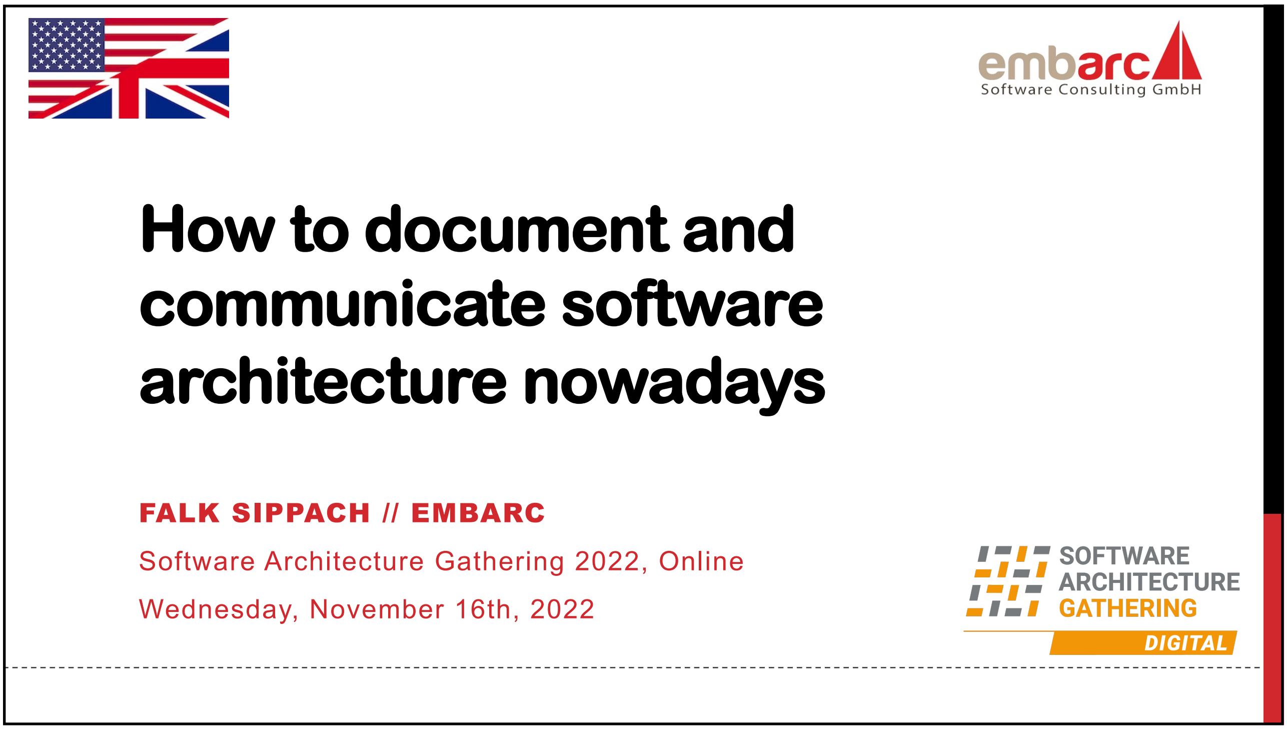
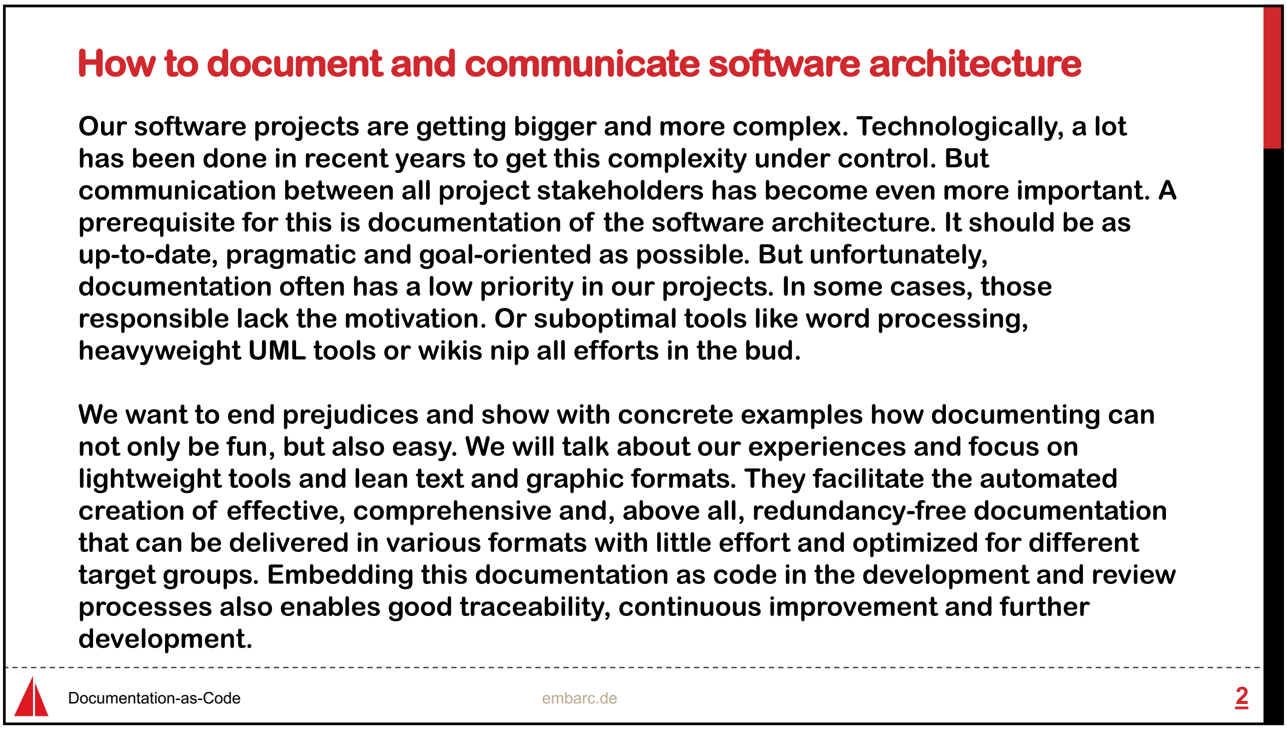


How to document and communicate software architecture nowadays

FALK SIPPACH // EMBARC

Software Architecture Gathering 2022, Online

Wednesday, November 16th, 2022



How to document and communicate software architecture

Our software projects are getting bigger and more complex. Technologically, a lot has been done in recent years to get this complexity under control. But communication between all project stakeholders has become even more important. A prerequisite for this is documentation of the software architecture. It should be as up-to-date, pragmatic and goal-oriented as possible. But unfortunately, documentation often has a low priority in our projects. In some cases, those responsible lack the motivation. Or suboptimal tools like word processing, heavyweight UML tools or wikis nip all efforts in the bud.

We want to end prejudices and show with concrete examples how documenting can not only be fun, but also easy. We will talk about our experiences and focus on lightweight tools and lean text and graphic formats. They facilitate the automated creation of effective, comprehensive and, above all, redundancy-free documentation that can be delivered in various formats with little effort and optimized for different target groups. Embedding this documentation as code in the development and review processes also enables good traceability, continuous improvement and further development.

Falk Sippach

- Software Architect, Consultant, Trainer at embarc
- formerly at Orientation in Objects (OIO), Trivadis

Focus:

- Architecture consulting and reviews
- Cloud and Java technologies

fs@embarc.de

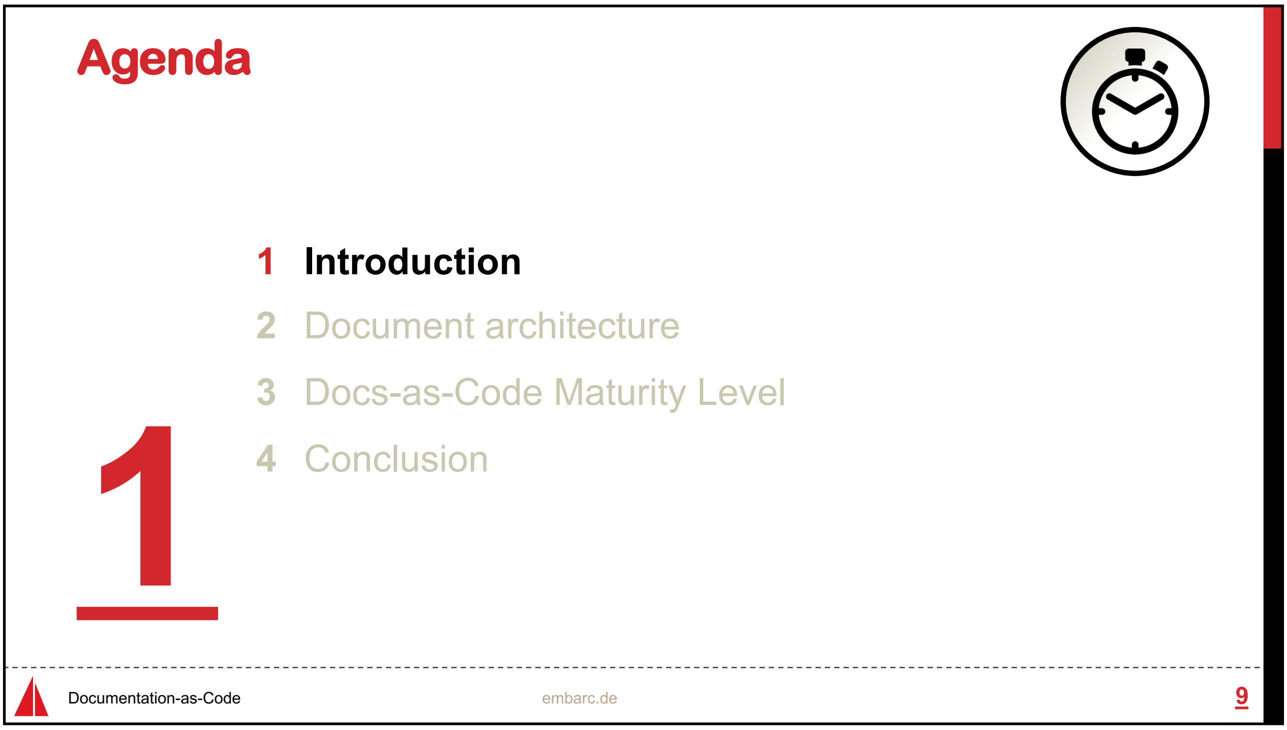
@sippsack

➔ xing.to/fsi

embarc
Software Consulting GmbH

Documentation-as-Code embarc.de 3

3



Agenda

1 **Introduction**

2 Document architecture

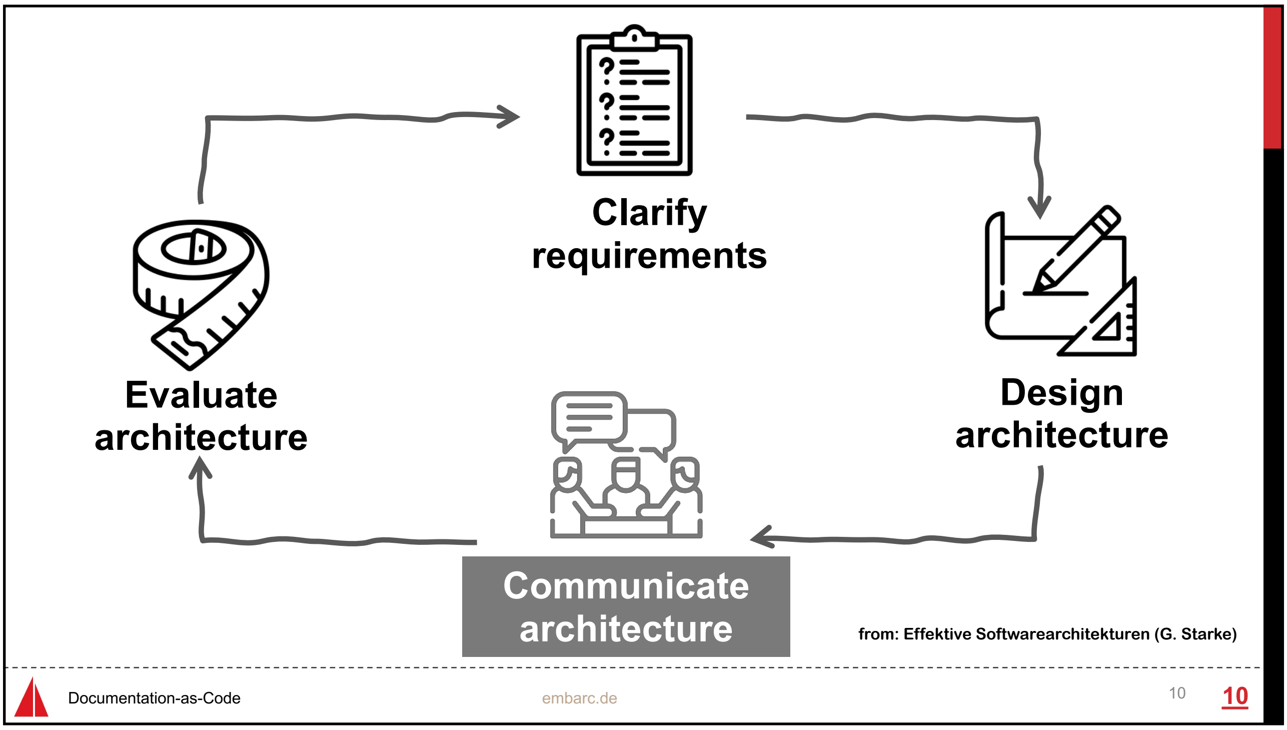
3 Docs-as-Code Maturity Level

4 Conclusion

1

Documentation-as-Code embarc.de 9

9



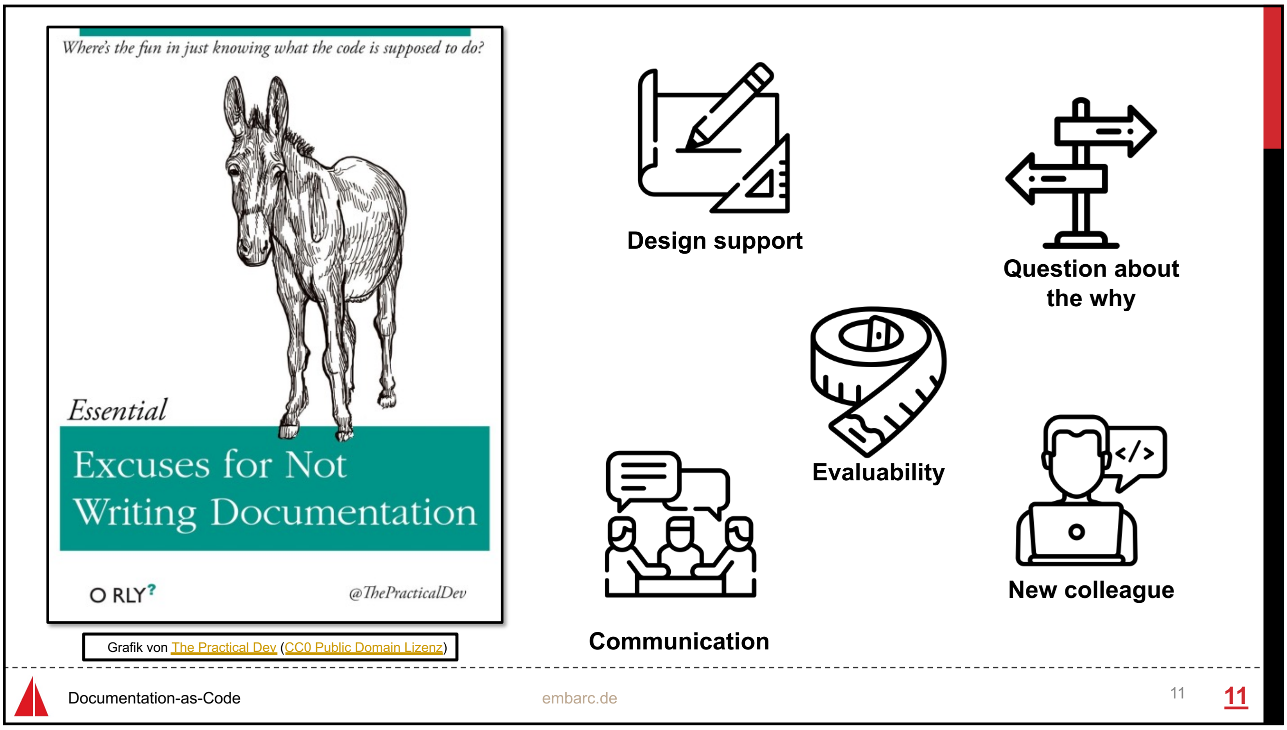
Clarify requirements

Design architecture

Communicate architecture

Evaluate architecture

from: Effektive Softwarearchitekturen (G. Starke)



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

Essential

Excuses for Not Writing Documentation

O RLY?

@ThePracticalDev

Grafik von The Practical Dev (CC0 Public Domain Lizenz)

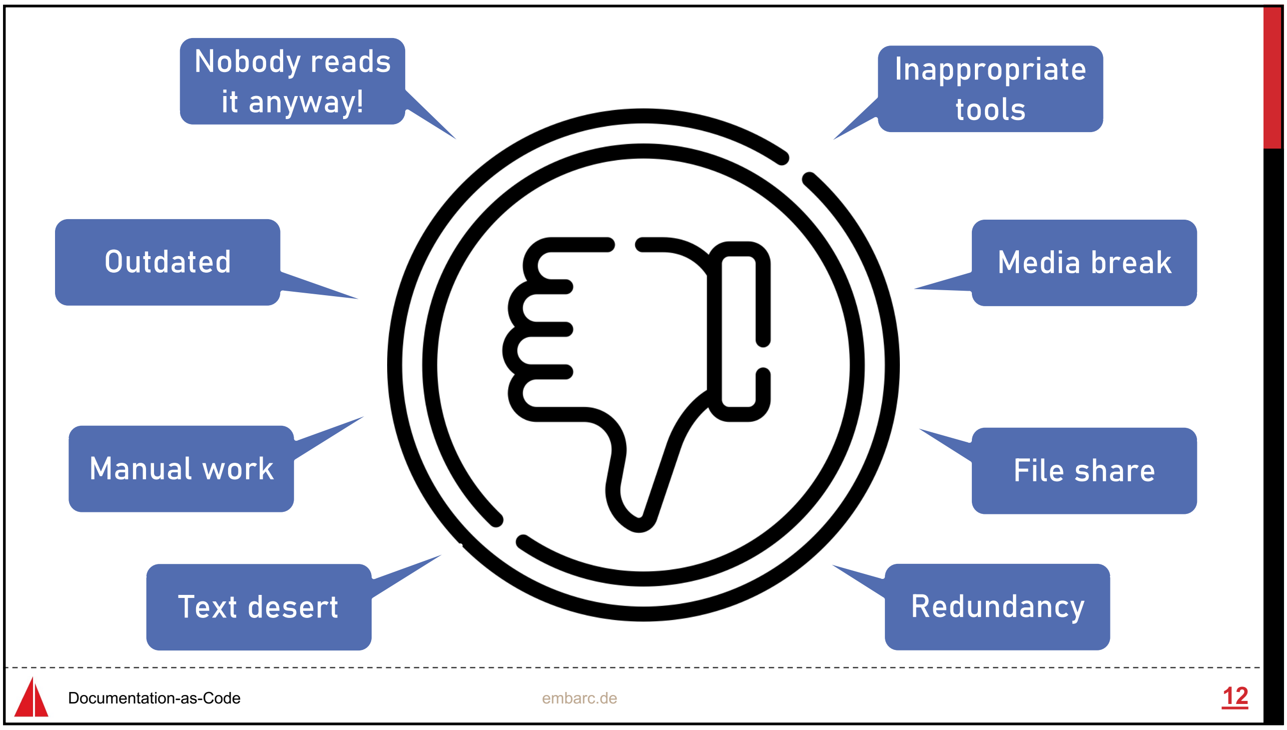
Design support

Question about the why

Evaluability

Communication

New colleague



Nobody reads it anyway!

Inappropriate tools

Outdated

Media break

Manual work

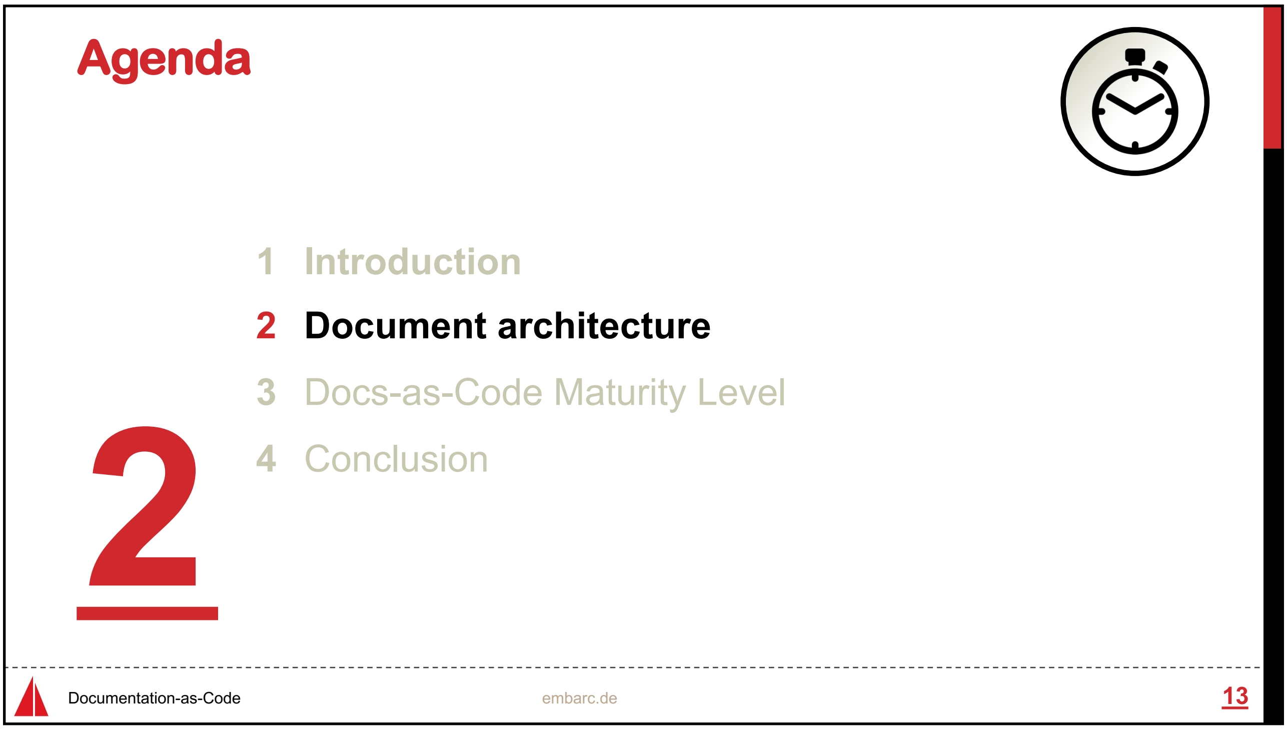
File share

Text desert

Redundancy

Documentation-as-Code embarc.de 12

12



Agenda

1 Introduction

2 Document architecture

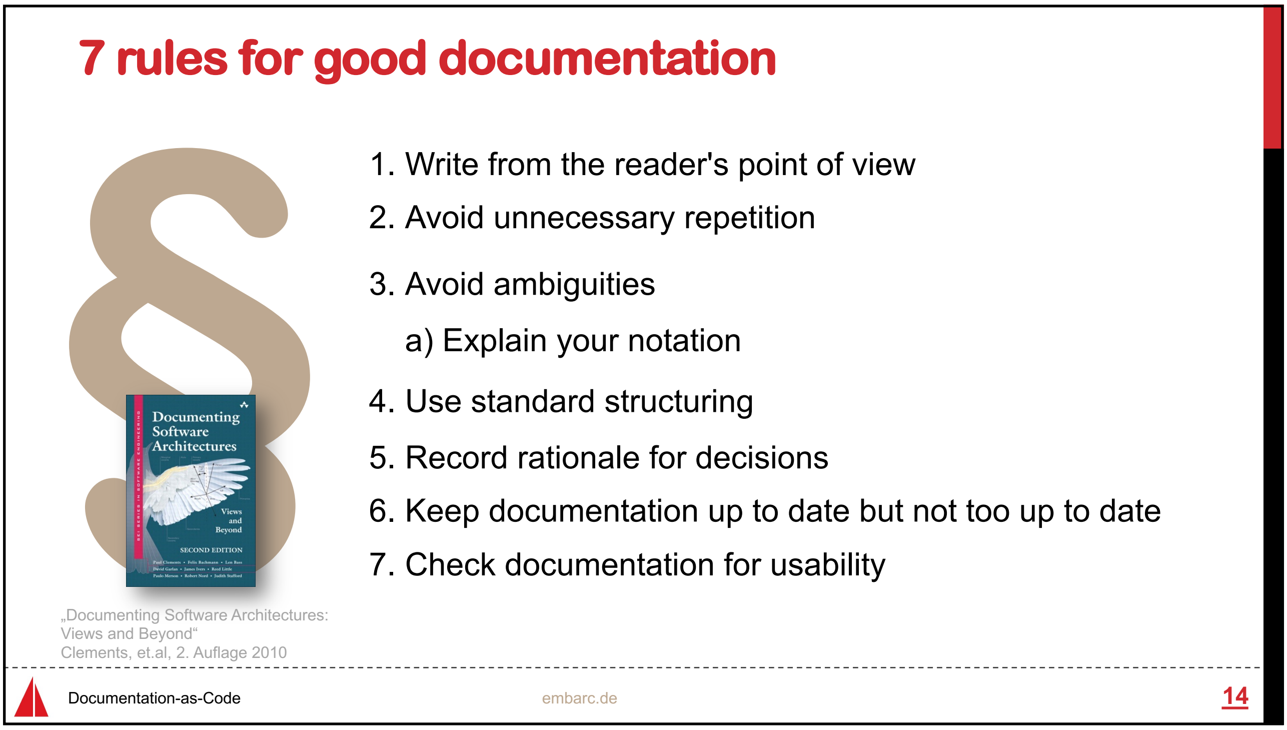
3 Docs-as-Code Maturity Level

4 Conclusion

2

Documentation-as-Code embarc.de 13

13



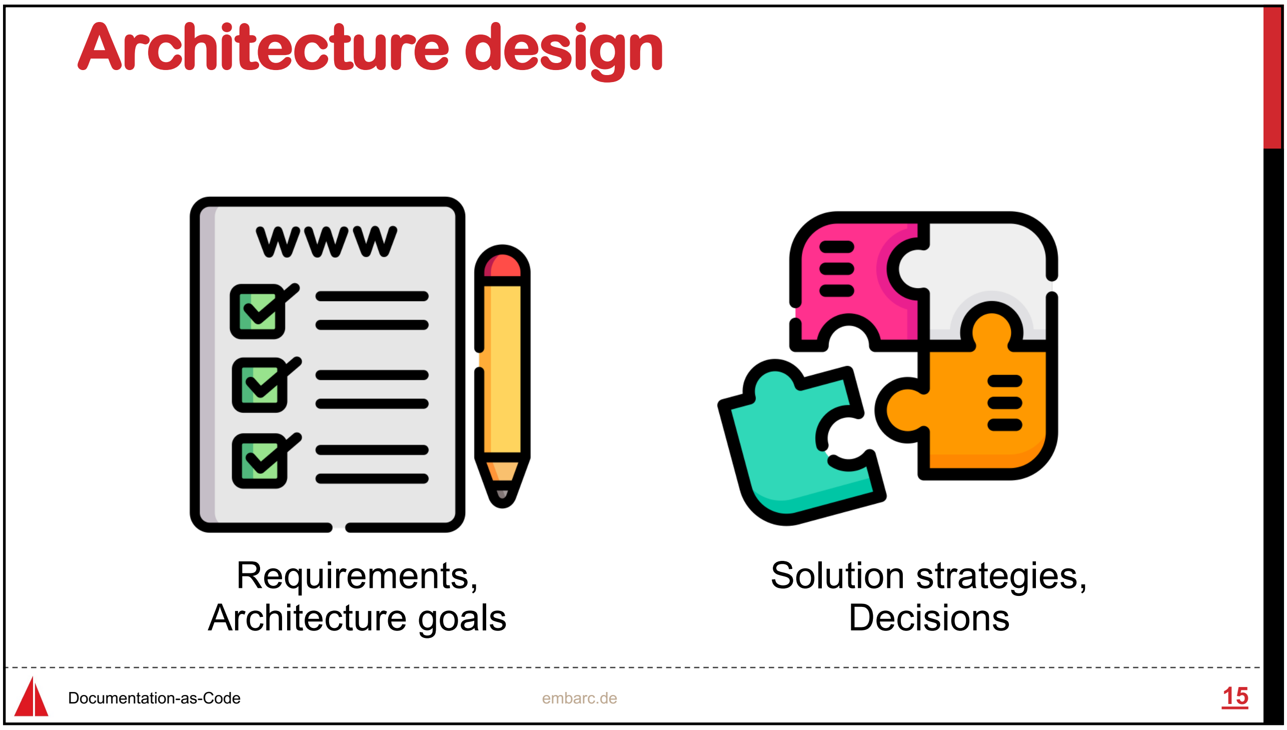
7 rules for good documentation

1. Write from the reader's point of view
2. Avoid unnecessary repetition
3. Avoid ambiguities
   a) Explain your notation
4. Use standard structuring
5. Record rationale for decisions
6. Keep documentation up to date but not too up to date
7. Check documentation for usability

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

Documentation-as-Code embarc.de 14

14



Architecture design

Requirements,
Architecture goals

Solution strategies,
Decisions

Documentation-as-Code embarc.de 15

15

Ingredients for an architectural overview

Quality goals

Constraints

System context

Mission Statement

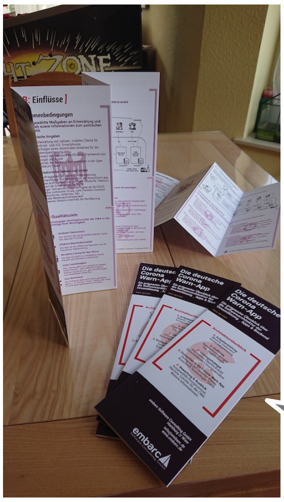
Informal overview

Architecture style/ patterns/principles

Decisisions

Technologies

Examples to look up (German only)



Slides

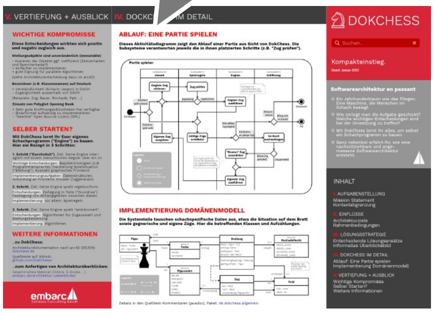
Die deutsche Corona Warn-App.
Ein prägnanter Überblick über die Softwarearchitektur der Gesamtlösung – Apps und Backend.

STAND: APRIL 2021

embarc Software Consulting GmbH

CWA Flyer

DokChess Flyer



V. VERTIEFUNG + AUSBLICK

WICHTIGE KOMPROMISSE

Diese Entscheidungen wirkten sich positiv und negativ zugleich aus.

SELBER STARTEN?

Mit DokChess lernt Ihr Euer eigenes Schachprogramm ("Engine") zu bauen. Hier ein Rezept in 3 Schritten:

WEITERE INFORMATIONEN

...zu DokChess:

Architekturdokumentation nach arc42 (DE/EN): dokchess.de

Quelltexte auf GitHub: github.com/DokChess/

...zum Anfertigen von Architekturüberblicken:

Gesammeltes Material (Videos, E-Books, ...) embarc.de/architektur-ueberblicke/

IV. DOKCHESS IM DETAIL

ABLAUF: EINE PARTIE SPIELEN

Dieses Aktivitätsdiagramm zeigt den Ablauf einer Partie aus Sicht von DokChess. Die Subsysteme verantworten jeweils die in ihnen platzierten Schritte (z.B. "Zug prüfen").

IMPLEMENTIERUNG DOMÄNENMODELL

Die Systemteile tauschen schachspezifische Daten aus, etwa die Situation auf dem Brett sowie gegnerische und eigene Züge. Hier die betreffenden Klassen und Aufzählungen.

Details in den Quelltext-Kommentaren (javadoc), Paket: de.dokchess.allgemein

DOKCHESS

Kompakteinstieg.

Softwarearchitektur en passant

- Ein Jahrhunderttraum wie das Fliegen. Eine Maschine, die Menschen im Schach besiegt.
- Wie zerlegt man die Aufgabe geschickt? Welche wichtigen Entscheidungen sind bei der Umsetzung zu treffen?
- Mit DokChess lernt Ihr alles, um selbst ein Schachprogramm zu bauen.
- Ganz nebenbei erfahrt Ihr, wie eine nachvollziehbare und angemessene Softwarearchitektur entsteht.

INHALT

I. AUFGABENSTELLUNG
Mission Statement
Kontextabgrenzung

II. EINFLÜSSE
Architekturziele
Rahmenbedingungen

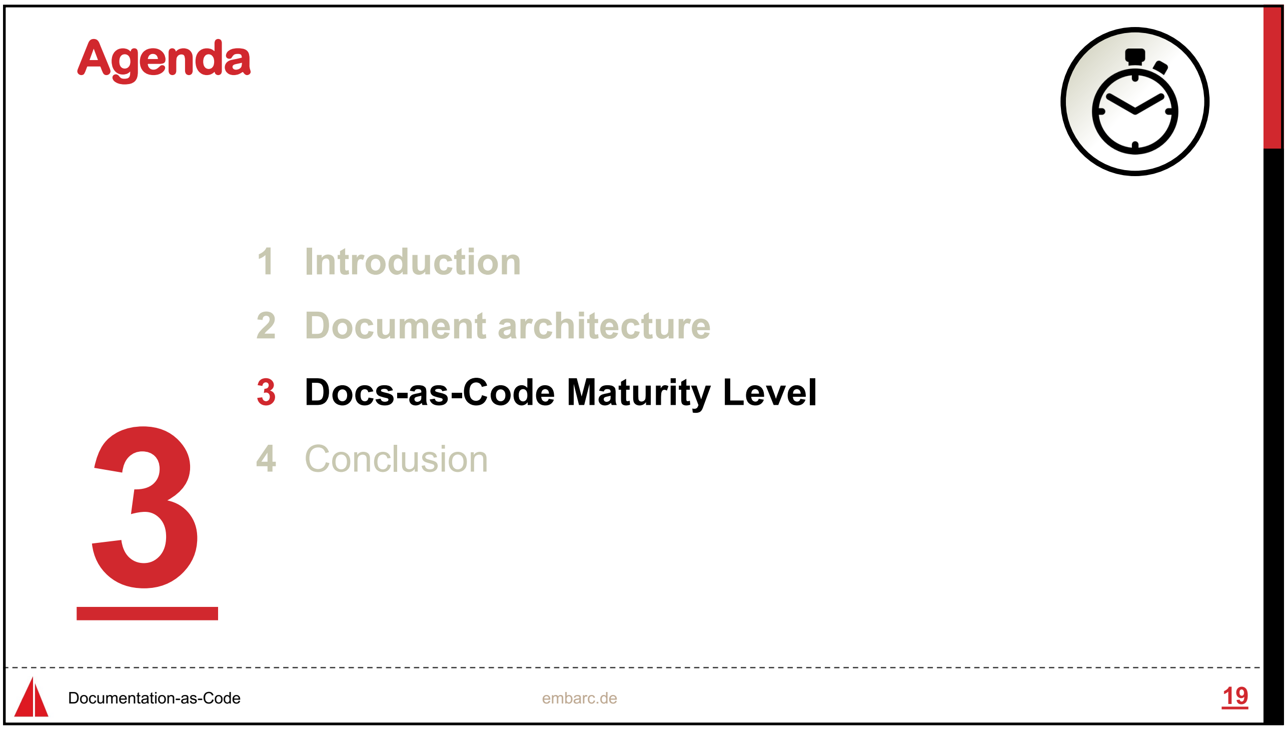
III. LÖSUNGSSTRATEGIE
Entscheidende Lösungsansätze
Informelles Überblicksbild

IV. DOKCHESS IM DETAIL
Ablauf: Eine Partie spielen
Implementierung Domänenmodell

V. VERTIEFUNG + AUSBLICK
Wichtige Kompromisse
Selber Starten?
Weitere Informationen

Author: Stefan Zörner

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



Agenda

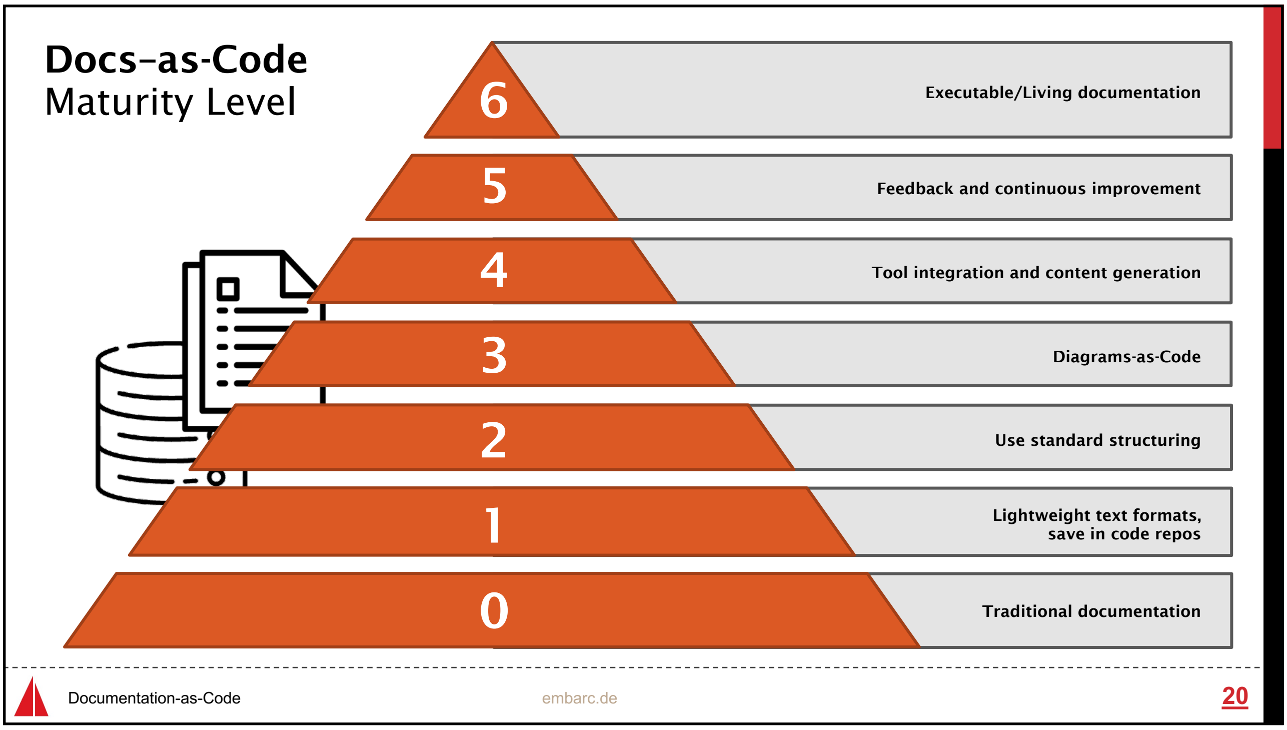
3

1 Introduction

2 Document architecture

3 Docs-as-Code Maturity Level

4 Conclusion



Docs-as-Code Maturity Level

6 Executable/Living documentation

5 Feedback and continuous improvement

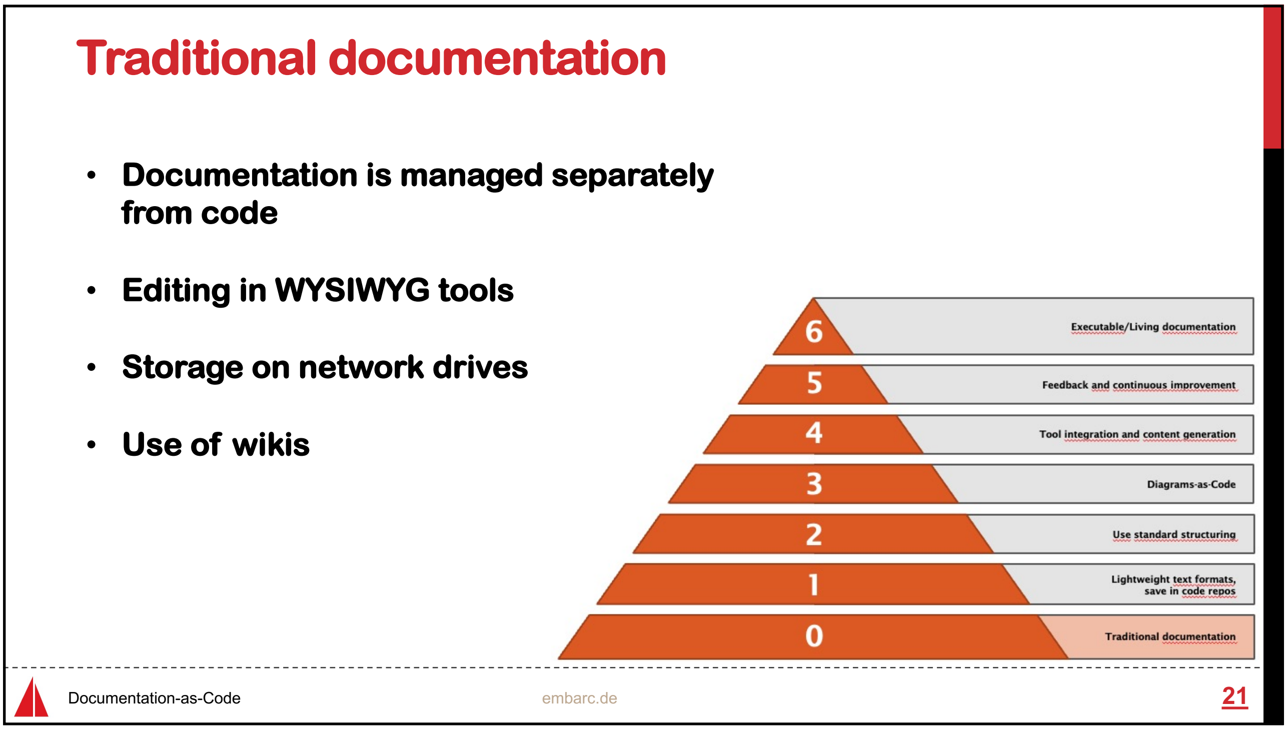
4 Tool integration and content generation

3 Diagrams-as-Code

2 Use standard structuring

1 Lightweight text formats, save in code repos

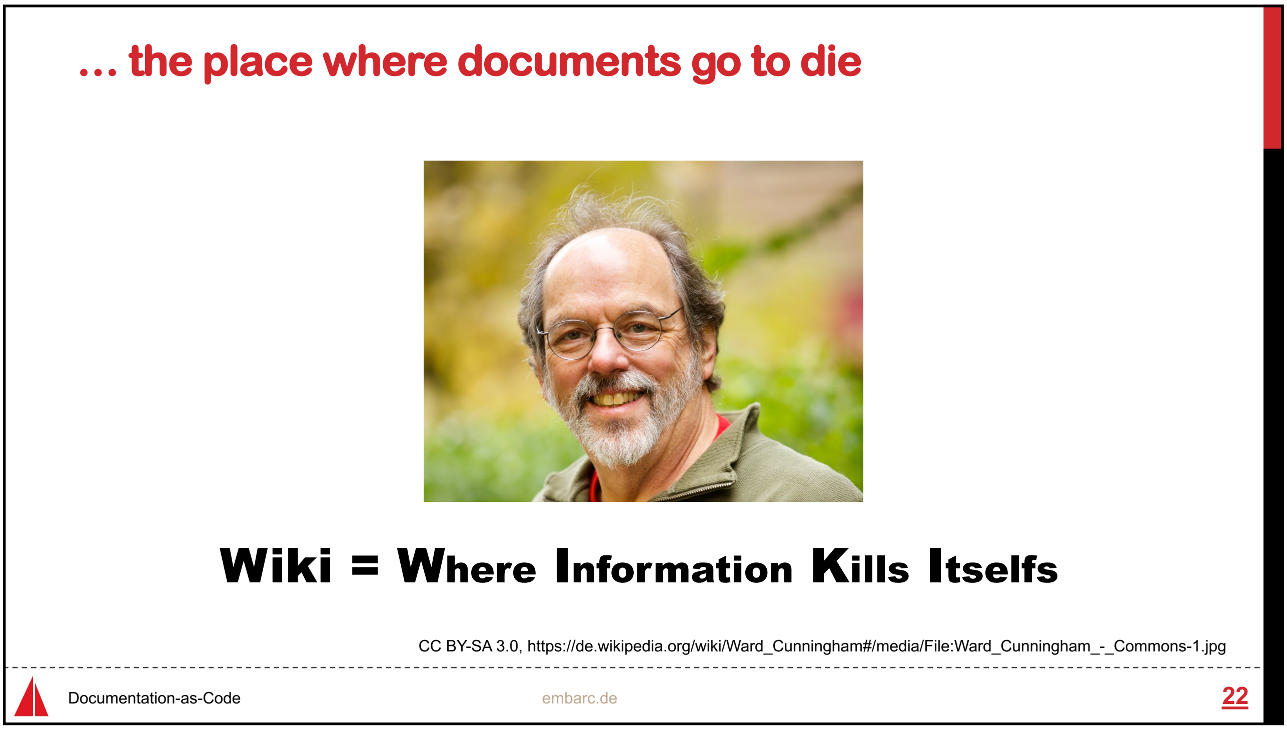
0 Traditional documentation



Traditional documentation

- Documentation is managed separately from code
- Editing in WYSIWYG tools
- Storage on network drives
- Use of wikis

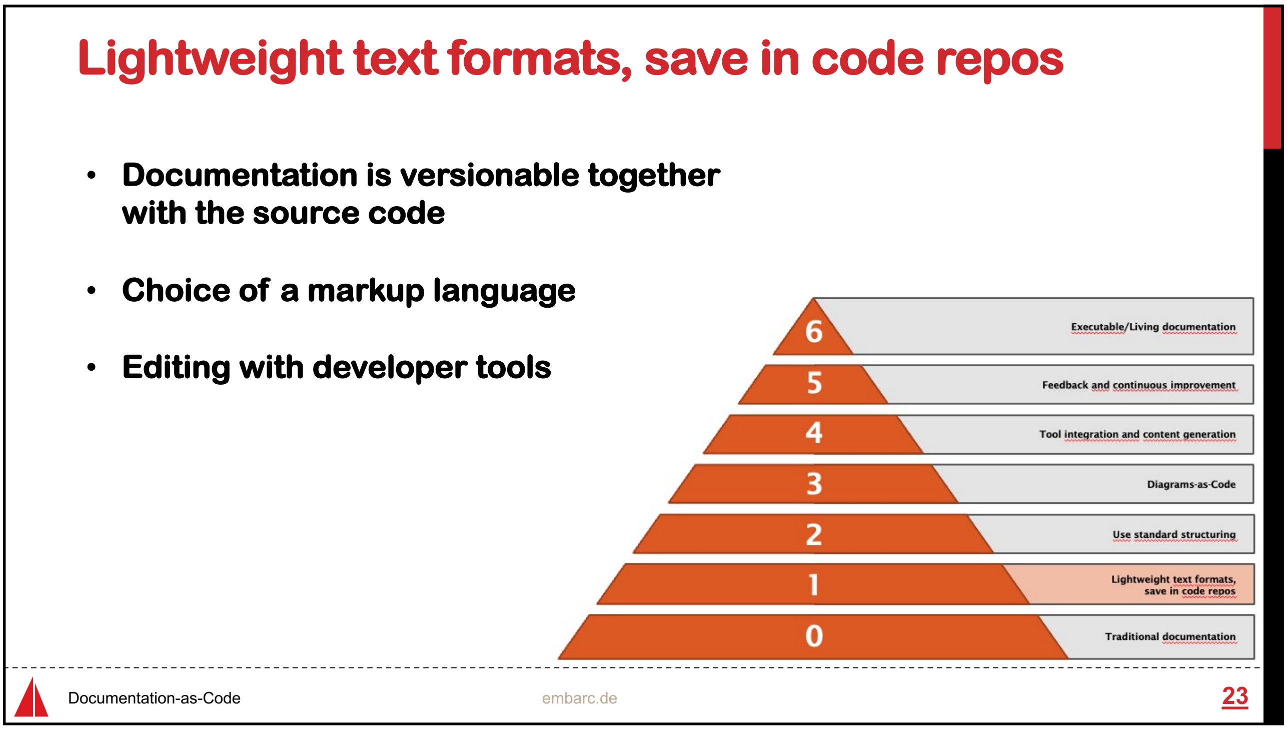
6 Executable/Living documentation
5 Feedback and continuous improvement
4 Tool integration and content generation
3 Diagrams-as-Code
2 Use standard structuring
1 Lightweight text formats, save in code repos
0 Traditional documentation



… the place where documents go to die

Wiki = Where Information Kills Itselfs

CC BY-SA 3.0, https://de.wikipedia.org/wiki/Ward_Cunningham#/media/File:Ward_Cunningham_-_Commons-1.jpg



Lightweight text formats, save in code repos

- **Documentation is versionable together with the source code**
- **Choice of a markup language**
- **Editing with developer tools**

6 Executable/Living documentation

5 Feedback and continuous improvement

4 Tool integration and content generation

3 Diagrams-as-Code

2 Use standard structuring

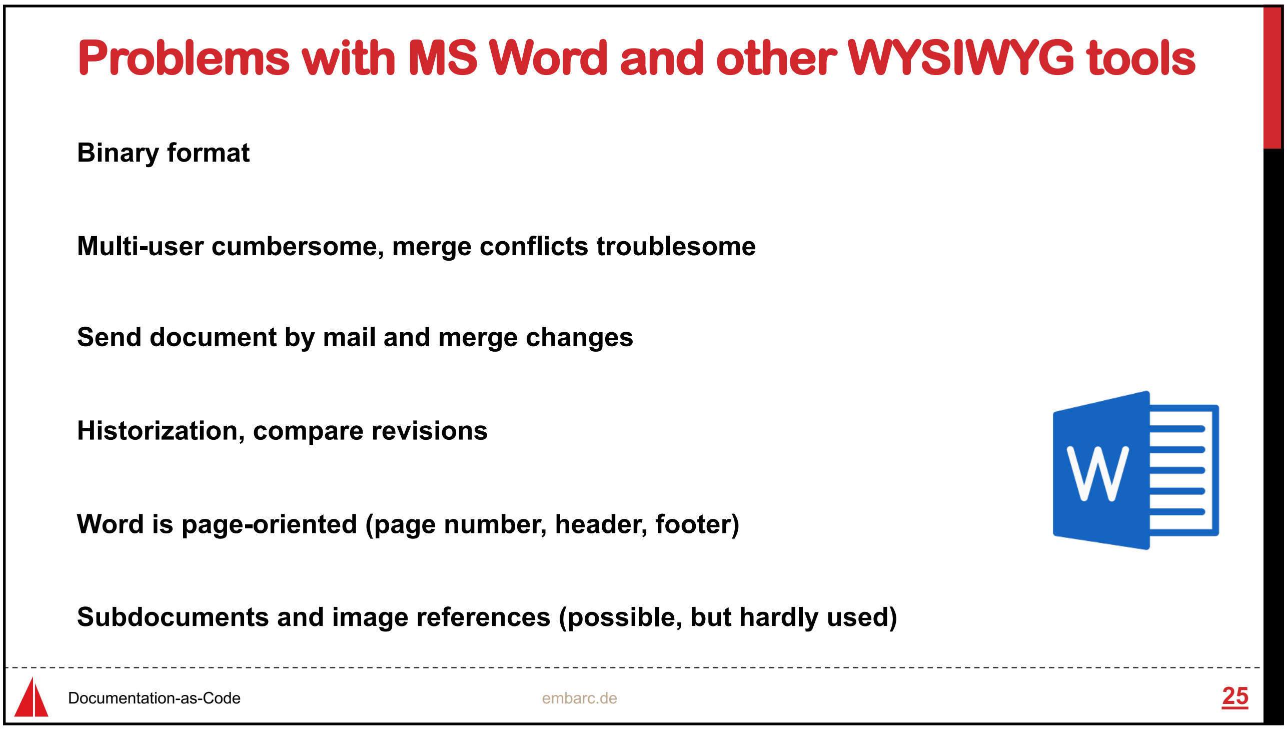
1 Lightweight text formats, save in code repos

0 Traditional documentation

The main thing, you don't do it with Word!

Spoiler: condoms are made of Latex

Ich mach's mit Latex.
Kondome schützen.
Für eine Nacht oder mehr, mach's mit.
GIB AIDS KEINE CHANCE
STI
Sexuell übertragbare Infektionen



Problems with MS Word and other WYSIWYG tools

Binary format

Multi-user cumbersome, merge conflicts troublesome

Send document by mail and merge changes

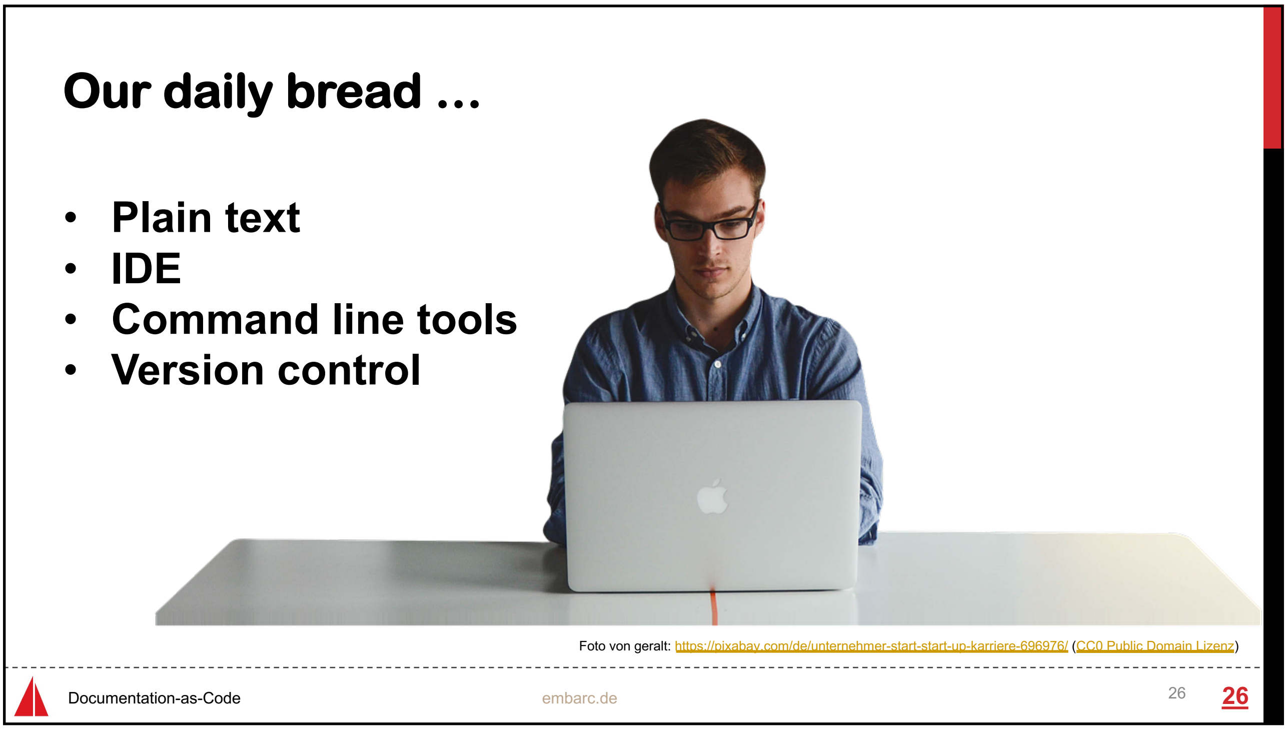
Historization, compare revisions

Word is page-oriented (page number, header, footer)

Subdocuments and image references (possible, but hardly used)

Documentation-as-Code embarc.de 25

25



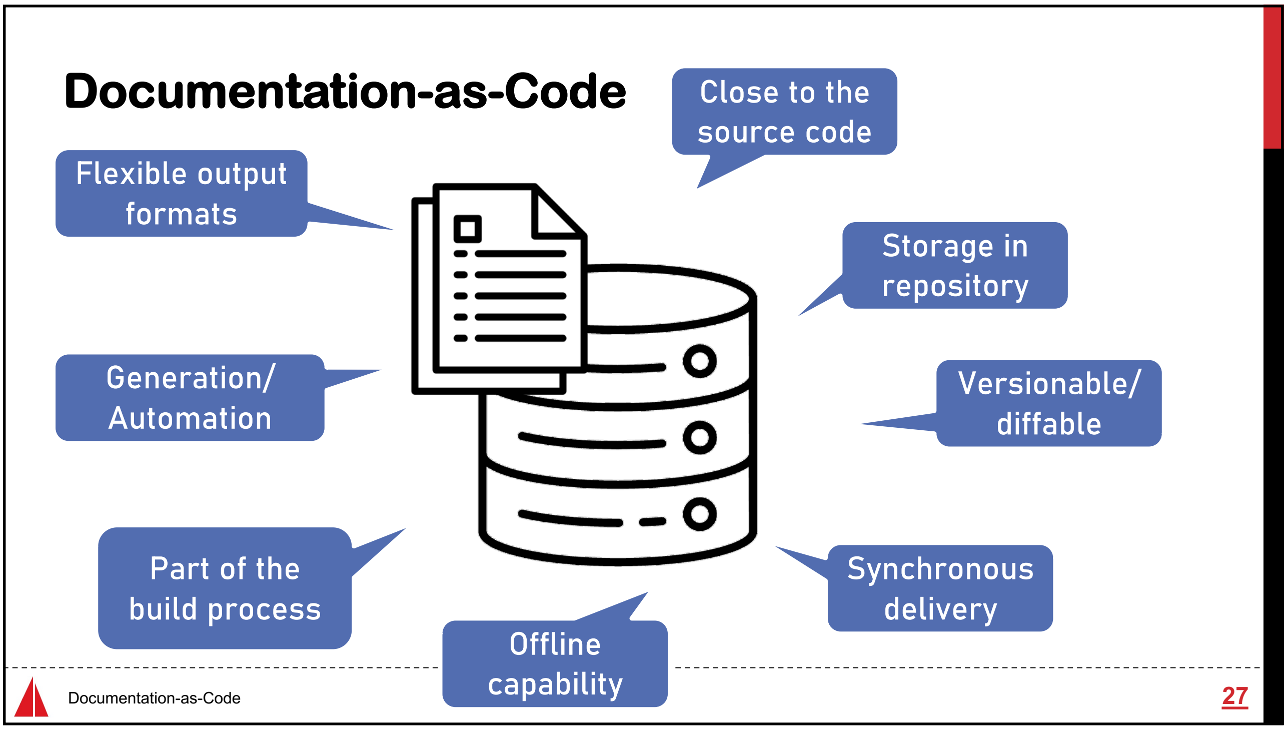
Our daily bread …

- Plain text
- IDE
- Command line tools
- Version control

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

Documentation-as-Code embarc.de 26 26

26



Documentation-as-Code

Close to the source code

Flexible output formats

Storage in repository

Generation/ Automation

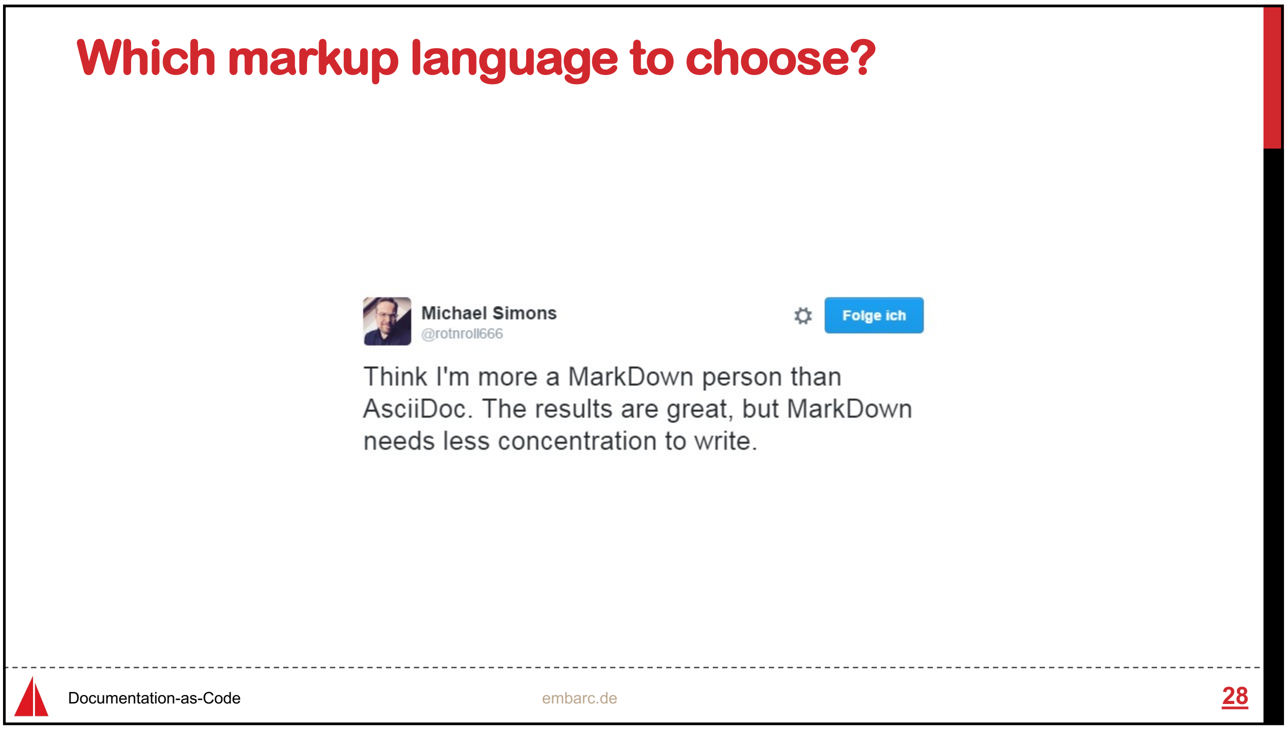
Versionable/ diffable

Part of the build process

Synchronous delivery

Offline capability

27

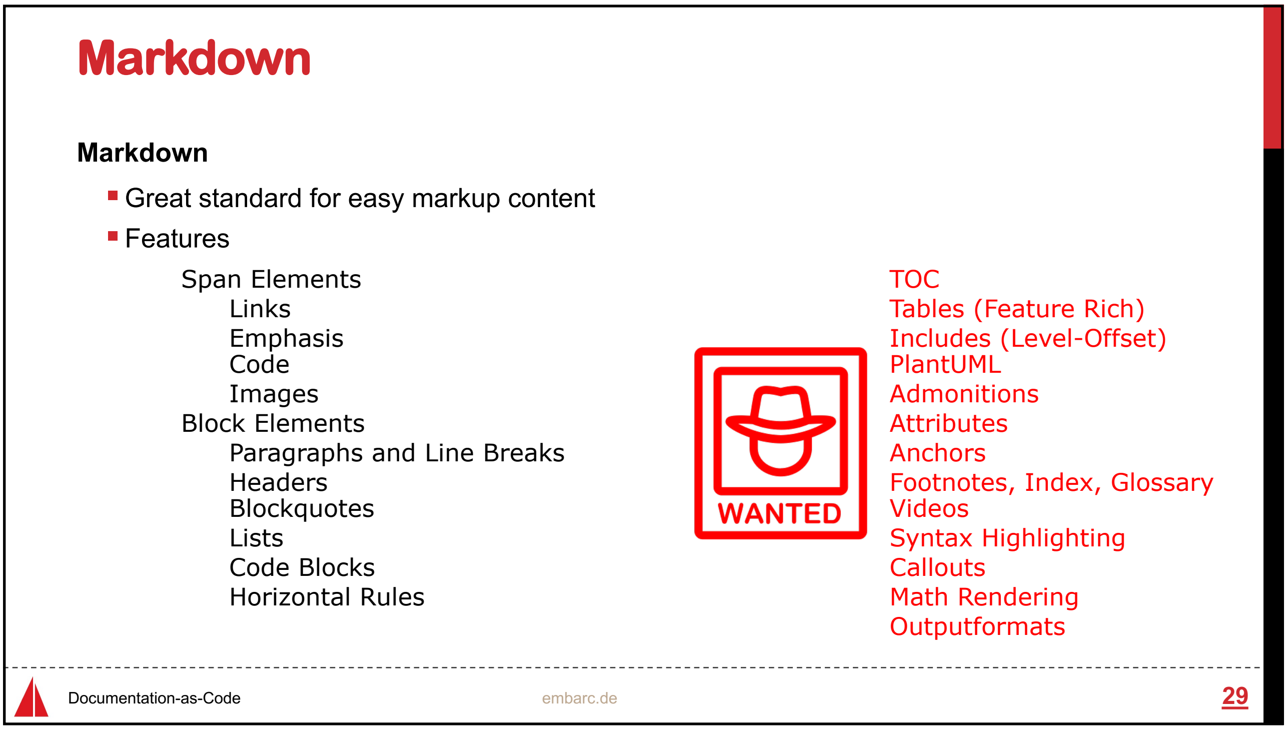


Which markup language to choose?

Michael Simons
@rotnroll666

Think I'm more a MarkDown person than AsciiDoc. The results are great, but MarkDown needs less concentration to write.

28



Markdown

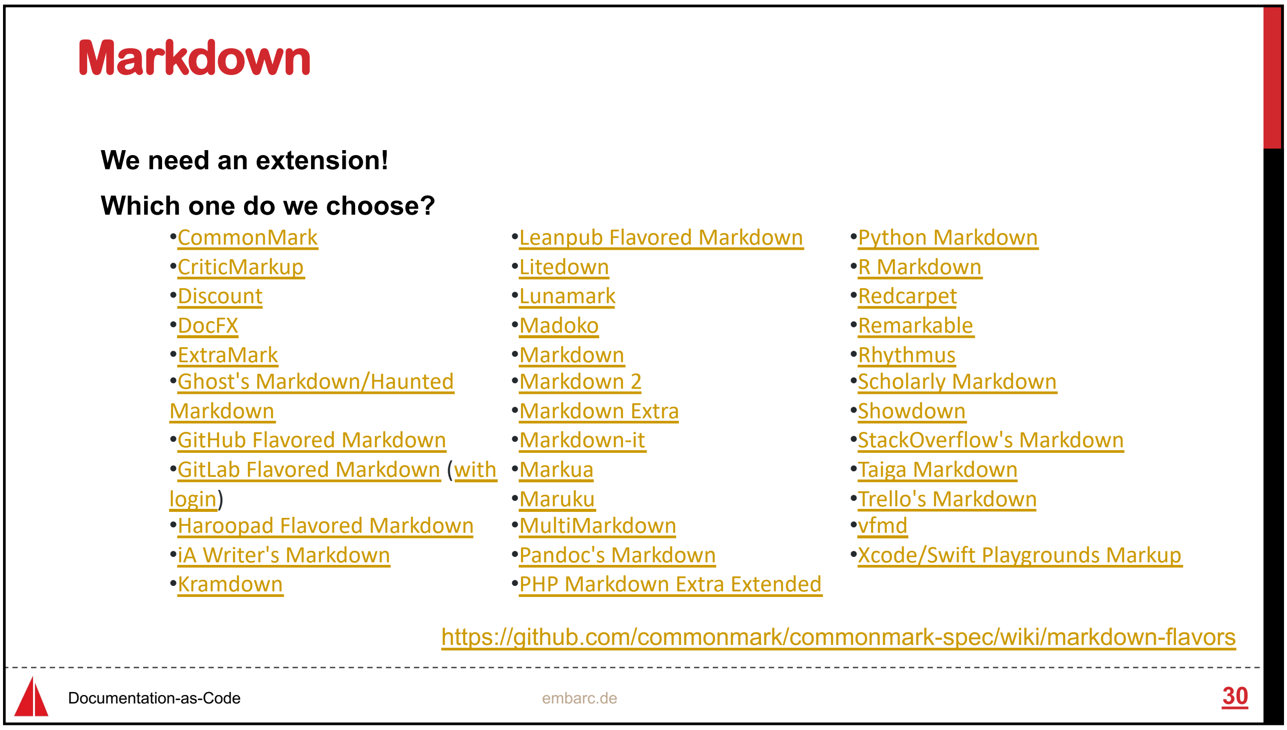
Markdown

- Great standard for easy markup content
- Features

Span Elements
Links
Emphasis
Code
Images
Block Elements
Paragraphs and Line Breaks
Headers
Blockquotes
Lists
Code Blocks
Horizontal Rules

WANTED

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



Markdown

We need an extension!

Which one do we choose?

- •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

We need an extension!

Which one do we choose?

Will our toolchain still work then?

Is there an editor preview?

AsciiDoc / Asciidoctor

powerful syntax for technical documentation

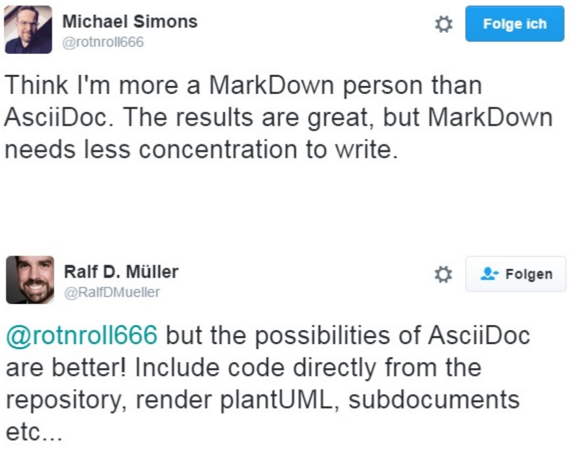
written in Ruby

transpiled to JavaScript with Opal

wrapped on the JVM with jRuby

=> No dialects!

Markdown or AsciiDoc?



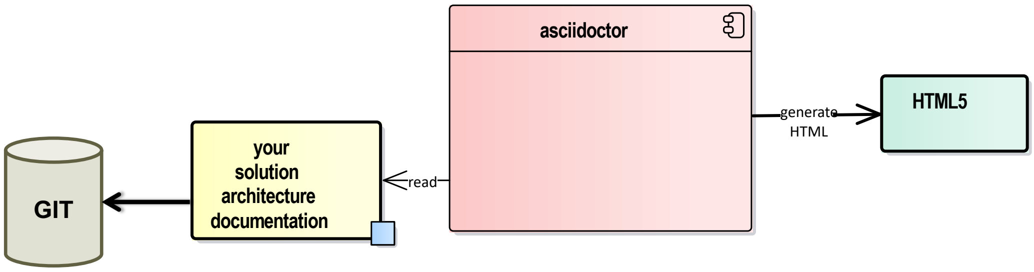
Michael Simons
@rotnroll666

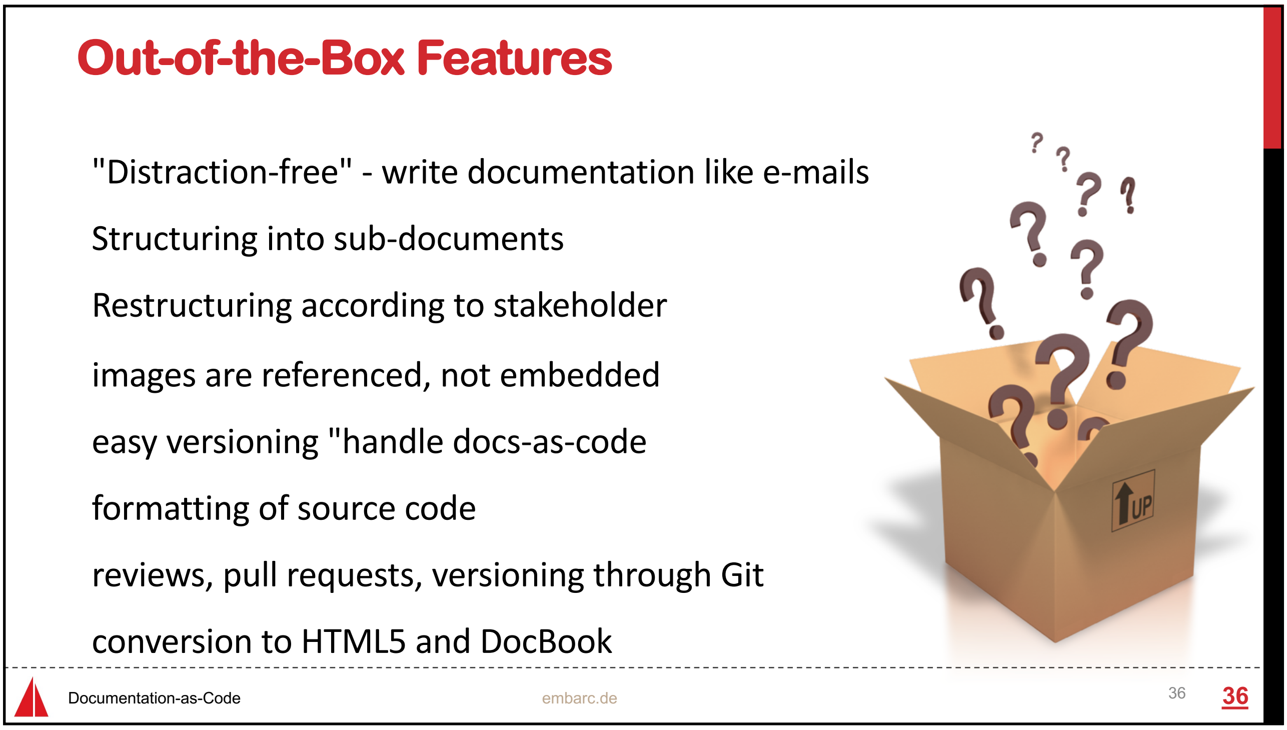
Think I'm more a MarkDown person than AsciiDoc. The results are great, but MarkDown needs less concentration to write.

Ralf D. Müller
@RalfDMueller

@rotnroll666 but the possibilities of AsciiDoc are better! Include code directly from the repository, render plantUML, subdocuments etc...

Basic Docs-as-Code with AsciiDoc





Out-of-the-Box Features

"Distraction-free" - write documentation like e-mails

Structuring into sub-documents

Restructuring according to stakeholder

images are referenced, not embedded

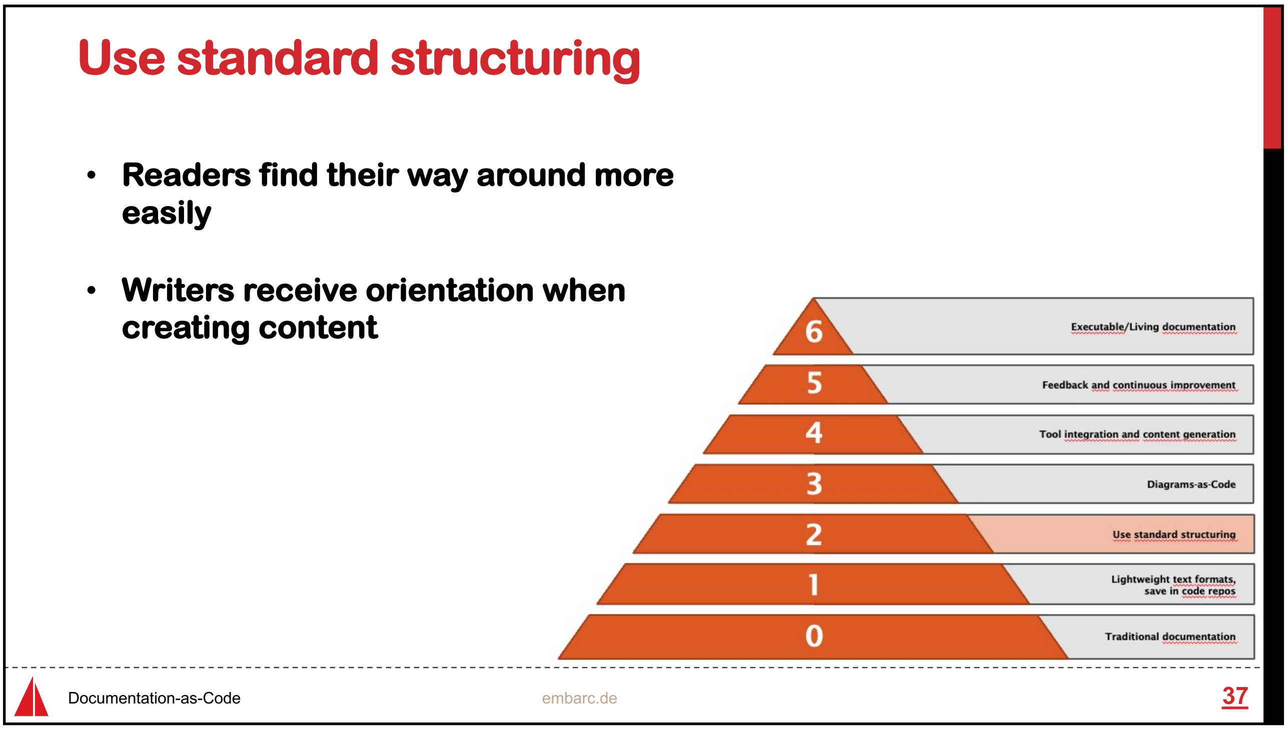
easy versioning "handle docs-as-code

formatting of source code

reviews, pull requests, versioning through Git

conversion to HTML5 and DocBook

36



Use standard structuring

- **Readers find their way around more easily**
- **Writers receive orientation when creating content**

6 Executable/Living documentation

5 Feedback and continuous improvement

4 Tool integration and content generation

3 Diagrams-as-Code

2 Use standard structuring

1 Lightweight text formats, save in code repos

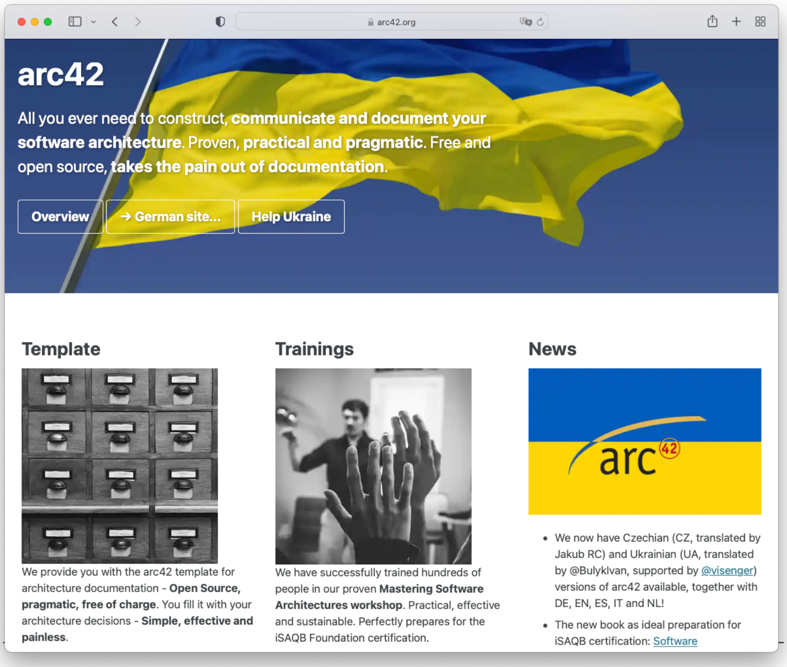
0 Traditional documentation

37

7 rules for good documentation

1. Write from the reader's point of view
2. Avoid unnecessary repetition
3. Avoid ambiguities
   a) Explain your notation
4. **Use standard structuring**
5. Record rationale for decisions
6. Keep documentation up to date but not too up to date
7. Check documentation for usability

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



arc42

All you ever need to construct, **communicate and document your software architecture**. Proven, **practical and pragmatic**. Free and open source, **takes the pain out of documentation**.

Template

We provide you with the arc42 template for architecture documentation - **Open Source, pragmatic, free of charge**. You fill it with your architecture decisions - **Simple, effective and painless**.

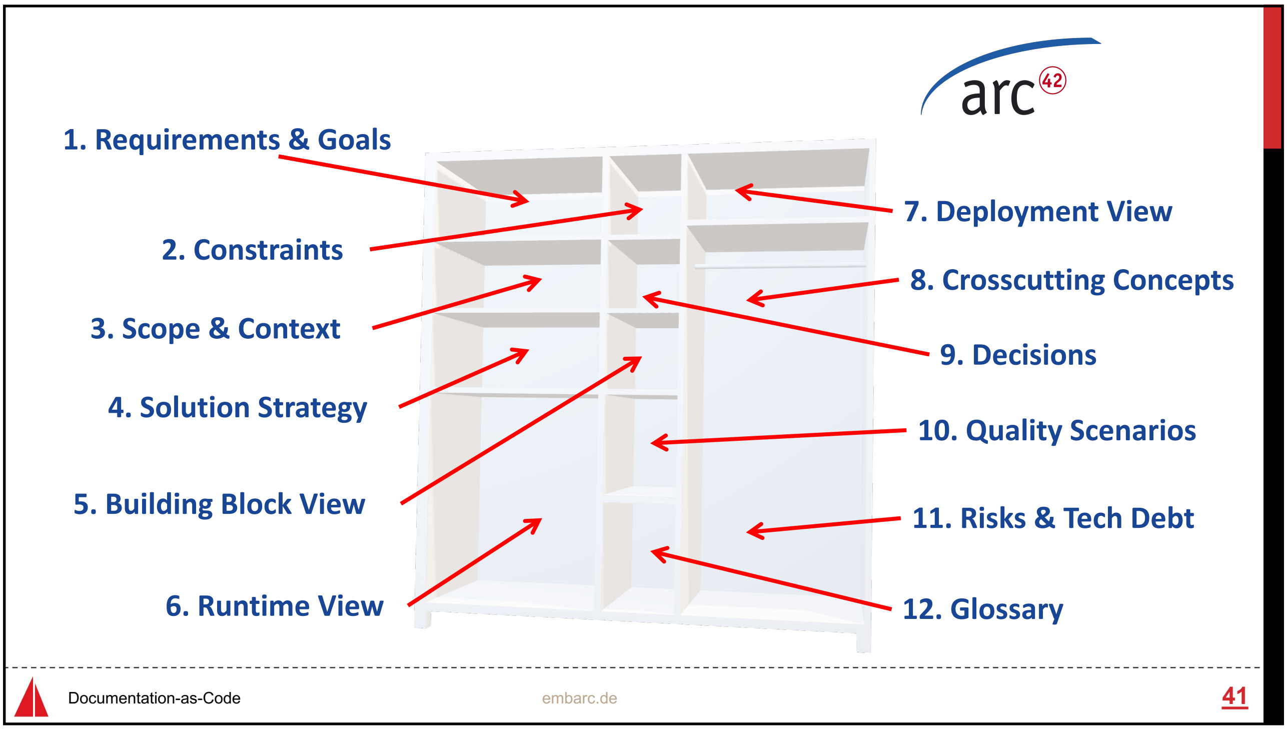
Trainings

We have successfully trained hundreds of people in our proven **Mastering Software Architectures workshop**. Practical, effective and sustainable. Perfectly prepares for the iSAQB Foundation certification.

News

- We now have Czechian (CZ, translated by Jakub RC) and Ukrainian (UA, translated by @Bulyklvan, supported by @visenger) versions of arc42 available, together with DE, EN, ES, IT and NL!
- The new book as ideal preparation for iSAQB certification: Software

https://arc42.org



arc42

1. Requirements & Goals

2. Constraints

3. Scope & Context

4. Solution Strategy

5. Building Block View

6. Runtime View

7. Deployment View

8. Crosscutting Concepts

9. Decisions

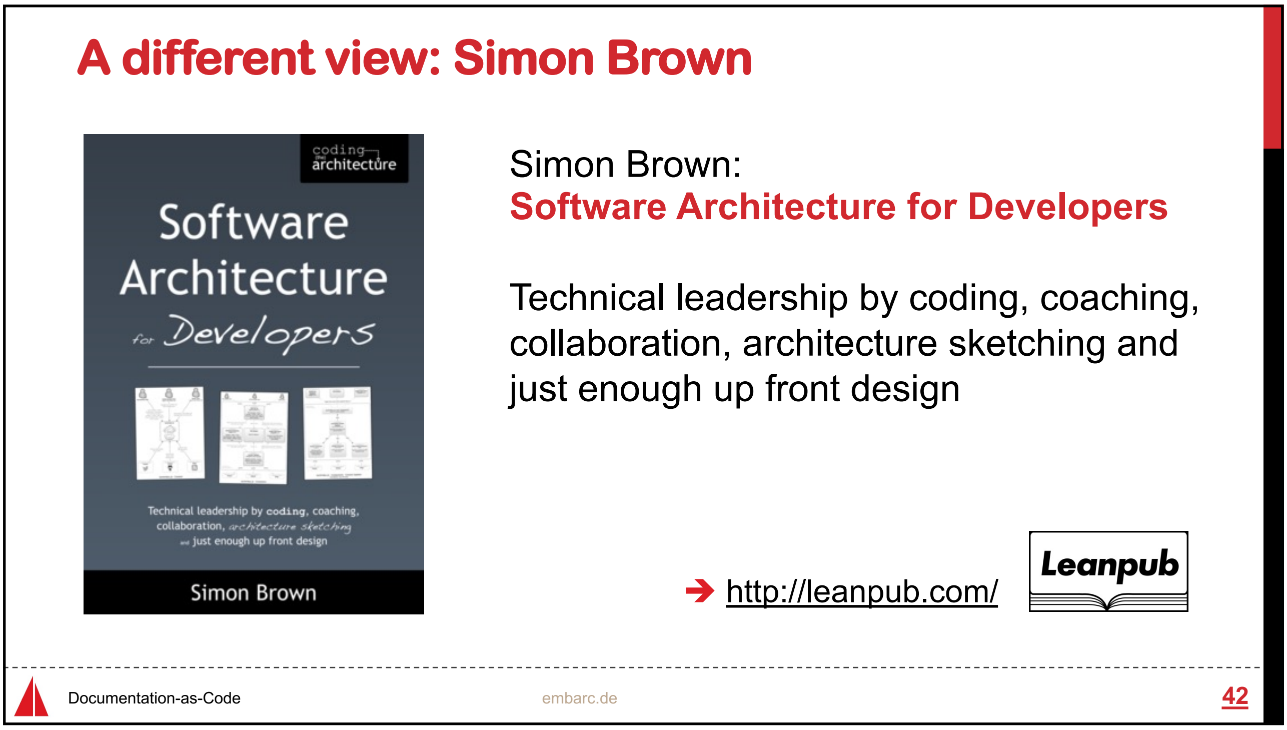
10. Quality Scenarios

11. Risks & Tech Debt

12. Glossary

Documentation-as-Code embarc.de 41

41



A different view: Simon Brown

coding the architecture
Software Architecture for Developers
Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design
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

Documentation-as-Code embarc.de 42

42

A different view: Simon Brown

Software Architecture for Developers

Volume 2

Visualise, document and explore your software architecture

Simon Brown

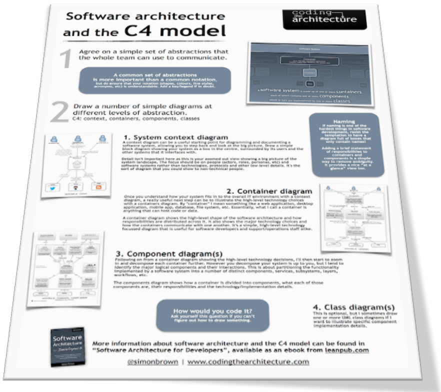
Simon Brown:
Software Architecture for Developers, Volume 2

Visualize, document and explore your software architecture

➔ http://leanpub.com/

Leanpub

Simon Brown: C4 Model



Software architecture and the C4 model

1 Agree on a simple set of abstractions that the whole team can use to communicate.

A common set of abstractions is more important than a common notation, but do ensure that your notation (shapes, colours, line styles, acronyms, etc) is understandable. Add a key/legend if in doubt.

A software system is made up of one or more containers, each of which contains one or more components, which in turn are implemented by one or more classes.

2 Draw a number of simple diagrams at different levels of abstraction.
C4: context, containers, components, classes

Naming
If naming is one of the hardest things in software development, resist the temptation to have a diagram full of boxes that only contain names!

Adding a brief statement of responsibilities to containers and components is a simple way to remove ambiguity. It provides a nice "at a glance" view too.

1. System context diagram

A context diagram can be a useful starting point for diagramming and documenting a software system, allowing you to step back and look at the big picture. Draw a simple block diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interfaces with.

Detail isn't important here as this is your zoomed out view showing a big picture of the system landscape. The focus should be on people (actors, roles, personas, etc) and software systems rather than technologies, protocols and other low-level details. It's the sort of diagram that you could show to non-technical people.

2. Container diagram

Once you understand how your system fits in to the overall IT environment with a context diagram, a really useful next step can be to illustrate the high-level technology choices with a containers diagram. By "container" I mean something like a web application, desktop application, mobile app, database, file system, etc. Essentially, what I call a container is anything that can host code or data.

A container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another. It's a simple, high-level technology focussed diagram that is useful for software developers and support/operations staff alike.

3. Component diagram(s)

Following on from a container diagram showing the high-level technology decisions, I'll then start to zoom in and decompose each container further. However you decompose your system is up to you, but I tend to identify the major logical components and their interactions. This is about partitioning the functionality implemented by a software system into a number of distinct components, services, subsystems, layers, workflows, etc.

The components diagram shows how a container is divided into components, what each of those components are, their responsibilities and the technology/implementation details.

How would you code it?
Ask yourself this question if you can't figure out how to draw something.

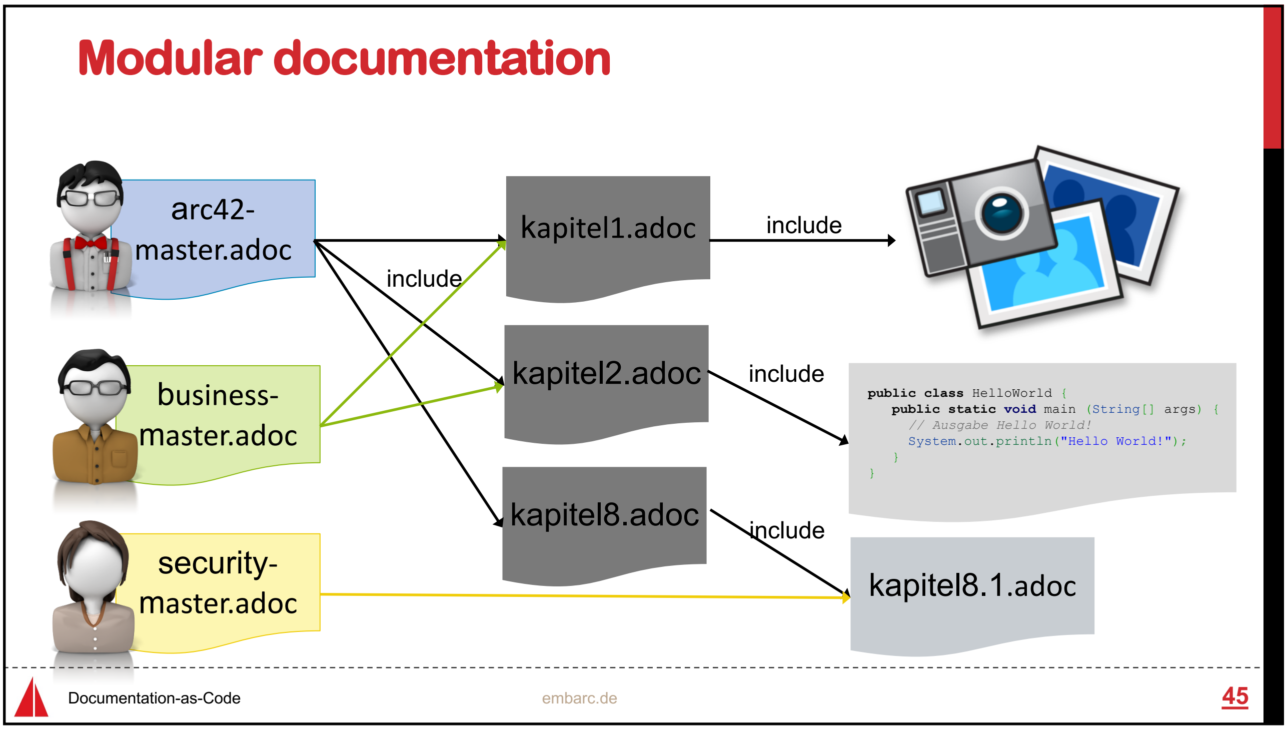
4. Class diagram(s)
This is optional, but I sometimes draw one or more UML class diagrams if I want to illustrate specific component implementation details.

More information about software architecture and the C4 model can be found in "Software Architecture for Developers", available as an ebook from leanpub.com

@simonbrown | www.codingthearchitecture.com

- 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

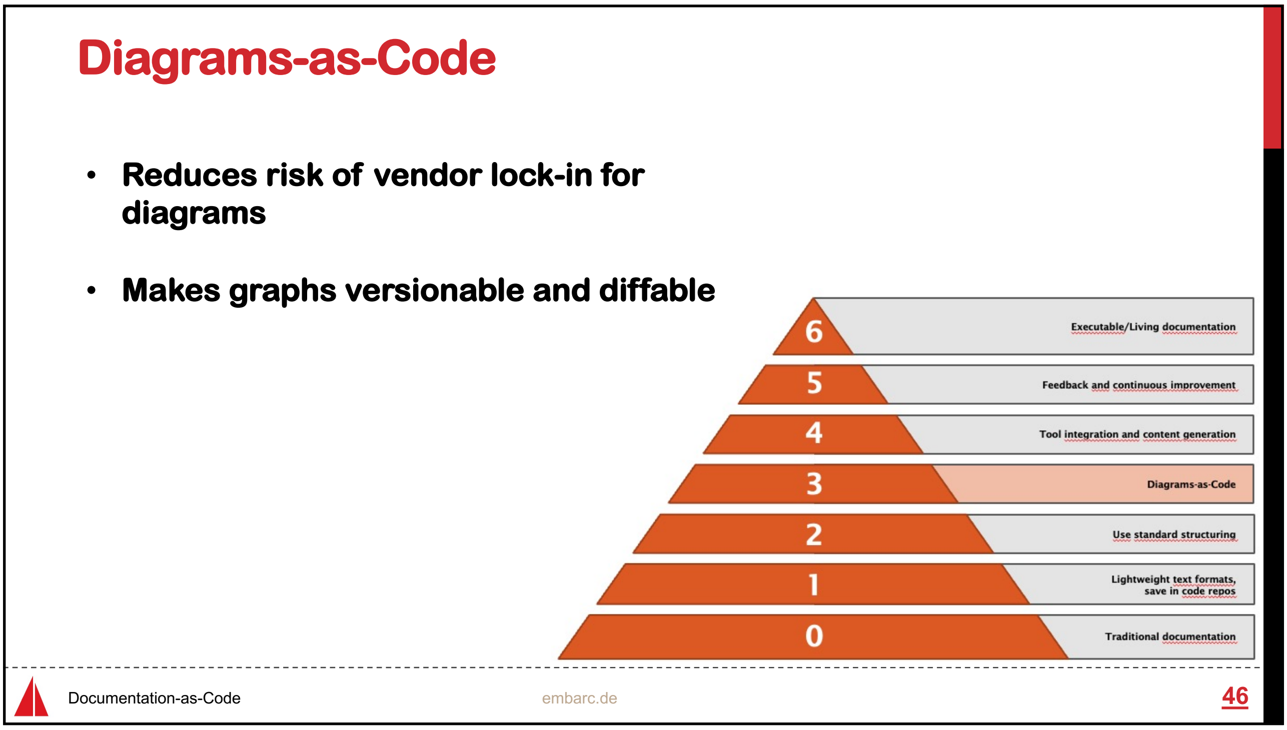


Modular documentation

```
public class HelloWorld {
   public static void main (String[] args) {
      // Ausgabe Hello World!
      System.out.println("Hello World!");
   }
}
```

Documentation-as-Code embarc.de 45

45



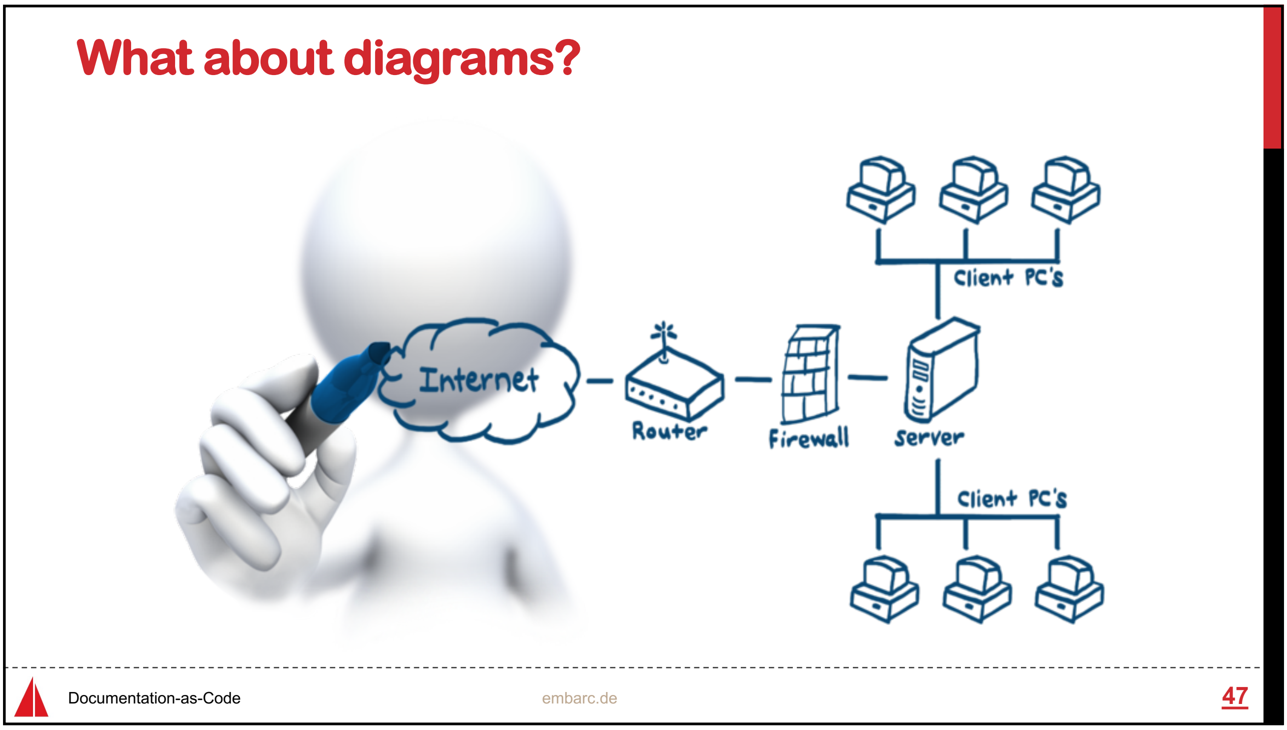
Diagrams-as-Code

- **Reduces risk of vendor lock-in for diagrams**
- **Makes graphs versionable and diffable**

6 Executable/Living documentation
5 Feedback and continuous improvement
4 Tool integration and content generation
3 Diagrams-as-Code
2 Use standard structuring
1 Lightweight text formats, save in code repos
0 Traditional documentation

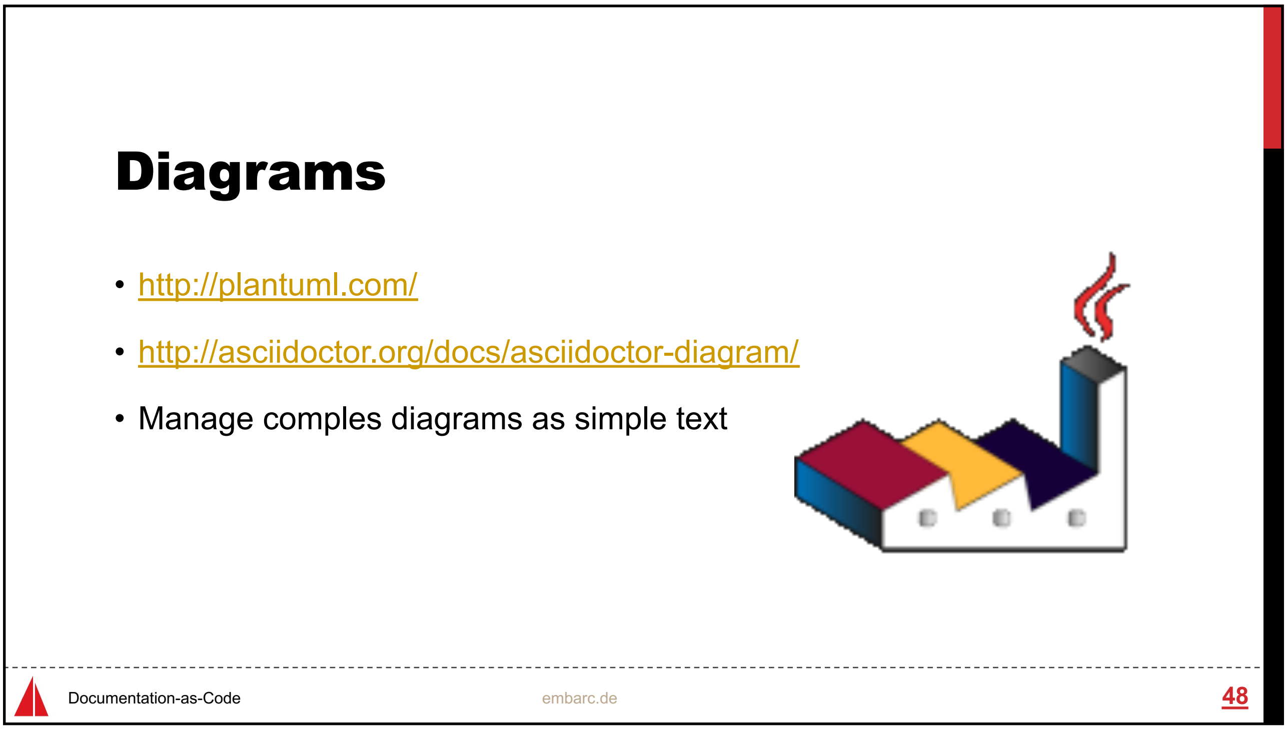
Documentation-as-Code embarc.de 46

46



What about diagrams?

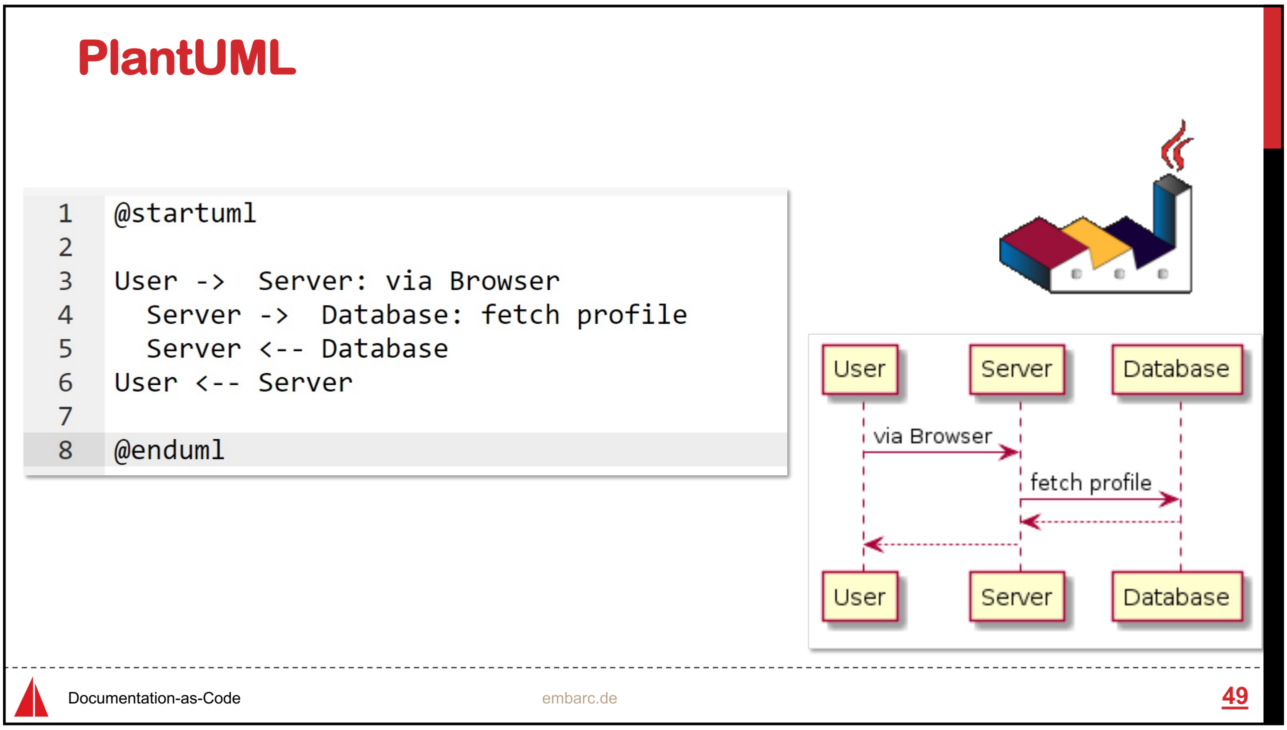
47



Diagrams

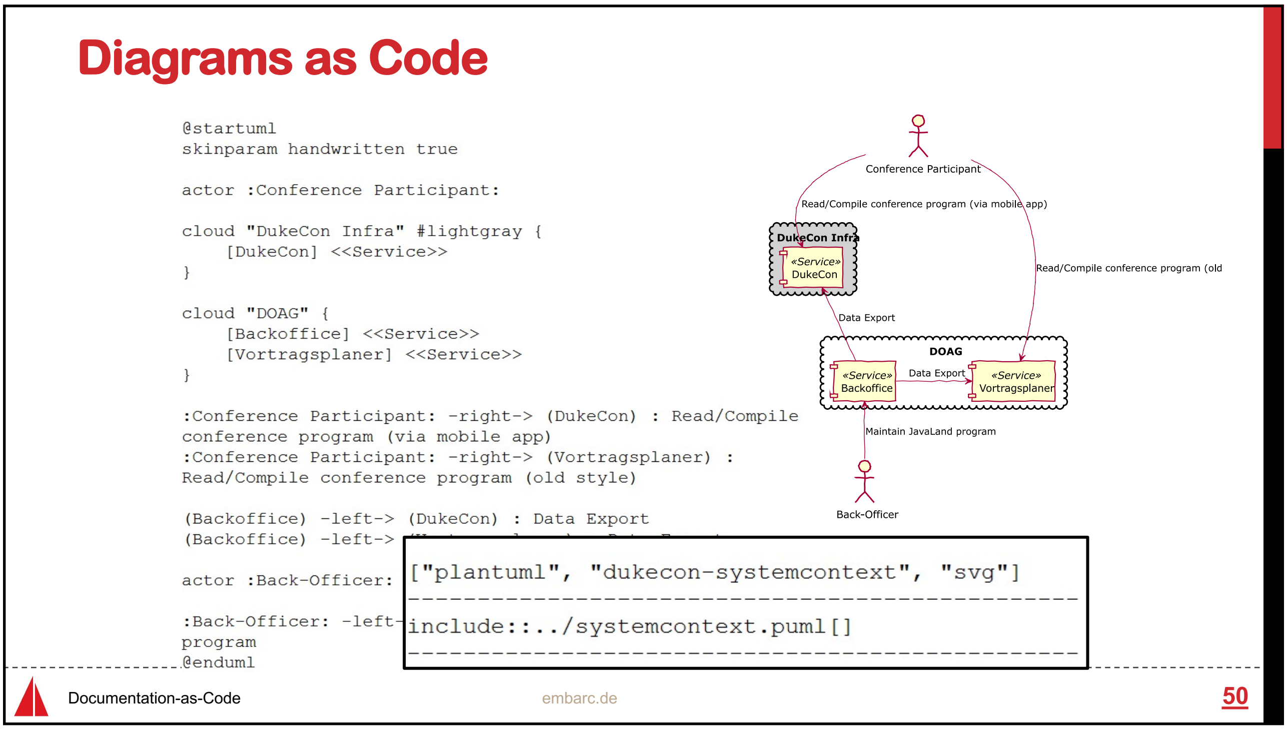
- http://plantuml.com/
- http://asciidoctor.org/docs/asciidoctor-diagram/
- Manage comples diagrams as simple text

48



PlantUML

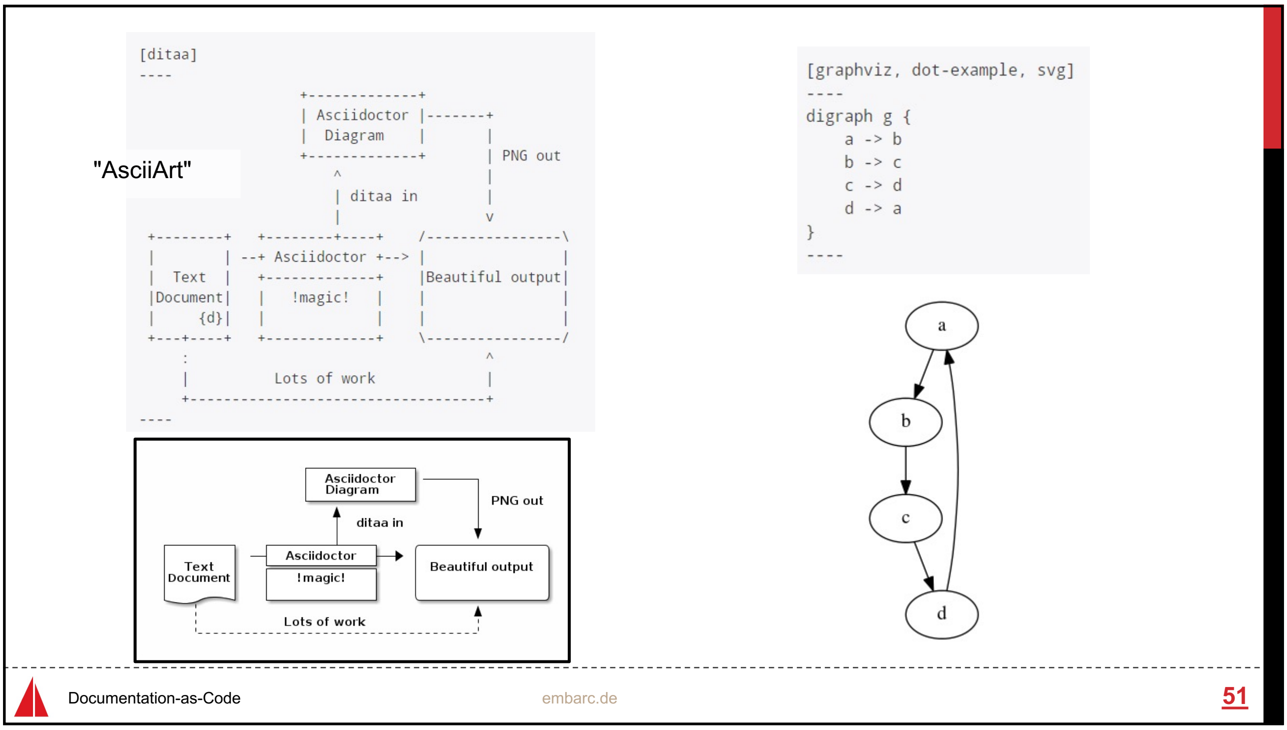
```
@startuml

User ->  Server: via Browser
   Server ->  Database: fetch profile
   Server <-- Database
User <-- Server

@enduml
```



Diagrams as Code

```
@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[]
---------------------------------------------
```

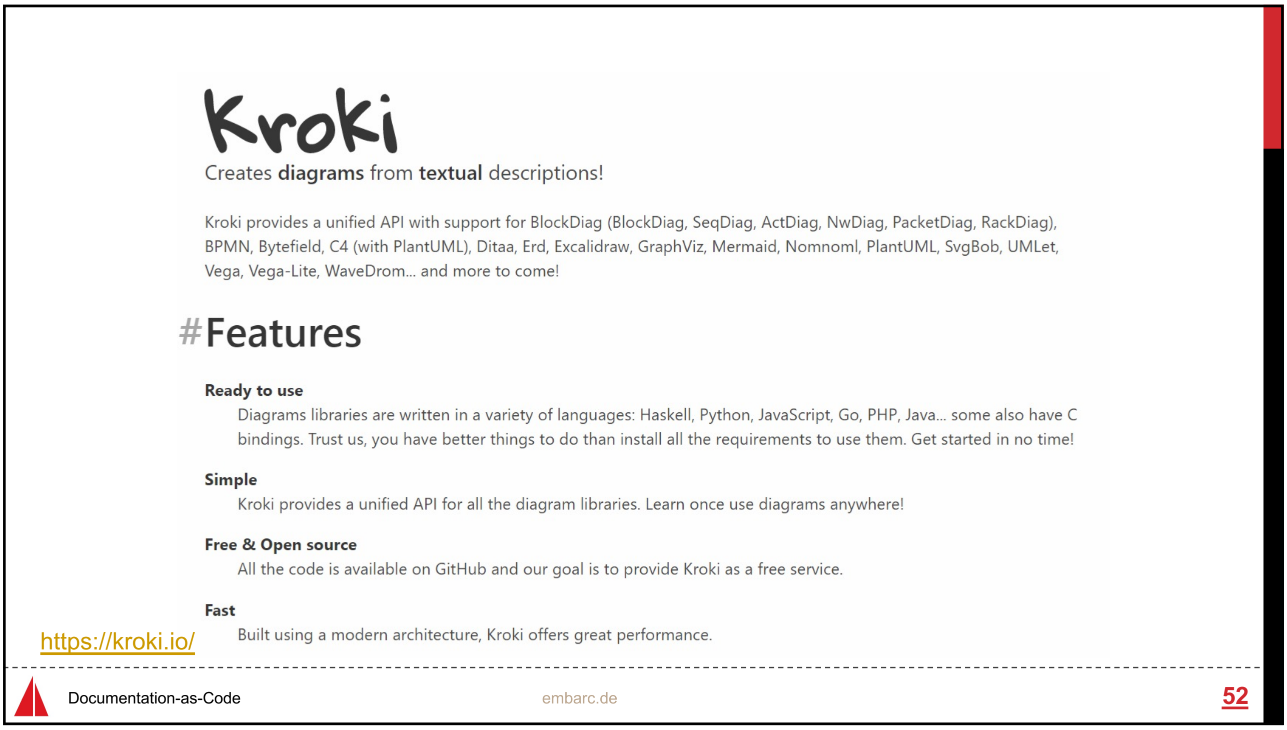


"AsciiArt"

```
[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
}
----
```

51



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!

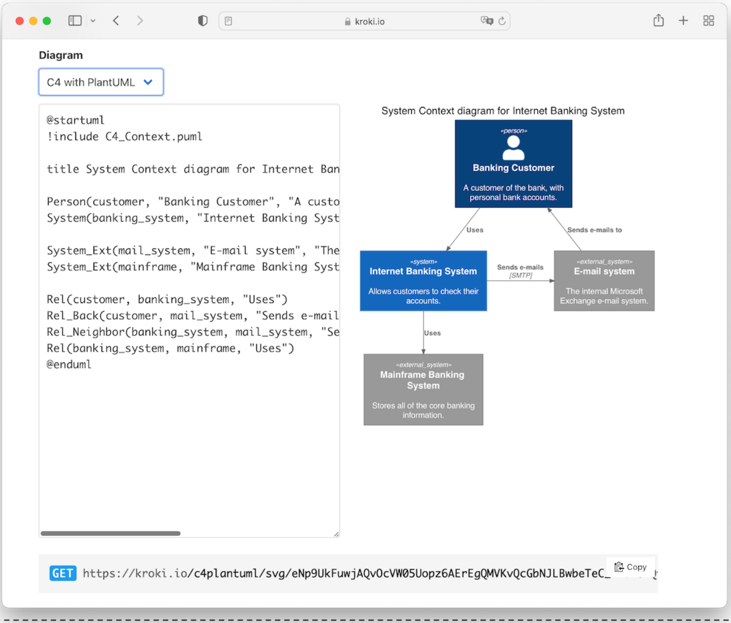
Simple
Kroki provides a unified API for all the diagram libraries. Learn once use diagrams anywhere!

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/

52



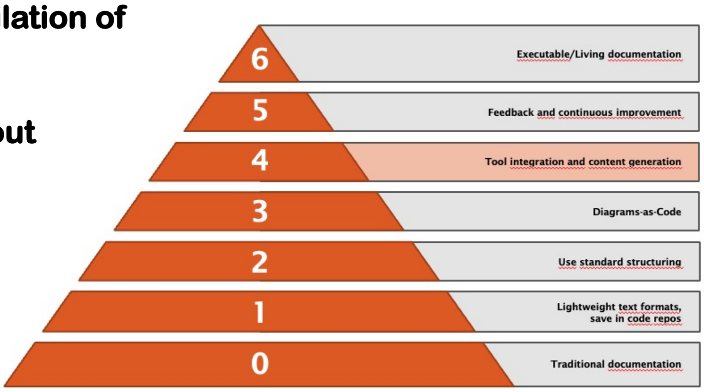
```
@startuml
!include C4_Context.puml

title System Context diagram for Internet Ban

Person(customer, "Banking Customer", "A custo
System(banking_system, "Internet Banking Syst

System_Ext(mail_system, "E-mail system", "The
System_Ext(mainframe, "Mainframe Banking Syst

Rel(customer, banking_system, "Uses")
Rel_Back(customer, mail_system, "Sends e-mail
Rel_Neighbor(banking_system, mail_system, "Se
Rel(banking_system, mainframe, "Uses")
@enduml
```

System Context diagram for Internet Banking System
«person»
Banking Customer
A customer of the bank, with personal bank accounts.
Uses
Sends e-mails to
«system»
Internet Banking System
Allows customers to check their accounts.
Sends e-mails
[SMTP]
«external_system»
E-mail system
The internal Microsoft Exchange e-mail system.
Uses
«external_system»
Mainframe Banking System
Stores all of the core banking information.

```
GET https://kroki.io/c4plantuml/svg/eNp9UkFuwjAQvOcVW05Uopz6AErEgQMVKvQcGbNJLBwbeTeC_
```

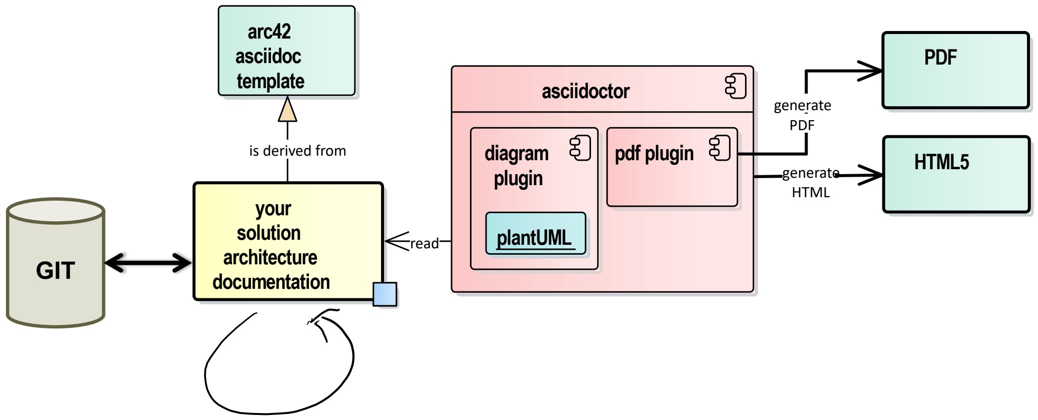
Tool integration und content generation

- **automated creation in the build process**
- **target group oriented compilation of results**
- **use of developer tools without context switching**
- **easy import and export**

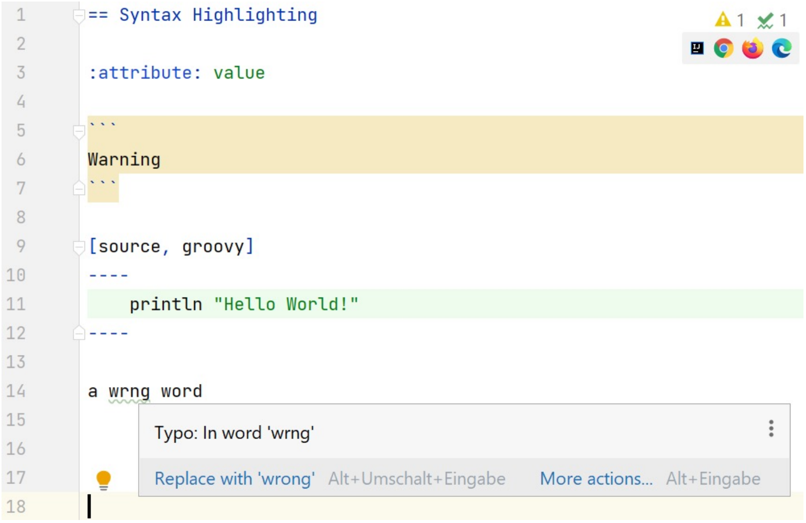


6
Executable/Living documentation
5
Feedback and continuous improvement
4
Tool integration and content generation
3
Diagrams-as-Code
2
Use standard structuring
1
Lightweight text formats, save in code repos
0
Traditional documentation

Simple workflow with AsciiDoctor



Docs-as-Code in your IDE

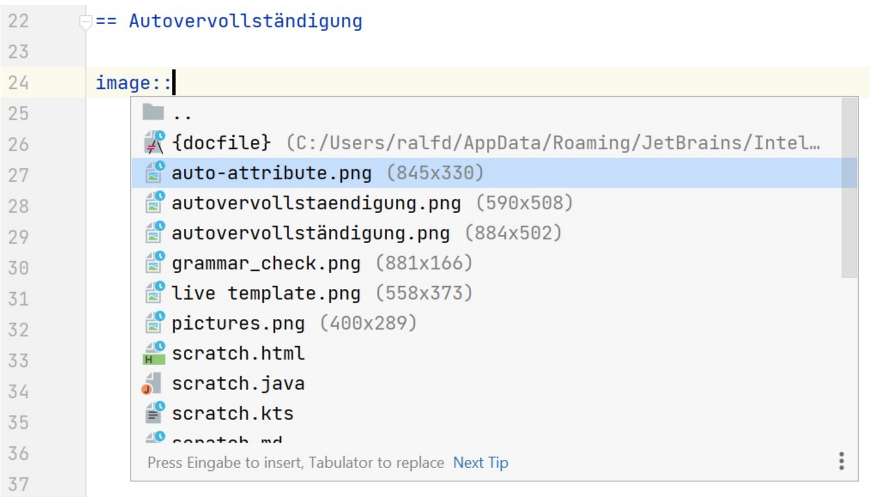


```
== Syntax Highlighting

:attribute: value

```
Warning
```

[source, groovy]
----
    println "Hello World!"
----

a wrng word
```

Typo: In word 'wrng'

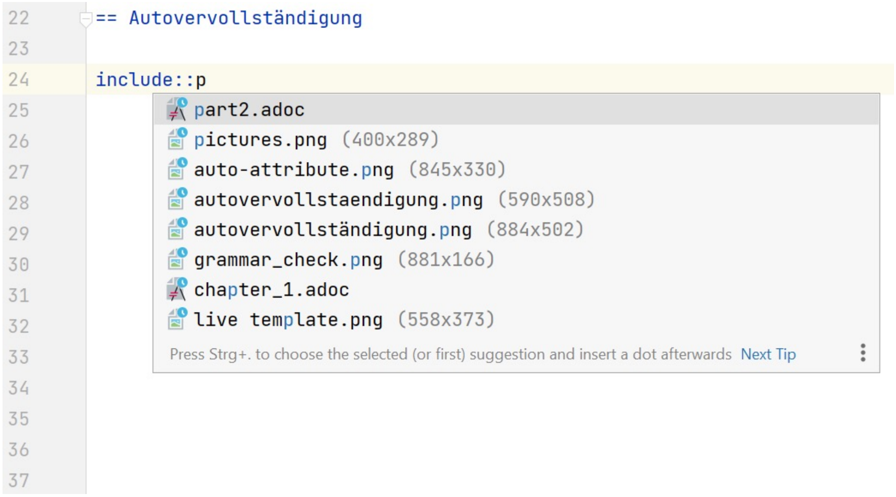
Replace with 'wrong' Alt+Umschalt+Eingabe
More actions... Alt+Eingabe
1
1

Docs-as-Code in your IDE



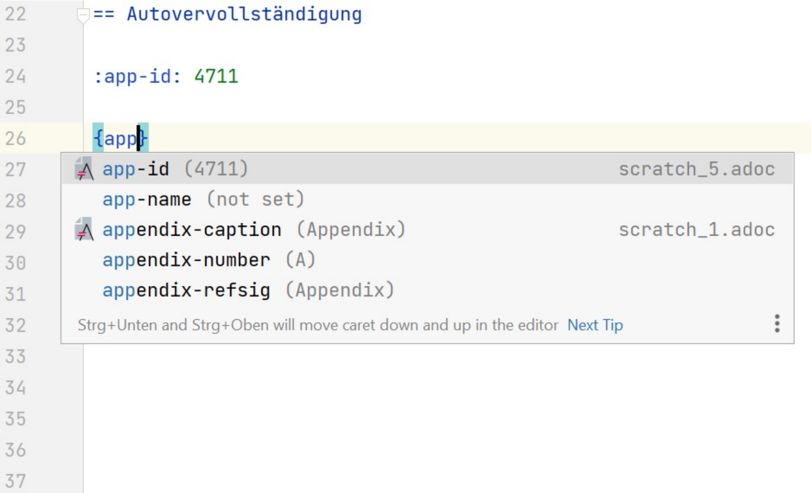
```
== Autovervollständigung

image::
  ..
  {docfile} (C:/Users/ralfd/AppData/Roaming/JetBrains/Intel…
  auto-attribute.png (845x330)
  autovervollstaendigung.png (590x508)
  autovervollständigung.png (884x502)
  grammar_check.png (881x166)
  live template.png (558x373)
  pictures.png (400x289)
  scratch.html
  scratch.java
  scratch.kts
  scratch.md
  Press Eingabe to insert, Tabulator to replace  Next Tip
```

Docs-as-Code in your IDE



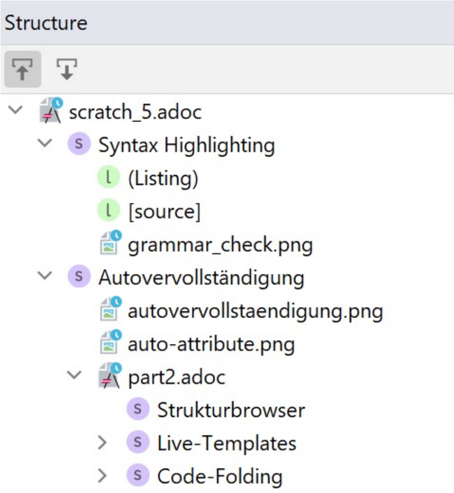
```
== Autovervollständigung

include::p
  part2.adoc
  pictures.png (400x289)
  auto-attribute.png (845x330)
  autovervollstaendigung.png (590x508)
  autovervollständigung.png (884x502)
  grammar_check.png (881x166)
  chapter_1.adoc
  live template.png (558x373)
  Press Strg+. to choose the selected (or first) suggestion and insert a dot afterwards  Next Tip
```

Docs-as-Code in your IDE



```
== Autovervollständigung

:app-id: 4711

{app}
app-id (4711)                              scratch_5.adoc
app-name (not set)
appendix-caption (Appendix)                scratch_1.adoc
appendix-number (A)
appendix-refsig (Appendix)
Strg+Unten and Strg+Oben will move caret down and up in the editor  Next Tip
```

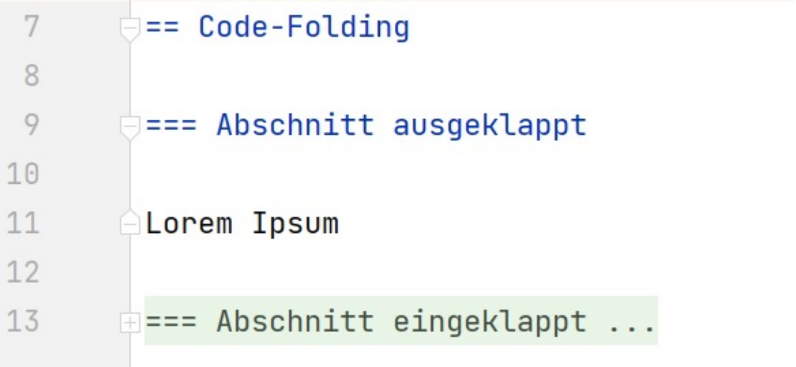
Docs-as-Code in your IDE



Structure

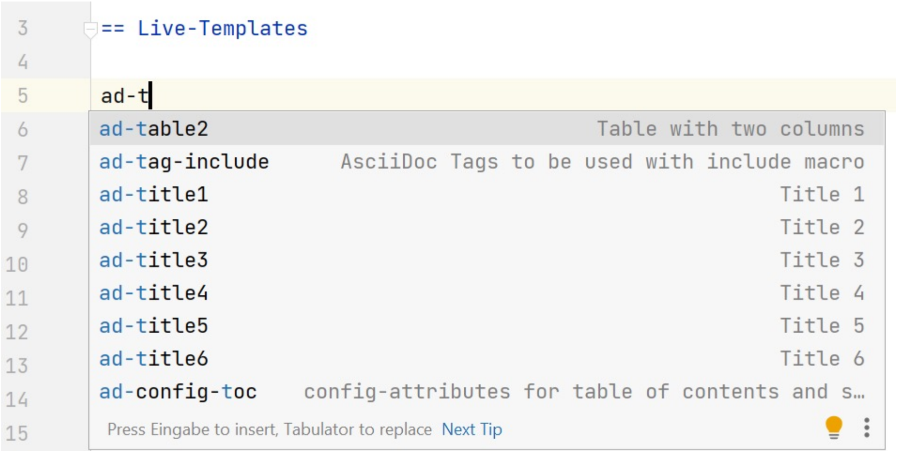
- scratch_5.adoc
  - Syntax Highlighting
    - (Listing)
    - [source]
    - grammar_check.png
  - Autovervollständigung
    - autovervollstaendigung.png
    - auto-attribute.png
    - part2.adoc
      - Strukturbrowser
      - Live-Templates
      - Code-Folding

Docs-as-Code in your IDE



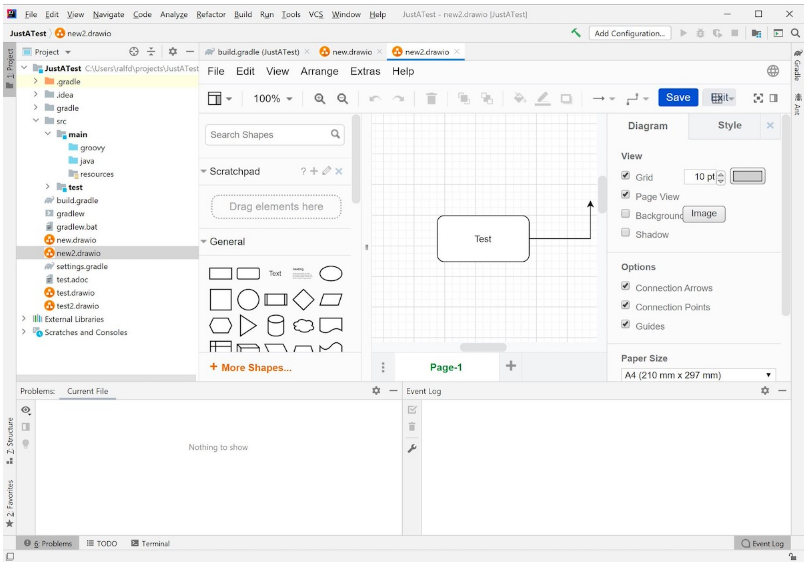
```
== Code-Folding

=== Abschnitt ausgeklappt

Lorem Ipsum

=== Abschnitt eingeklappt ...
```

Docs-as-Code in your IDE



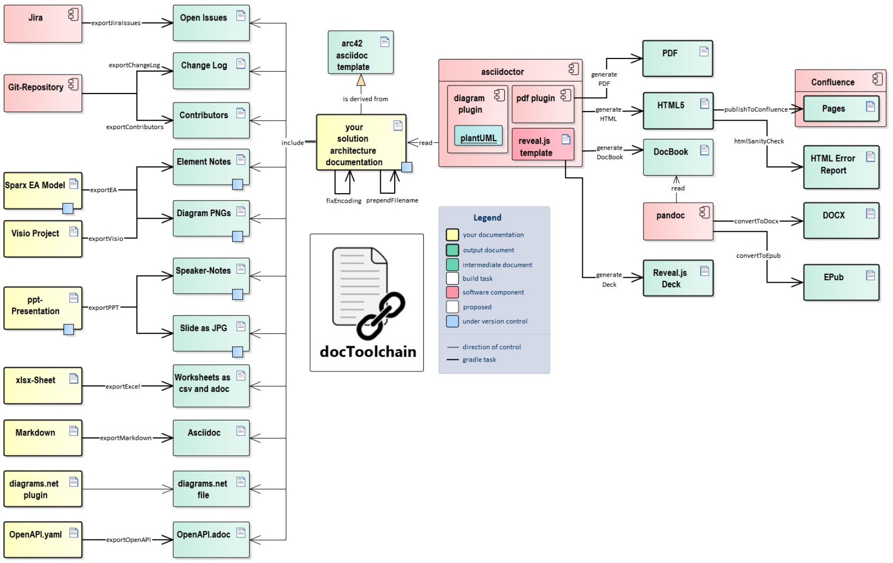
```
== Live-Templates

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

Draw.io/Diagrams.net Integration

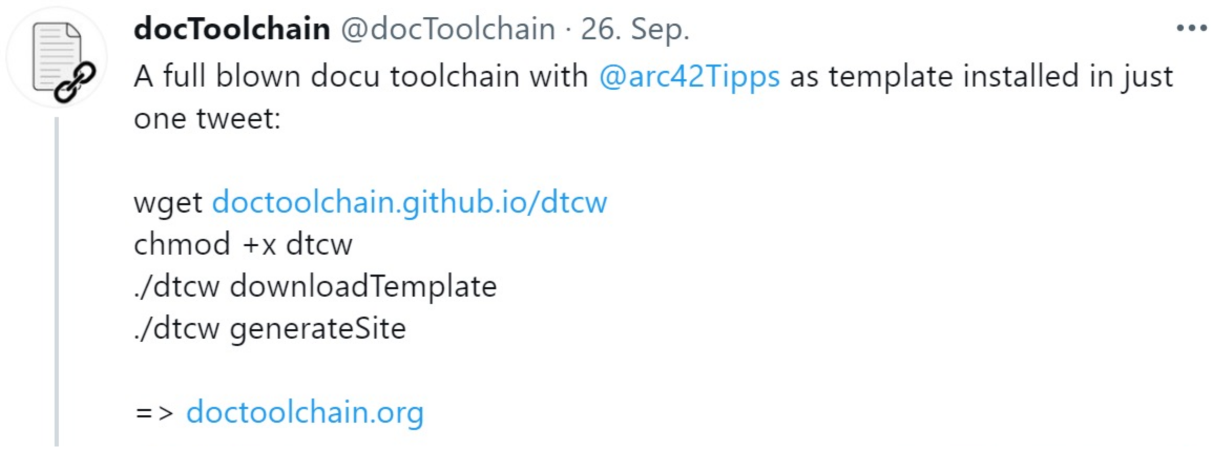


63

Tool chain: docToolchain



64



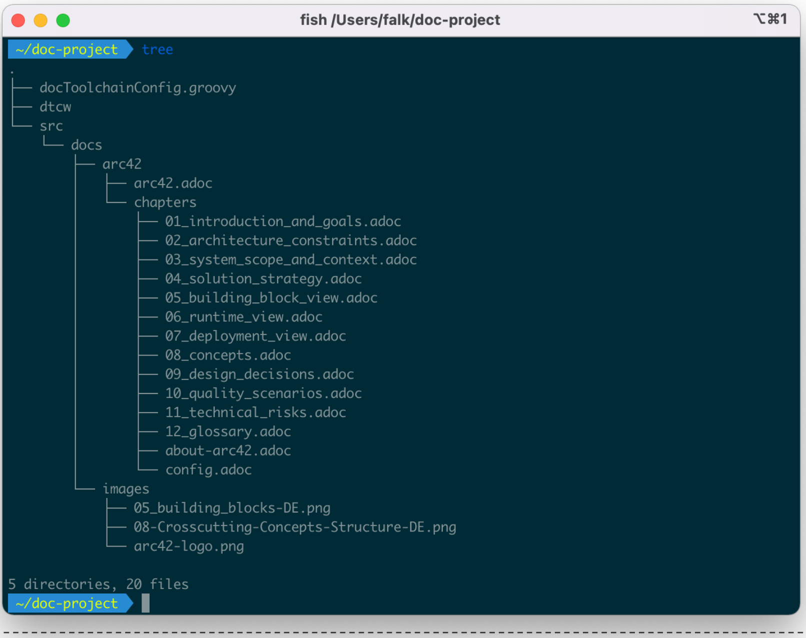
docToolchain @docToolchain · 26. Sep.

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

Documentation-as-Code embarc.de 65

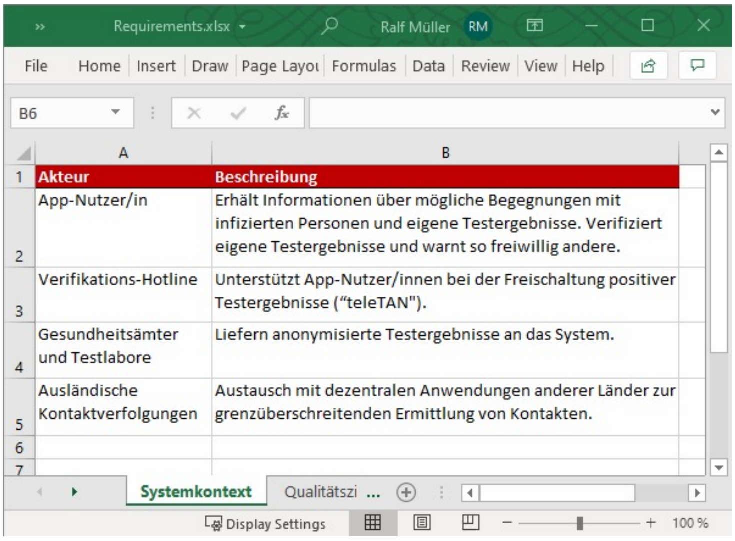


fish /Users/falk/doc-project ⌥⌘1

```
~/doc-project  tree
.
├── docToolchainConfig.groovy
├── dtcw
└── src
    └── docs
        ├── arc42
        │   ├── arc42.adoc
        │   └── chapters
        │       ├── 01_introduction_and_goals.adoc
        │       ├── 02_architecture_constraints.adoc
        │       ├── 03_system_scope_and_context.adoc
        │       ├── 04_solution_strategy.adoc
        │       ├── 05_building_block_view.adoc
        │       ├── 06_runtime_view.adoc
        │       ├── 07_deployment_view.adoc
        │       ├── 08_concepts.adoc
        │       ├── 09_design_decisions.adoc
        │       ├── 10_quality_scenarios.adoc
        │       ├── 11_technical_risks.adoc
        │       ├── 12_glossary.adoc
        │       ├── about-arc42.adoc
        │       └── config.adoc
        └── images
            ├── 05_building_blocks-DE.png
            ├── 08-Crosscutting-Concepts-Structure-DE.png
            └── arc42-logo.png

5 directories, 20 files
~/doc-project 
```

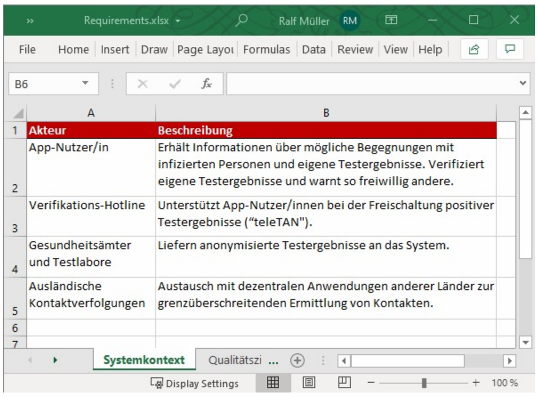
Documentation-as-Code embarc.de 66

Tabular input!?

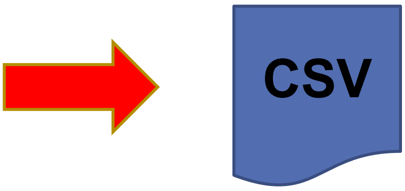


Akteur	Beschreibung
App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse ("teleTAN").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

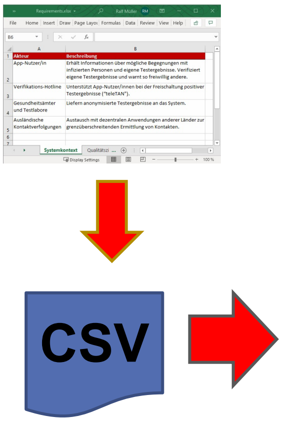
Tabular input!?



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.
Ausländische Kontaktverfolgungen	Austausch mit dezentralen Anwendungen anderer Länder zur grenzüberschreitenden Ermittlung von Kontakten.

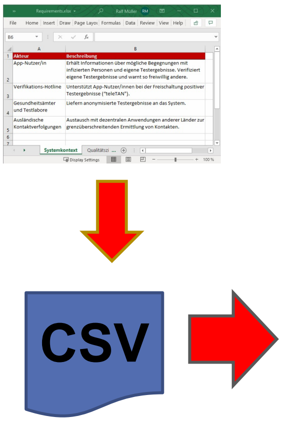


Tabular input!?



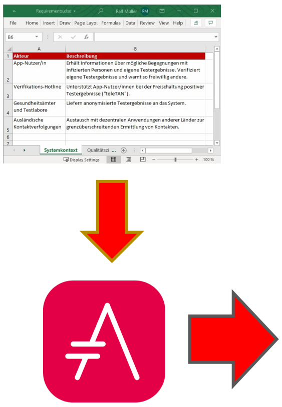
Akteur	Beschreibung
App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse (“teleTAN").
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.

Tabular input!?



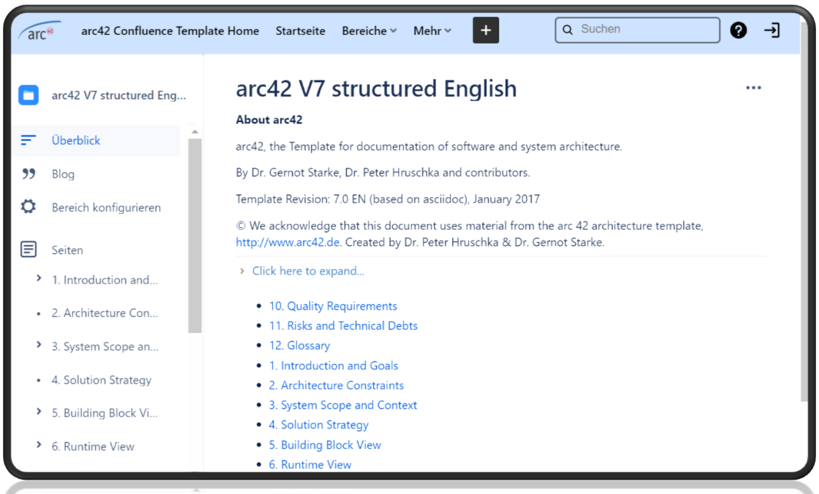
Akteur	Beschreibung
App-Nutzer/in	Erhält Informationen über mögliche Begegnungen mit infizierten Personen und eigene Testergebnisse. Verifiziert eigene Testergebnisse und warnt so freiwillig andere.
Verifikations-Hotline	Unterstützt App-Nutzer/innen bei der Freischaltung positiver Testergebnisse (“teleTAN").
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.

Tabular input!?



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.

publishToConfluence



arc42 V7 structured English

About arc42

arc42, the Template for documentation of software and system architecture.

By Dr. Gernot Starke, Dr. Peter Hruschka and contributors.

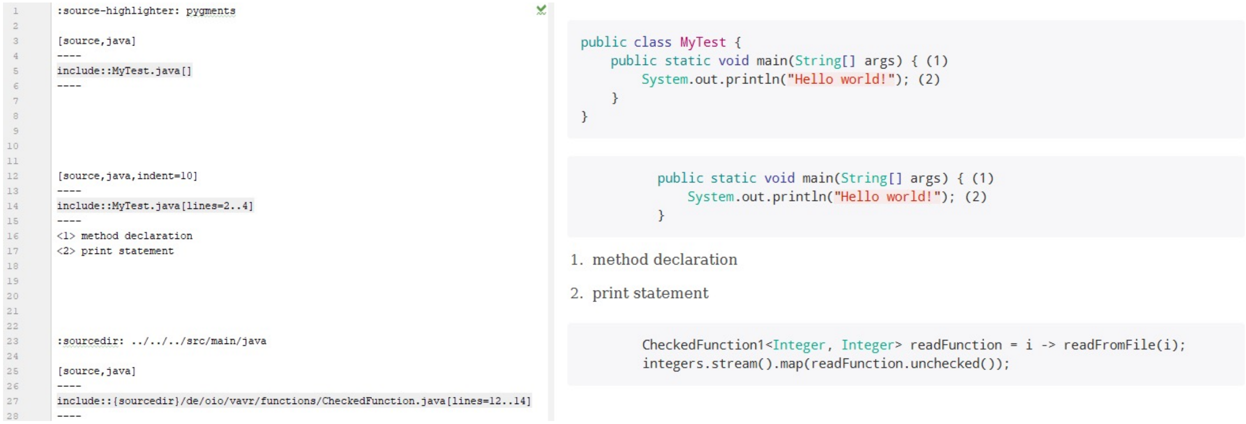
Template Revision: 7.0 EN (based on asciidoc), January 2017

© We acknowledge that this document uses material from the arc 42 architecture template, http://www.arc42.de. Created by Dr. Peter Hruschka & Dr. Gernot Starke.

› Click here to expand...

- 10. Quality Requirements
- 11. Risks and Technical Debts
- 12. Glossary
- 1. Introduction and Goals
- 2. Architecture Constraints
- 3. System Scope and Context
- 4. Solution Strategy
- 5. Building Block View
- 6. Runtime View

Embed source code



```
: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]
----
```

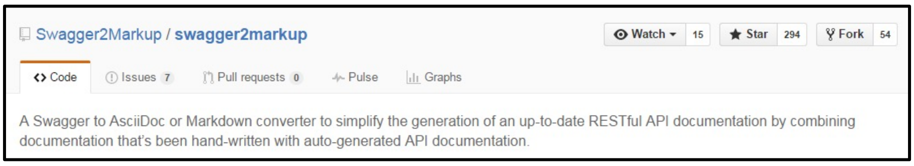
```
public class MyTest {
    public static void main(String[] args) { (1)
        System.out.println("Hello world!"); (2)
    }
}
```

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

1. method declaration
2. print statement

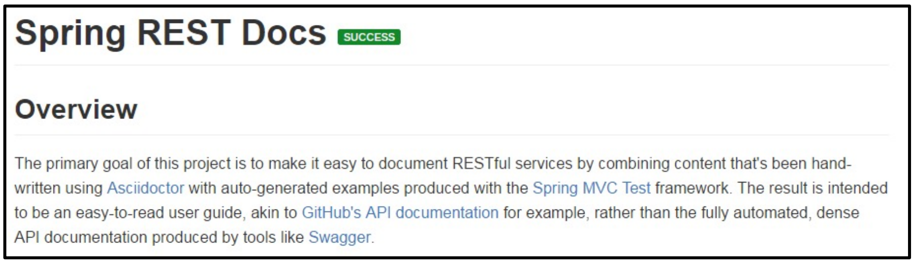
```
          CheckedFunction1<Integer, Integer> readFunction = i -> readFromFile(i);
          integers.stream().map(readFunction.unchecked());
```

Generate interface definitions/descriptions



Swagger2Markup / **swagger2markup**

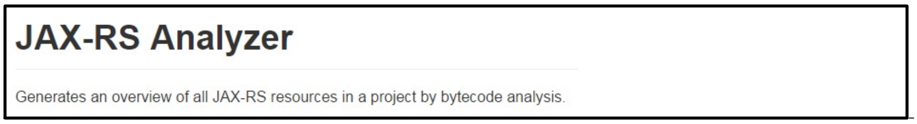
A Swagger to AsciiDoc or Markdown converter to simplify the generation of an up-to-date RESTful API documentation by combining documentation that's been hand-written with auto-generated API documentation.



Spring REST Docs SUCCESS

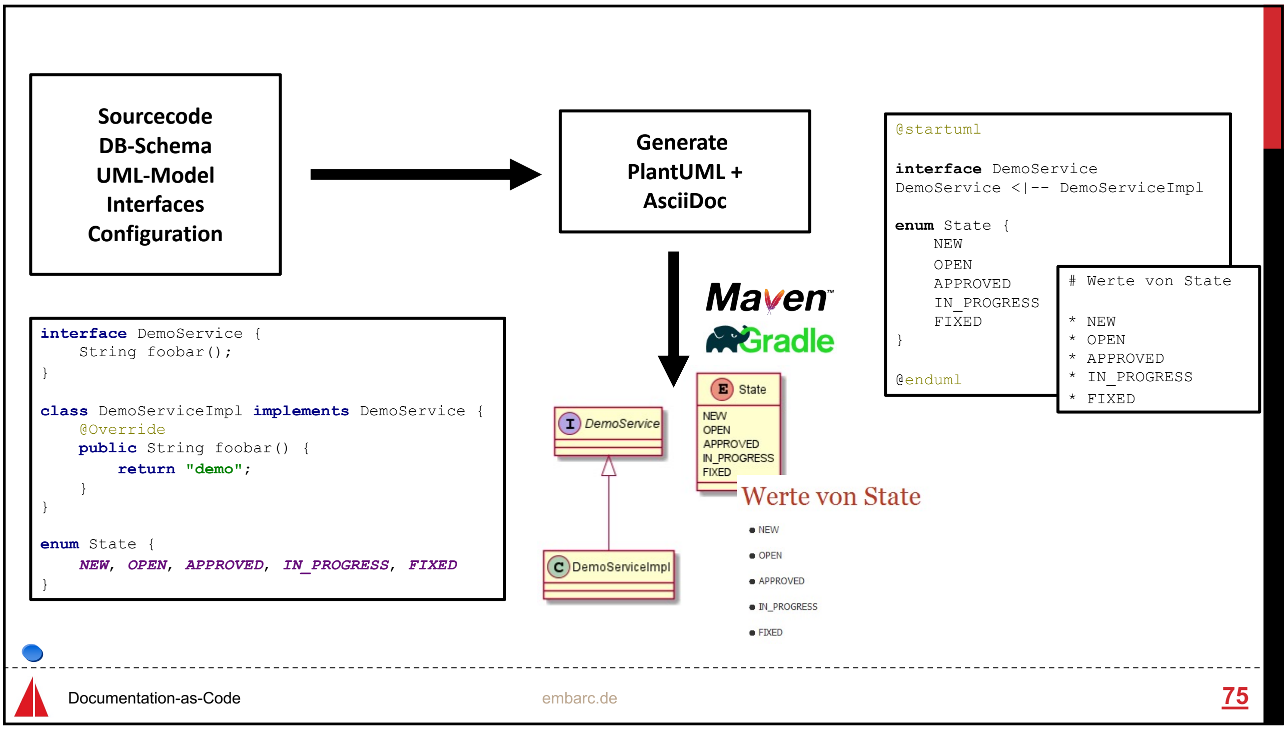
Overview

The primary goal of this project is to make it easy to document RESTful services by combining content that's been hand-written using Asciidoctor with auto-generated examples produced with the Spring MVC Test framework. The result is intended to be an easy-to-read user guide, akin to GitHub's API documentation for example, rather than the fully automated, dense API documentation produced by tools like Swagger.



JAX-RS Analyzer

Generates an overview of all JAX-RS resources in a project by bytecode analysis.



Sourcecode
DB-Schema
UML-Model
Interfaces
Configuration

Generate
PlantUML +
AsciiDoc

```
interface DemoService {
    String foobar();
}

class DemoServiceImpl implements DemoService {
    @Override
    public String foobar() {
        return "demo";
    }
}

enum State {
    NEW, OPEN, APPROVED, IN_PROGRESS, FIXED
}
```

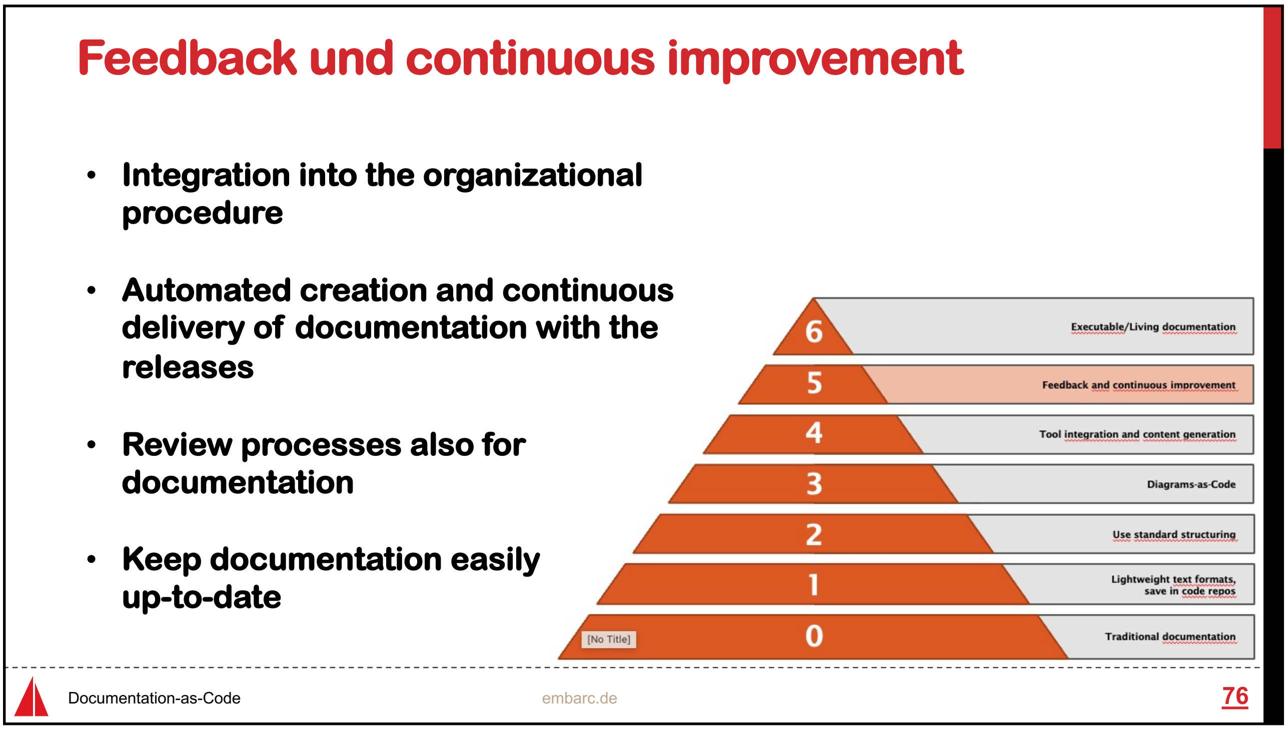
```
@startuml

interface DemoService
DemoService <|-- DemoServiceImpl

enum State {
    NEW
    OPEN
    APPROVED
    IN_PROGRESS
    FIXED
}

@enduml
```

```
# Werte von State

* NEW
* OPEN
* APPROVED
* IN_PROGRESS
* FIXED
```

Werte von State

- NEW
- OPEN
- APPROVED
- IN_PROGRESS
- FIXED

75

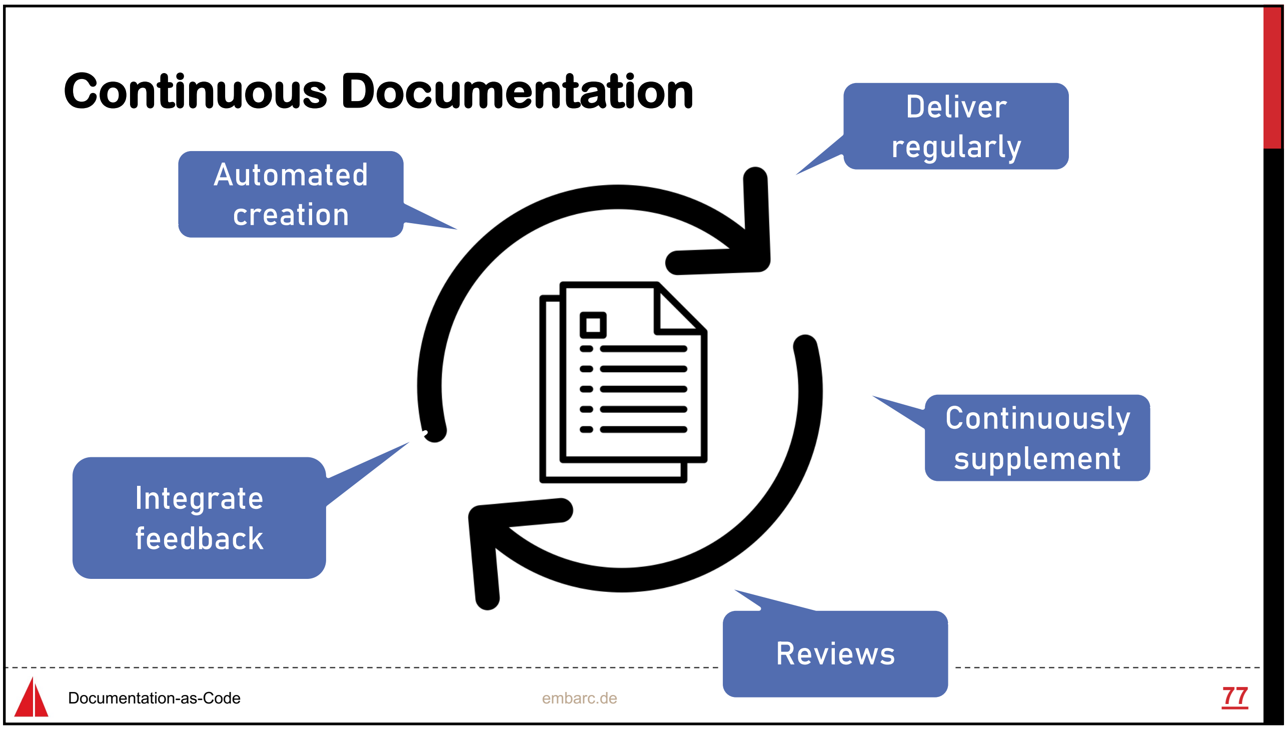


Feedback und continuous improvement

- Integration into the organizational procedure
- Automated creation and continuous delivery of documentation with the releases
- Review processes also for documentation
- Keep documentation easily up-to-date

6 Executable/Living documentation
5 Feedback and continuous improvement
4 Tool integration and content generation
3 Diagrams-as-Code
2 Use standard structuring
1 Lightweight text formats, save in code repos
0 Traditional documentation
[No Title]

76



Continuous Documentation

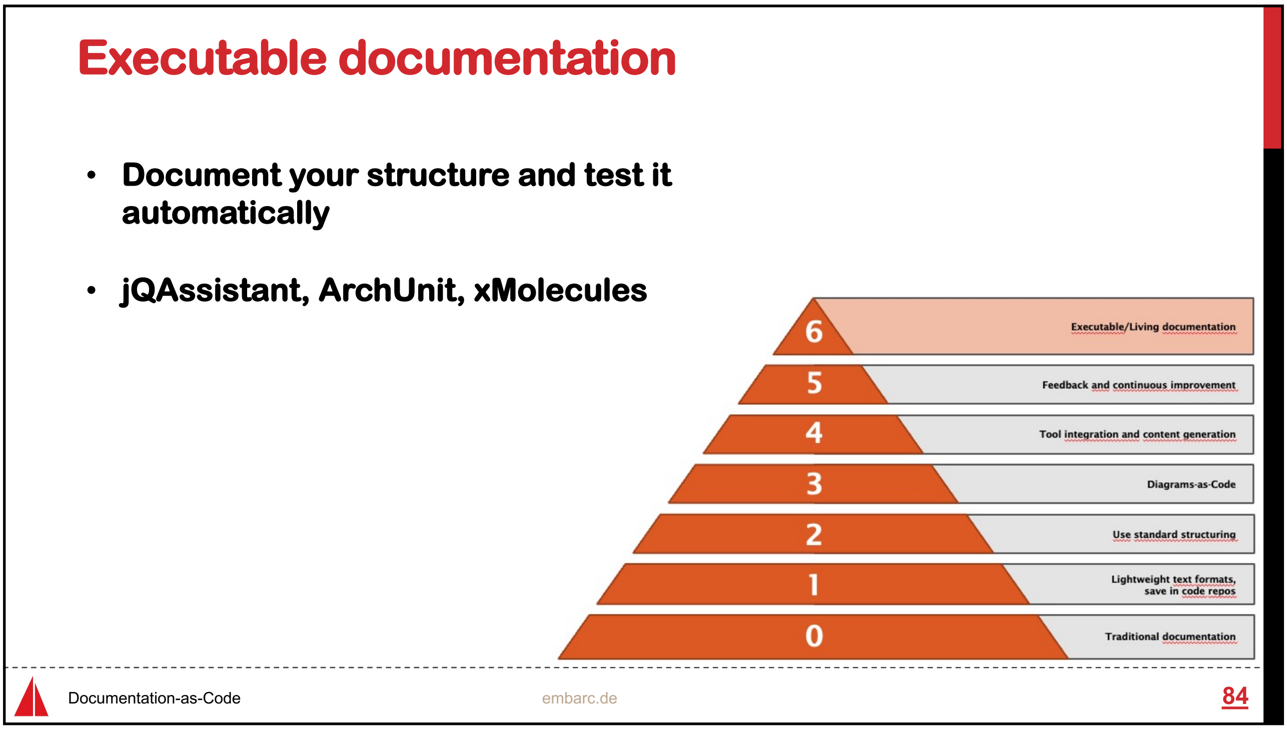
Automated creation

Deliver regularly

Continuously supplement

Reviews

Integrate feedback



Executable documentation

- Document your structure and test it automatically
- jQAssistant, ArchUnit, xMolecules

6 Executable/Living documentation

5 Feedback and continuous improvement

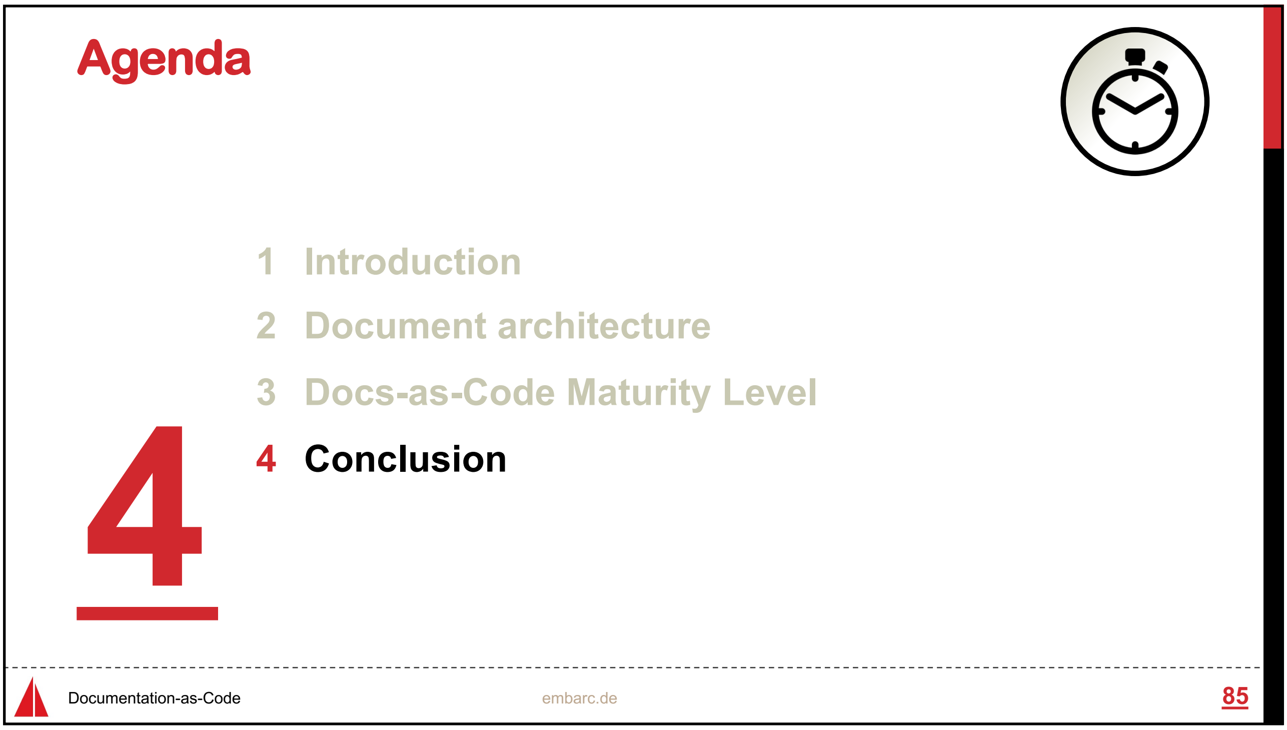
4 Tool integration and content generation

3 Diagrams-as-Code

2 Use standard structuring

1 Lightweight text formats, save in code repos

0 Traditional documentation



Agenda

4

1 Introduction

2 Document architecture

3 Docs-as-Code Maturity Level

4 Conclusion

Documentation-as-Code embarc.de 85

85

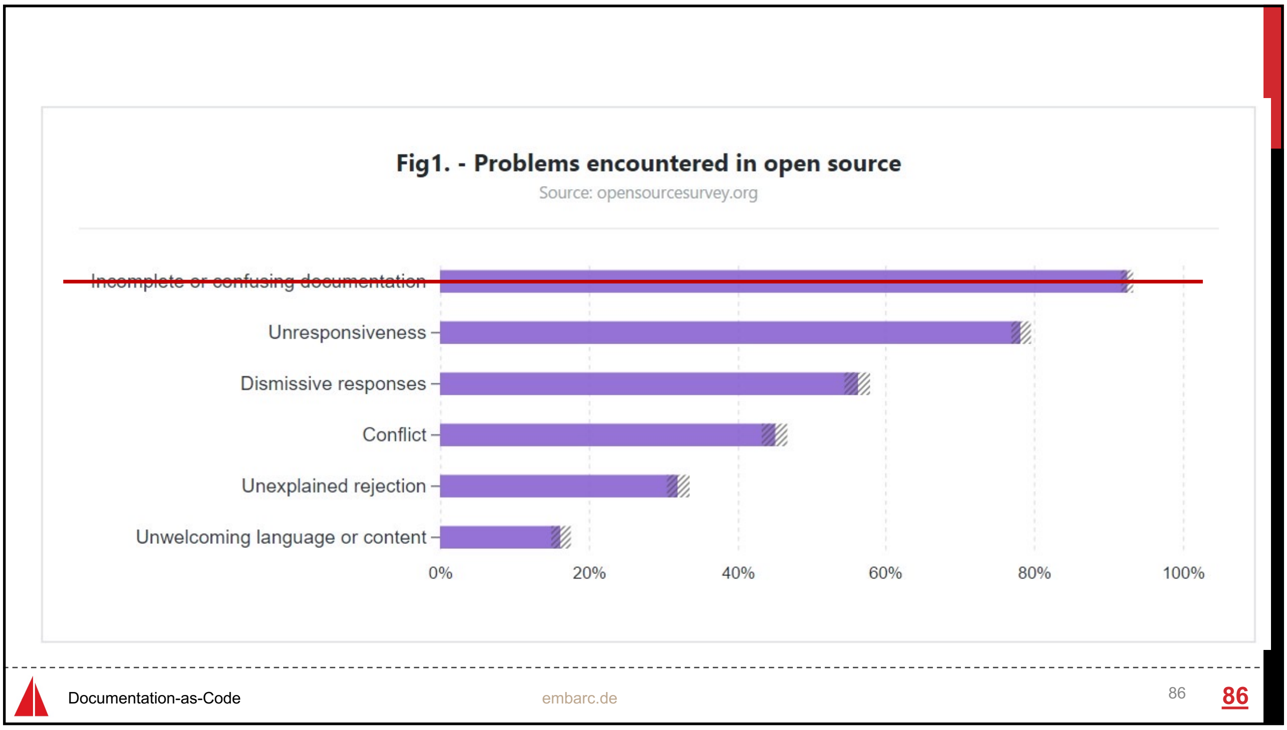
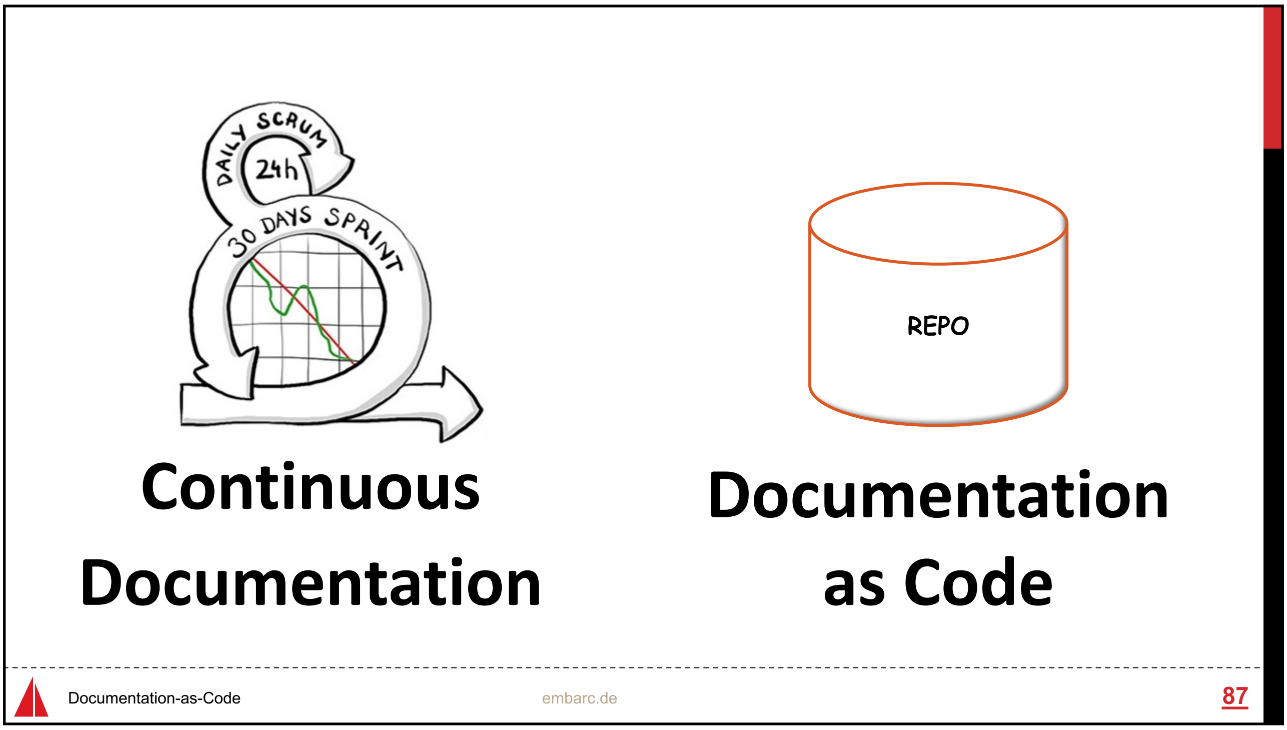


Fig1. - Problems encountered in open source

Source: opensourcesurvey.org

Documentation-as-Code embarc.de 86 86

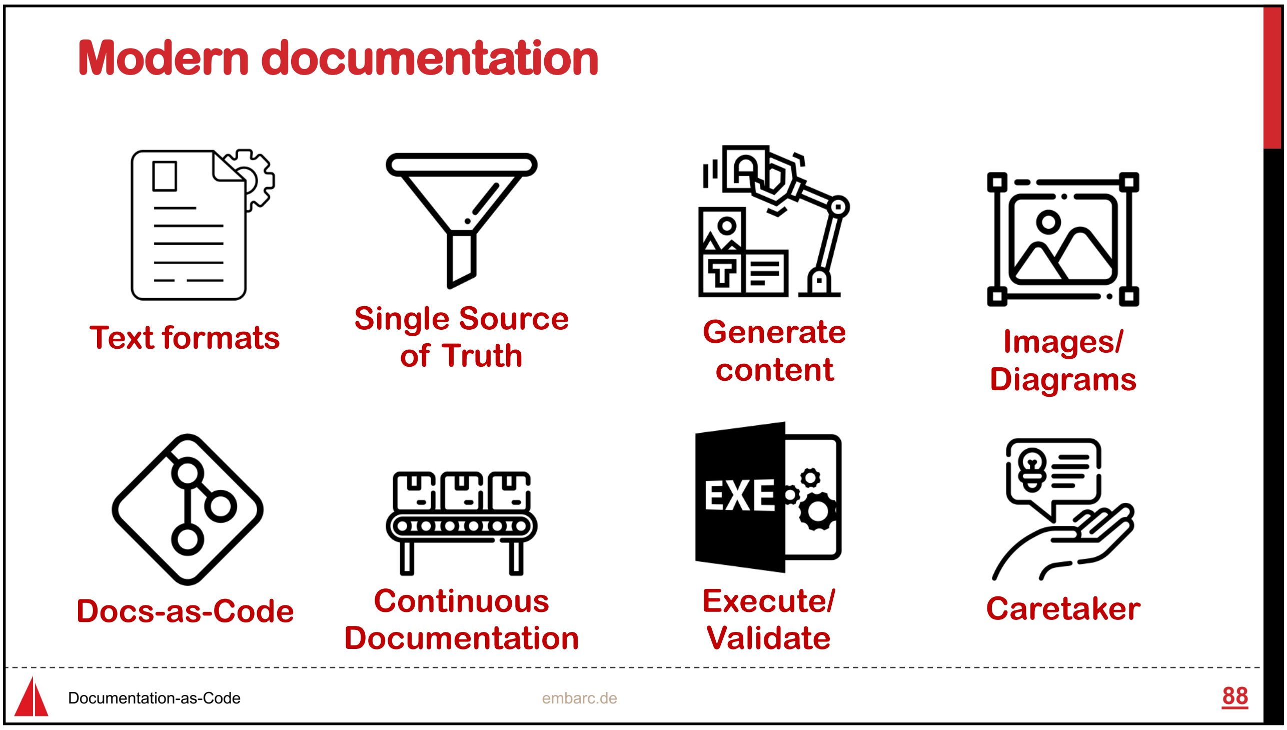
86



Continuous Documentation

Documentation as Code

87

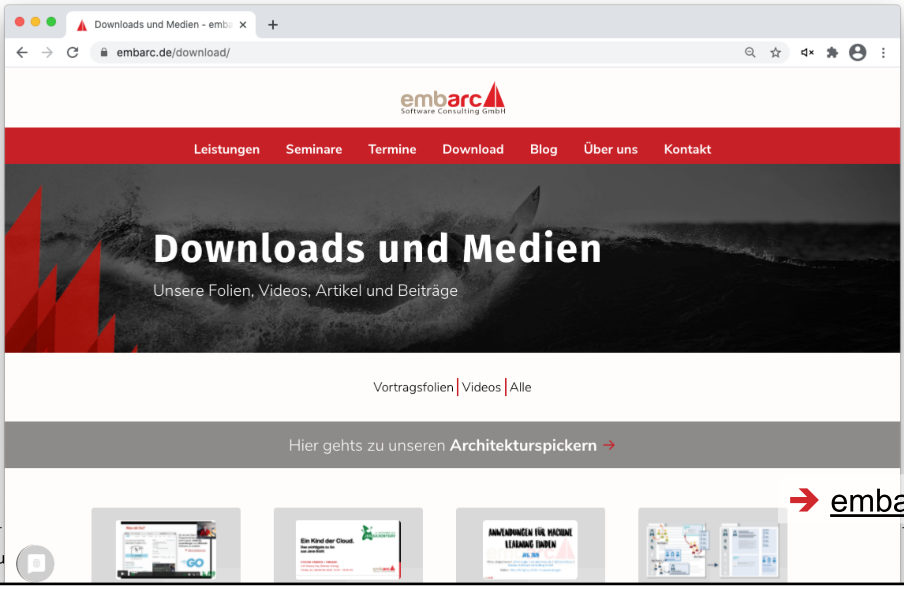


Modern documentation

- Text formats
- Single Source of Truth
- Generate content
- Images/ Diagrams
- Docs-as-Code
- Continuous Documentation
- Execute/ Validate
- Caretaker

88

Slides for download



Downloads und Medien - emb…

embarc.de/download/

Downloads und Medien

Unsere Folien, Videos, Artikel und Beiträge

Vortragsfolien | Videos | Alle

Hier gehts zu unseren **Architekturspickern** →

➔ embarc.de/download/

Falk Sippach

- Software Architect, Consultant, Trainer at embarc
- formerly at Orientation in Objects (OIO), Trivadis

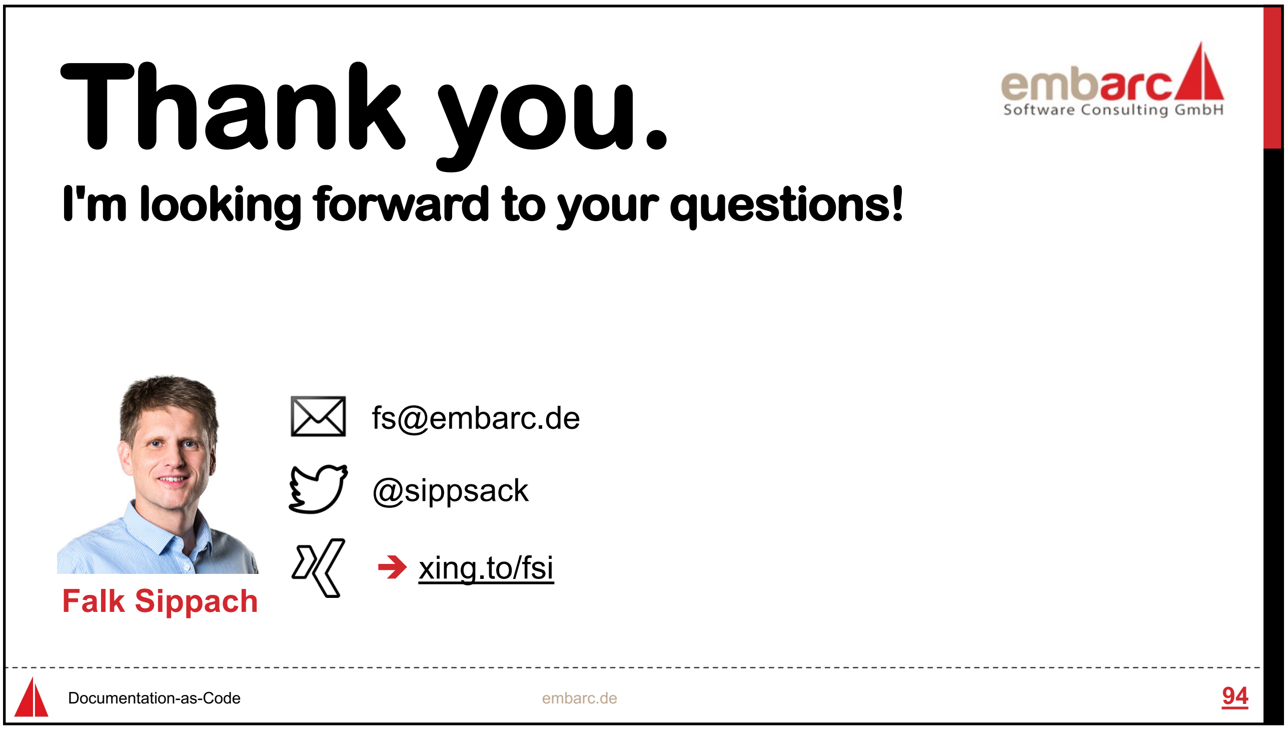
Focus:

- Architecture consulting and reviews
- Cloud and Java technologies

✉ fs@embarc.de

@sippsack

➔ xing.to/fsi



Thank you.

I'm looking forward to your questions!

Falk Sippach

✉ fs@embarc.de

@sippsack

➔ xing.to/fsi