Ein interaktiver Einstieg in Docs-as-Code

FALK SIPPACH

embarc Architektur-Punsch 2021, Online

Montag, 13.12.2021

1

Architekturen dokumentieren und kommunizieren

Im Gegensatz zu den klassischen Ansätzen (Textverarbeitung, Wiki, Netzlaufwerke, ...) verfolgt Docs-as-Code das Ziel, die in Softwareprojekten relevante Dokumentation genau wie den Quelltext zu behandeln. Für die entwickelnden Team-Mitglieder entsteht somit kein Medienbruch. Sie können die gleichen Werkzeuge (Texteditor, IDE, Build-Tools, CI/CD-Pipeline) verwenden, um die Inhalte als leichtgewichtige Textformate neben dem Sourcecode in der Versionsverwaltung abzulegen, die Änderungen gemeinsam mit dem Quellen zu reviewen, als Release zu markieren und auszuliefern sowie die zielgruppenorientierte Zusammenstellung der Ergebnisse in den Build-Prozess einzubinden. Um Redundanzen zu vermeiden, können aus den vorhandenen Projektmodellen (Sourcecode, DB-, UML-Modell, ...) automatisiert textuelle Inhalte und Diagramme generiert werden.

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. In dieser interaktiven Session erfahrt ihr, wie ihr mit Documentation as Code starten könnt, welche typischen Fallstricke es zu umschiffen gilt und welche konkreten Tools man kenne sollte.

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

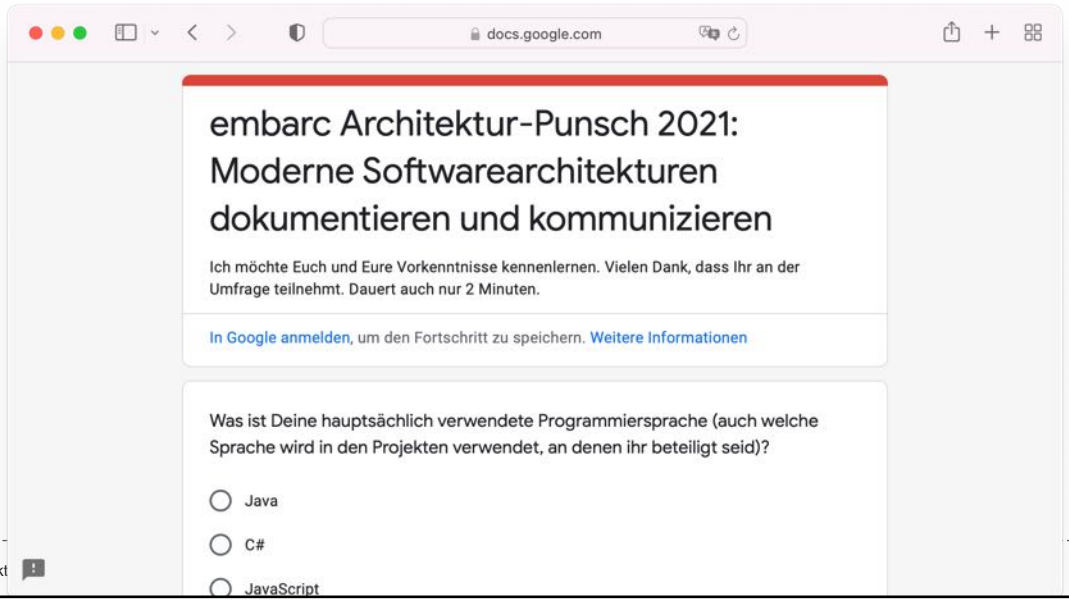
@sippsack

➔ 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

Agenda

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

8

Agenda

1 **Einstieg und Motivation**

2 Architektur dokumentieren

3 Docs-as-Code Maturity Level

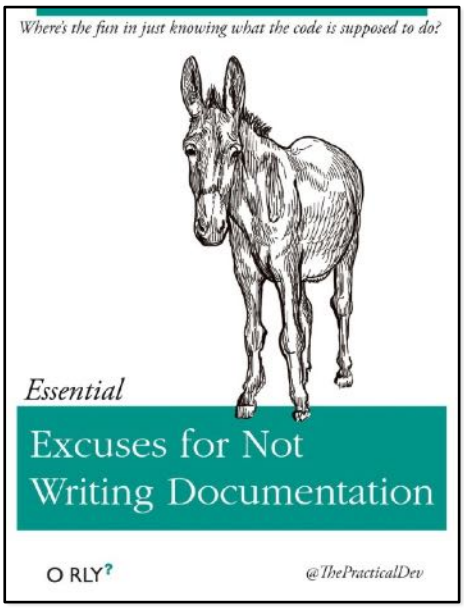
4 Fazit und Ausblick

1

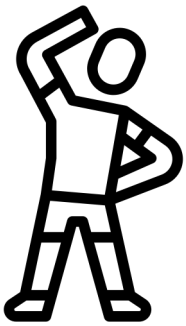
9

aus: Effektive Softwarearchitekturen (G. Starke)

10



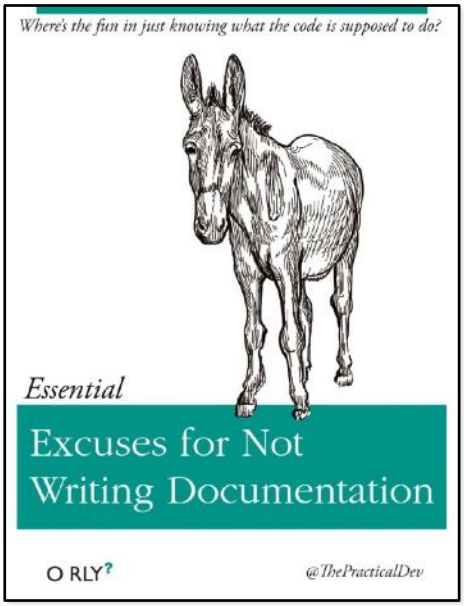
Grafik von The Practical Dev (CC0 Public Domain Lizenz)



Bitte folgt dem Link für eine kleine Übung

https://tinyurl.com/arcpunsch-docs-as-code

11



Grafik von The Practical Dev (CC0 Public Domain Lizenz)

Entwurfs-unterstützung

Frage nach dem Warum

Bewertbarkeit

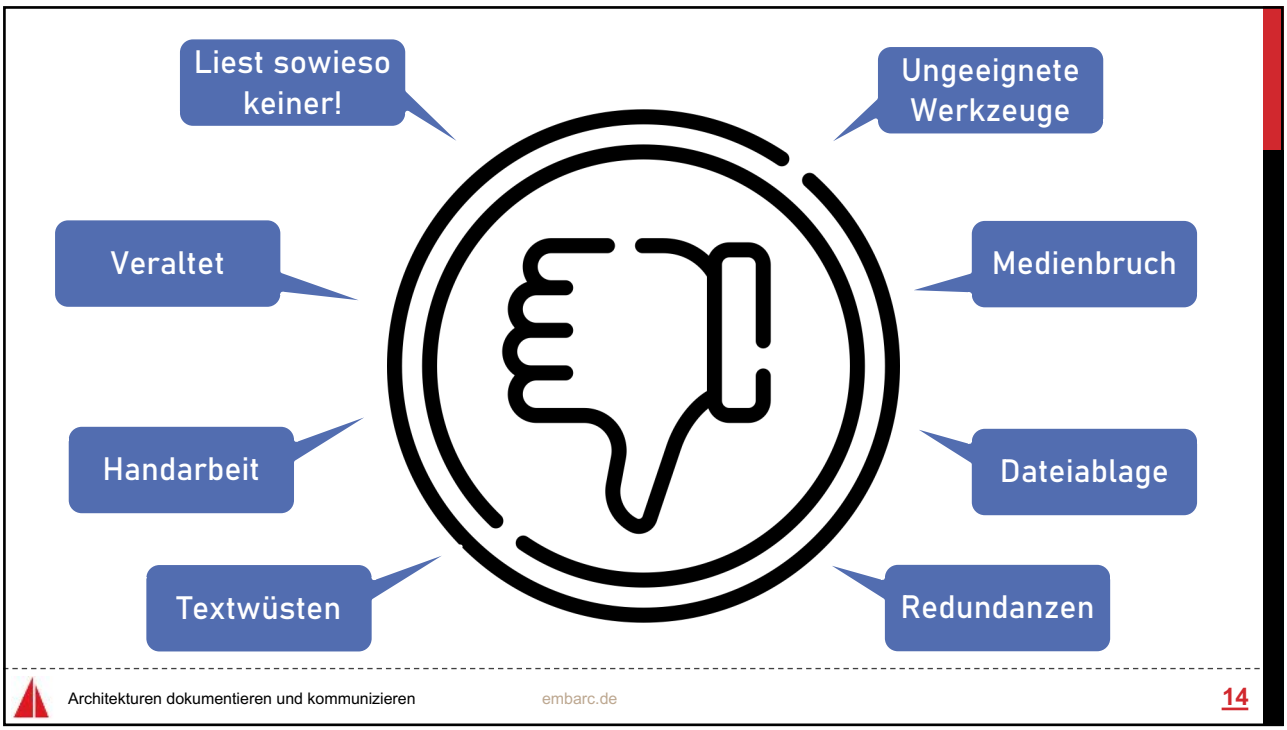
Kommunikation

Neue Mitarbeiter

12


Hauptsache, du machst es nicht mit Word!
Ich mach's mit Latex.
Kondome schützen.
Für eine Nacht oder mehr, mach's mit.
GIB AIDS KEINE CHANCE
STI
Bitte folgt dem Link für eine kleine Übung
https://tinyurl.com/arcpunsch-docs-as-code
Architekturen dokumentieren und kommunizieren
embarc.de
13

13


Liest sowieso keiner!
Ungeeignete Werkzeuge
Veraltet
Medienbruch
Handarbeit
Dateiablage
Textwüsten
Redundanzen
Architekturen dokumentieren und kommunizieren
embarc.de
14

14

Unser täglich Entwickler-Brot

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

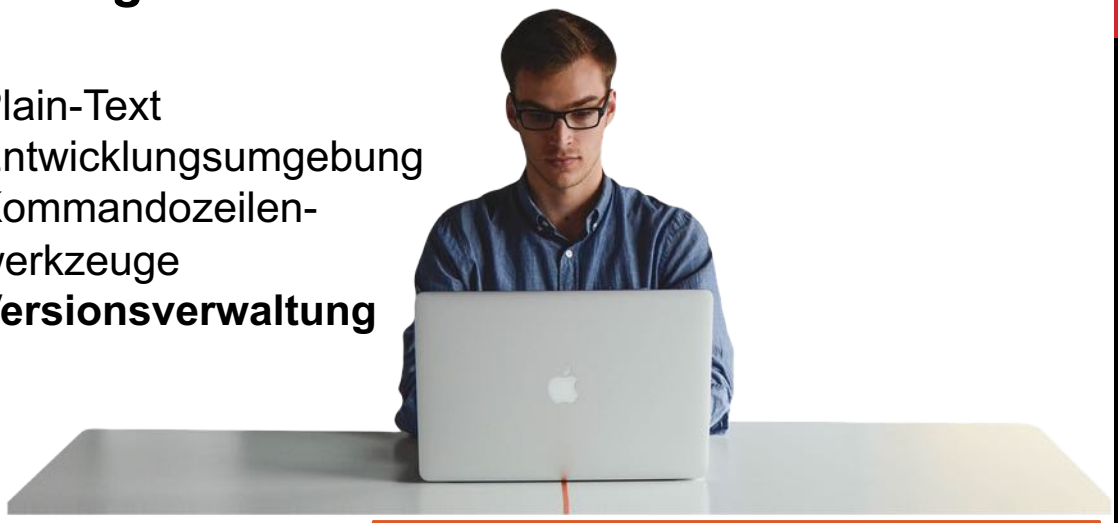
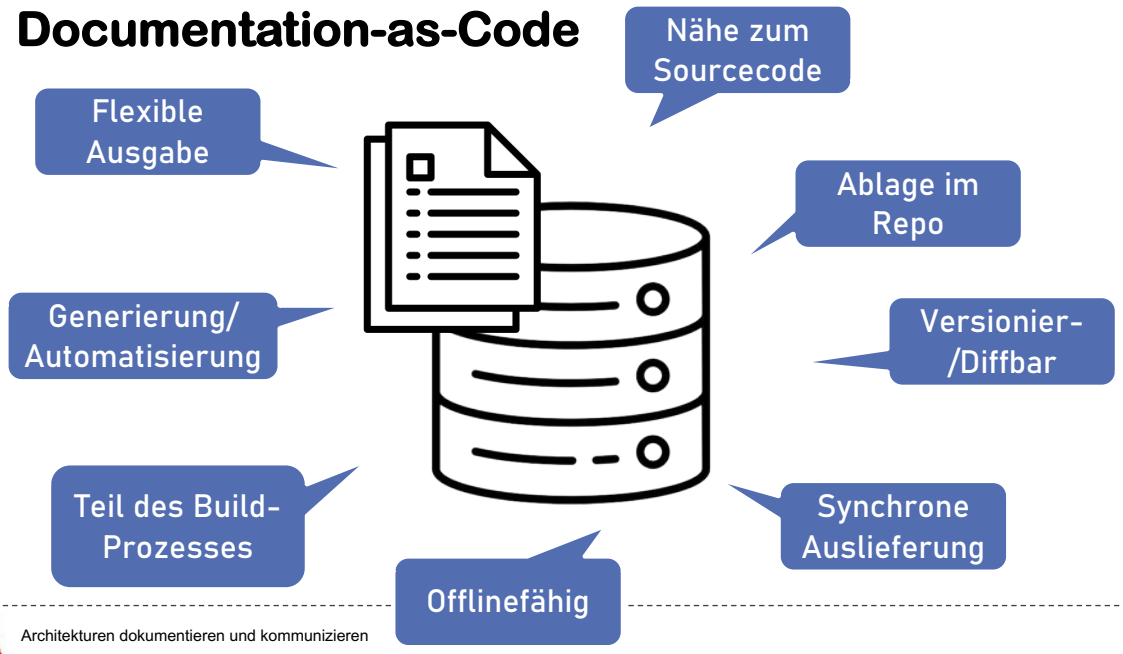


Foto von geralt: https://pixabay.com/de/unternehmer-start-start-up-karriere-696976/ (CC0 Public Domain Lizenz)

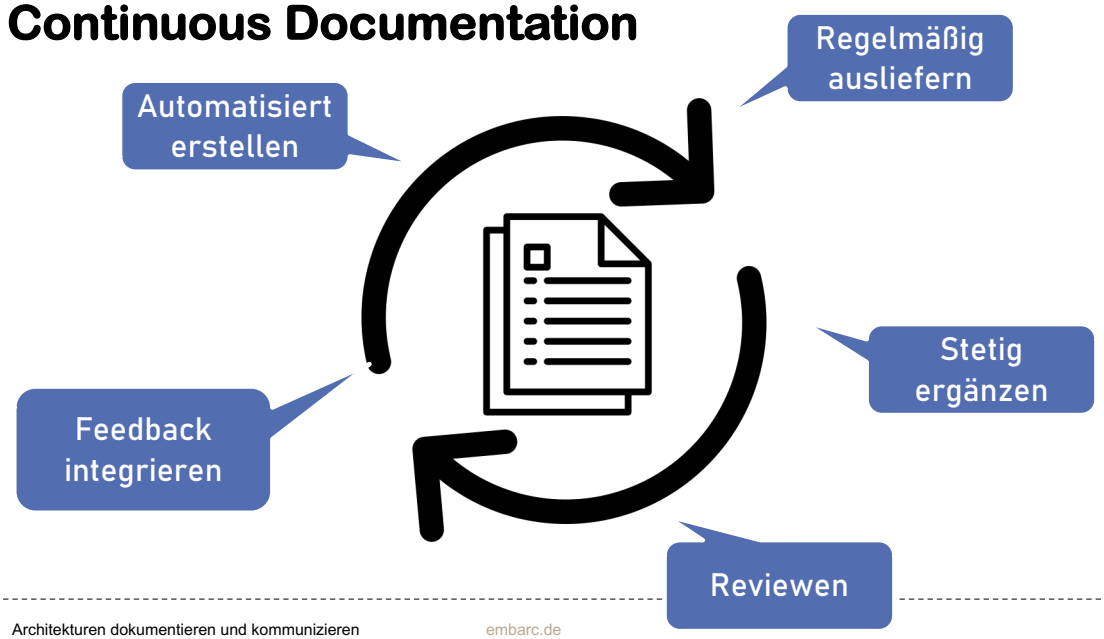
15

Documentation-as-Code



16

Continuous Documentation



17

Agenda

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

2

18

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

19

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

20

„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

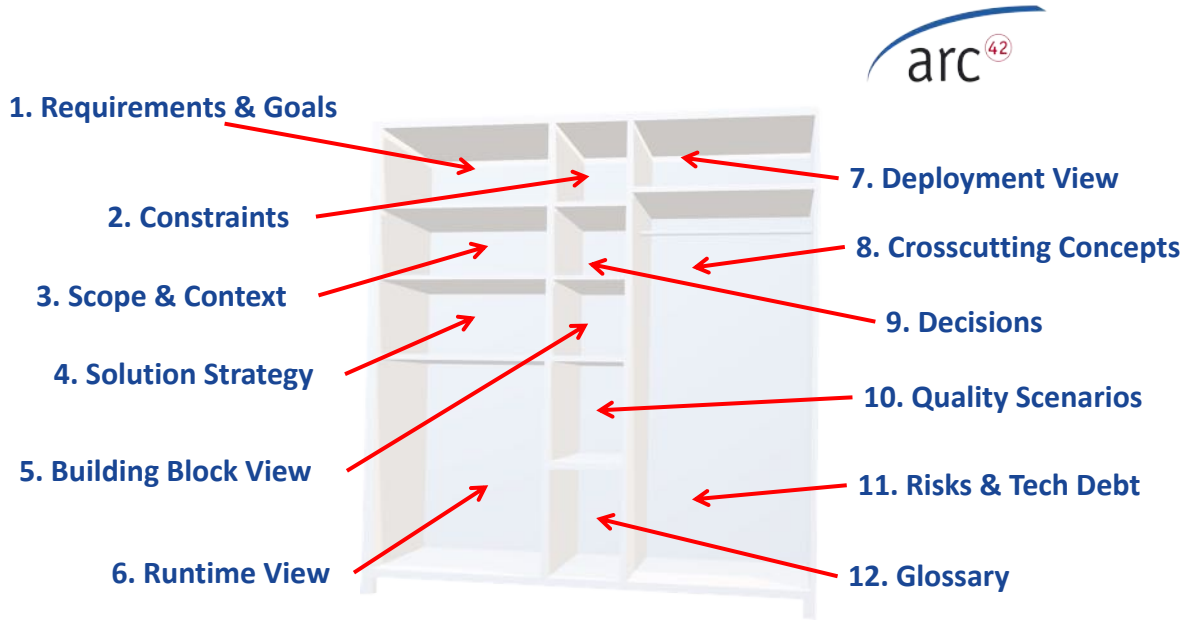
Dr. Peter Hruschka
http://www.peterhruschka.eu/

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

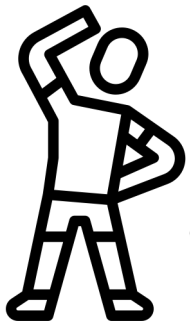
arc42

➔https://arc42.de/

21



22

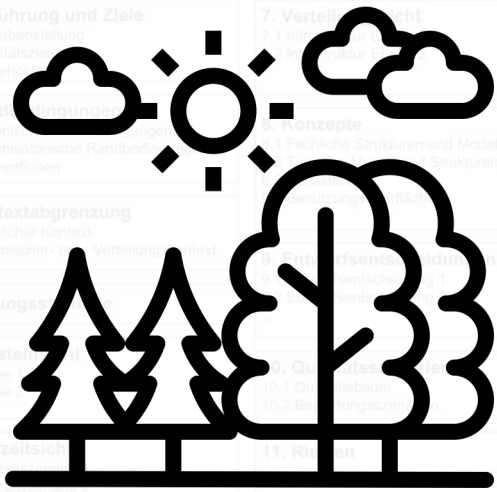


Bitte folgt dem Link für eine kleine Übung

https://tinyurl.com/arcpunsch-docs-as-code

23

Wald vor lauter Bäumen …



24

Architekturentwurf

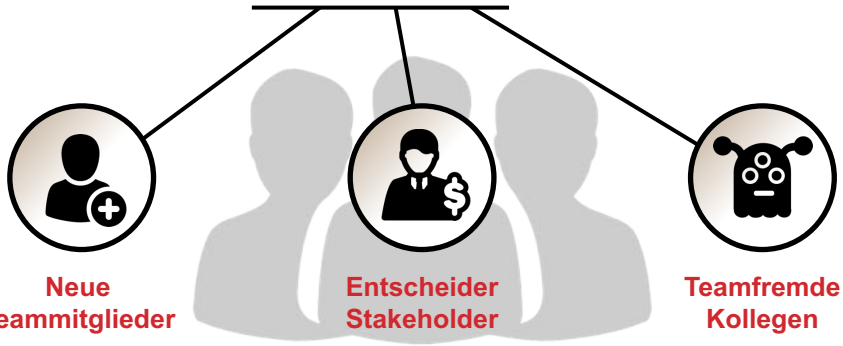
Anforderungen/
Architekturziele

Lösungsansätze/
Entscheidungen

25

Was ist ein Architekturüberblick?

Ein Architekturüberblick macht die zentralen Lösungsansätze Eurer Softwarearchitektur für Außenstehende nachvollziehbar.



Neue Teammitglieder

Entscheider Stakeholder

Teamfremde Kollegen

26

Zutaten für einen Architekturüberblick

Qualitätsziele

Vorgaben

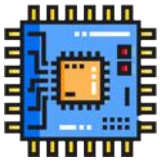
Systemkontext

Metapher “Produktkarton“

Informeller Überblick

Architektur-Stil/-Muster/…

Entscheidungen



Technologien

27

„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

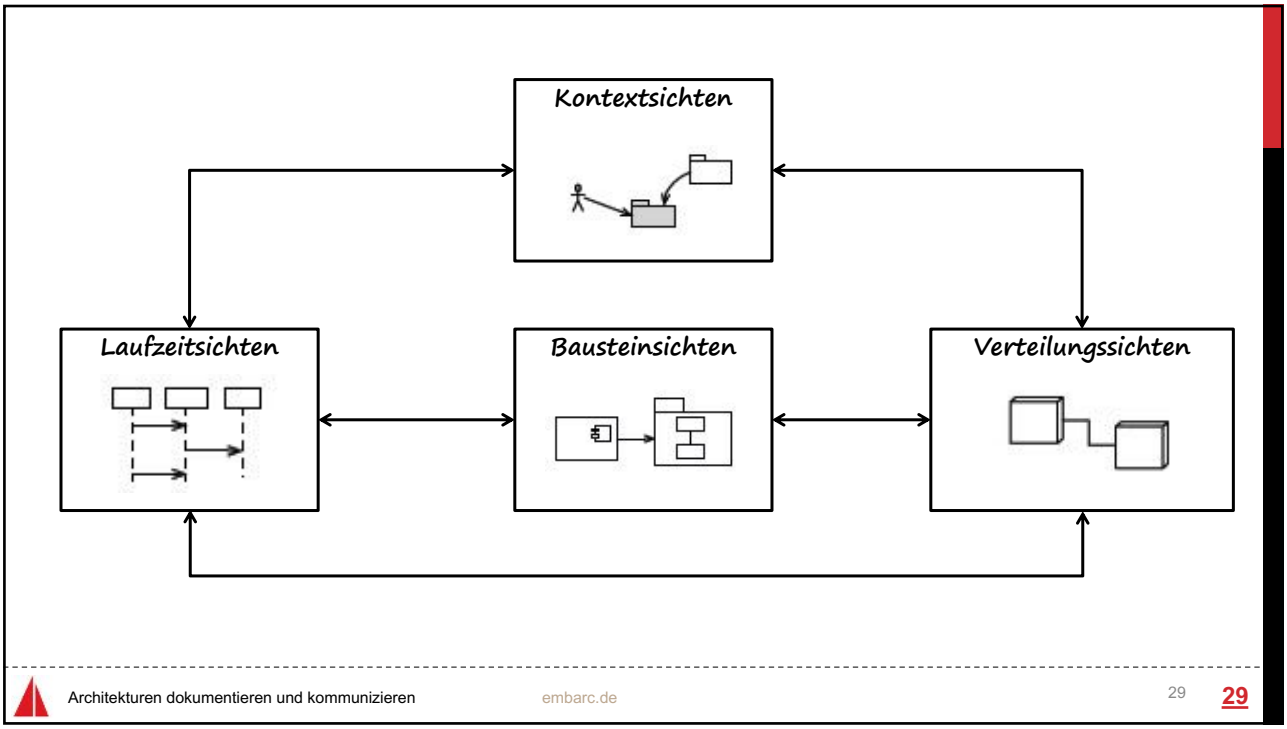
11. Risiken

12. Glossar

arc42

➔https://arc42.de/

28



29

„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

arc42

➔https://arc42.de/

30

„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

arc42

➔https://arc42.de/

41

Agenda

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

3

42

Docs-as-code / Architecture-as-code

43

Ein Reifegradmodell

0

Dokumentation wird getrennt vom Code verwaltet

Es gibt keine Anzeichen des Docs-as-Code Ansatzes

44

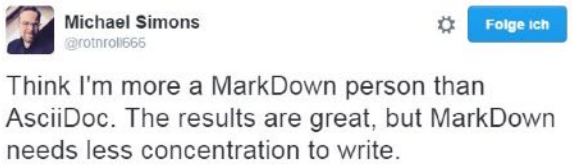
0 -> 1

Dokumentation mit dem Code in Git verwalten

~~**Word in Git**~~

Wahl einer Auszeichnungssprache

45



46

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

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

Markdown

einen Dialekt

Wir brauchen eine ~~Erweiterung!~~

Welche wählen wir?

Funktioniert dann unsere Toolchain noch?

Haben wir ein Editor-Preview?

49

AsciiDoc / Asciidoctor

leistungsfähige Syntax für technische Dokumentation

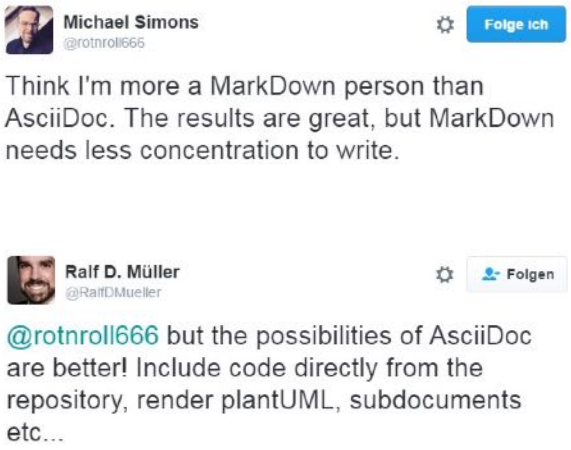
in Ruby geschrieben

mit Opal nach JavaScript transpiliert

mit jRuby auf der JVM gewrapped

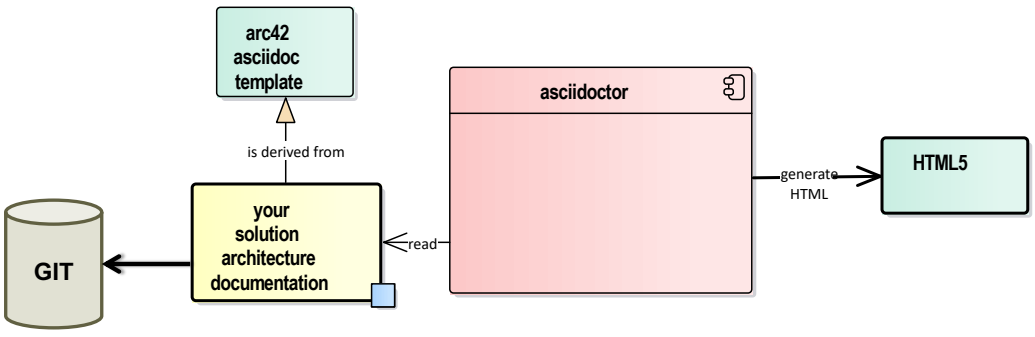
=> Keine Dialekte!

51



52

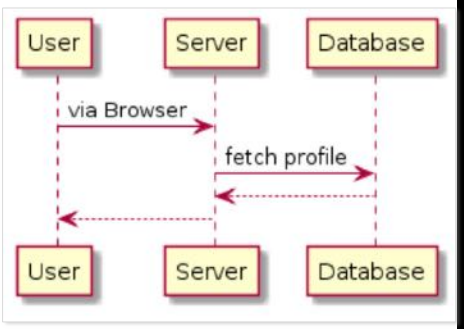
Basic Docs-as-Code with AsciiDoc



53

PlantUML

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



54

Ein Reifegradmodell

1

Dokumentation wird in Git mit dem Sourcecode verwaltet

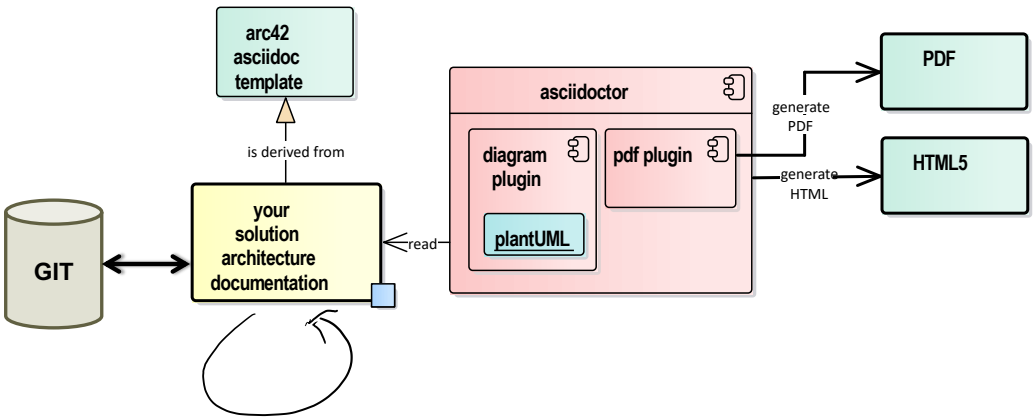
55

1 -> 2

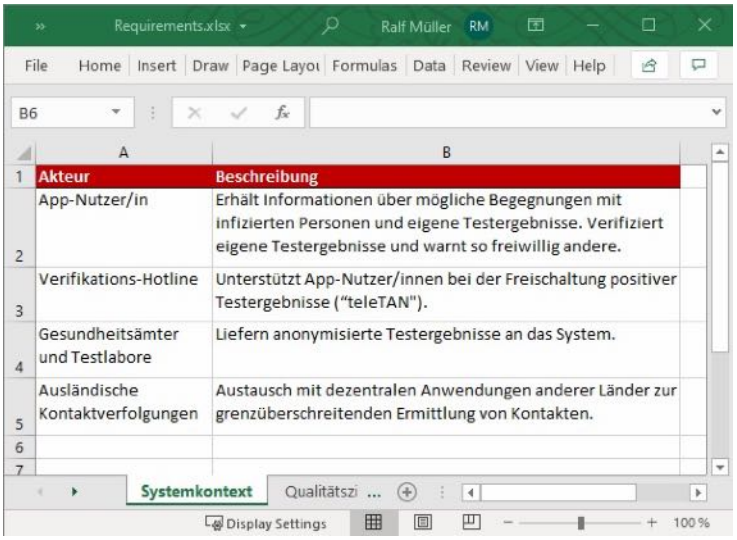
Maintain Docs like Code?

Treat Docs as Code!

Plugins + Scripted Docs-as-Code with AsciiDoc



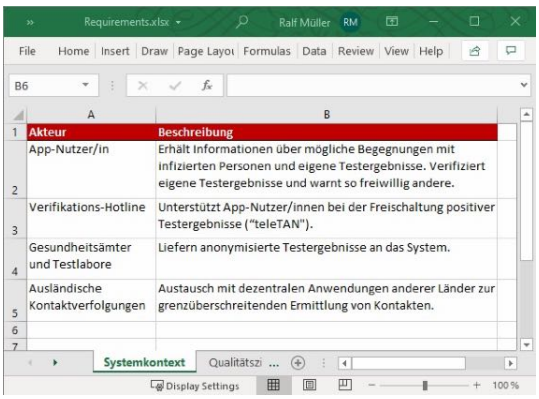
Tabellen!?



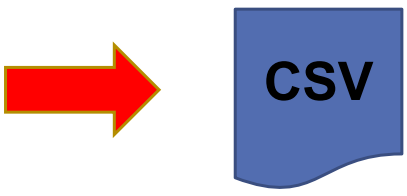
	A	B
1	**Akteur**	**Beschreibung**
2	App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
3	Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse ("teleTAN").
4	Gesundheitsämter und Testlabore	Liefern anonymisierte Testergebnisse an das System.
5	Ausländische Kontaktverfolgungen	Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.
6		
7		

58

Tabellen!?

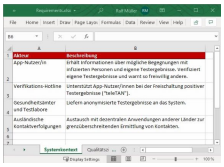


	A	B
1	**Akteur**	**Beschreibung**
2	App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
3	Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse ("teleTAN").
4	Gesundheitsämter und Testlabore	Liefern anonymisierte Testergebnisse an das System.
5	Ausländische Kontaktverfolgungen	Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.
6		
7		



59

Tabellen!?

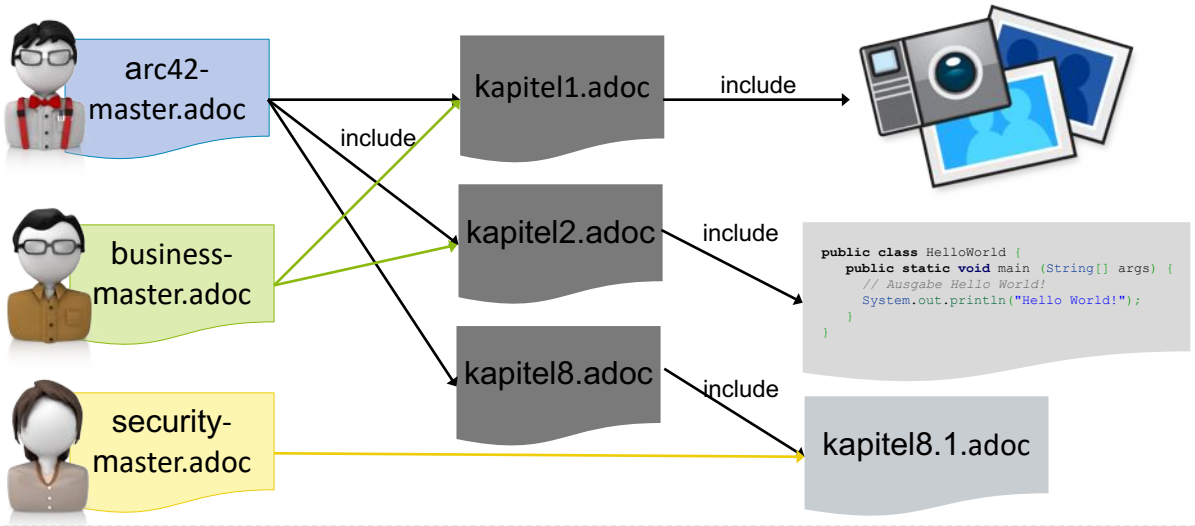


CSV

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.

60

Modulare Dokumentation



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

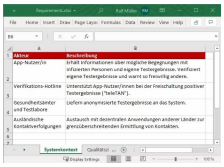
Ein Reifegradmodell

Dokumentation wird nicht nur als HTML gerendert, sondern auch mit eignen Skripten extrahiert und transformiert.

…wie Code

63

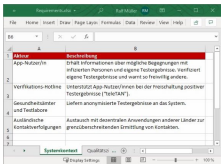
Tabellen!?



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.

64

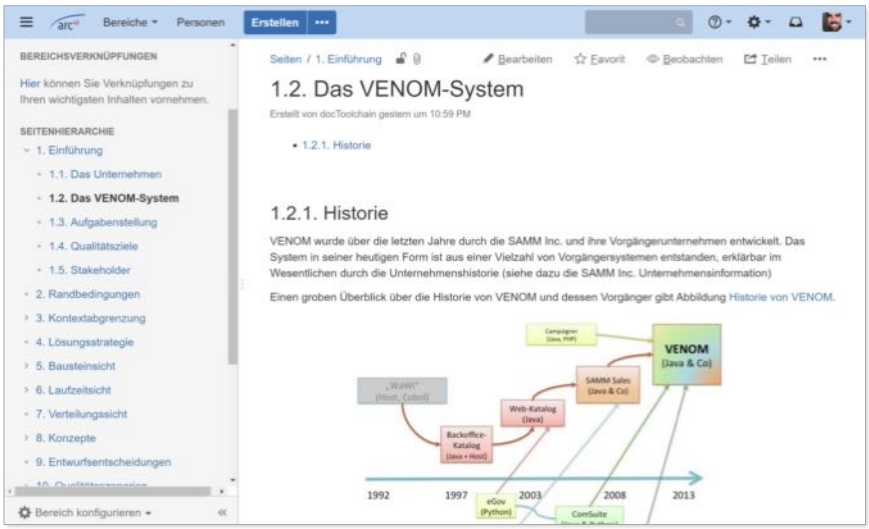
Tabellen!?



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.

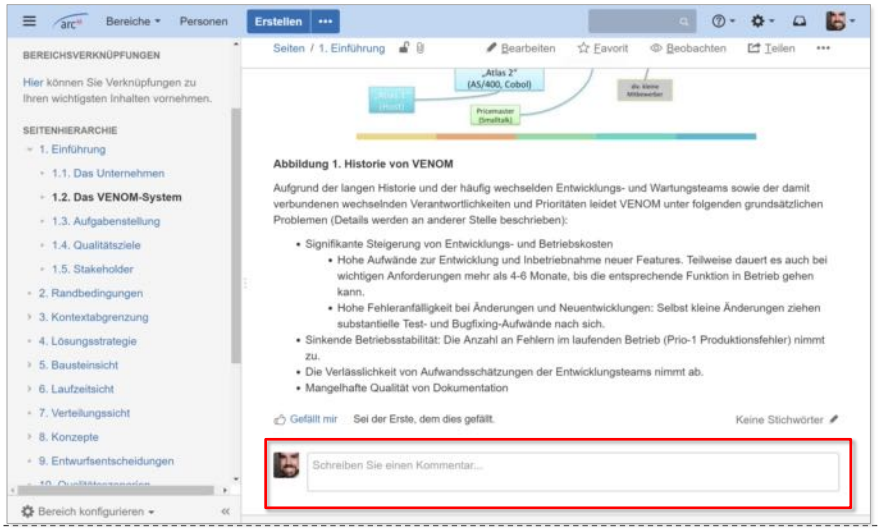
65

Zusammenarbeit



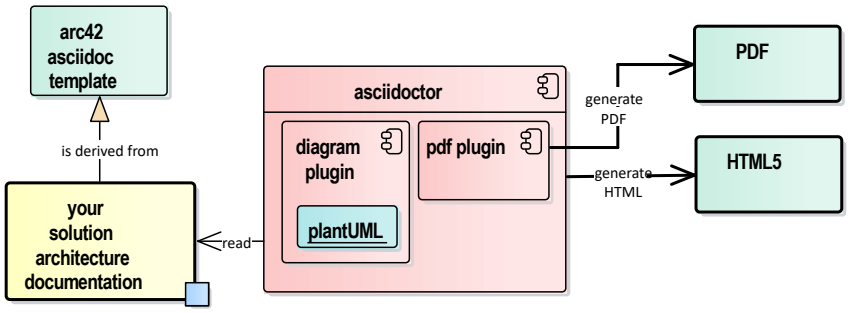
66

Zusammenarbeit



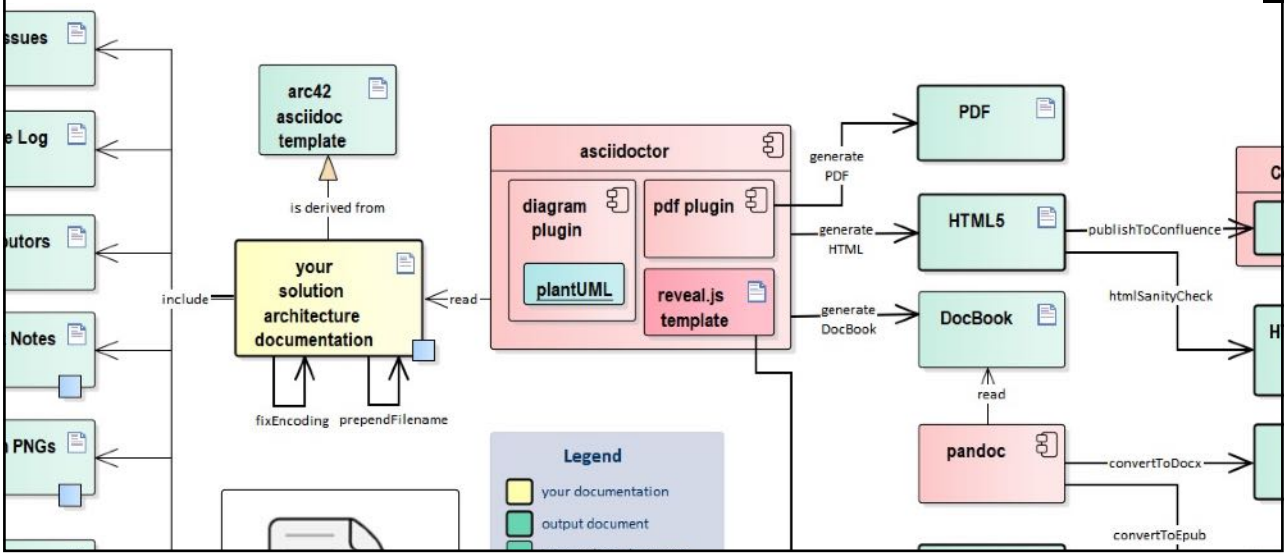
67

Basic Docs-as-Code with AsciiDoc



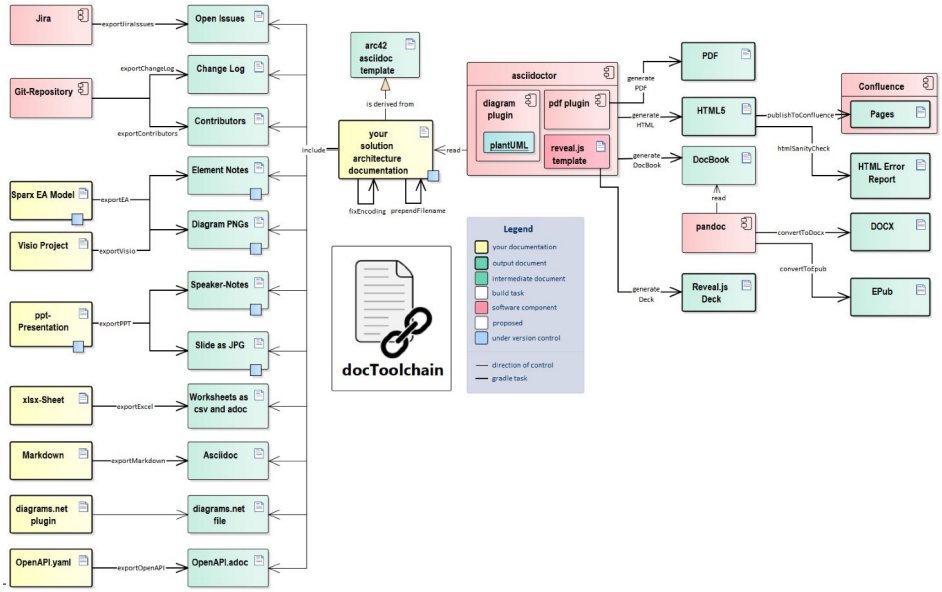
68

Basic Docs-as-Code with AsciiDoc & docToolchain



69

Basic Docs-as-Code with AsciiDoc & docToolchain



70

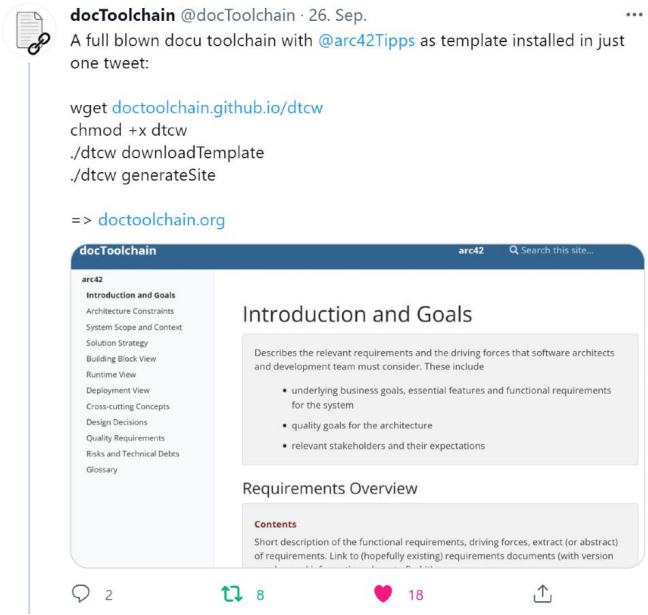
docToolchain @docToolchain · 26. Sep.

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
```


75



76

Ein Reifegradmodell

2.5

Es werden Standard-Skripte zum Extrahieren und Transformieren verwendet,

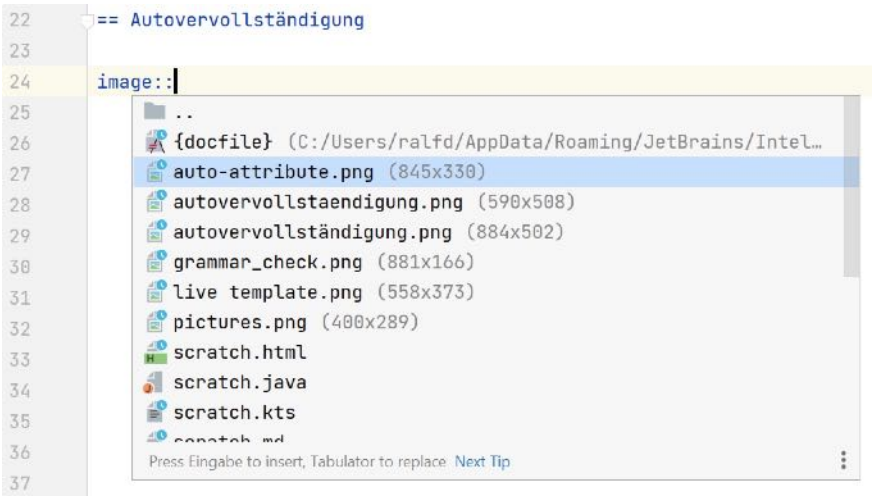
wie z.B. docToolchain

77

Docs-as-Code in der IDE

78

Docs-as-Code in der IDE



79

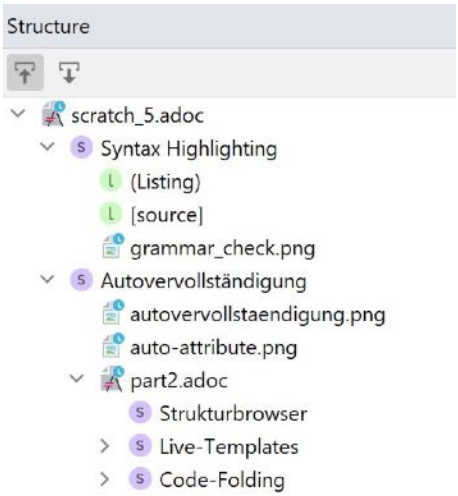
Docs-as-Code in der IDE

80

Docs-as-Code in der IDE

81

Docs-as-Code in der IDE



82

Docs-as-Code in der IDE

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

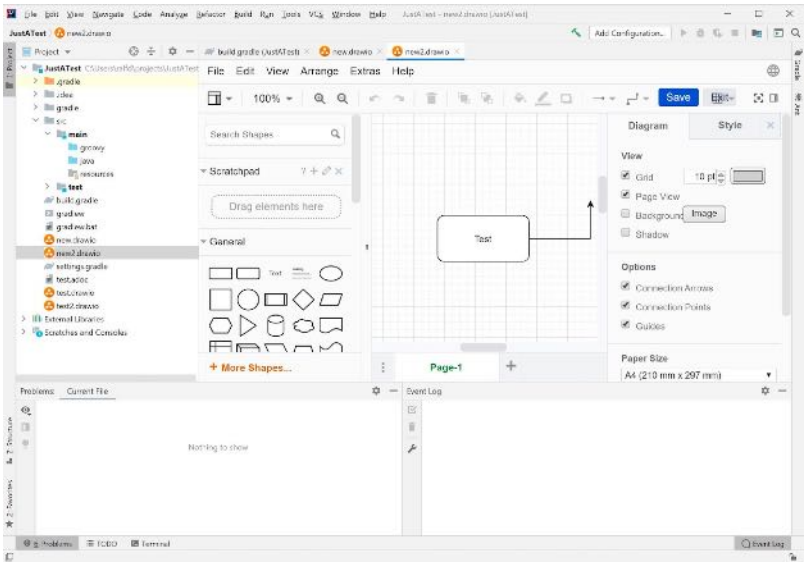

83

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
```


84

Draw.io/Diagrams.net Integration



85

Ein Reifegradmodell

3

Der Docs-as-Code Ansatz wird durch Plugins auch in die IDE getragen

Autovervollständigung, Syntax Highlighting, Strukturbrowser, Live-Templates, Code-Folding etc.

86

Agenda

1 Einstieg und Motivation

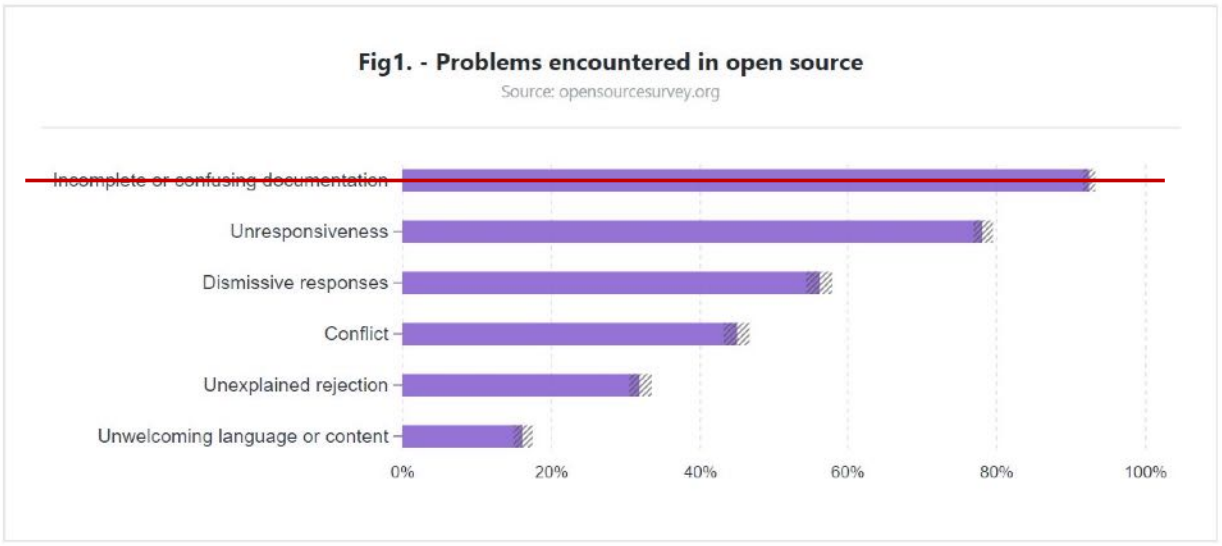
2 Architektur dokumentieren

3 Docs-as-Code Maturity Level

4 **Fazit und Ausblick**

4

89



90

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“

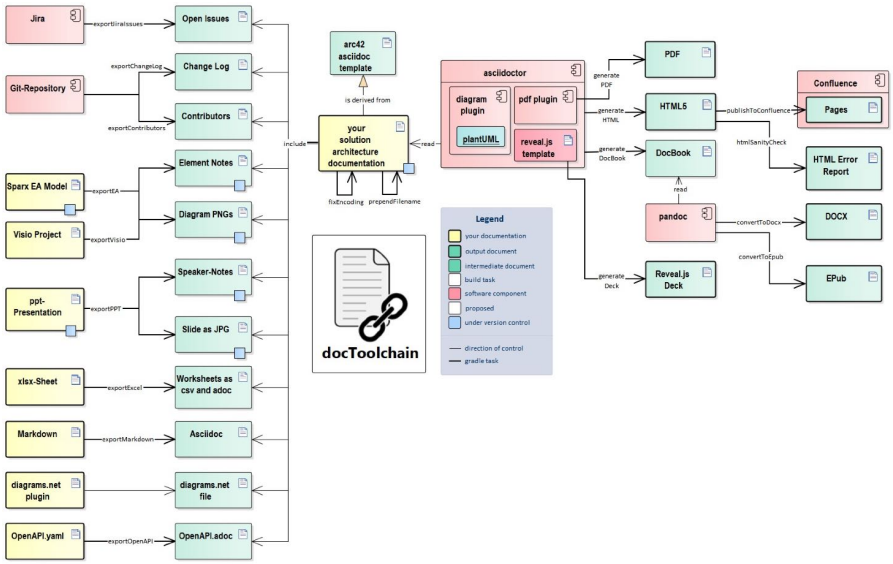
Formatierung von Source-Code

Reviews, Pull-Requests, Versionierung durch Git

Konvertierung nach HTML5 und DocBook

91

docToolchain Übersicht



92

Moderne (Architektur-)Dokumentation

Textformate

Single Source of Truth

Inhalte generieren

Bilder/Grafiken

Docs-as-Code

Continuous Documentation

Ausführen/ Validieren

Kümmerer

93

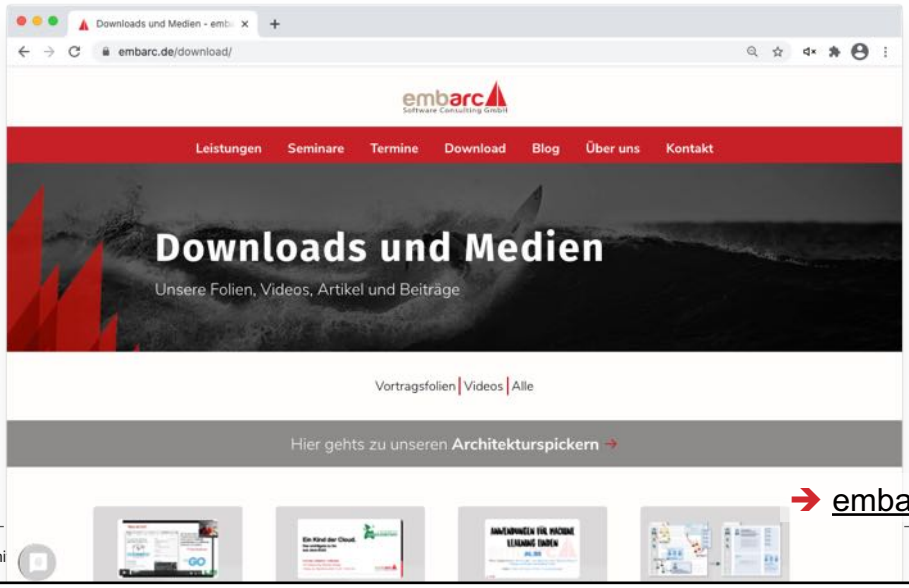
Inhalte generieren

Ausführen/ Validieren

Kümmerer

94

Folien von heute als PDF zum Download



➔ embarc.de/download/

95

Gibt es auch gedruckt!

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

info@embarc.de

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

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

Vielen Dank.

embarc Software Consulting GmbH

Ich freue mich auf Eure Fragen!

Falk Sippach

fs@embarc.de

@sippsack

➔ xing.to/fsi