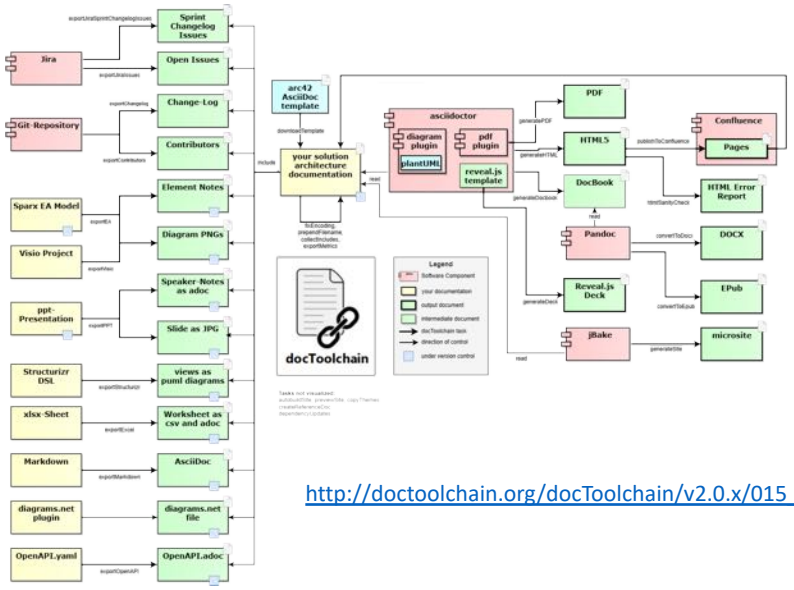
A **A** Schriftgrößen und Stile bewusst einsetzen

Verbindungen und Elemente beschriften

Unterschiedliche Größen bewusst einsetzen

Verbindungen nicht überlappen und nur in eine Richtung

Mit Diagrammen kommunizieren

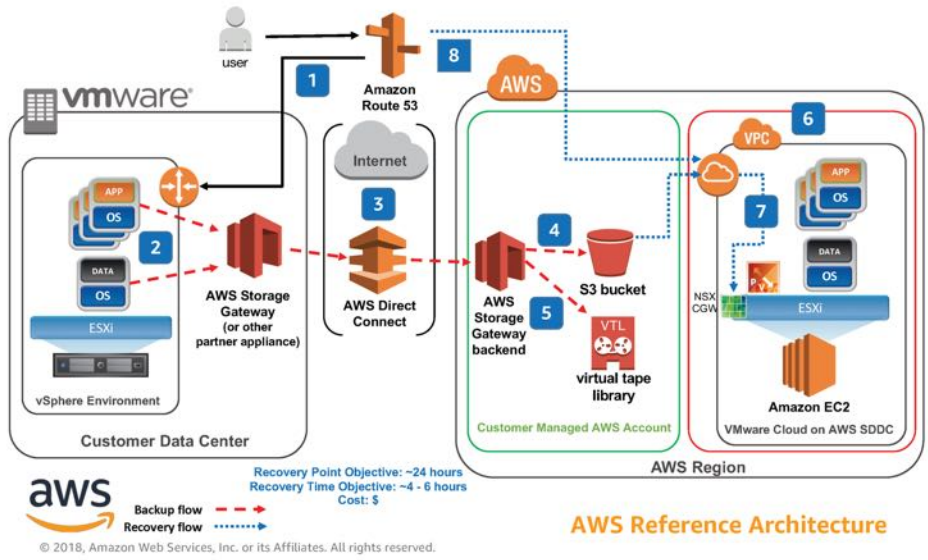


http://doctoolchain.org/docToolchain/v2.0.x/015_tasks/03_tasks.html

Visuelle Sprachen: "Meta-Modelle"

Backup and Restore to VMware Cloud on AWS

Native Services Integration: Storage Gateway, S3, Direct Connect, and Route53

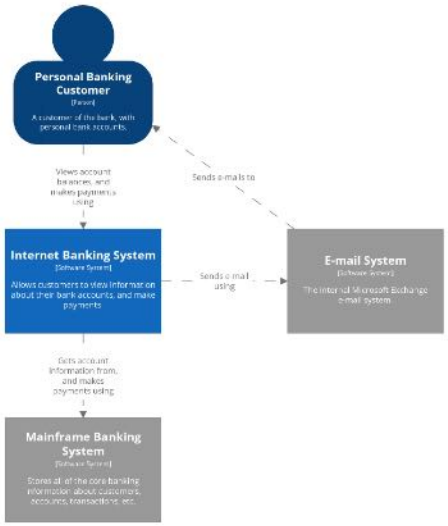


AWS Reference Architecture

#	Description
1	Amazon Route 53 routes DNS requests to the primary domain controller on-premises.
2	VM and application backups are stored in Amazon S3 using an AWS Storage Gateway or other storage appliance using a partner-integrated solution or application-level backup software.
3	The on-premises Storage Gateway securely transfers the backup data to the Storage Gateway backend using Direct Connect or through an SSL Internet connection.
4	File gateway uses an AWS Identity and Access Management role to access the customer backup data and securely store it in Amazon S3.
5	Use the virtual tape library configuration in the Storage Gateway for long term data archiving to AWS Glacier or other archive service.
6	The recovery process starts by launching and configuring a VMware SDDC cluster in AWS with the web portal or through automation scripts using AWS CloudFormation, VMware vRA, or vCLI.
7	After VMware Cloud on AWS SDDC is ready, deploy the software to restore the backed up application and VM data from Amazon S3.
8	The final recovery step is updating the Route 53 DNS records to route new requests to secondary domain controller in AWS.

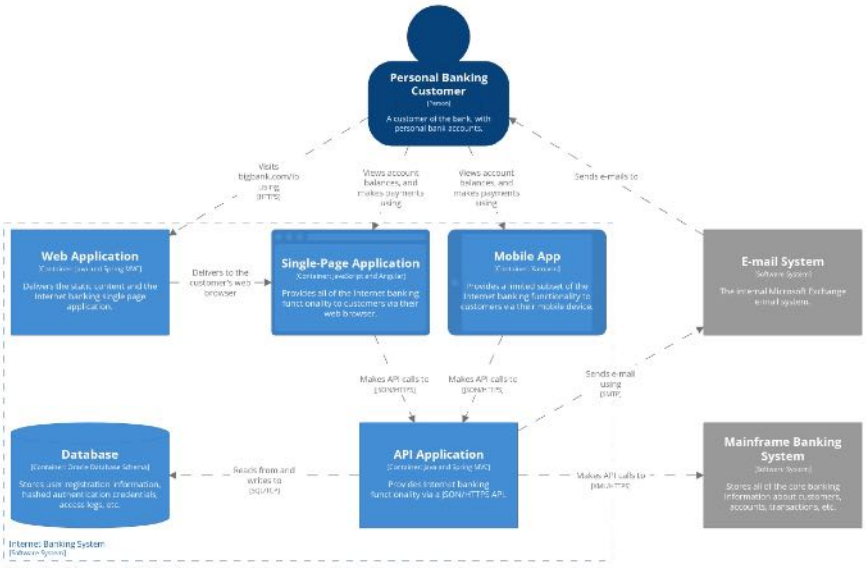
c4model.com

c4model.com: Context



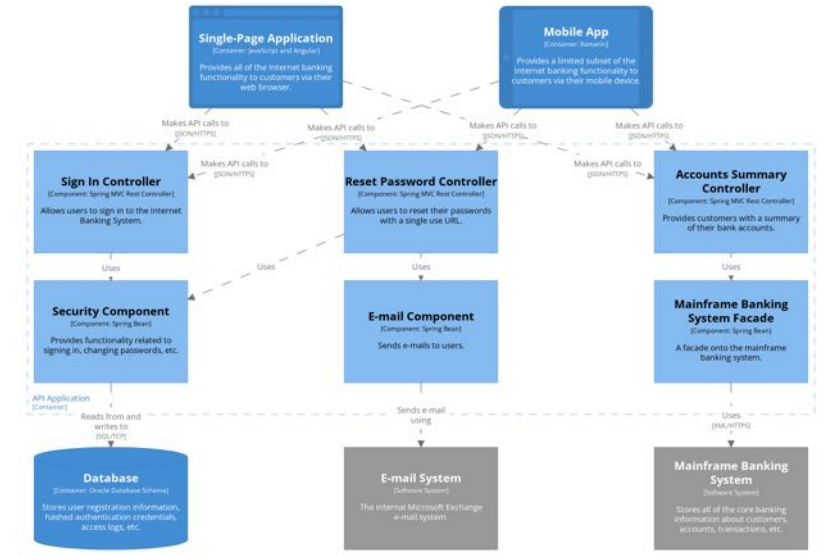
[System Context] Internet Banking System

c4model.com: Containers



[Container] Internet Banking System

c4model.com: Components

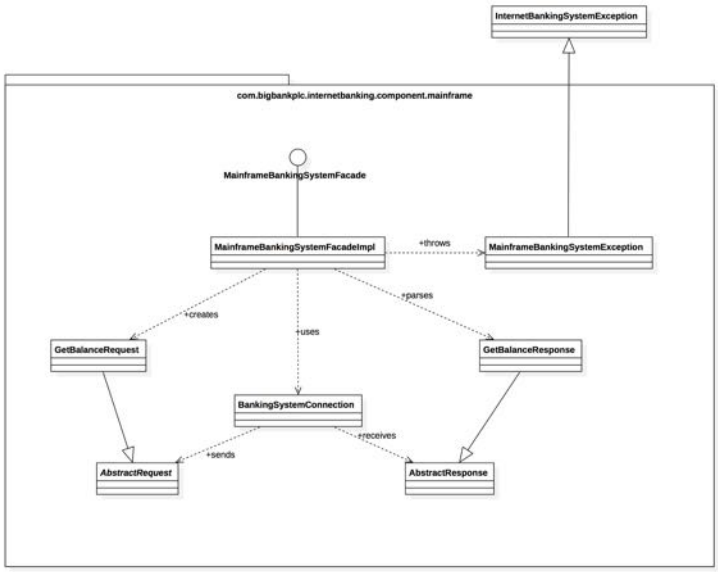


[Component] Internet Banking System - API Application

48

48

c4model.com: Classes



49

Quelle der C4-Diagramme c4model.com

49

Agenda

Probleme/Schwächen von Diagrammen

Diagrams-as-Code

Sichten der Softwarearchitektur

Einbettung in Dokumentation

51

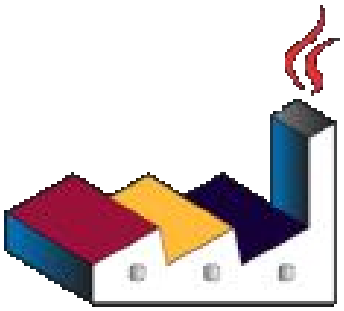
Interaktiv

https://github.com/docToolchain/empty-workshop

52

Diagramme

- http://plantuml.com/
- http://asciidoctor.org/docs/asciidoctor-diagram/
- Komplexe Diagramme als einfachen Text verwalten



PlantUML

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

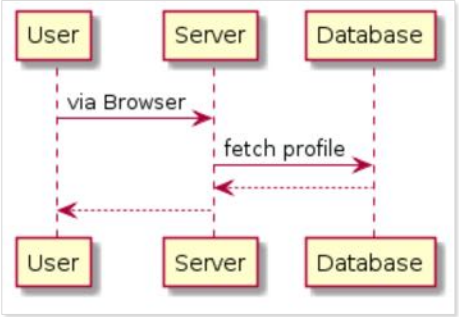
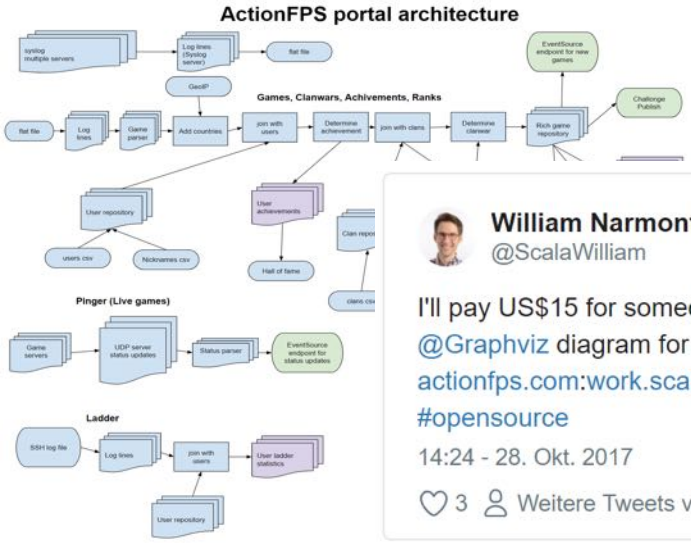


Diagramme: PlantUML



William Narmontas
@ScalaWilliam

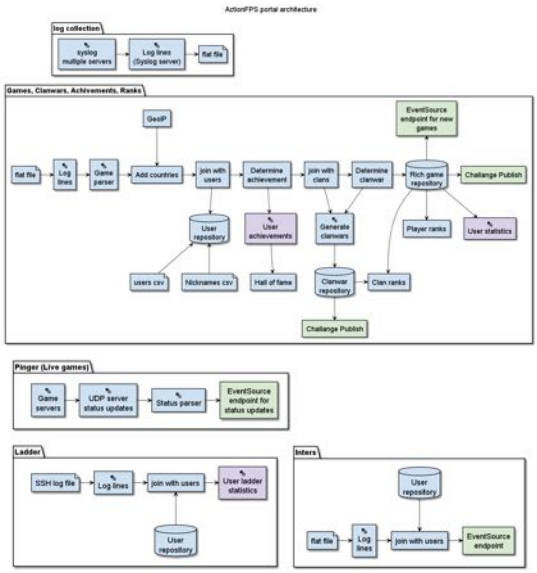
I'll pay US$15 for someone who can do this @PlantUML or @Graphviz diagram for actionfps.com:work.scalawilliam.com/graphviz-plant… #opensource

14:24 - 28. Okt. 2017

♡ 3 Weitere Tweets von William Narmontas ansehen

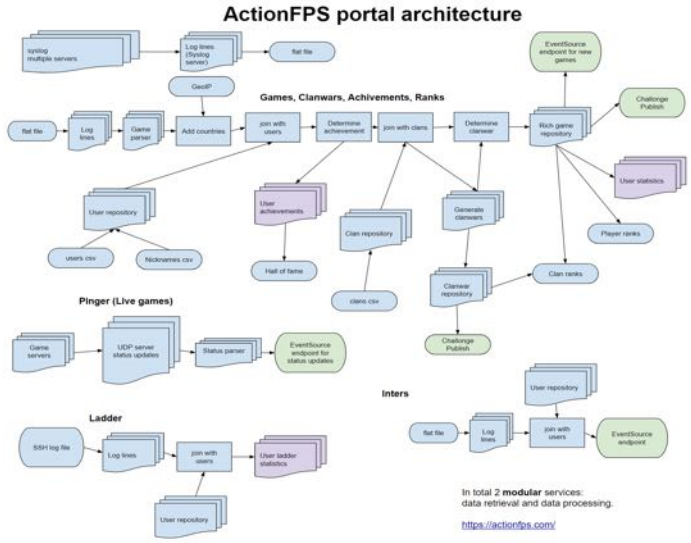
55

Diagramme: PlantUML

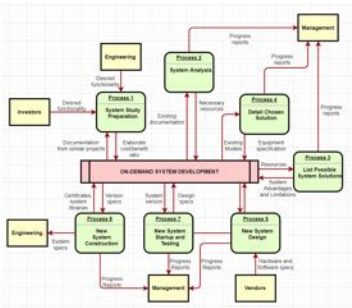


56

Diagramme: PlantUML



Diagrams.net

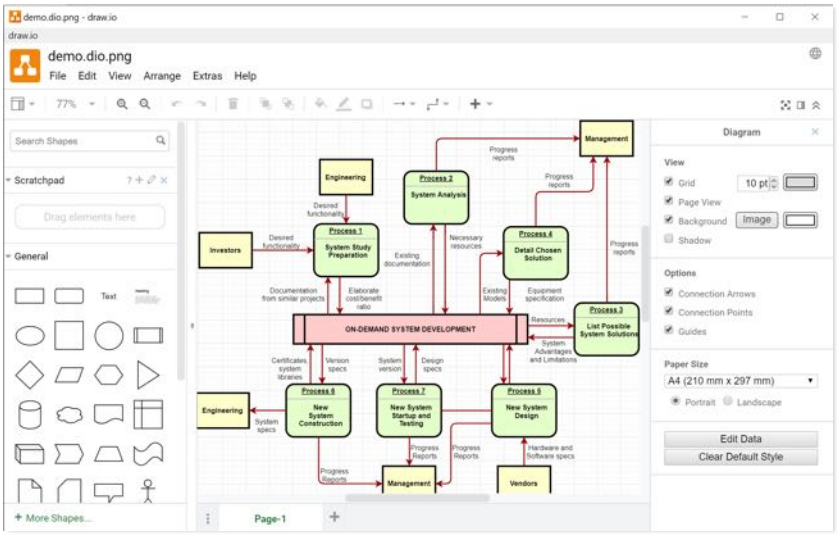


<xml>
...
</xml>

PNG-Diagramm

Meta-Daten

Diagrams.net



Demo Time!

- Diagrams.net

Vega – A Visualization Grammar

https://vega.github.io/vega/

61

Kroki

Creates **diagrams** from **textual** descriptions!

Kroki provides a unified API with support for BlockDiag (BlockDiag, SeqDiag, ActDiag, NwDiag, PacketDiag, RackDiag), BPMN, Bytefield, C4 (with PlantUML), Ditaa, 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.

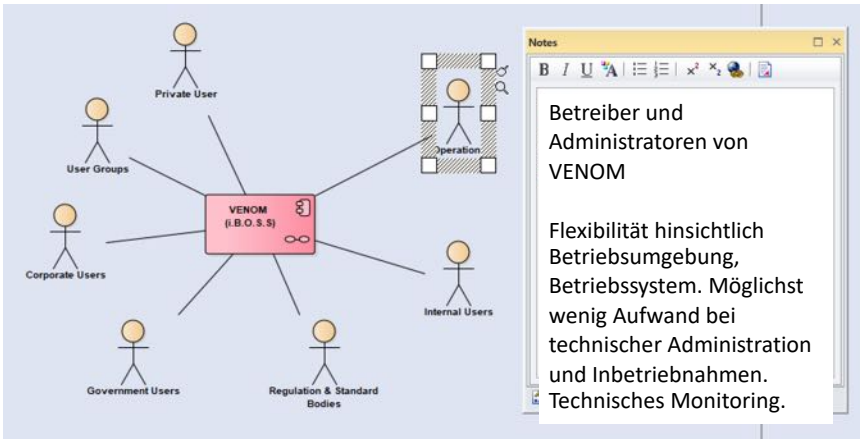
https://kroki.io/

62

Diagramme: Nicht malen, modellieren!

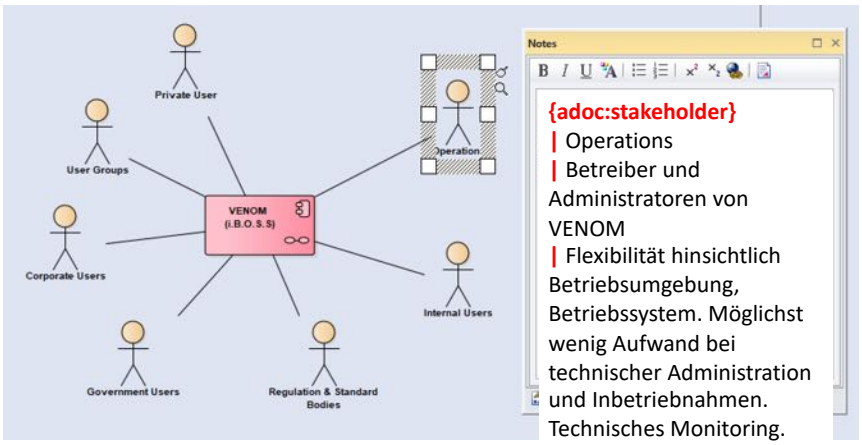
63

treat Docs-as-Code: automate!

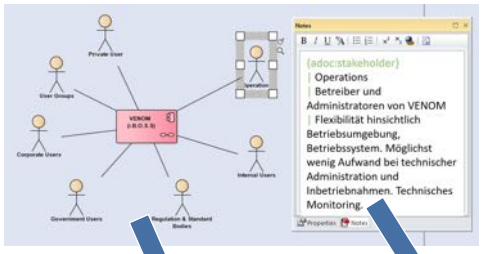


64

treat Docs-as-Code: automate!



treat Docs-as-Code: automate!

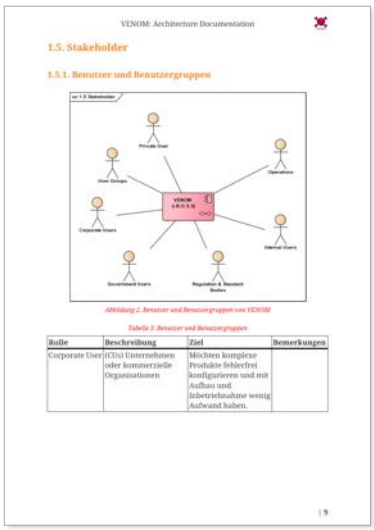



```
=== Stakeholder

==== 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[]

|===
```

Demo Time!

- Enterprise Architect

treat Docs-as-Code: automate!


VENOM Architecture Documentation

1.5. Stakeholder

1.5.1. Benutzer und Benutzergruppen

Rolle	Beschreibung	Ziel	Bemerkungen
Corporate User	(CUs) Unternehmen oder kommerzielle Organisationen	Möchten komplexe Produkte fehlerfrei konfigurieren und mit Aufbau und Inbetriebnahme wenig Aufwand haben.	

Agenda

Probleme/Schwächen von Diagrammen

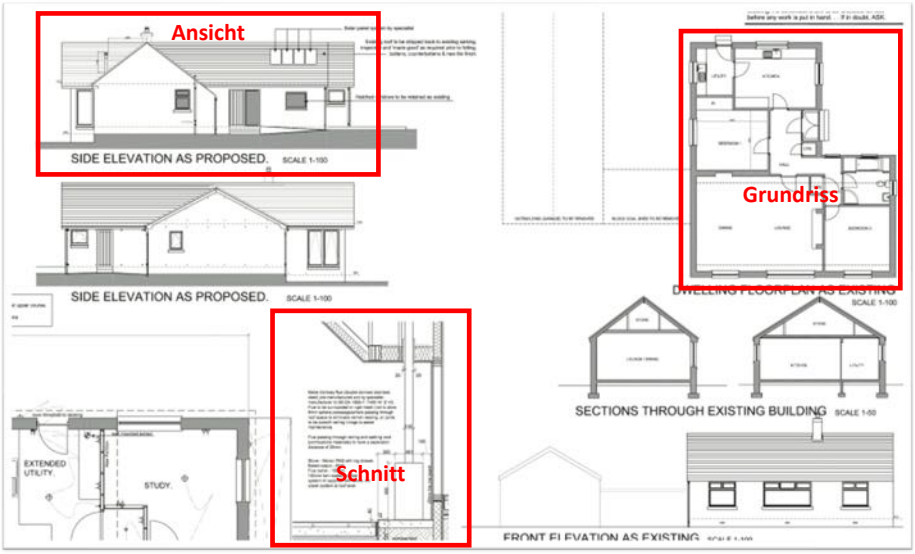
Diagrams-as-Code

Sichten der Softwarearchitektur

Einbettung in Dokumentation

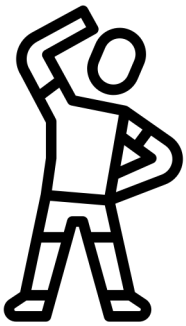
70

Analogie Bauwesen



71

Übung: Welche Diagramme nutzt Ihr?



Bitte folgt dem Link für eine kleine Übung (02)

https://tinyurl.com/sas-muenchen-2023

72

Sichten in 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/

73

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.

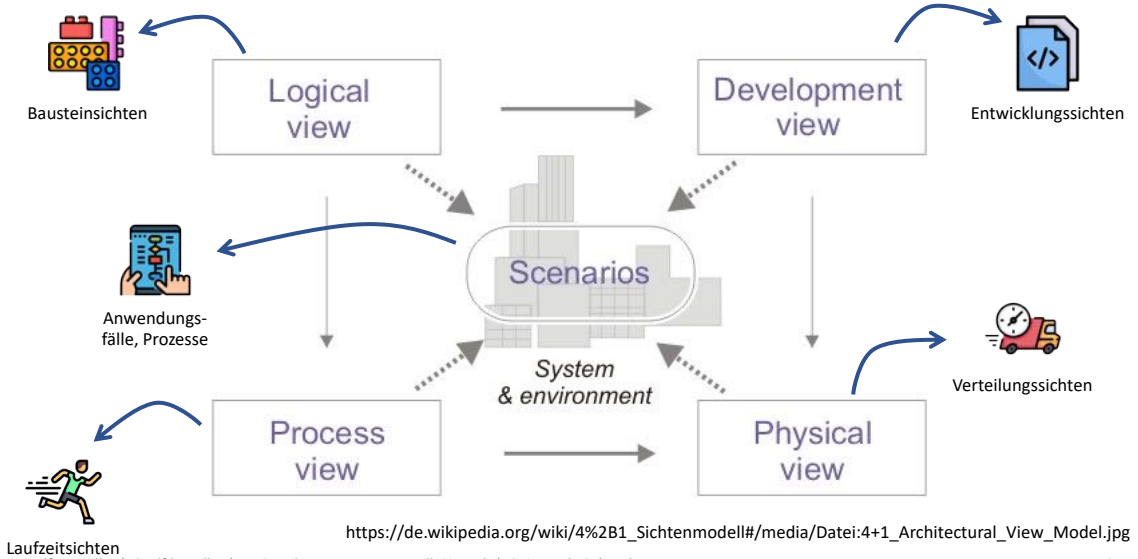
Ursprung: 4+1 Sichten

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

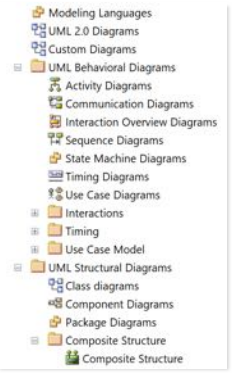
Elemente des 4+1



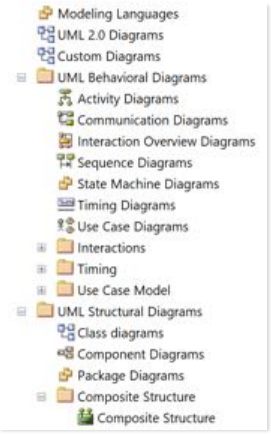
https://de.wikipedia.org/wiki/4%2B1_Sichtenmodell#/media/Datei:4+1_Architectural_View_Model.jpg

To UML or to not UML?

> ***"Wir benutzen für unsere Architekturdiagramme UML. Ist standardisiert. Kann jeder lesen."***
>
> *-- unbekannte Softwarearchitekt:in*



UML - Überblick



Strukturdiagramme

- Klassendiagramm
- Paketdiagramm
- Komponentendiagramm
- Kompositionsstrukturdiagramm
- Verteilungsdiagramm

Verhaltensdiagramme

- Sequenzdiagramm
- Aktivitätsdiagramm
- Zustandsdiagramm
- Anwendungsfalldiagramm

78

Mögliche UML-Diagrammarten

Klassendiagramm
Kommunikations-diagramm
Sequenzdiagramm

Komponenten-diagramm
Paketdiagramm

Anwendungsfall-diagramm

Sequenzdiagramm
Aktivitätsdiagramm

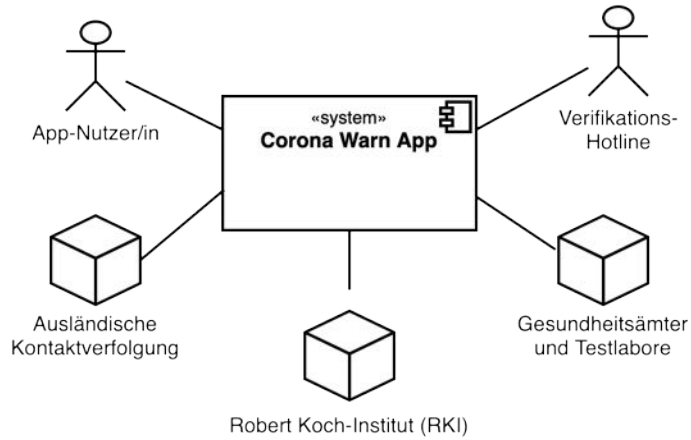
Verteilungs-diagramm

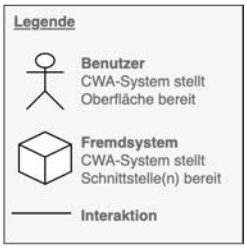
79

Kontextdiagramme

Kontextabgrenzung

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



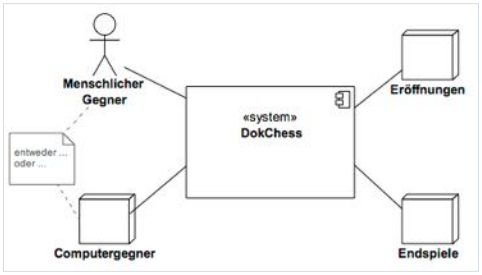


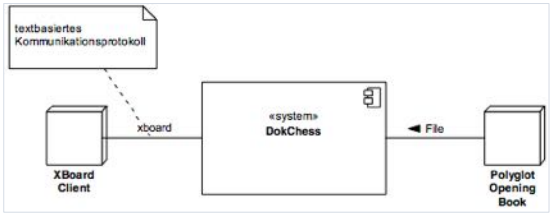
Fachlicher Systemkontext, Akteure

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.

Fachlich vs. Technischer Kontext





fachliche Sicht auf Benutzer, Fremdsysteme und Prozesse		technische Sicht (Schnittstellen, …)
Fachanwender, Product-Owner, Tester, Entwickler		Entwickler, Betrieb
Geschäftsprozesse bleiben typischerweise stabil		Könnte in ein paar Jahren schon ganz anders aussehen …

DOKCHESS

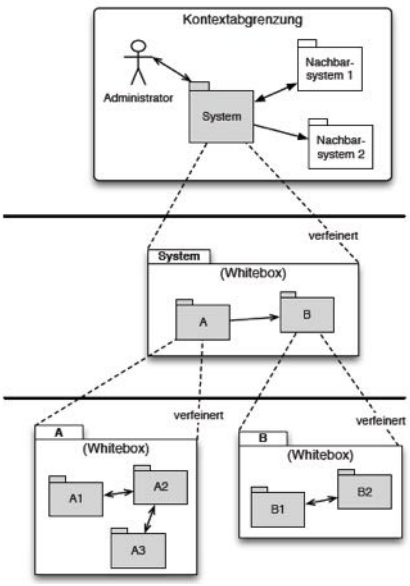
https://www.dokchess.de/

Bausteinsichten

Level 0

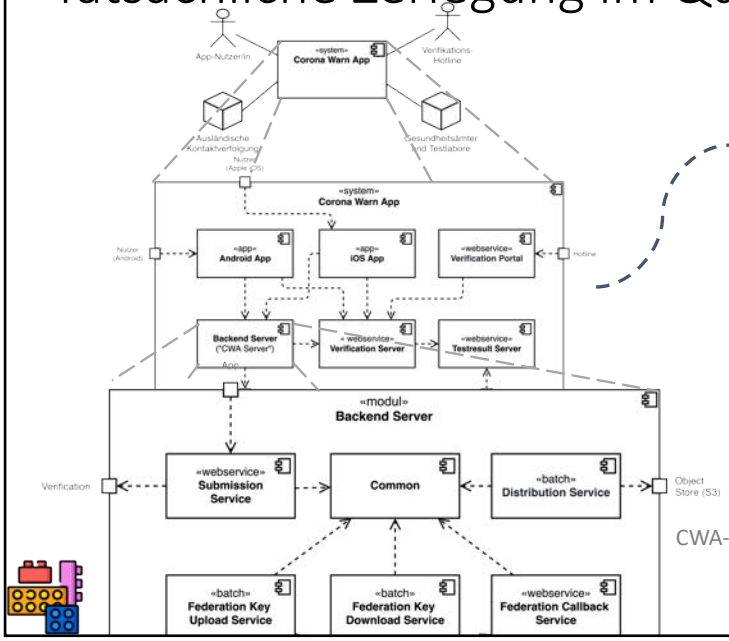
Level 1

Level 2



Nicht zu detailliert, sondern pragmatisch und effizient!

Tatsächliche Zerlegung im Quelltext



„Bausteinsicht, Ebene 1“

GitHub-Repositories

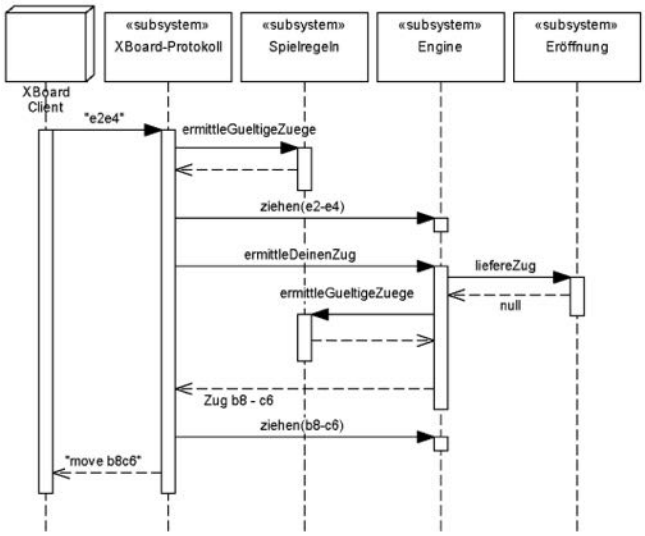
https://github.com/corona-warn-app/...

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

CWA-Server (Backend), Bausteinsicht, Ebene 2

Laufzeitsichten

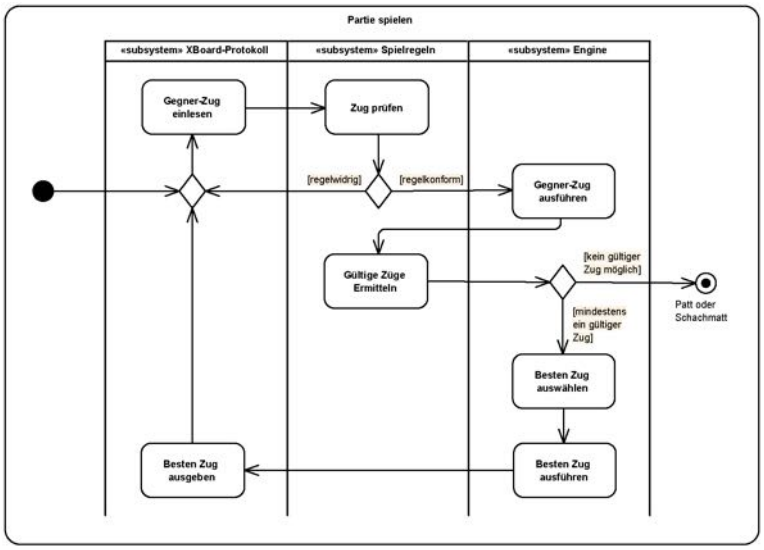
Beispiel eines Ablaufes



DOKCHESS

https://www.dokchess.de/

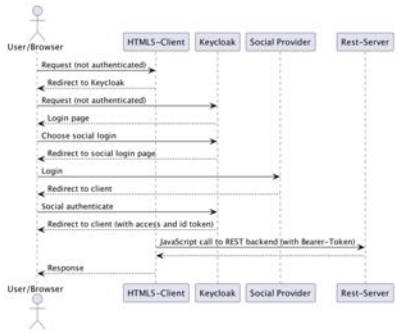
Beispiel eines Ablaufes



DOKCHESS

https://www.dokchess.de/

Sequenzdiagramm mit PlantUML



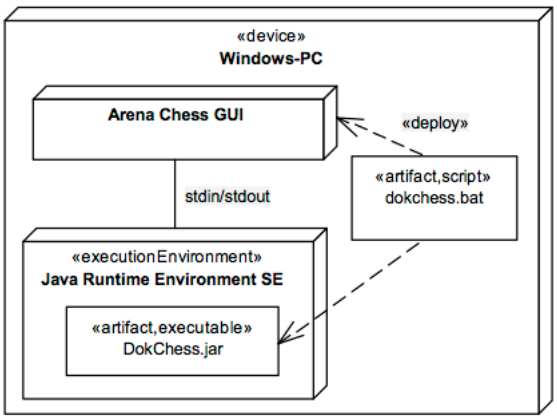

```
@startuml
actor browser as "User/Browser"
participant html5 as "HTML5-Client"
participant Keycloak
participant socialprovider as "Social Provider"
participant server as "Rest-Server"

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

91

Verteilungssichten

92

Beispiel Verteilungssicht

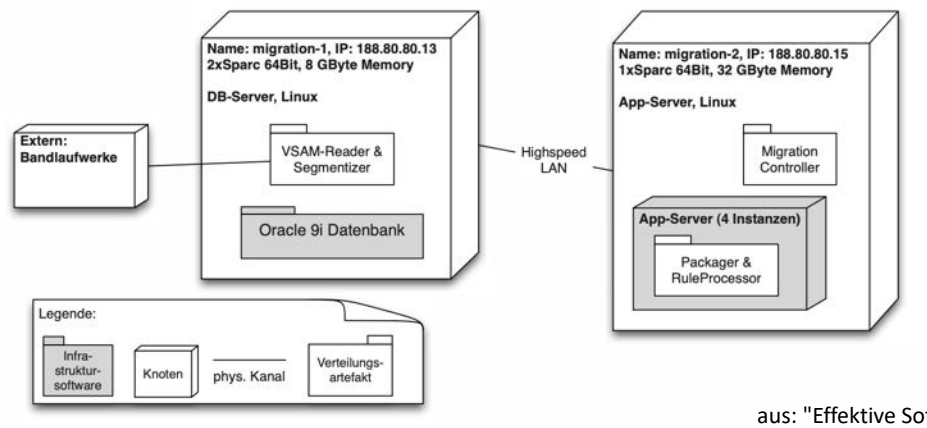


DOKCHESS

https://www.dokchess.de/

93

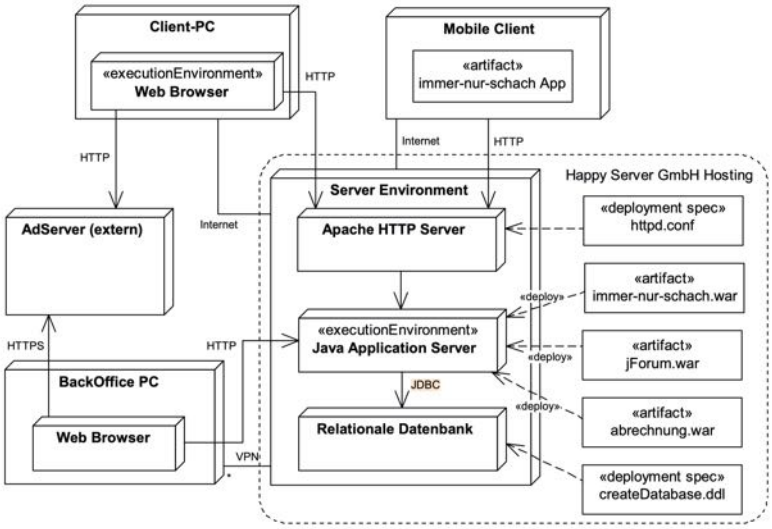
Beispiel Verteilungssicht



aus: "Effektive Software-architekturen"
von Gernot Starke

94

Verteilungssicht "immer-nur-schach"



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

95

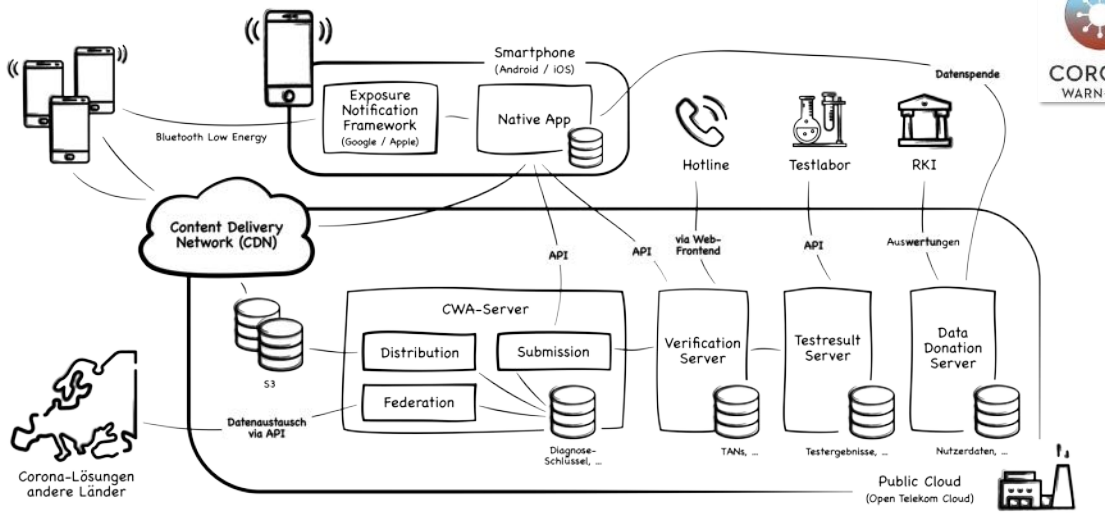
Was noch für Diagramme?

96

Waren das schon alle Sichten?

97

Informelles Überblicksbild

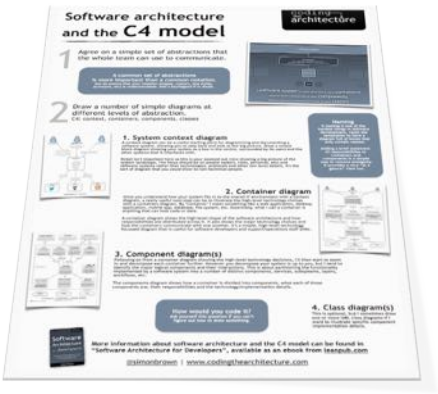


Quelle der Abbildung: S. Zörner, F. Sippach: „So gehen Architektur-Reviews! Entlang der Corona-Warn-App", OOP 2021

98

Alternativen

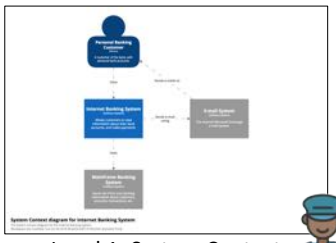
Alternative: C4 Model (Simon Brown)



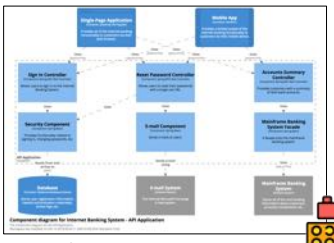
- Agree on a simple set of abstractions that the whole team can use to communicate.
- Draw a number of simple diagrams at different levels of abstraction.
- C4: context, containers, components, classes (code)

➔ http://static.codingthearchitecture.com/c4.pdf

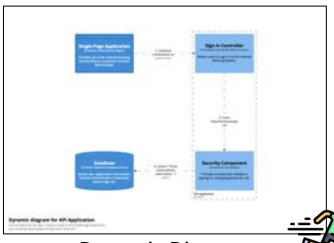
Alternative C4: Inhalte



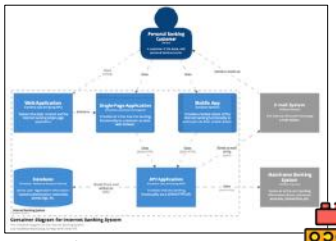
Level 1: System Context



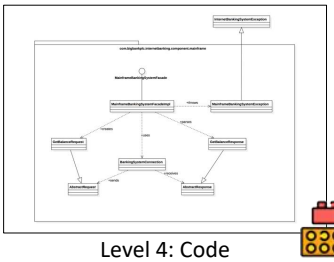
Level 3: Component Diagram



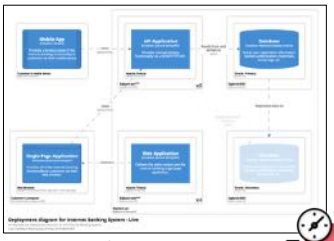
Dynamic Diagram



Level 2: Container Diagram



Level 4: Code



Deployment Diagram

101

Alle Sichten auf einen Blick

4 + 1 (Kruchten)

Bausteinsichten

Entwicklungssichten

Kontextsichten

Anwendungsfälle/
Prozesse

Informelles
Überblicksbild

Laufzeitsichten

Verteilungssichten

102

Diagramm-Tools - Empfehlungen

Bausteinsichten

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

Entwicklungssichten

draw.io/diagrams.net

Kontextsichten

draw.io/diagrams.net

Anwendungsfälle/ Prozesse

PlantUML (Anwendungsfall-diagramm)

Informelles Überblicksbild

draw.io/diagrams.net

Laufzeitsichten

PlantUML (Sequenzdiagramm, Aktivitätsdiagramme)

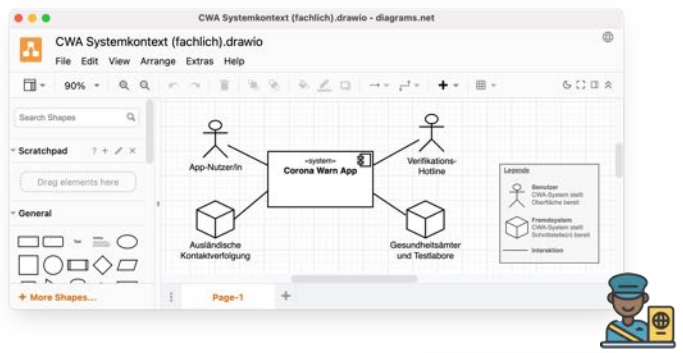
Verteilungssichten

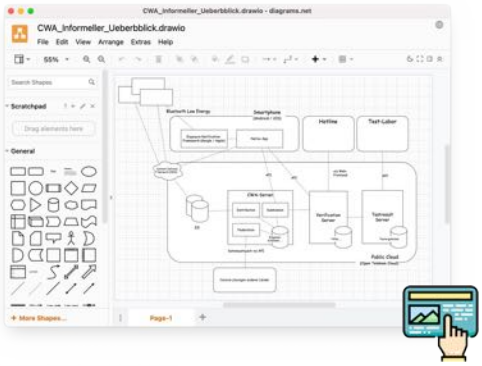
PlantUML (Sequenzdiagramm, Aktivitätsdiagramme)

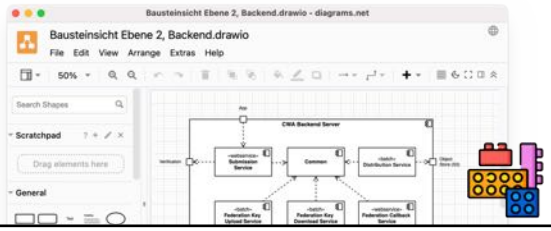
103

103

Beispiele Umsetzung in **diagrams.net**







CORONA WARN-APP

104

104

Agenda

Probleme/Schwächen von Diagrammen

Diagrams-as-Code

Sichten der Softwarearchitektur

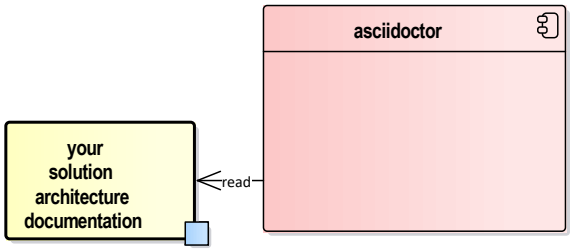
Einbettung in Dokumentation

105

Basic Docs-as-Code with AsciiDoc

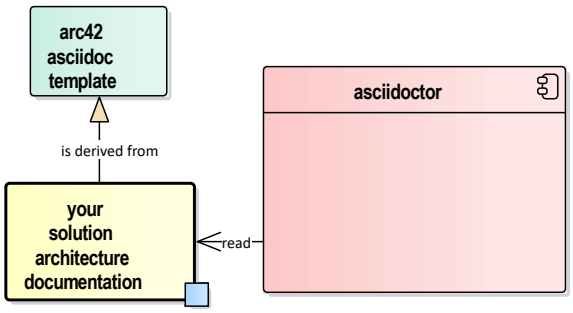
106

Basic Docs-as-Code with AsciiDoc



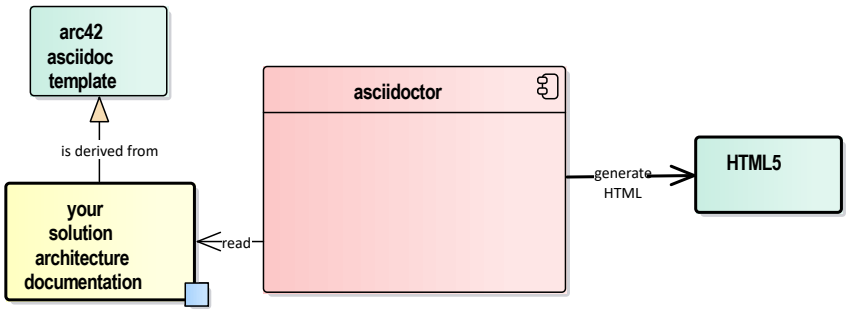
107

Basic Docs-as-Code with AsciiDoc



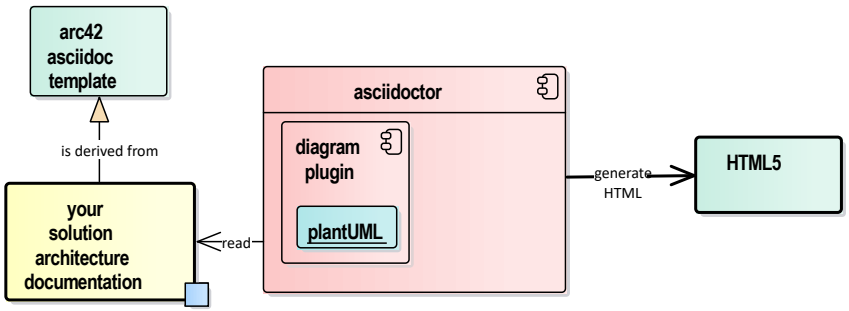
108

Basic Docs-as-Code with AsciiDoc



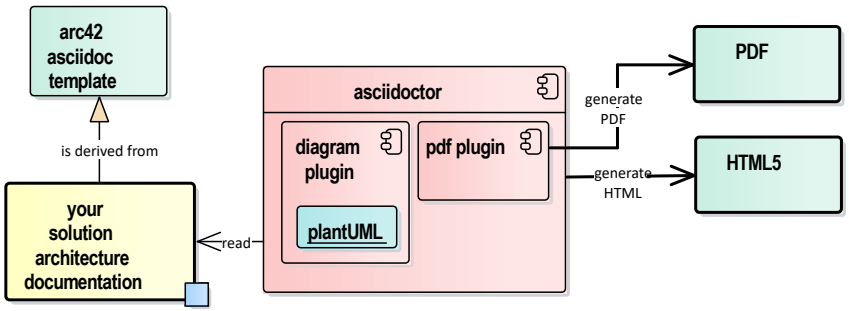
109

Basic Docs-as-Code with AsciiDoc



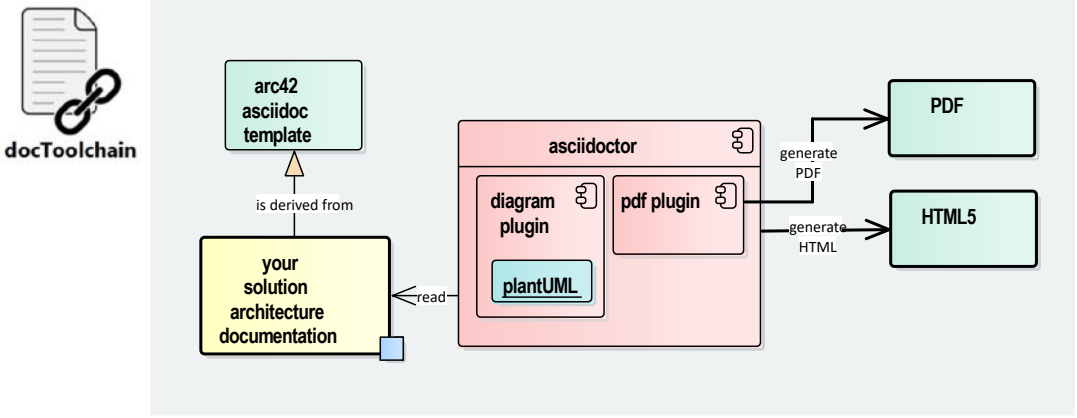
110

Basic Docs-as-Code with AsciiDoc



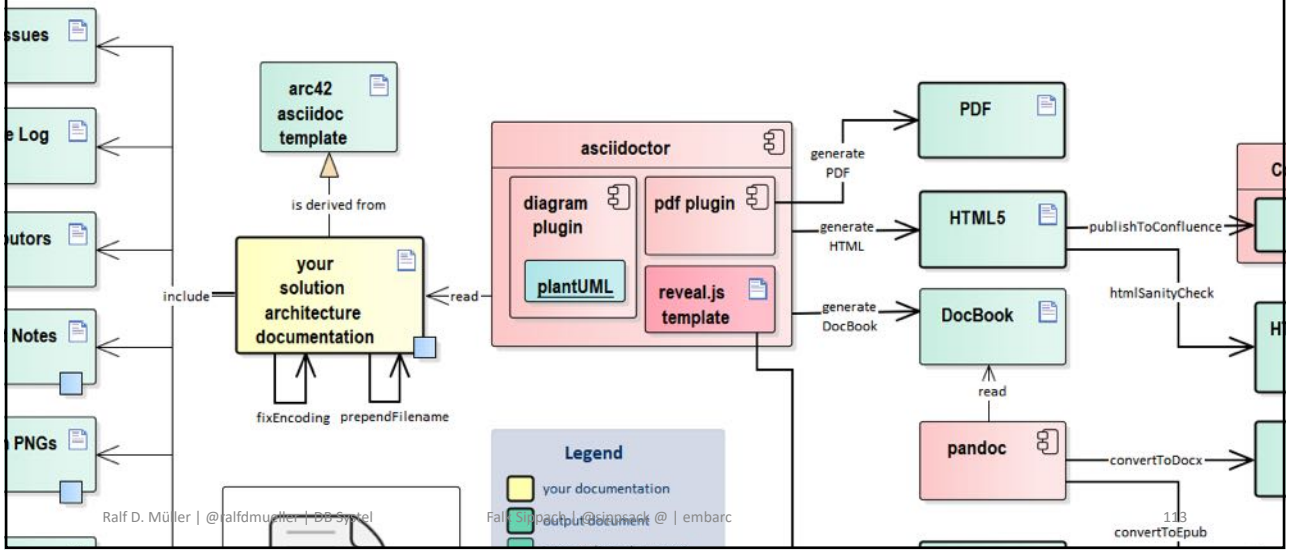
111

Basic Docs-as-Code with AsciiDoc & docToolchain



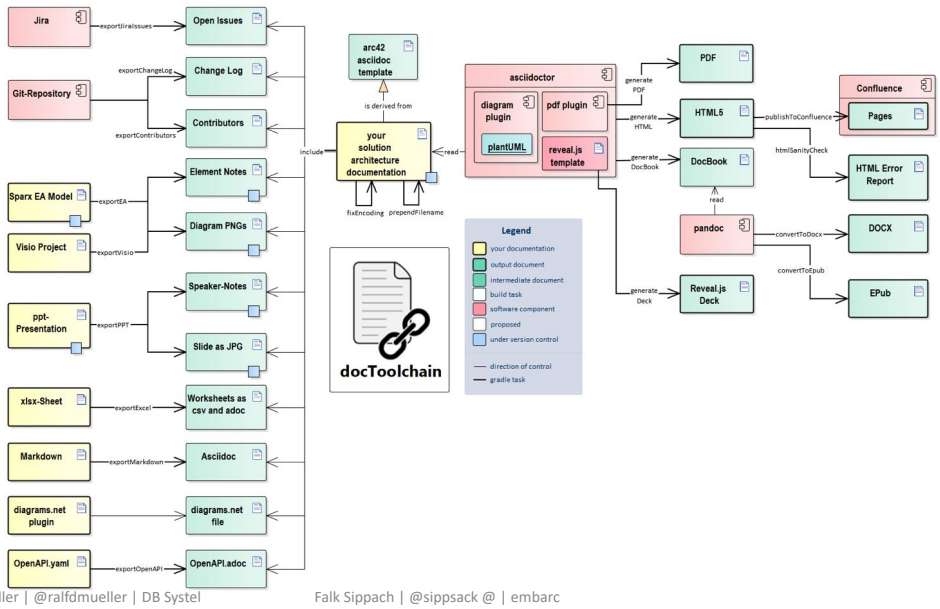
112

Basic Docs-as-Code with AsciiDoc & docToolchain



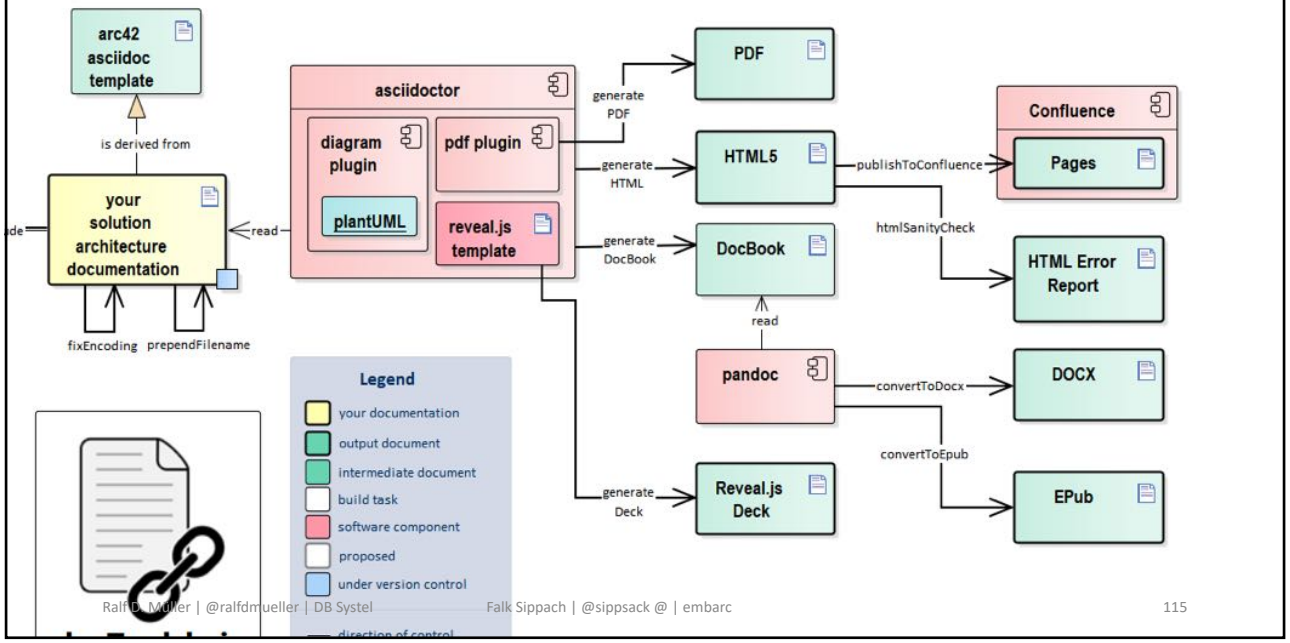
113

Basic Docs-as-Code with AsciiDoc & docToolchain



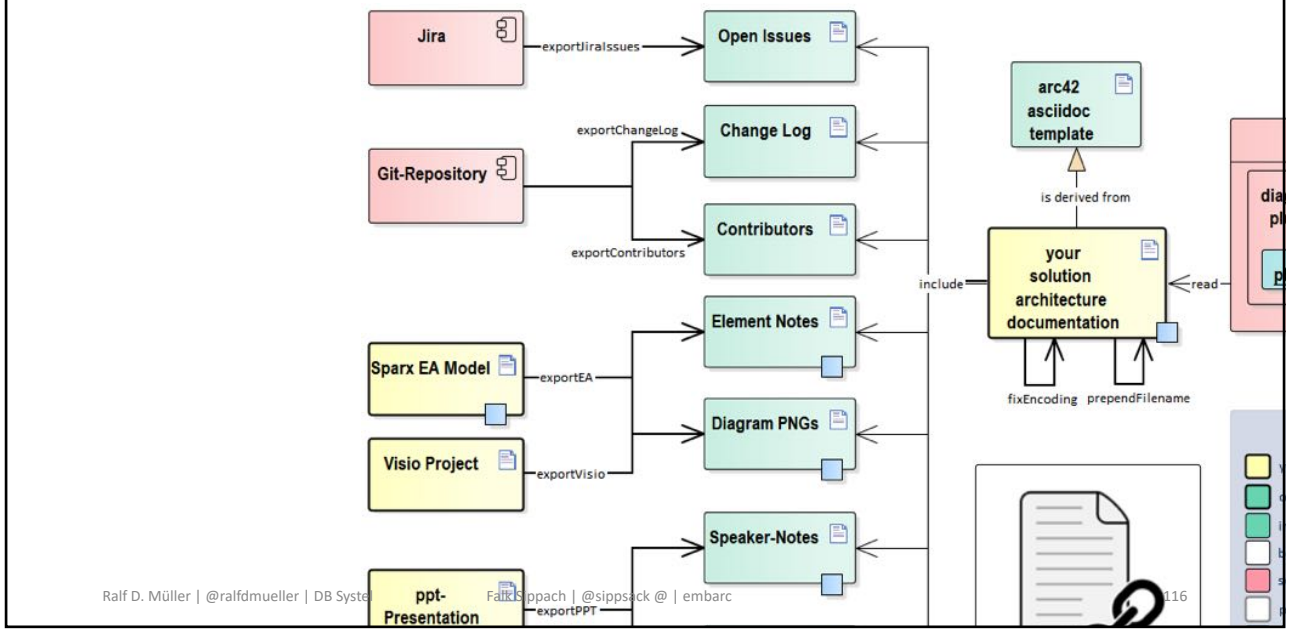
114

Basic Docs-as-Code with AsciiDoc & docToolchain



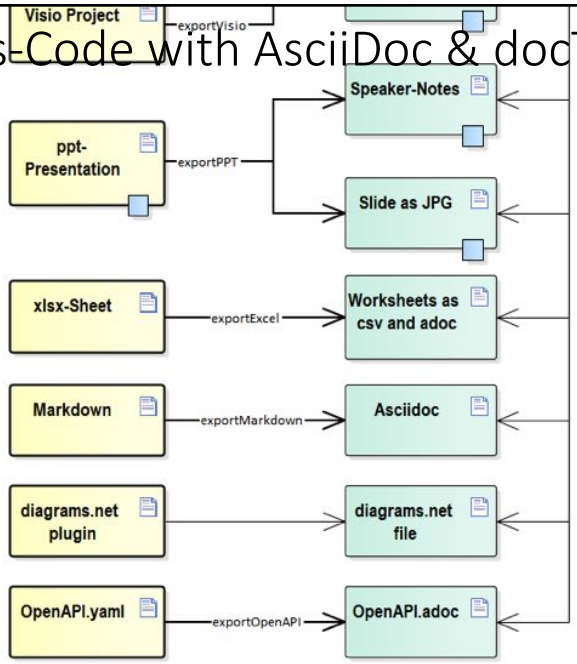
115

Basic Docs-as-Code with AsciiDoc & docToolchain



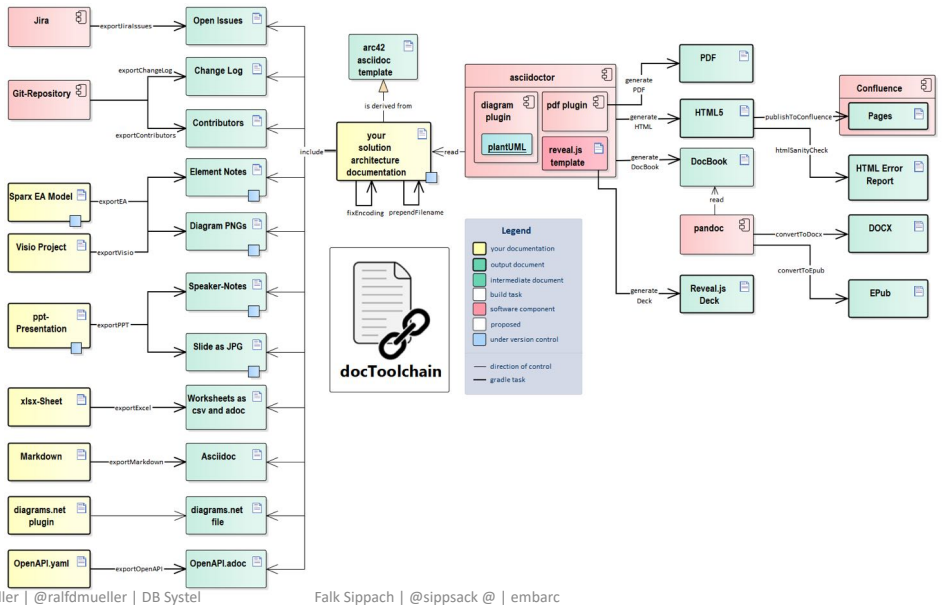
116

Basic Docs-as-Code with AsciiDoc & docToolchain



117

Basic Docs-as-Code with AsciiDoc & docToolchain



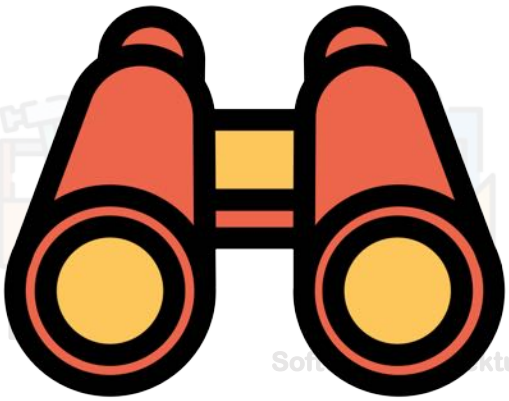
118

Demo Time!

- komplette arc42 Dokumentation mit docToolchain

119

Ausblick



Probleme/Schwächen von Diagrammen

Einbettung in Dokumentation

120

Checkliste für Architekturdiagramme

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

Diagrammzweck ersichtlich?

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

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

Sind die verwendeten **Abkürzungen ersichtlich**?

Bedeutung aller Farben und Formen ersichtlich?

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

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

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

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

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

Wurde ein **passendes Werkzeug** verwendet?

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

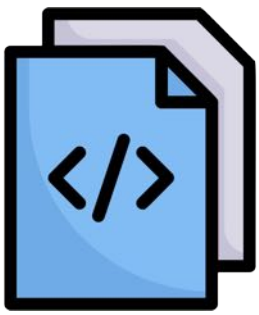
121

Vorteile Diagrams-as-Code (und leichtgewichtige Diagramme)

Jederzeit und von jedem editierbar.

Versionier- und historisierbar.

Leicht zu erstellen.



Direkte Unterstützung der Notation.

Entwicklertools verwendbar.

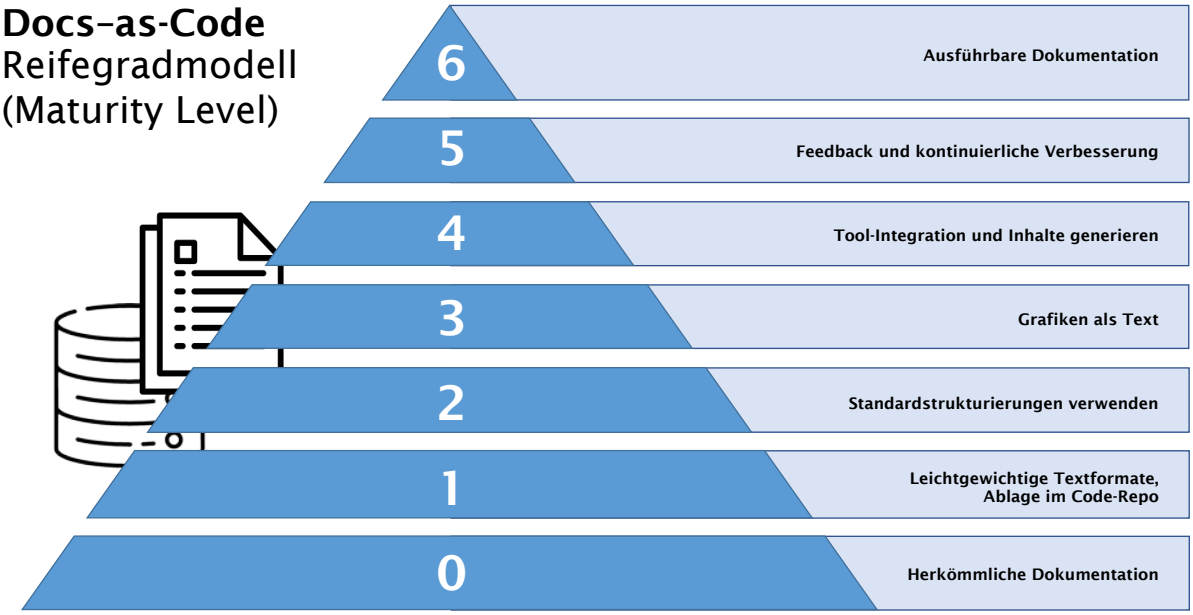
Leicht zu aktualisieren

Automatisches Layouting und Rendering.

einheitlicher Stil

122

Docs-as-Code Reifegradmodell (Maturity Level)

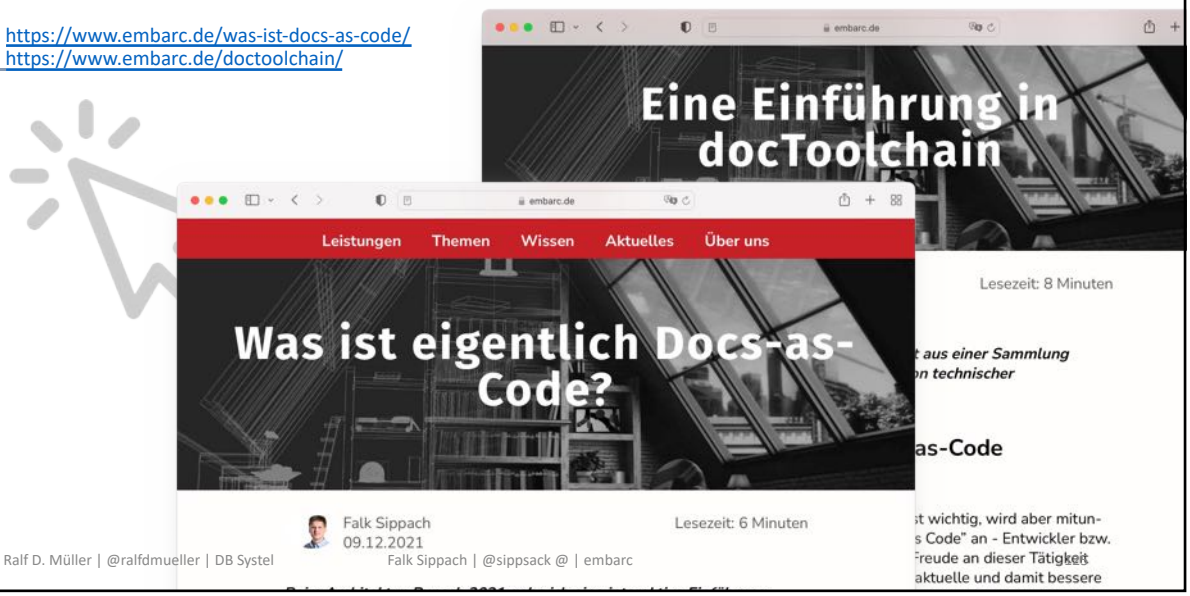


123

Weiterführende Blog-Posts

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

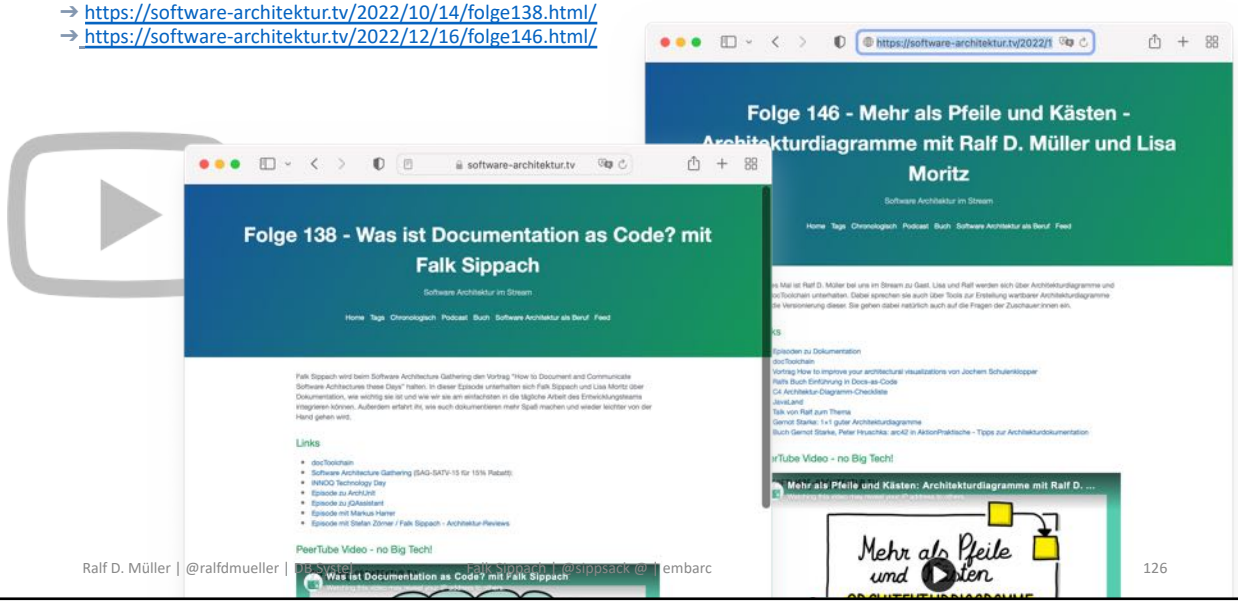
➔ https://www.embarc.de/was-ist-docs-as-code/
➔ https://www.embarc.de/doctoolchain/



125

Weiterführende Videos/Podcasts

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



126

Einführung in Docs-as-Code

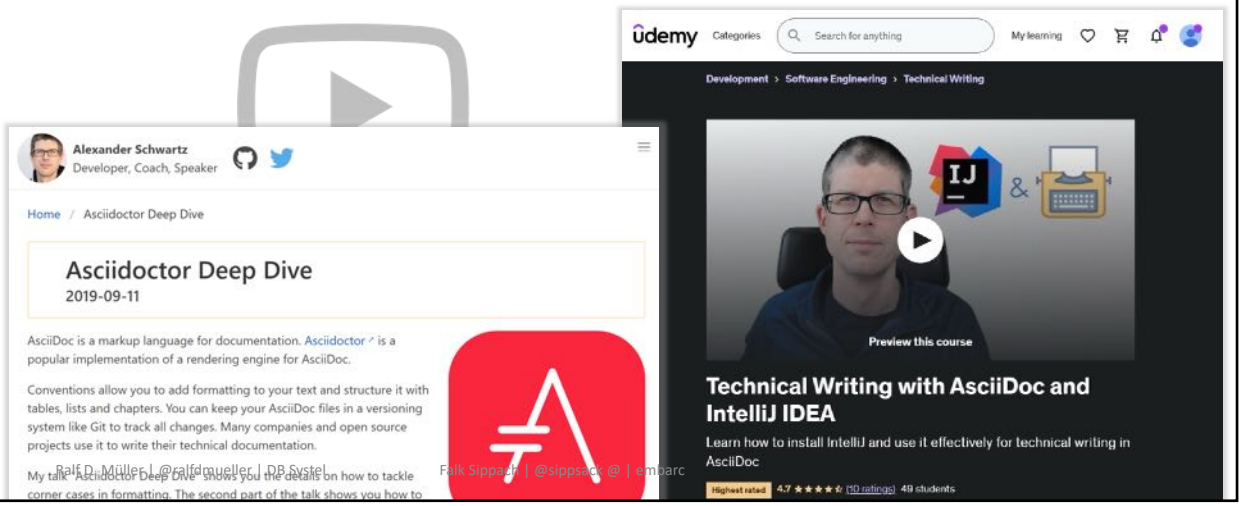
https://leanpub.com/praxisbuchdocs-as-code/c/SAS-Munich

127

Asciidoc

https://www.udemy.com/course/technical-writing-with-asciidoc-and-intellij-idea/?couponCode=10E18DB7D7561FBBAC25

https://www.ahus1.de/post/asciidoctor-intro-and-deep-dive



128

Spicken erlaubt!

PDF, 4 Seiten
Kostenloser Download.

Unsere Architektur-Spicker beleuchten die konzeptionelle Seite der Softwareentwicklung.

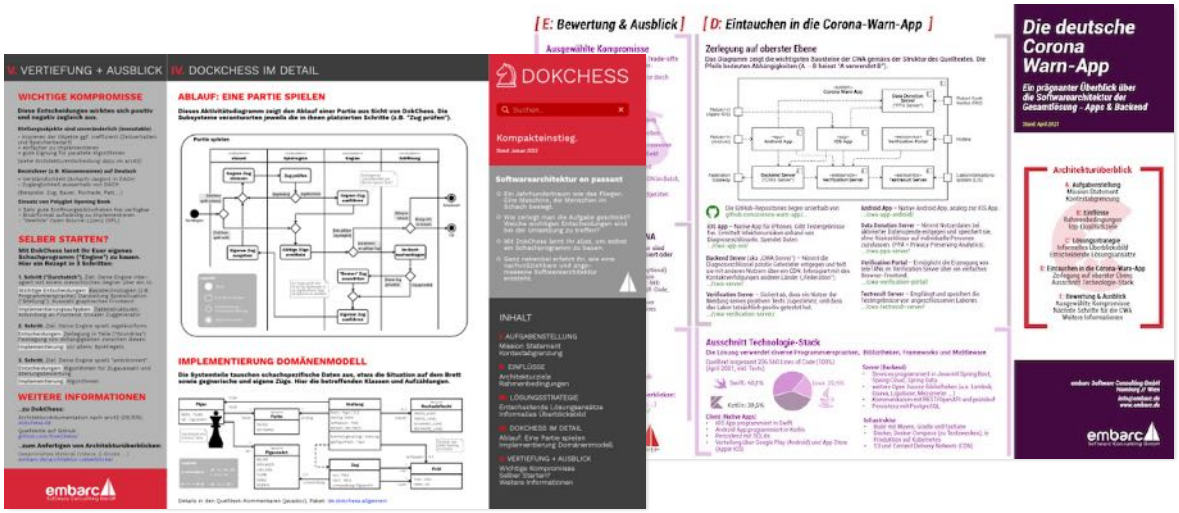
Spicker #1:
„Der Architekturüberblick“

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

➔ embarc.de/architektur-spicker/

129

Architekturüberblicke als Vorlage



➔ embarc.de/architektur-ueberblicke/

130

Vielen Dank.

Wir freuen uns auf Eure Fragen!

falk.sippach@embarc.de

@sippsack

ralf.d.mueller@deutschebahn.com

@RalfDMueller

131

Links/Referenzen (1)

- 1×1 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=QC7vjEZ5ZFcKviUWBbTk_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/asciidoctor-intro-and-deep-dive