Kontinuierlich und automatisiert dokumentieren

Docs-as-Code in der Praxis

FALK SIPPACH // EMBARC

OOP 2023, Online

Donnerstag, 09.02.2023

embarc Software Consulting GmbH

OOP DIGITAL software meets business

1

Kontinuierlich und automatisiert dokumentieren

Dokumentation wird häufig vernachlässigt. Mit dem Docs-as-Code-Ansatz wird die in Softwareprojekten relevante Dokumentation genau wie Quellcode behandelt, in der Versionsverwaltung abgelegt, mit leichtgewichtigen Entwicklerwerkzeugen (IDE/Texteditor, Build-Tools, CI/CD-Pipelines) bearbeitet und in die Softwareentwicklungsprozesse integriert. Die Inhalte können redundanzfrei verwaltet und manche Informationen aus Modellen oder dem Quellcode generiert werden. Durch die Verwendung leichtgewichtiger Text- und Grafikformate lassen sich die Ergebnisse einfach zielgruppenorientiert zusammenstellen. Die Verarbeitung erfolgt automatisiert über die schon vorhandenen Build-Prozesse.

Jedwede Art von Dokumentation gewinnt somit an Sichtbarkeit, durch die Eingliederung in die Entwicklungsprozesse und die damit verbundene kontinuierliche Weiterentwicklung steigt die Qualität und damit die Akzeptanz bei den Lesern. Dokumentation kann sogar ausgeführt werden, um zum Beispiel eingebettete Architekturregeln regelmäßig automatisiert zu testen. Die Zuhörer erfahren in diesem Vortrag an konkreten Beispielen, wie sie mit Documentation as Code starten können, welche typischen Fallstricke sie umschiffen und mit welchen konkreten Tools sie am besten arbeiten sollten.

2

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

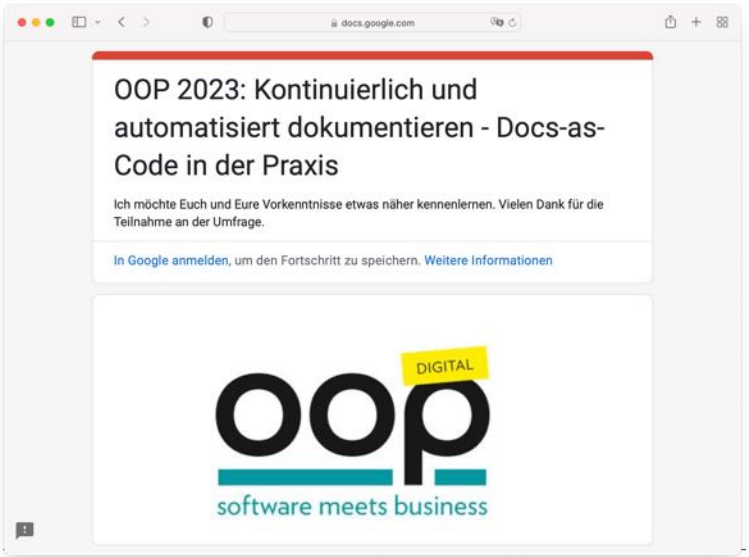
@sippsack

➔ xing.to/fsi

embarc Software Consulting GmbH

3

Wer seid Ihr?



6

sw_architecture_handson
@HandsonSw

Antwort an @HandsonSw @grafandreas und 3 weitere Personen

I think this "docs-as-code", "architecture-as-code" thing is currently at the beginning. People just start to find out how much sense it makes to embed the architecture documents into your git repo in a branch/mergable way. Sure we will see some great solutions in the future.

Tweet übersetzen

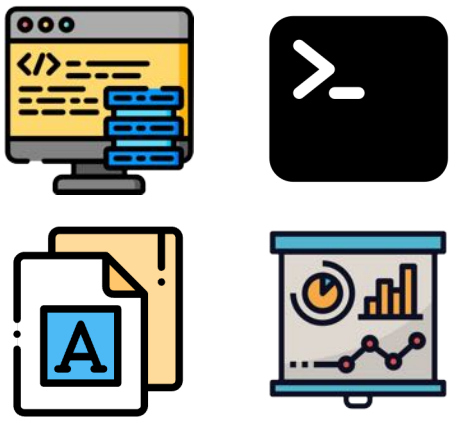
11:56 vorm. · 2. Okt. 2021 · Twitter Web App

4 Retweets **1** Tweet zitieren **24** „Gefällt mir"-Angaben

7

Inhalte

Docs-as-Code Reifegrad-Modell



Werkzeuge

8

Agenda

1 Einstieg und Motivation
2 Docs-as-Code Maturity Level
3 Werkzeuge
4 Fazit und Ausblick

9

Agenda

1 **Einstieg und Motivation**
2 Docs-as-Code Maturity Level
3 Werkzeuge
4 Fazit und Ausblick

1

11

7 Regeln für gute Dokumentation

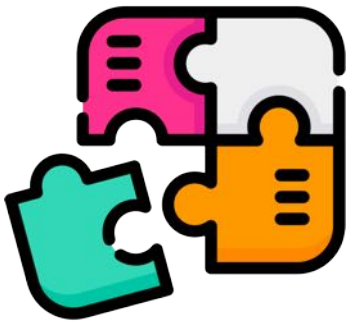
„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
   3. a) Erkläre Deine Notation
4. Verwende eine Standardstrukturierung
5. Halte Begründungen für Entscheidungen fest
6. Halte die Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

12

Architekturentwurf

Anforderungen/
Architekturziele



Lösungsansätze/
Entscheidungen

13

Zutaten für einen Architekturüberblick


Qualitätsziele


Vorgaben

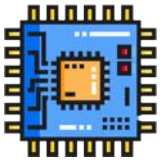

Systemkontext


Metapher
“Produktkarton“

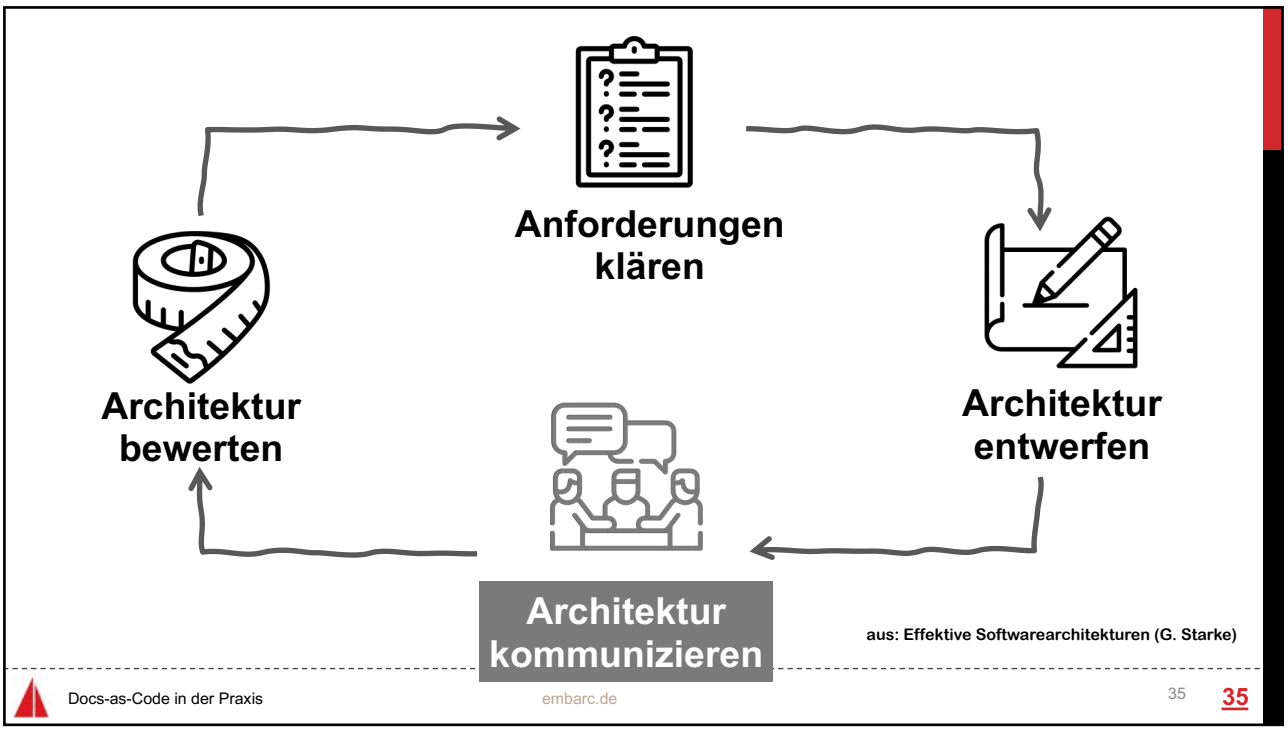

Informeller
Überblick


Architektur-Stil/-
Muster/…

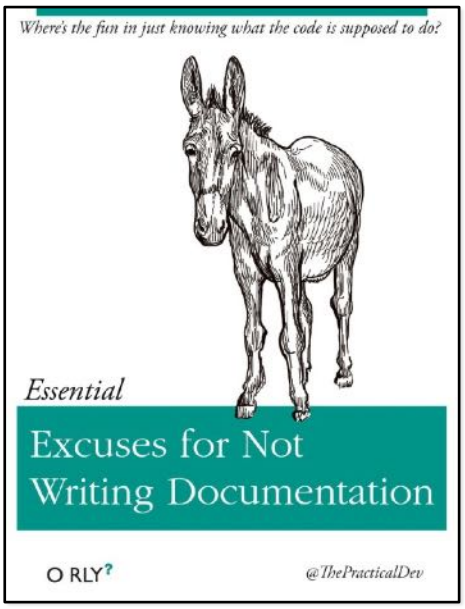

Entscheidungen


Technologien

15



35



Grafik von The Practical Dev (CC0 Public Domain Lizenz)

Entwurfs-unterstützung

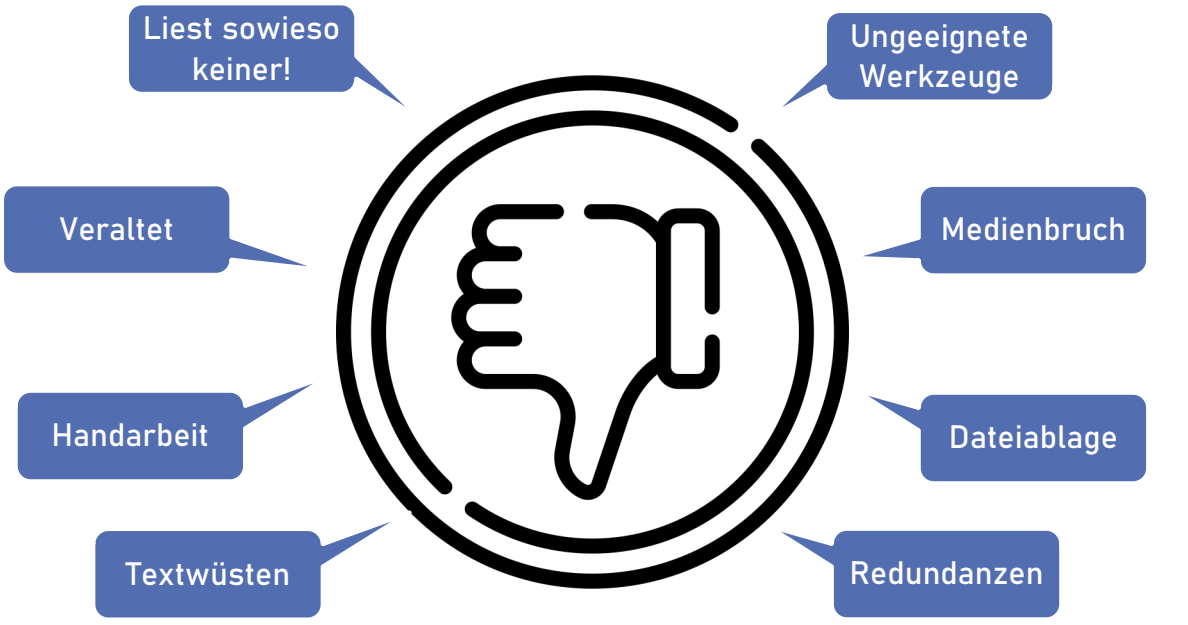
Frage nach dem Warum

Bewertbarkeit

Kommunikation

Neue Mitarbeiter

36



37

Agenda

1 Einstieg und Motivation

2 **Docs-as-Code Maturity Level**

3 Werkzeuge

4 Fazit und Ausblick

2

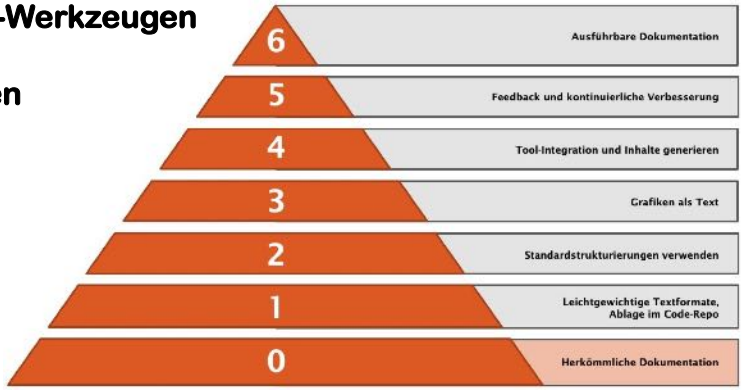
38

Docs-as-Code Reifegradmodell (Maturity Level)

39

Herkömmliche Dokumentation

- **Dokumentation wird getrennt vom Code verwaltet**
- **Bearbeitung in WYSIWYG-Werkzeugen**
- **Ablage auf Netzlaufwerken**
- **Verwendung von Wikis**



40

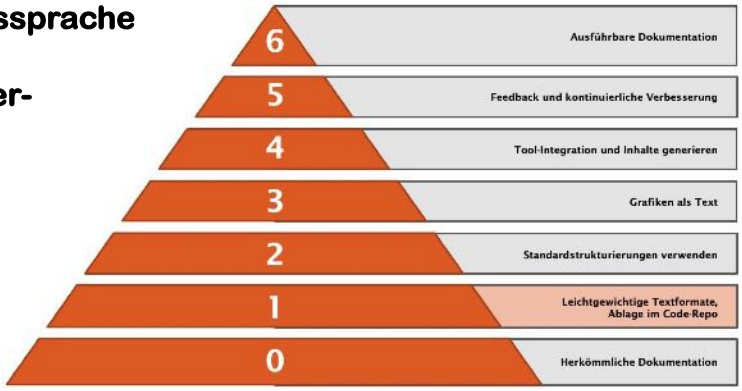
… der Ort, wo die Dokumente zum Sterben hingehen

Wiki = Where Information Kills Itselfs

41

Leichtgewichtige Textformate, Ablage im Repo

- **Dokumentation ist gemeinsam mit dem Quelltext versionierbar**
- **Wahl einer Auszeichnungssprache**
- **Bearbeitung mit Entwickler-Werkzeugen**



42

43

Probleme mit Word und Co.

Binärformat

Mehrbenutzer schwierig

Dokument per Mail verschicken und Änderungen wieder zusammenbringen

Historisierung, Versionen vergleichen

Word ist seitenorientiert (Seitenzahl, Header, Footer)

Subdokumente und Bild-Referenzen (möglich, aber wird kaum verwendet)

Unser täglich Entwickler-Brot

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

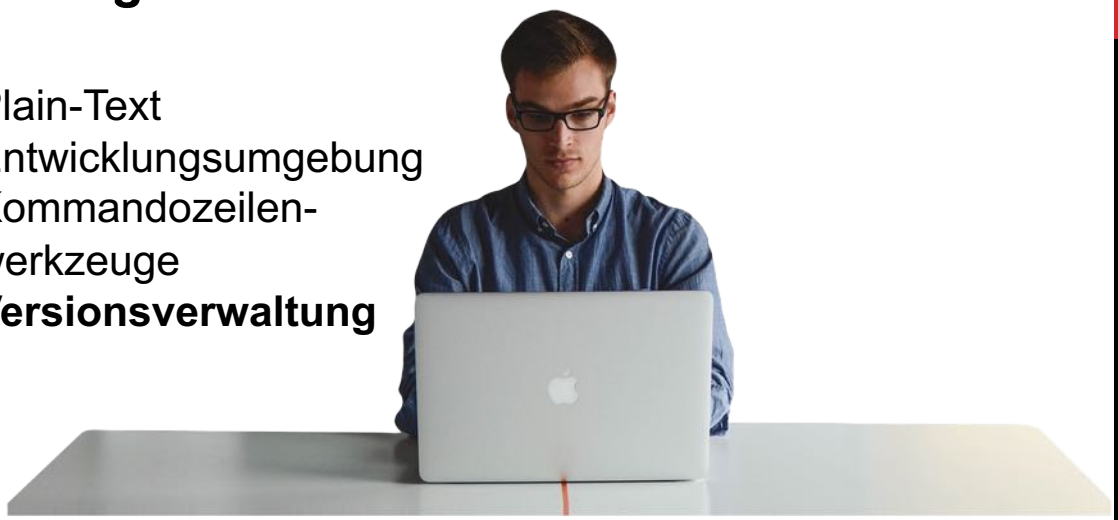
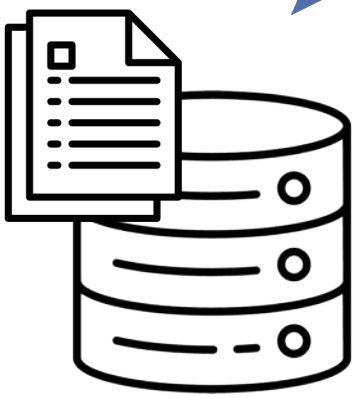


Foto von geralt: https://pixabay.com/de/unternehmer-start-start-up-karriere-696976/ (CC0 Public Domain Lizenz)

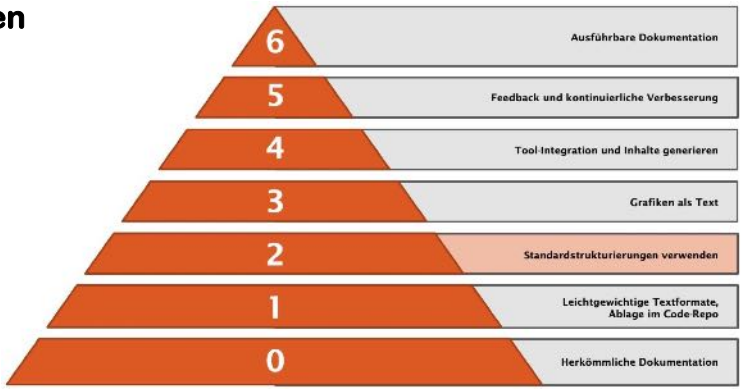
Documentation-as-Code



46

Standardstrukturierungen verwenden

- **Leser finden sich leichter zurecht**
- **Schreiber erhalten Orientierung bei der Erstellung von Inhalten**



47

7 Regeln für gute Dokumentation

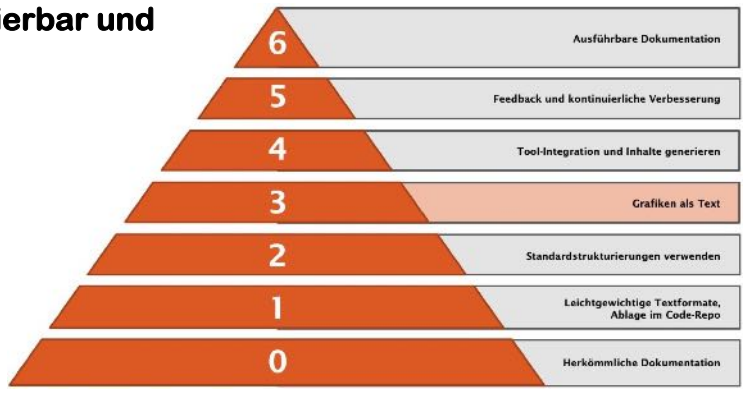
„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
   3. a) Erkläre Deine Notation
4. **Verwende eine Standardstrukturierung**
5. Halte Begründungen für Entscheidungen fest
6. Halte die Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

48

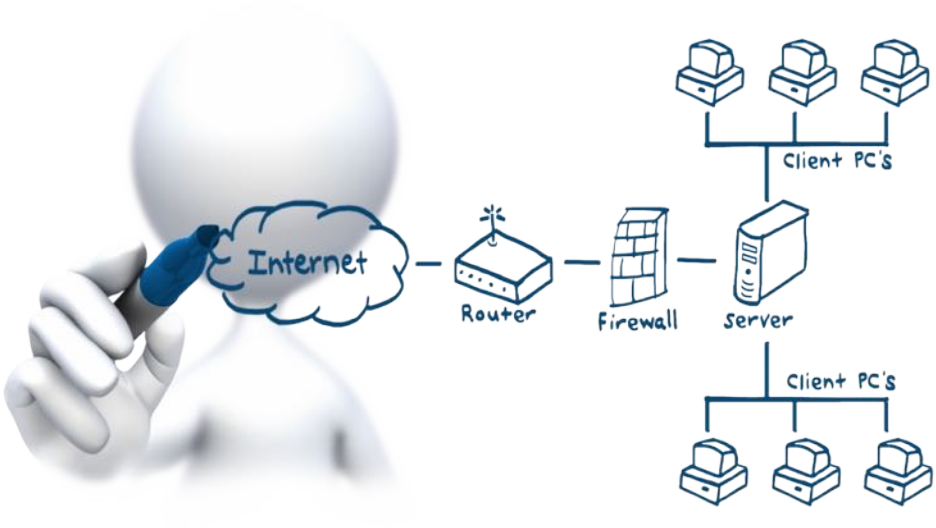
Grafiken als Text

- **Mindert Risiko des Vendor-Lock-In bei Diagrammen**
- **macht Graphiken versionierbar und diffbar**



49

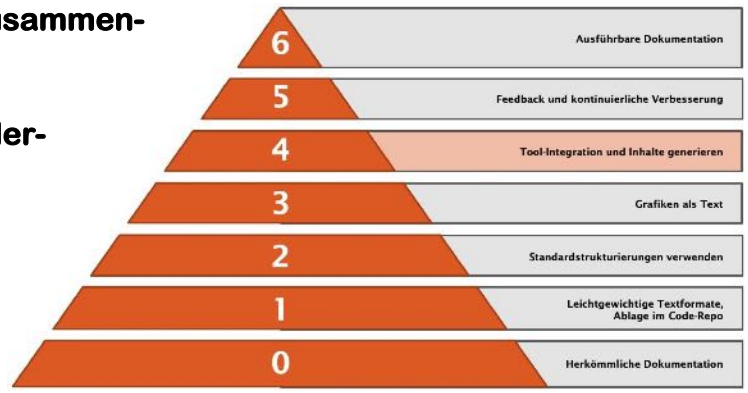
Diagramme



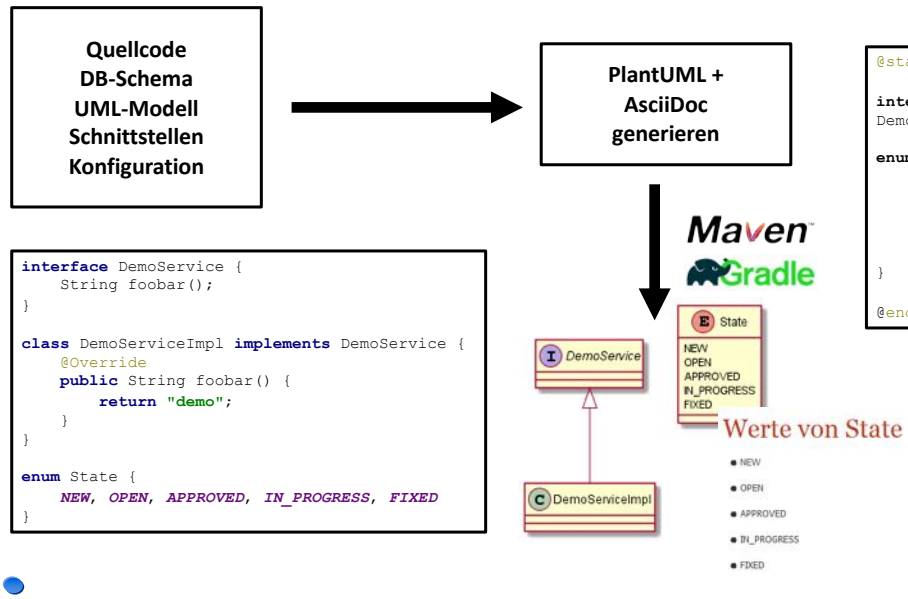
50

Tool-Integration und Inhalte generieren

- **automatisierte Erstellung im Build-Prozess**
- **zielgruppenorientierte Zusammen-stellung der Ergebnisse**
- **Verwendung der Entwickler-werkzeuge ohne Kontext-wechsel**
- **einfache Im- und Exporte**



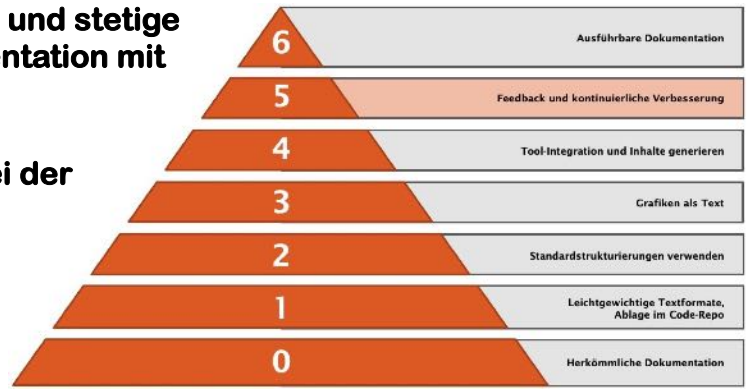
51



52

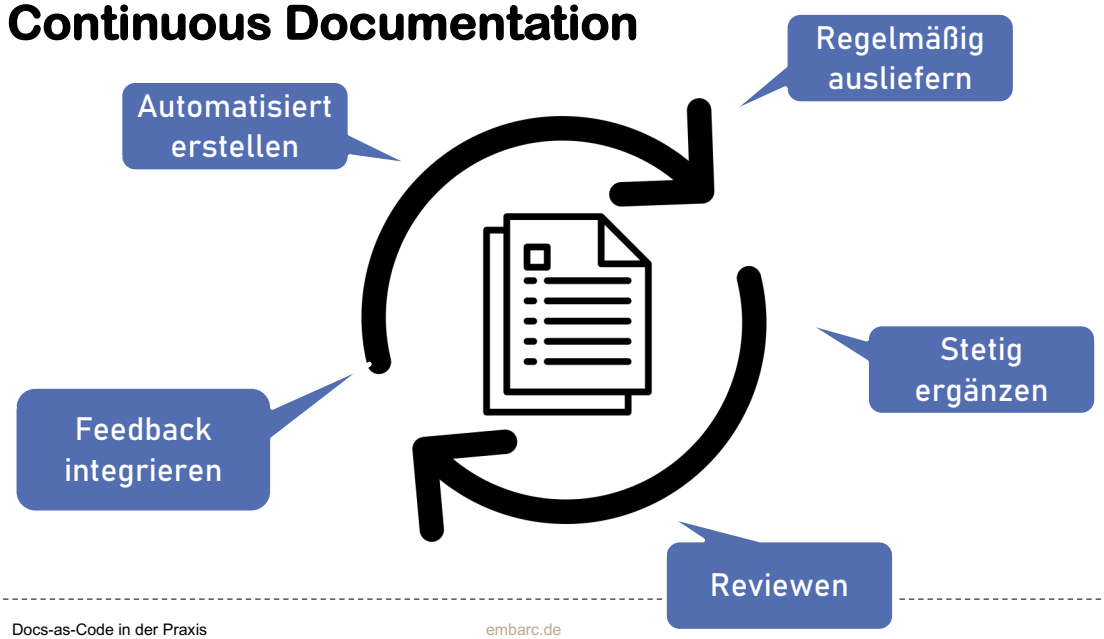
Feedback und kontinuierliche Verbesserung

- **Einbinden in das organisatorische Vorgehen**
- **automatisierte Erstellung und stetige Auslieferung der Dokumentation mit den Releases**
- **Review-Prozesse auch bei der Dokumentation**
- **Dokumentation leicht up-to-date halten**



53

Continuous Documentation



54

Ausführbare Dokumentation

- **Strukturvorgaben dokumentieren und automatisiert testen**
- **jQAssistant, ArchUnit, xMolecules**

61

Agenda

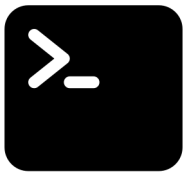
1 Einstieg und Motivation
2 Docs-as-Code Maturity Level
3 **Werkzeuge**
4 Fazit und Ausblick

3

62

Werkzeuge

IDEs, Texteditoren



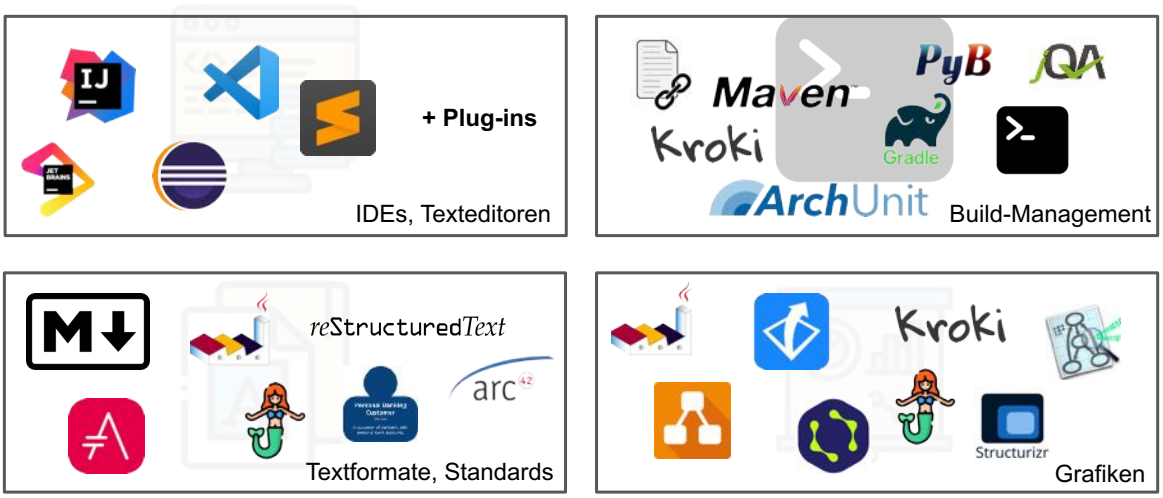
Build-Management

Textformate, Standards

Grafiken

64

Werkzeuge



65

Werkzeuge

IDEs, Texteditoren

Build-Management

Textformate, Standards

Grafiken

67

Docs-as-Code in der IDE

68

Docs-as-Code in der IDE

69

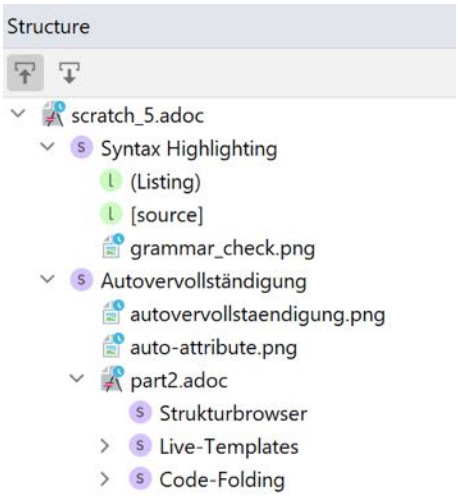
Docs-as-Code in der IDE

70

Docs-as-Code in der IDE

71

Docs-as-Code in der IDE



72

Docs-as-Code in der IDE

```
7   == Code-Folding
8
9   === Abschnitt ausgeklappt
10
11  Lorem Ipsum
12
13  === Abschnitt eingeklappt ...
```


73

Docs-as-Code in der IDE

```
3   == Live-Templates
4
5   ad-t
6   ad-table2                                  Table with two columns
7   ad-tag-include      AsciiDoc Tags to be used with include macro
8   ad-title1                                                Title 1
9   ad-title2                                                Title 2
10  ad-title3                                                Title 3
11  ad-title4                                                Title 4
12  ad-title5                                                Title 5
13  ad-title6                                                Title 6
14  ad-config-toc     config-attributes for table of contents and s…
15  Press Eingabe to insert, Tabulator to replace  Next Tip
```


74

Code einbetten

```
:source-highlighter: pygments

[source,java]
----
include::MyTest.java[]
----



[source,java,indent=10]
----
include::MyTest.java[lines=2..4]
----
<1> method declaration
<2> print statement



:sourcedir: ../../../src/main/java

[source,java]
----
include::{sourcedir}/de/oio/vavr/functions/CheckedFunction.java[lines=12..14]
----
```

```java
public class MyTest {
    public static void main(String[] args) { (1)
        System.out.println("Hello world!"); (2)
    }
}
```

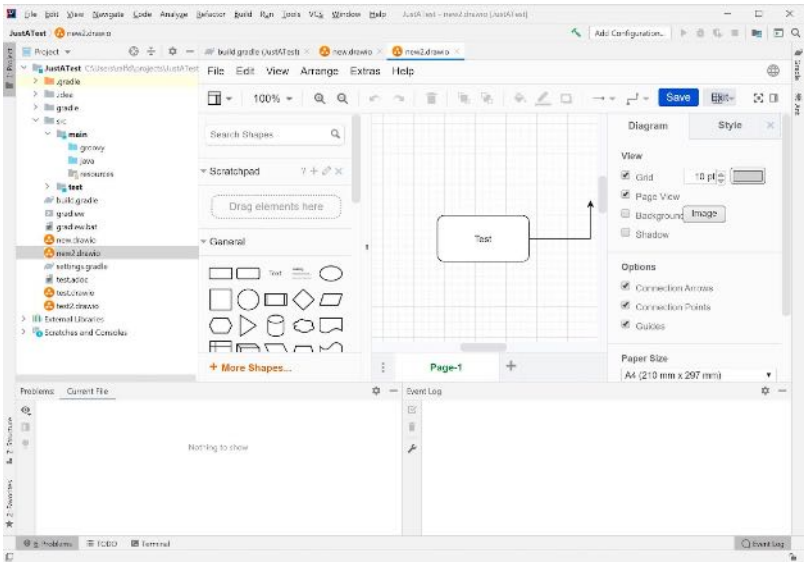
```java
        public static void main(String[] args) { (1)
            System.out.println("Hello world!"); (2)
        }
```

1. method declaration
2. print statement

```java
        CheckedFunction1<Integer, Integer> readFunction = i -> readFromFile(i);
        integers.stream().map(readFunction.unchecked());
```


75

Erweiterung über Plugins, z. B. draw.io



76

Werkzeuge

IDEs, Texteditoren

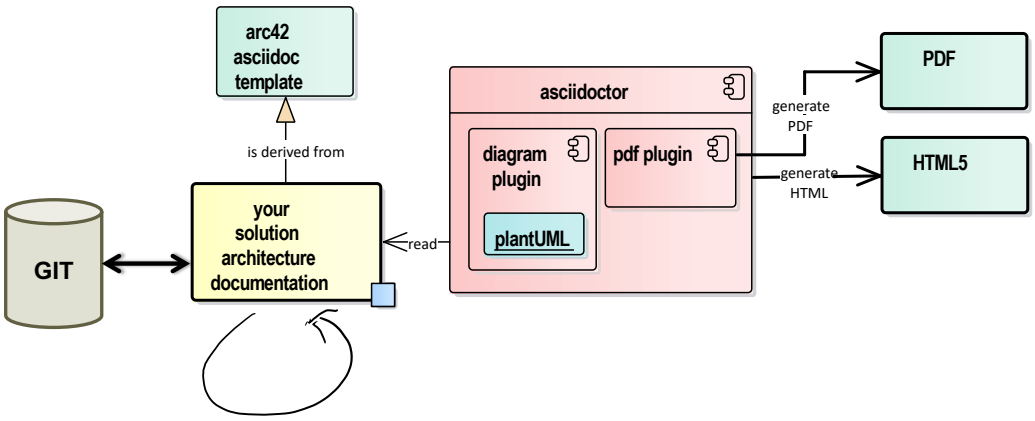
Build-Management

Textformate, Standards

Grafiken

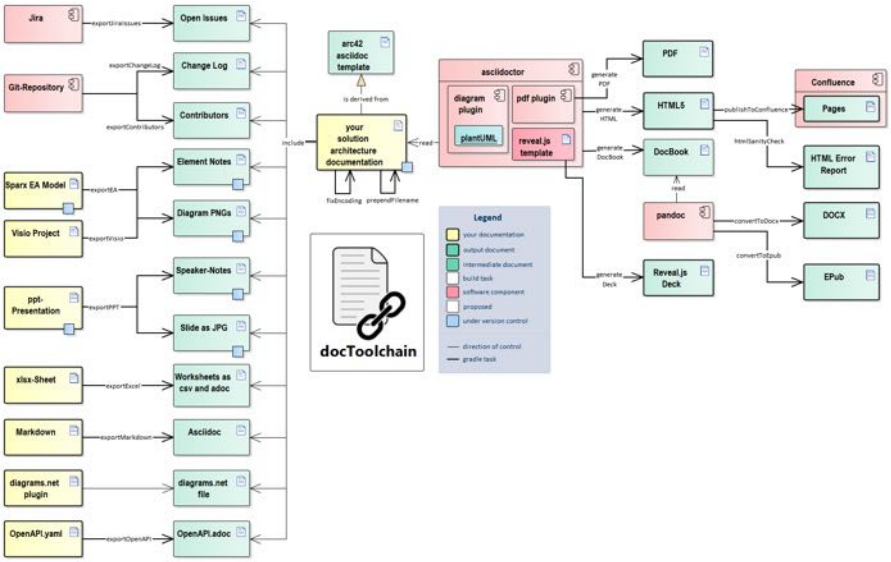
77

Einfacher Workflow mit AsciiDoctor



78

Werkzeugkette: docToolchain



79

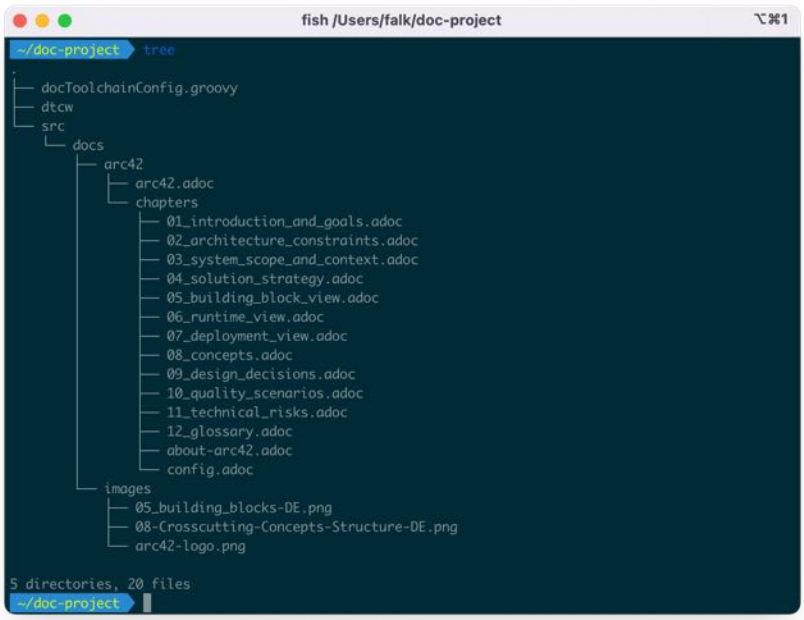
docToolchain @docToolchain · 26. Sep.

A full blown docu toolchain with @arc42Tipps as template installed in just one tweet:

```
wget doctoolchain.github.io/dtcw
chmod +x dtcw
./dtcw downloadTemplate
./dtcw generateSite
```

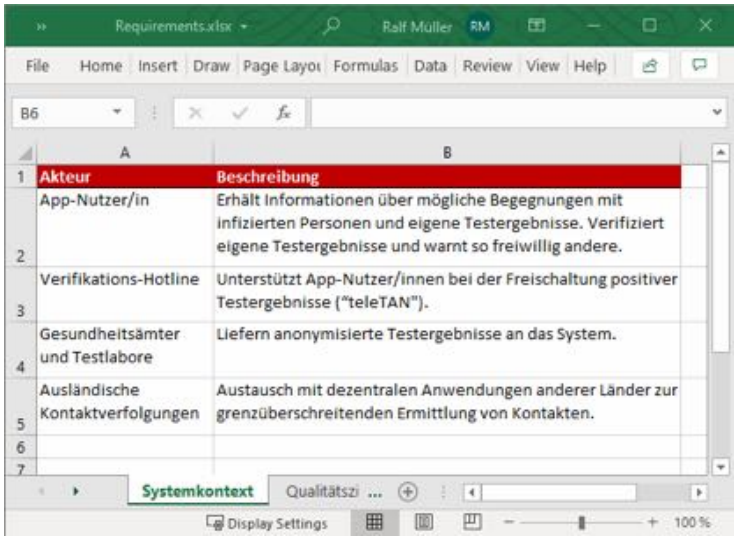
=> doctoolchain.org

80



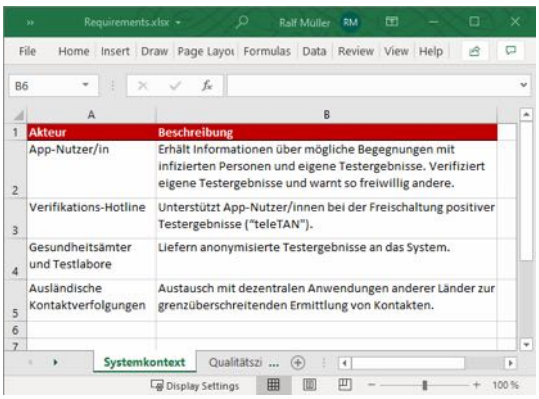
81

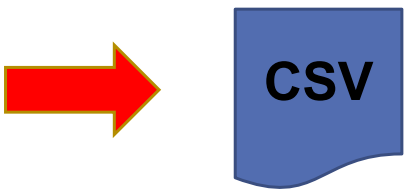
Tabellen!?



82

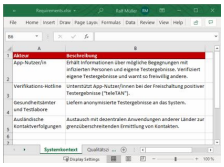
Tabellen!?





83

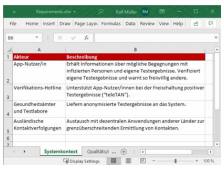
Tabellen!?



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.

84

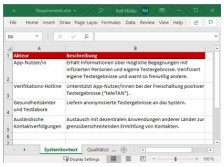
Tabellen!?



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.

85

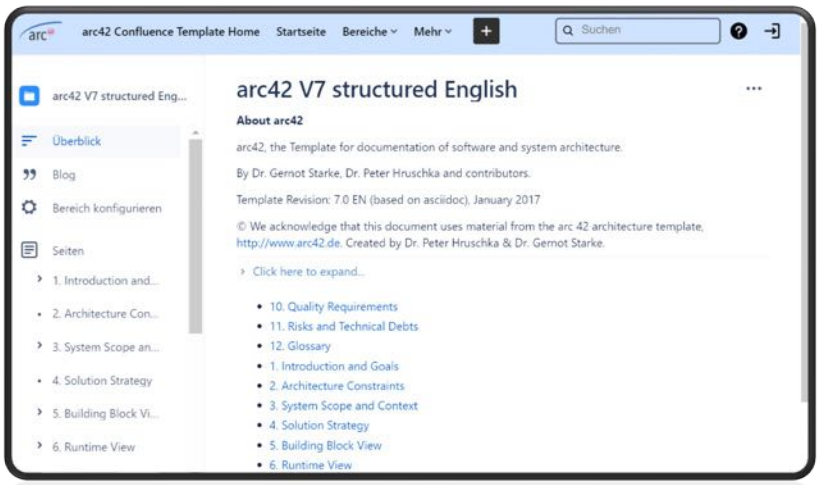
Tabellen!?



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.

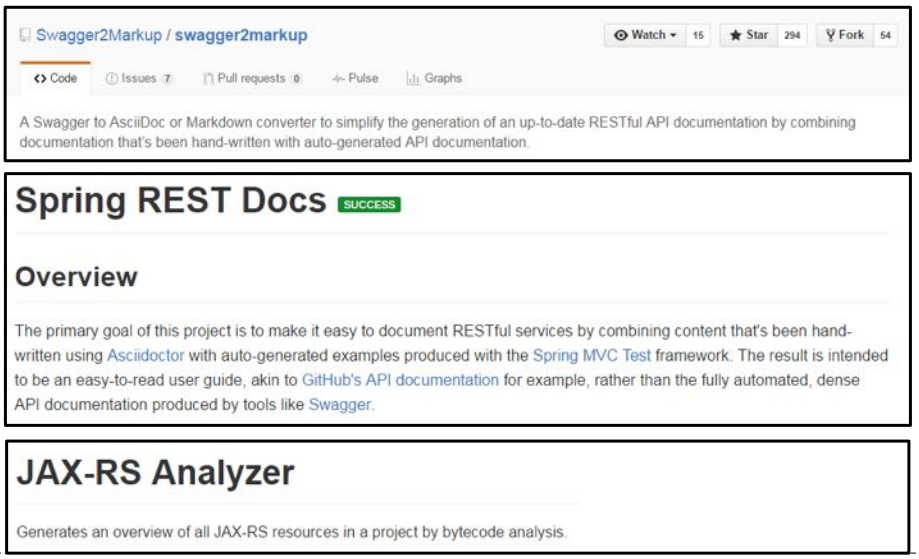
86

publishToConfluence



87

Schnittstellenbeschreibung generieren



88

Werkzeuge

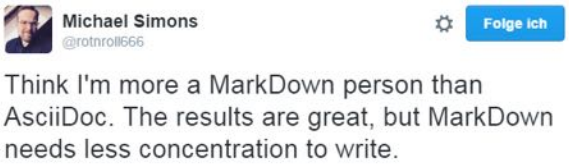
IDEs, Texteditoren

Build-Management

Textformate, Standards

Grafiken

89



90

Markdown

Markdown

- Toller Standard für einfache Auszeichnungen
- Features
  - Span Elements
    - Links
    - Emphasis
    - Code
    - Images
  - Block Elements
    - Paragraphs and Line Breaks
    - Headers
    - Blockquotes
    - Lists
    - Code Blocks
    - Horizontal Rules

- TOC
- Tables (Feature Rich)
- Includes (Level-Offset)
- PlantUML
- Admonitions
- Attributes
- Anchors
- Footnotes, Index, Glossary
- Videos
- Syntax Highlighting
- Callouts
- Math Rendering
- Outputformats

91

Markdown

Wir brauchen eine Erweiterung!

Welche wählen wir?

- CommonMark
- CriticMarkup
- Discount
- DocFX
- ExtraMark
- Ghost's Markdown/Haunted Markdown
- GitHub Flavored Markdown
- GitLab Flavored Markdown (with login)
- Haroopad Flavored Markdown
- iA Writer's Markdown
- Kramdown
- Leanpub Flavored Markdown
- Litedown
- Lunamark
- Madoko
- Markdown
- Markdown 2
- Markdown Extra
- Markdown-it
- Markua
- Maruku
- MultiMarkdown
- Pandoc's Markdown
- PHP Markdown Extra Extended
- Python Markdown
- R Markdown
- Redcarpet
- Remarkable
- Rhythmus
- Scholarly Markdown
- Showdown
- StackOverflow's Markdown
- Taiga Markdown
- Trello's Markdown
- vfmd
- Xcode/Swift Playgrounds Markup

https://github.com/commonmark/commonmark-spec/wiki/markdown-flavors

92

Markdown

einen Dialekt

Wir brauchen eine ~~Erweiterung!~~

Welche wählen wir?

Funktioniert dann unsere Toolchain noch?

Gibt es einen Editor-Preview?

93

AsciiDoc / Asciidoctor

leistungsfähige Syntax für technische Dokumentation

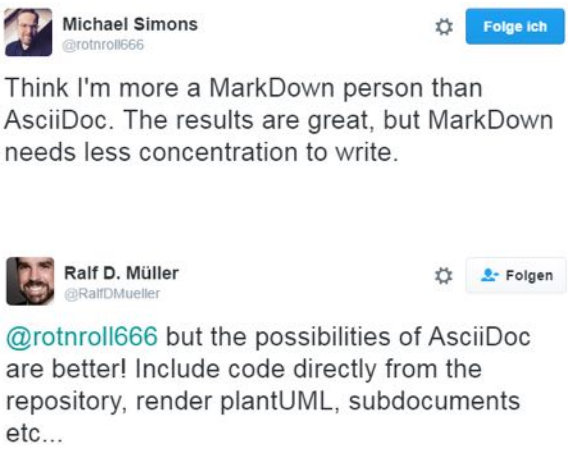
in Ruby geschrieben

mit Opal nach JavaScript transpiliert

mit jRuby auf der JVM gewrapped

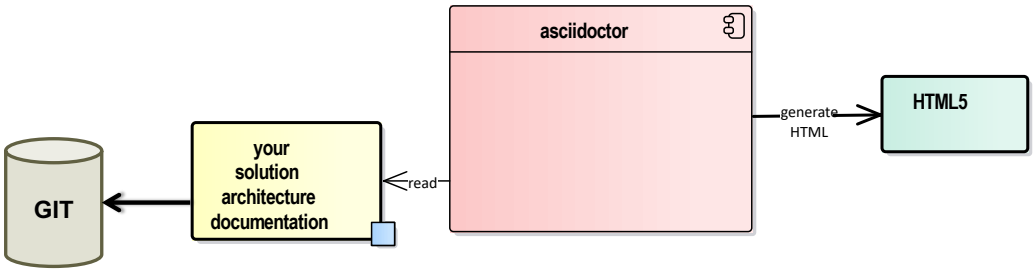
=> Keine Dialekte!

95



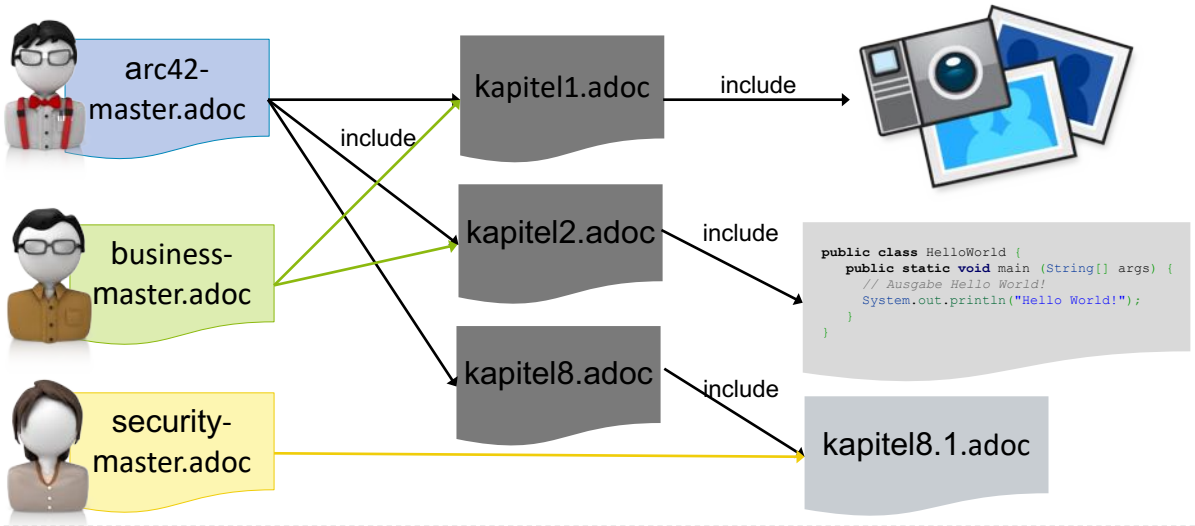
96

Basic Docs-as-Code with AsciiDoc



97

Modulare Dokumentation



98

Out-of-the-Box Features

„ablenkungsfrei“ – Dokumentation wie E-Mails schreiben

Gliederung in Unterdokumente

Neugliederung je nach Stakeholder

Bilder werden referenziert, nicht eingebettet

leichte Versionierung „handle Docs-as-Code“

Formatierung von Source-Code

Reviews, Pull-Requests, Versionierung durch Git

Konvertierung nach HTML5 und DocBook

7 Regeln für gute Dokumentation

„Documenting Software Architectures: Views and Beyond“
Clements, et.al, 2. Auflage 2010

1. Schreibe aus Sicht des Lesers
2. Vermeide unnötige Wiederholungen
3. Vermeide Mehrdeutigkeiten
   3. a) Erkläre Deine Notation
4. **Verwende eine Standardstrukturierung**
5. Halte Begründungen für Entscheidungen fest
6. Halte die Dokumentation aktuell, aber auch nicht zu aktuell
7. Überprüfe Dokumentation auf ihre Gebrauchstauglichkeit

„Abheften“ 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

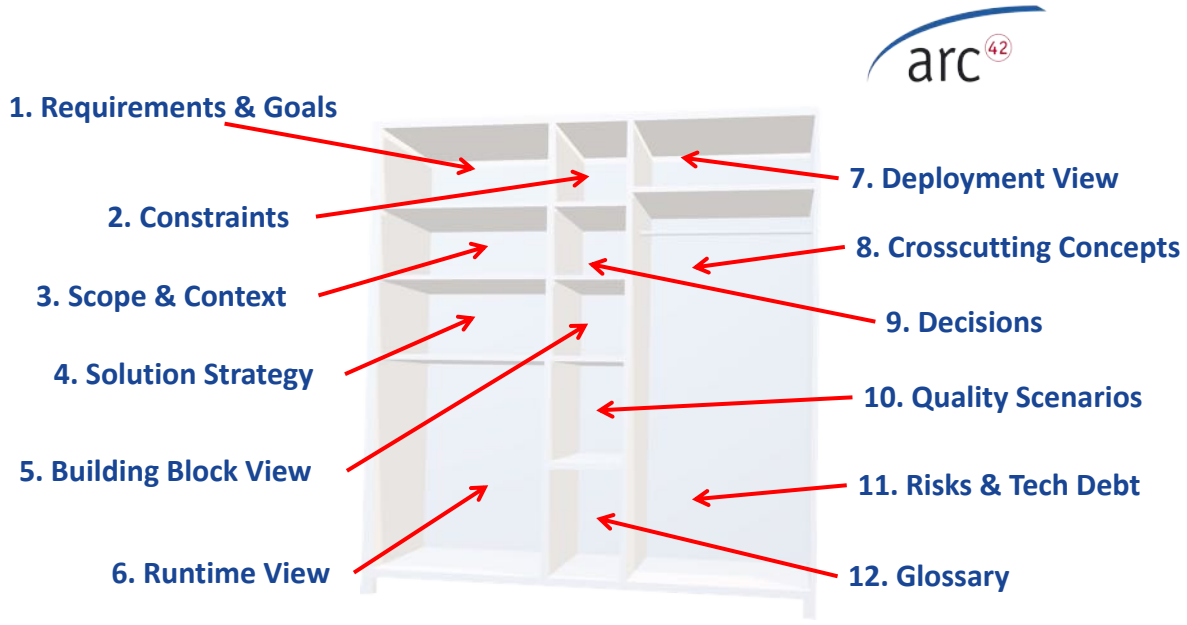
Dr. Peter Hruschka
http://www.peterhruschka.eu/

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

arc42

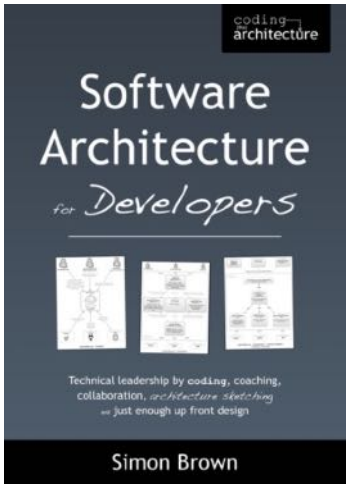
➔https://arc42.de/

101



102

Ein anderer Blick: Simon Brown



Simon Brown:
Software Architecture for Developers

Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design

➔ http://leanpub.com/ Leanpub

103

Ein anderer Blick: Simon Brown

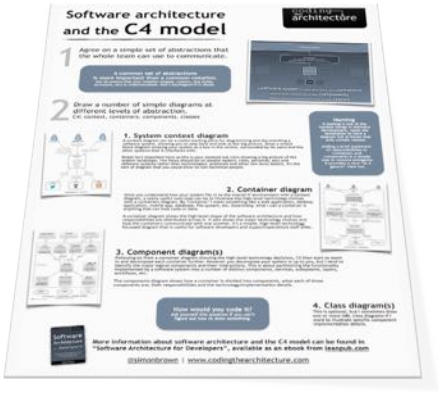
Simon Brown:
Software Architecture for Developers, Volume 2

Visualize, document and explore your software architecture

➔ http://leanpub.com/ Leanpub

104

Simon Brown: C4 Model



- 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

105

Werkzeuge

IDEs, Texteditoren

Build-Management

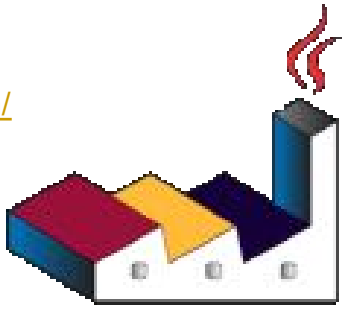
Textformate, Standards

Grafiken

106

Diagramme

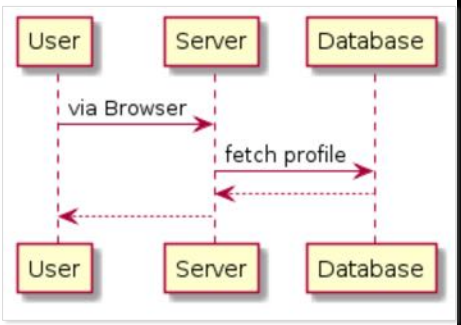
- http://plantuml.com/
- http://asciidoctor.org/docs/asciidoctor-diagram/
- Komplexe Diagramme als einfachen Text verwalten



107

PlantUML

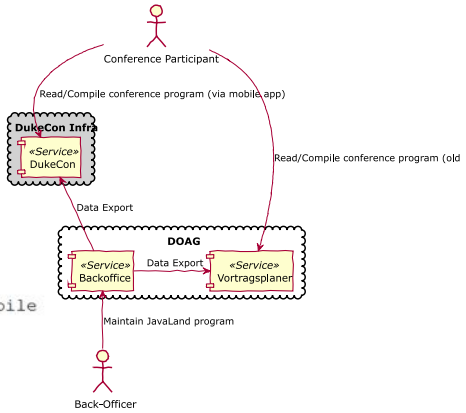
```
1  @startuml
2
3  User ->  Server: via Browser
4    Server ->  Database: fetch profile
5    Server <-- Database
6  User <-- Server
7
8  @enduml
```



108

Diagramme als textuelle Beschreibung

```
@startuml
skinparam handwritten true

actor :Conference Participant:

cloud "DukeCon Infra" #lightgray {
    [DukeCon] <<Service>>
}

cloud "DOAG" {
    [Backoffice] <<Service>>
    [Vortragsplaner] <<Service>>
}

:Conference Participant: -right-> (DukeCon) : Read/Compile
conference program (via mobile app)
:Conference Participant: -right-> (Vortragsplaner) :
Read/Compile conference program (old style)

(Backoffice) -left-> (DukeCon) : Data Export
(Backoffice) -left->

actor :Back-Officer:

:Back-Officer: -left-
program
@enduml
```

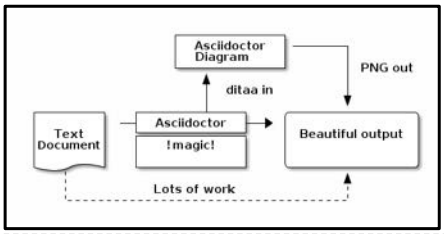



```
["plantuml", "dukecon-systemcontext", "svg"]
----------------------------------------------
include::../systemcontext.puml[]
----------------------------------------------
```

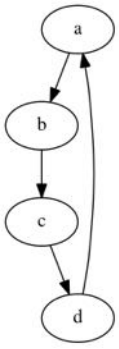

109

"AsciiArt"

```
[ditaa]
----
                +-------------+
                | Asciidoctor |-------+
                |  Diagram    |       |
                +-------------+       | PNG out
                    ^                 |
                    | ditaa in        |
                    |                 v
 +--------+   +--------+----+    /----------------\
 |        | --+ Asciidoctor +--> |                |
 |  Text  |   +-------------+    |Beautiful output|
 |Document|   |   !magic!   |    |                |
 |     {d}|   |             |    |                |
 +---+----+   +-------------+    \----------------/
     :                                   ^
     |          Lots of work             |
     +-----------------------------------+
----
```




```
[graphviz, dot-example, svg]
----
digraph g {
    a -> b
    b -> c
    c -> d
    d -> a
}
----
```



110

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!

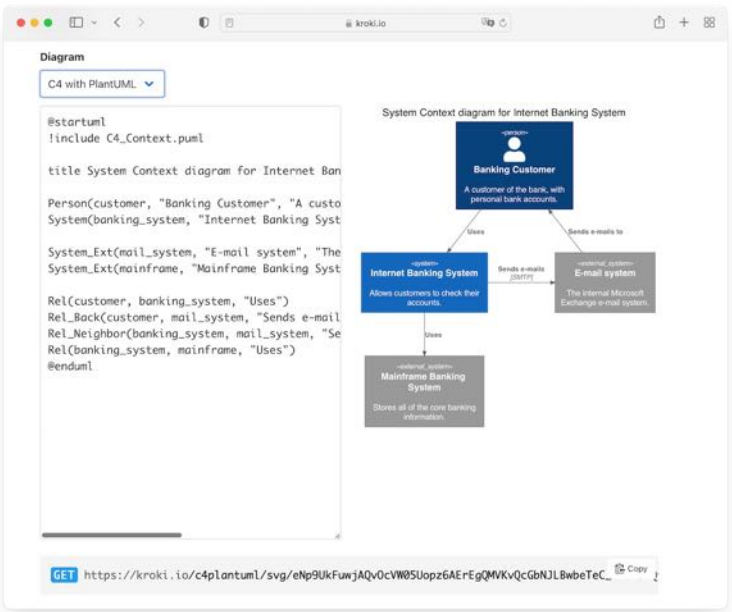
Simple
Kroki provides a unified API for all the diagram libraries. Learn once use diagrams anywhere!

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.

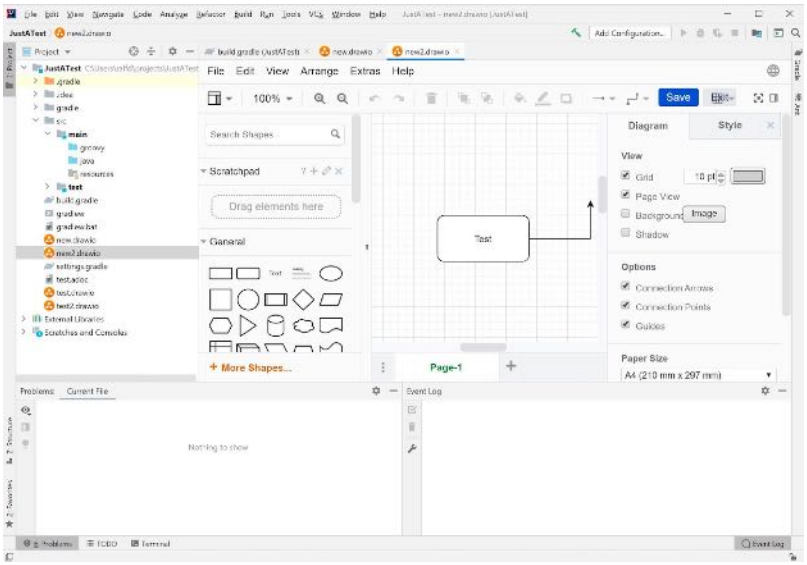
https://kroki.io/

111



112

Draw.io/Diagrams.net und Co.

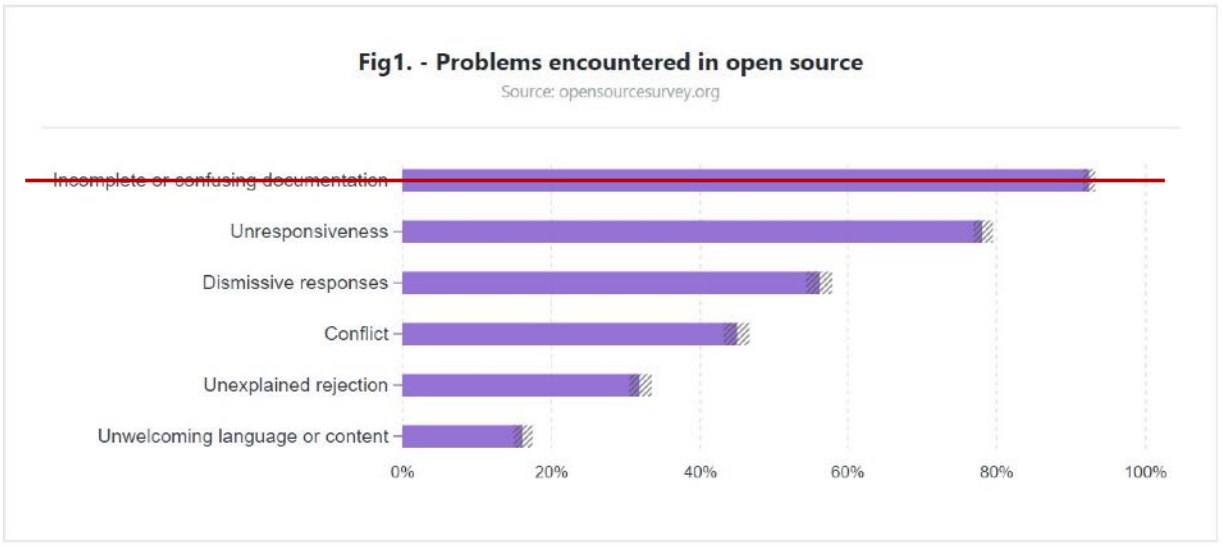


113

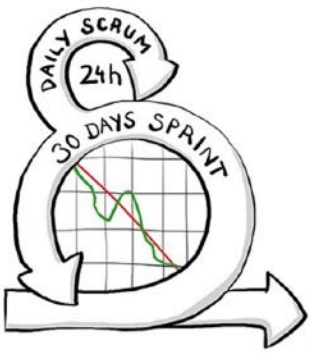
Agenda

4

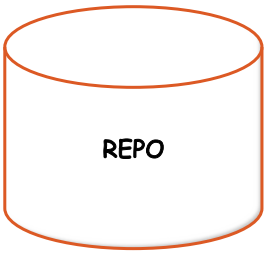
115



116



Continuous Documentation



Documentation as Code

117

Moderne (Architektur-)Dokumentation

Textformate

Single Source of Truth

Inhalte generieren

Bilder/Grafiken

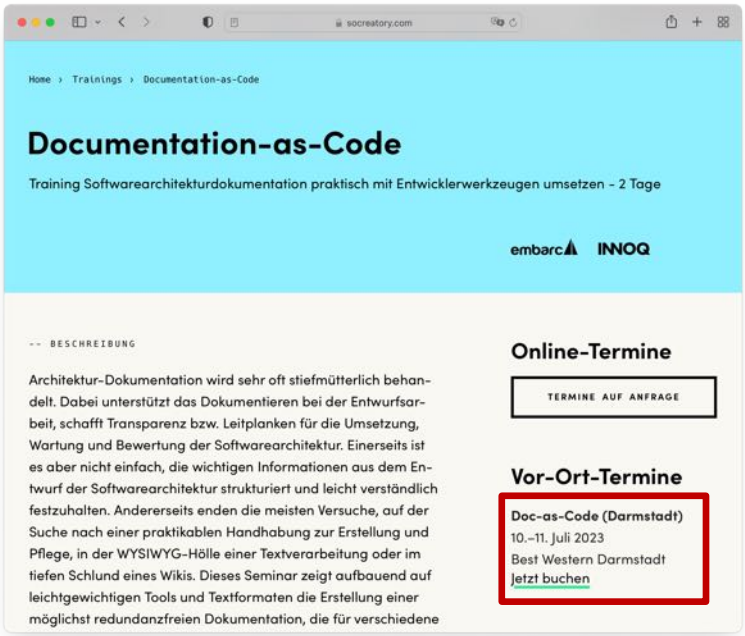
Docs-as-Code

Continuous Documentation

Ausführen/ Validieren

Kümmerer

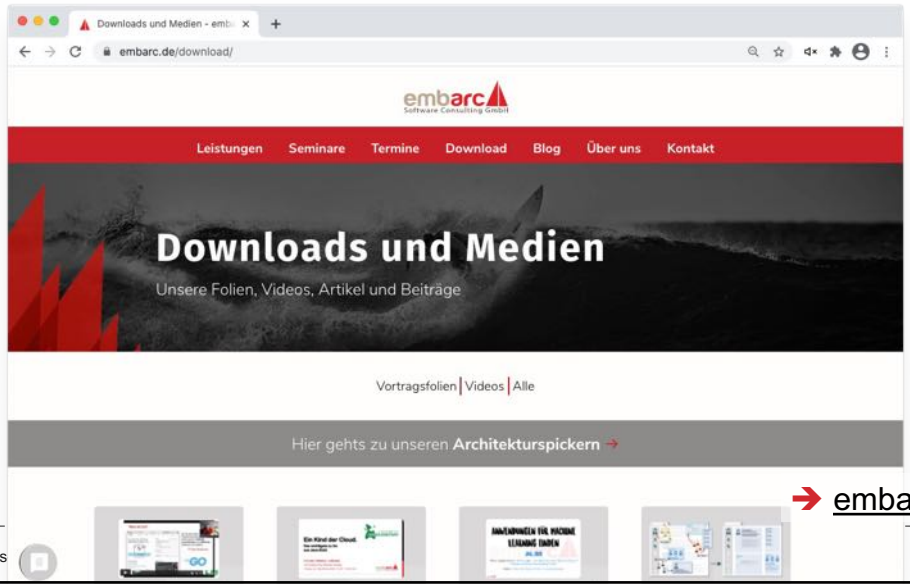
118



SOCREATORY

119

Folien von heute als PDF zum Download



➔ embarc.de/download/

120

Flyer zum CWA Architekturüberblick

Wir schicken Ihnen gerne ein gedrucktes Exemplar dieses Überblicks als Flyer Einfach eine E-Mail senden an

info@embarc.de

mit Betreff „CWA-Flyer“ und Eurer Postadresse im Text, dann geht es los …

https://www.embarc.de/architektur-ueberblicke/#cwa

121

Vielen Dank.

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi