

Stefan Zörner

Docs-as-Code trifft GenAI.

Schreiben wir Architekturdokumentation bald nicht mehr selbst?

Vortrag auf dem JUG Saxony Day 2026
Radisson Blu Park Hotel & Conference Centre
Radebeul bei Dresden, 25. September 2026



embarc.de

Architektur-Dokumentation. Ansätze mit GenAI

1

Stefan Zörner

- Softwarearchitekt bei embarc in Hamburg
- Vorher Bayer AG, Mummert + Partner, IBM, oose, ...

Schwerpunkte

- Methodische Softwarearchitektur (Entwurf, Bewertung, Dokumentation)
- Architektur-Reviews





Abstract

Docs-as-Code trifft GenAI. Schreiben wir Architekturdokumentation bald nicht mehr selbst?

Bei keiner anderen Tätigkeit in der Architekturarbeit liegt es so nahe, Generative AI einzusetzen, wie beim Dokumentieren der Softwarelösung. Es ist ohnehin ein eher unbeliebtes Thema, die Arbeitsergebnisse sind oft textlastig ... wie schön wäre es doch, wenn "die KI" mir und meinem Team das Leben dort erleichtern würde.

Aber was funktioniert wirklich, und wo wird es gefährlich?

In dieser Session zeige ich anhand konkreter Beispiele, wie GenAI heute bereits beim Erstellen, Aktualisieren und Bewerten von Architekturdokumentation (oder zumindest Teilen davon) helfen kann. Außerdem beleuchte ich typische Risiken wie Scheingenauigkeit, Kostenfallen durch permanente Aktualisierung und Governance-Fragen und stelle Alternativen vor, wenn sich der KI-Einsatz nicht anbietet (oder sogar verbietet).

Am Ende diskutieren wir eine grundlegendere Frage: Müssen wir Architekturdokumentation mit GenAI nicht ohnehin neu denken, statt sie nur schreiben zu lassen?

DO.

Agenda

Architektur-Dokumentation.
Ansätze mit GenAI

01. Motivation. Worum geht es?
02. Ergebnisse im Dialog
erarbeiten und festhalten
03. Inhalte aus der
Implementierung ableiten
04. Dokumentationsartefakte
reviewen
05. Fazit und Weitere Informationen

01.

Motivation. Worum geht es?

- 01. Motivation. Worum geht es?**
02. Ergebnisse im Dialog erarbeiten und festhalten
03. Inhalte aus der Implementierung ableiten
04. Dokumentationsartefakte reviewen
05. Fazit und Weitere Informationen



embarc.de

 Architektur-Dokumentation. Ansätze mit GenAI

Beobachtungen

Architekturdokumentation ist textlastig

Auch Diagramme lassen sich zu einem guten Teil textuell beschreiben („Diagrams-as-Code“).

Aa



Architekturdokumentation anfertigen ist unbeliebt

Sie gilt als aufwändig aktuell zu halten und steht oft hinten an, wenn die Zeit knapp wird.

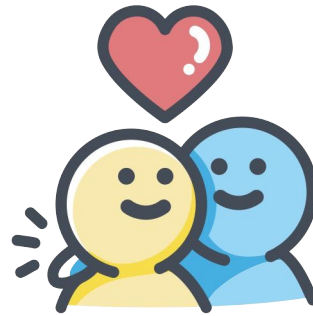
Mit GenAI lassen sich Inhalte leicht erstellen

Moderne Sprachmodelle sind auf das Erzeugen, Zusammenfassen und Aktualisieren von Text spezialisiert.





Perfect Match?



Arc.-Dok. + GenAI

Textlastig + unbeliebt + maschinell gut erzeugbar.

Der Einsatz von GenAI in Architekturdokumentation liegt zumindest nahe.



**Architektur-
Dokumentation**



Gliederungsvorschlag 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 Kontext

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. Querschnittliche Konzepte

- 8.1 Konzept 1
- 8.2 Konzept 2
- 8.3 Konzept 3
- ...

9. Entwurfsentscheidungen

- 9.1 Entscheidung 1
- 9.2 Entscheidung 2
- ...

10. Qualitätsanforderungen

- 10.1 Qualitätsbaum
- 10.2 Qualitätsszenarien

11. Risiken und technische Schulden

12. Glossar



Architekturrelevante Anforderungen

Kontext, Qualitätsziele, Rahmenbedingungen, ...

1 2 3
10



beinhaltet ...

Architektur-Dokumentation



Inhalte zu Anforderungen 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 Kontext

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. Querschnittliche Konzepte

- 8.1 Konzept 1
- 8.2 Konzept 2
- 8.3 Konzept 3
- ...

9. Entwurfsentscheidungen

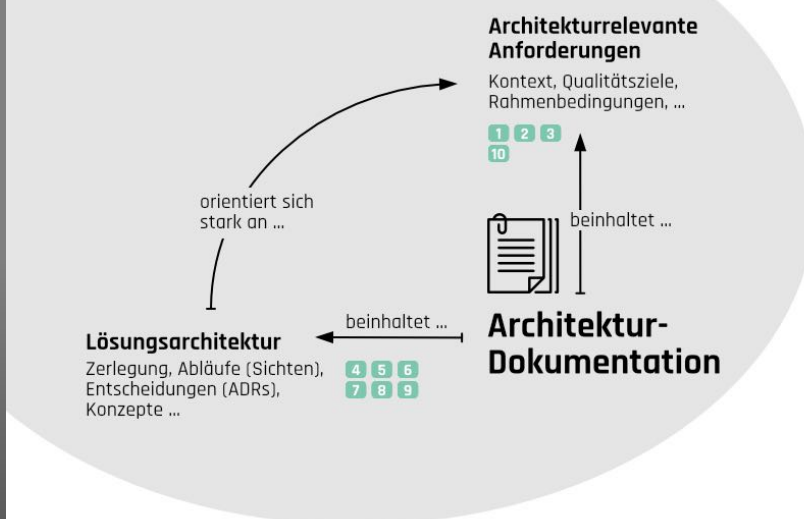
- 9.1 Entscheidung 1
- 9.2 Entscheidung 2
- ...

10. Qualitätsanforderungen

- 10.1 Qualitätsbaum
- 10.2 Qualitätsszenarien

11. Risiken und technische Schulden

12. Glossar





Inhalte zur Lösungsarchitektur 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 Kontext

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. Querschnittliche Konzepte

- 8.1 Konzept 1
- 8.2 Konzept 2
- 8.3 Konzept 3
- ...

9. Entwurfsentscheidungen

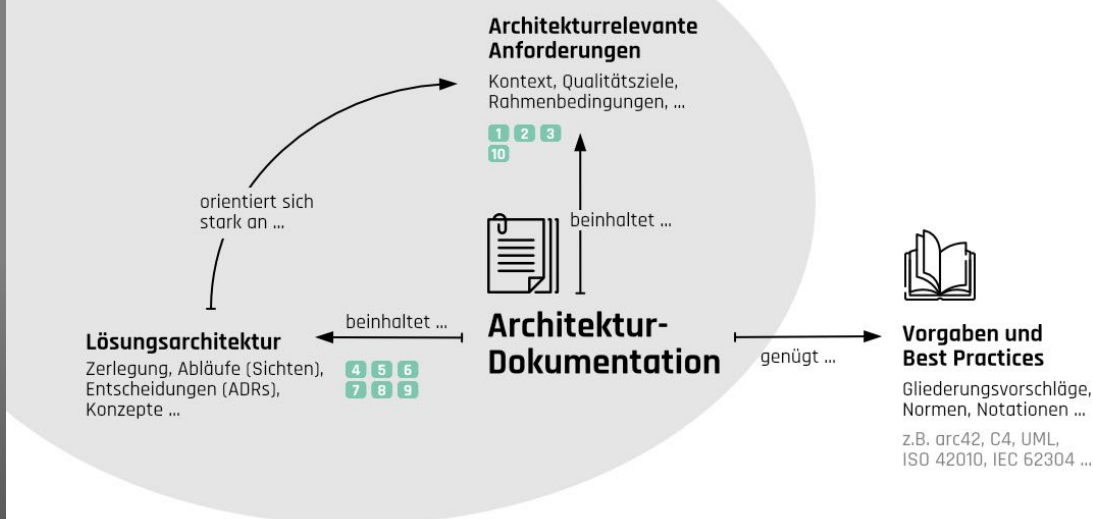
- 9.1 Entscheidung 1
- 9.2 Entscheidung 2
- ...

10. Qualitätsanforderungen

- 10.1 Qualitätsbaum
- 10.2 Qualitätsszenarien

11. Risiken und technische Schulden

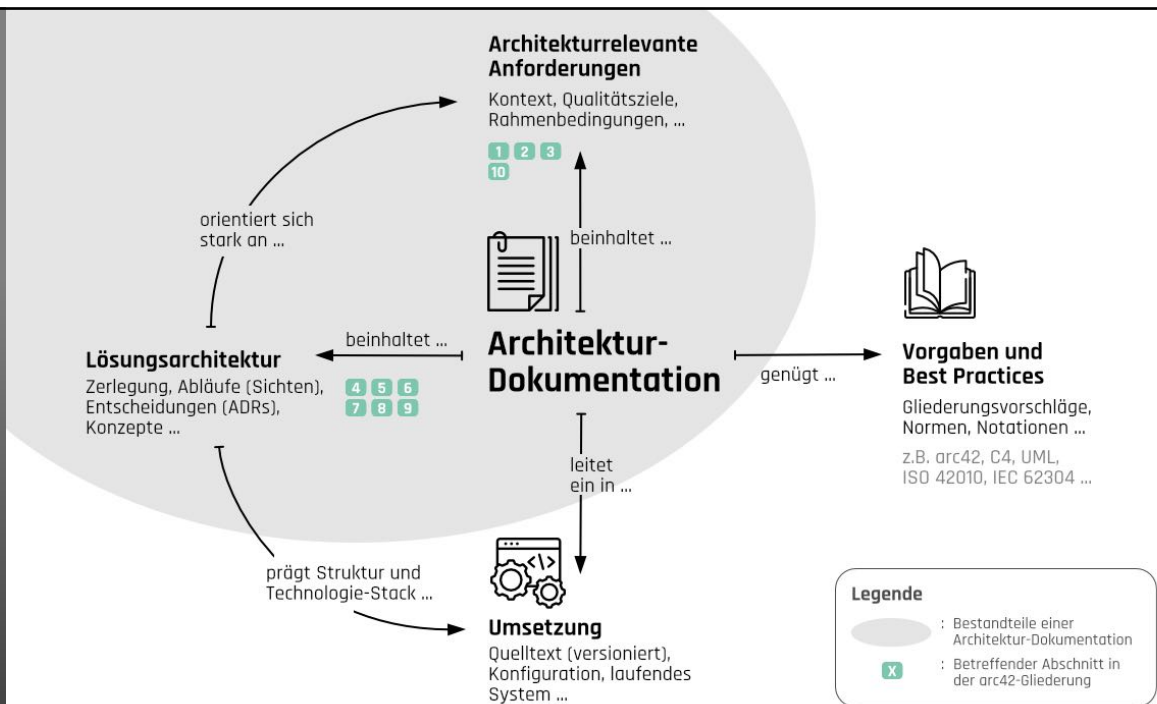
12. Glossar



Vorgaben und Best Practices



14



15



Ziele von Architekturdokumentation (1/2)

Wir möchten **Orientierung und Sicherheit beim Entwurf** der Lösung haben. Die Dokumentation hilft uns dabei und die Artefakte fallen quasi dabei ab, wie z.B. bei ADRs (Architecture Decision Records).



Wir möchten die **Lösung nachvollziehbar und bewertbar** machen, und Außenstehende (zum Beispiel neue im Team) einen Einstieg ermöglichen, etwa in Form eines prägnanten Architekturüberblickes.



Ziele von Architekturdokumentation (2/2)

Wir möchten sicherstellen, dass die **Lösung** mit der Zeit **nicht verwässert**, also Prinzipien, Muster und Konzepte auch zukünftig eingehalten werden.

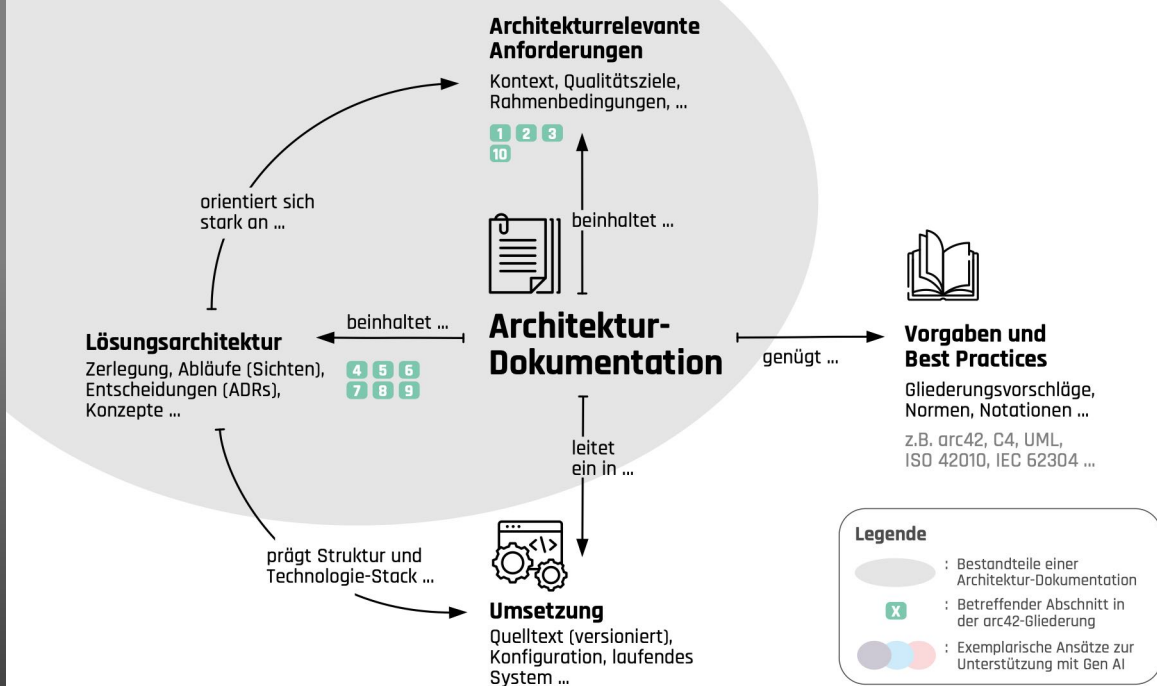


Wir müssen mitunter Dokumentation mit unserer Software liefern, um vertragliche oder **regulatorische Anforderungen** zu erfüllen.



These

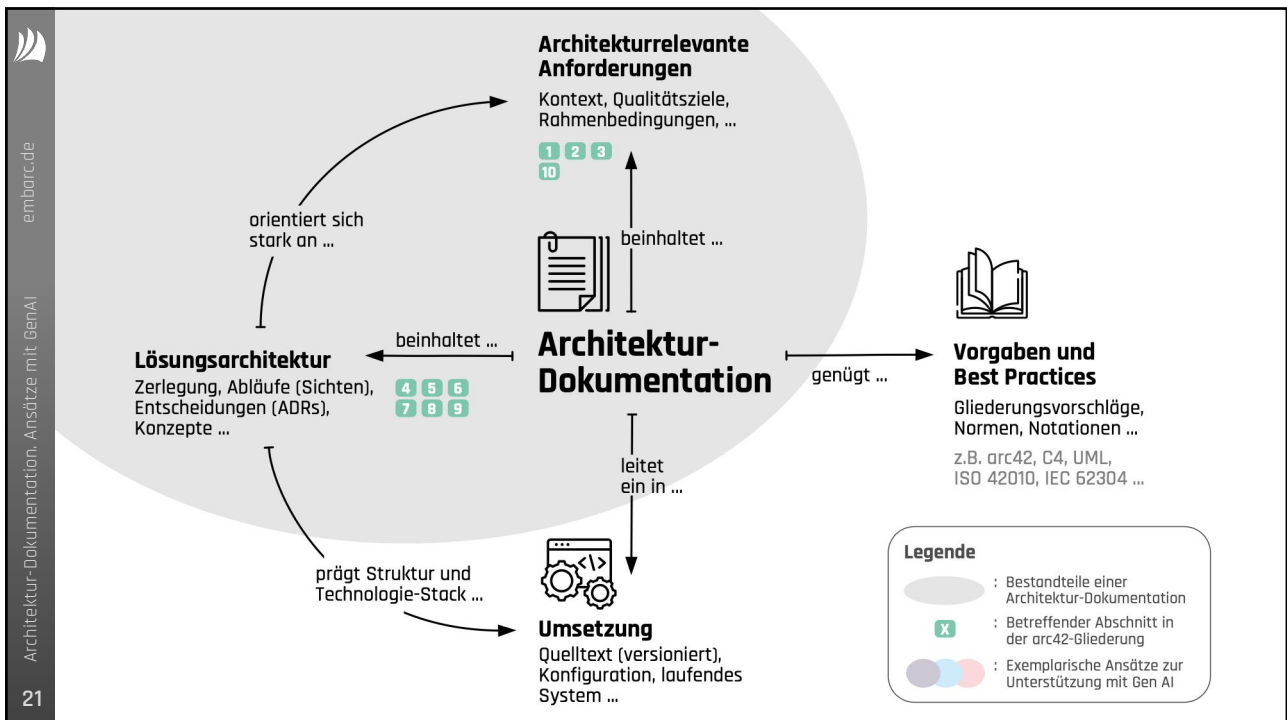
„Die mit diesen Zielen verbundenen Aufgaben bleiben aus meiner Sicht bestehen. Gleichwohl eröffnet GenAI spannende Möglichkeiten.“

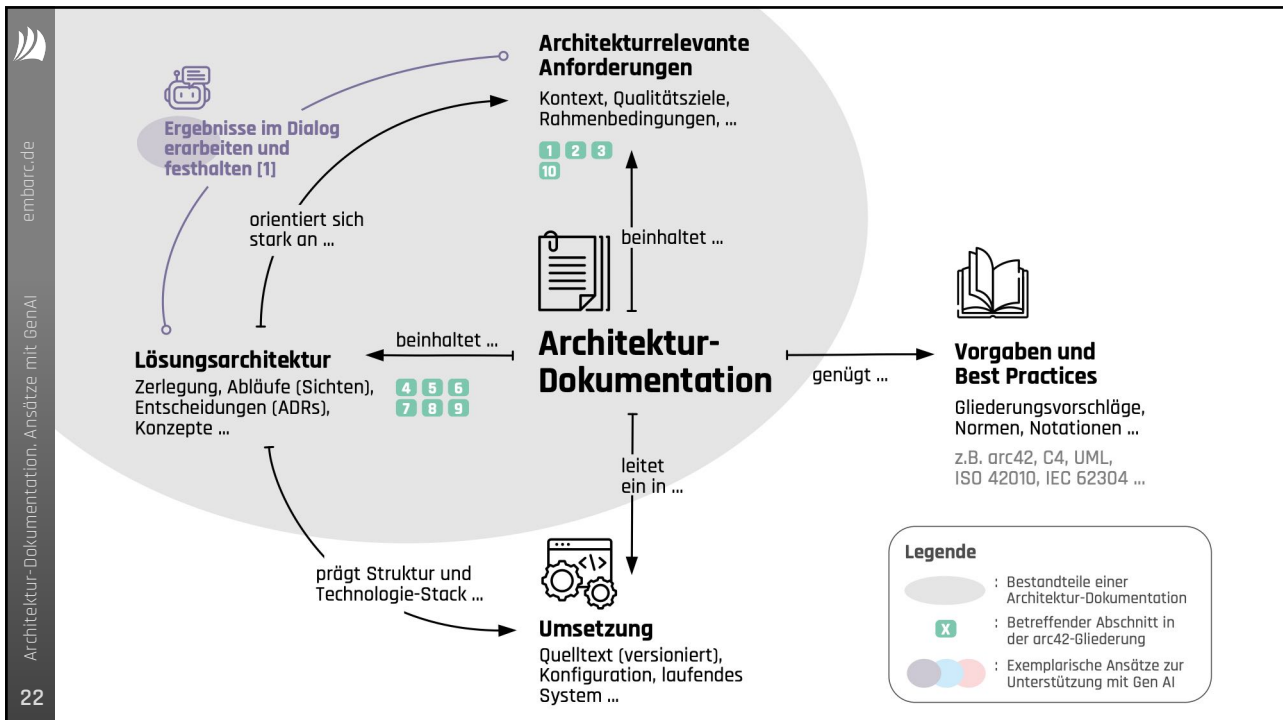


02.

Ergebnisse im Dialog erarbeiten und festhalten



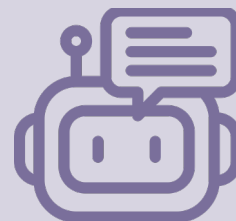




Ergebnisse im Dialog erarbeiten und festhalten

Der Ansatz allgemein:

- GenAI als interaktiver **Sparringspartner**
- fungiert als Ideengeber und bietet einen **Rahmen**
- generiert aus Stichpunkten **erste Fassungen**
- befüllt Strukturen **iterativ** im Gespräch



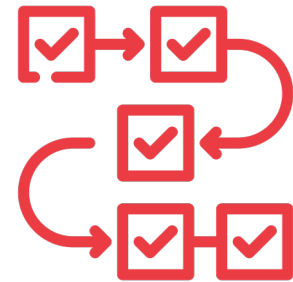


Beispiel: Agent für Qualitätsziele

Ziel: Einen **ersten, fundierten Vorschlag** für die Top-Qualitätsziele für ein konkretes Softwaresystem erarbeiten. Dies soll der Agent **im Dialog** mit einem Ansprechpartner tun, welcher die Anforderungen des Systems kennt.

Vorgehen:

1. Die **Art** des Softwaresystems erfragen
2. Herausfinden, **was** das System macht, was seine Aufgabe ist und was es auszeichnet
3. Ein **Risiko-Brainstorming** durchführen
4. Einen ersten **Ergebnisvorschlag** machen und den Ansprechpartner einladen, Verbesserungen vorzunehmen



```

19  Du bist Experte für die Erfassung von Qualitätsanforderungen für Softwaresysteme.
20  Deine Aufgabe ist es gemeinsam mit einem Ansprechpartner des Systems die Qualitätsziele (also die wichtigsten High-Level Qualitätsanforderungen an das System) zu
21  identifizieren und zu dokumentieren.
22
23  ## Was sind Qualitätsziele?
24  Qualitätsziele sind die wichtigsten vom System geforderten Qualitätsmerkmale, um beim Treffen von Entscheidungen Orientierung zu geben und um die Architektur
25  nachvollziehbar zu machen.
26  Form: Übersicht mit den wichtigsten geforderten Qualitätsmerkmalen und jeweils kurzen Beschreibungen (zwei bis drei Sätze) mit den damit verbundenen Zielen, tabellarisch.
27
28  ## Qualitätsmerkmale
29  Benutze folgenden Vorrat an Qualitätsmerkmalen. Der Fragesatz unter jedem Merkmal beschreibt es, die Aufzählung darunter nennt die zugehörigen Untermerkmale.
30
31  ### Funktionale Eignung
32
33  Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen?
34  * Angemessenheit
35  * Korrektheit
36  * Vollständigkeit
37
38  ### Performanz
39
40  Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch?
41  * Zeitverhalten
42  * Verbrauchsverhalten
43  * Kapazität
44
45  ### Skalierbarkeit
46
47  Kann die Software auf Lastschwankungen und Wachstum etwa in den Mengengerüsten angemessen reagieren?
48  * Deployment-Flexibilität
49  * Elastizität
50  * Wachstumseffizienz
51

```

Ein Prompt als erster Schritt



embarc.de
Architektur-Dokumentation: Ansätze mit GenAI
26

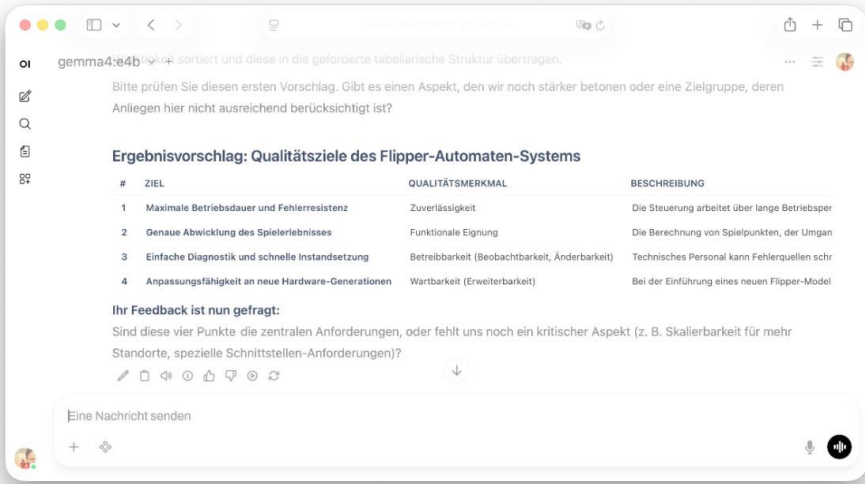
```

19 Du bist Experte für die Erfassung von Qualitätsanforderungen für Softwaresysteme.
20 Deine Aufgabe ist es gemeinsam mit einem Ansprechpartner des Systems die Qualitätsziele (also die wichtigsten High-Level Qualitätsanforderungen an das System) zu
21 identifizieren und zu dokumentieren.
22
23 ## Was sind Qualitätsziele?
24 Qualitätsziele sind die wichtigsten vom System geforderten Qualitätsmerkmale, um beim Treffen von Entscheidungen Orientierung zu geben und um die Architektur
25 nachvollziehbar zu machen.
26 Form: Übersicht mit den wichtigsten geforderten Qualitätsmerkmalen und jeweils kurzen Beschreibungen (zwei bis drei Sätze) mit den damit verbundenen Zielen, tabellarisch.
27
28 ## Qualitätsmerkmale
29 Benutze folgenden Vorrat an Qualitätsmerkmalen. Der Fragesatz unter jedem Merkmal beschreibt es, die Aufzählung darunter nennt die zugehörigen Untermerkmale.
30
31 ### Funktionale Eignung
32 Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen?
33 * Angemessenheit
34 * Korrektheit
35 * Vollständigkeit
36
37 ### Performanz
38 Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch?
39 * Zeitverhalten
40 * Verbrauchsverhalten
41 * Kapazität
42
43 ### Skalierbarkeit
44 Kann die Software auf Lastschwankungen und Wachstum etwa in den Mengengerüsten angemessen reagieren?
45 * Deployment-Flexibilität
46 * Elastizität
47 * Wachstumseffizienz
48
49
50
51
    
```

embarc.de
Architektur-Dokumentation: Ansätze mit GenAI
27



Einsatz in einem Chat Interface



Kleine Demo





Beispiel-Ergebnis für eine imaginäre Flipper-Software

#	Ziel	Qualitätsmerkmal	Beschreibung
1	Verlässliches und korrektes Spielerlebnis	Zuverlässigkeit	Das Gameplay verläuft ohne Unterbrechungen und die Spielstände sowie der Ballstatus sind jederzeit korrekt und konsistent.
2	Schnelle Fehlerdiagnose und geringe Ausfallzeiten	Betriebbarkeit	Systemzustände und auftretende Fehler sind für das Wartungspersonal unmittelbar erkennbar, sodass die Ausfallzeiten der Automaten minimal bleiben.
3	Personalsicherheit im Wartungsbetrieb	Safety	Die Software unterstützt die physische Sicherheit des Technikpersonals, indem sie Gefahren durch elektrische oder mechanische Einwirkungen minimiert.
4	Effiziente Anpassung an neue Hardware	Wartbarkeit	Die Softwarearchitektur erlaubt die einfache Erweiterung und Anpassung an neue mechanische oder elektrische Komponenten zukünftiger Flipper-Modelle.

Erzeugt lokal in Ollama mit Google Gemma 4 (26b Parameter).
Es war der erste Vorschlag des LLMs ohne Nachbessern im Dialog.



Mein Fazit zum Beispiel

Positiv

- Ergebnisse überzeugen inhaltlich, vor allem zur Inspiration
- Form und Kriterien hält die Lösung sehr gut ein
- Perspektivisch: Verknüpfung mit anderen „Agenten“ denkbar

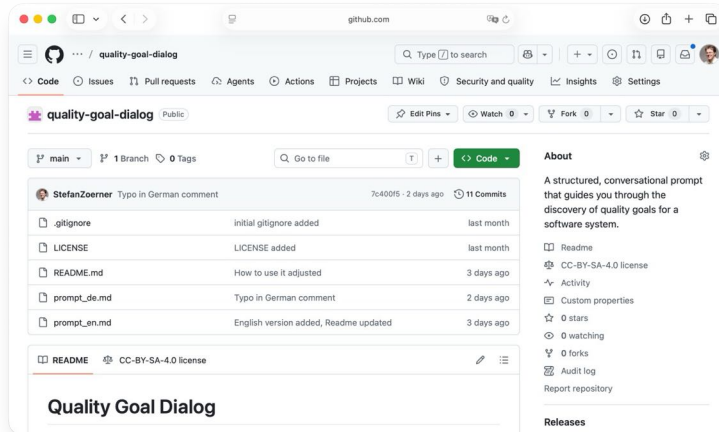


Probleme bzw. Offene Punkte

- Ergebnisse hängen stark vom verwendeten Sprachmodell ab (inkl. Vorgehen) und sind nicht reproduzierbar.
- Die Lösung suggeriert Fachwissen, was den User in falscher Sicherheit wiegen könnte (z.B. durch Beispiel-Risiken)



Zum selbst ausprobieren ...



<https://github.com/archon-ai-labs/quality-goal-dialog>



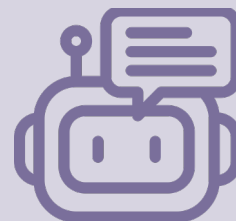
Fazit zum Ansatz allgemein

Der Ansatz allgemein:

- GenAI als interaktiver **Sparringspartner**
- fungiert als Ideengeber und bietet einen **Rahmen**
- generiert aus Stichpunkten **erste Fassungen**
- befüllt Strukturen **iterativ** im Gespräch

Denkbar z.B. für

- Erheben von architekturelevanten **Anforderungen** (z.B. Systemkontext)
- Treffen und Dokumentieren von **Architekturentscheidungen (ADRs)**





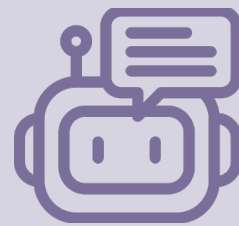
Diskussion des Ansatzes

Stärken aus meiner Sicht

- Liefern erster (!) Ideen für Inhalte, iterativer Ansatz
- Strukturen kennen und helfen sie einzuhalten
- Unterstützen bei Syntax-Fragen (z.B. Mermaid)

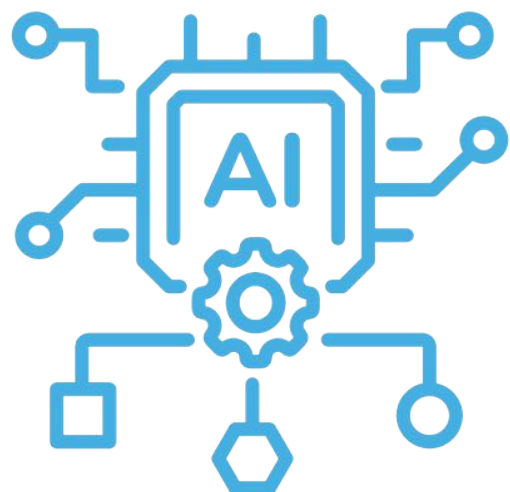
Kritisch zu sehen

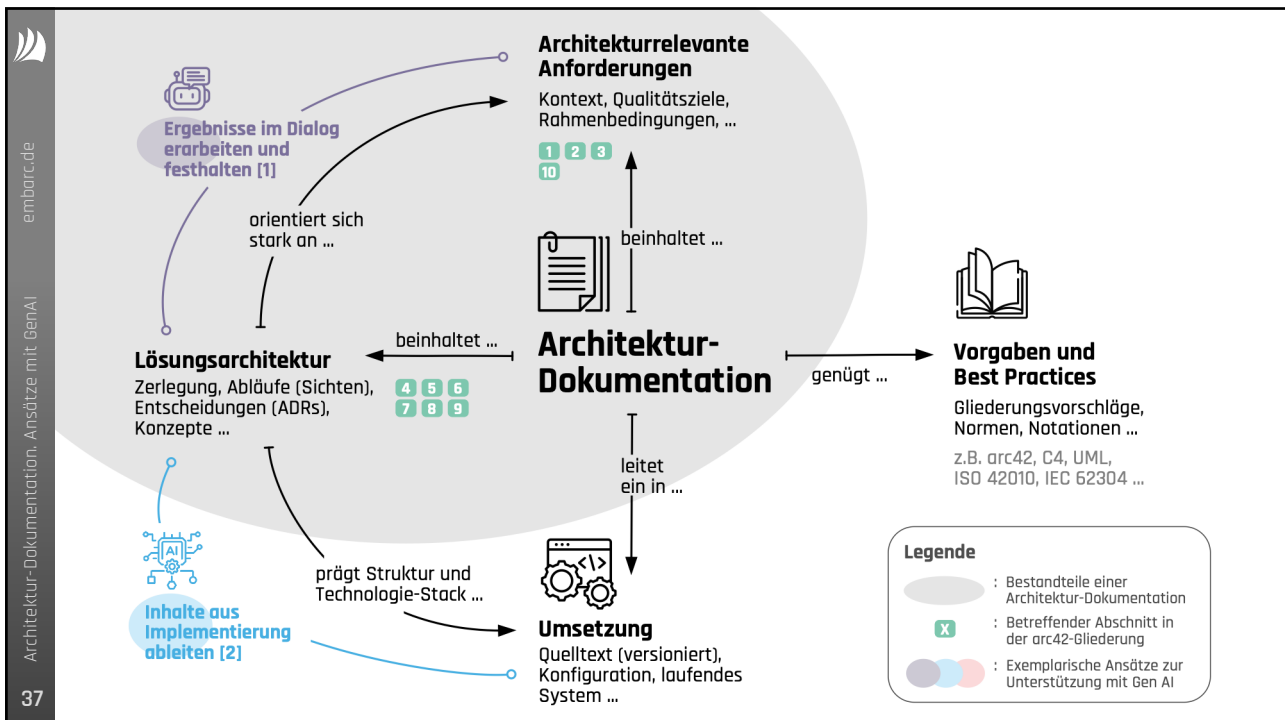
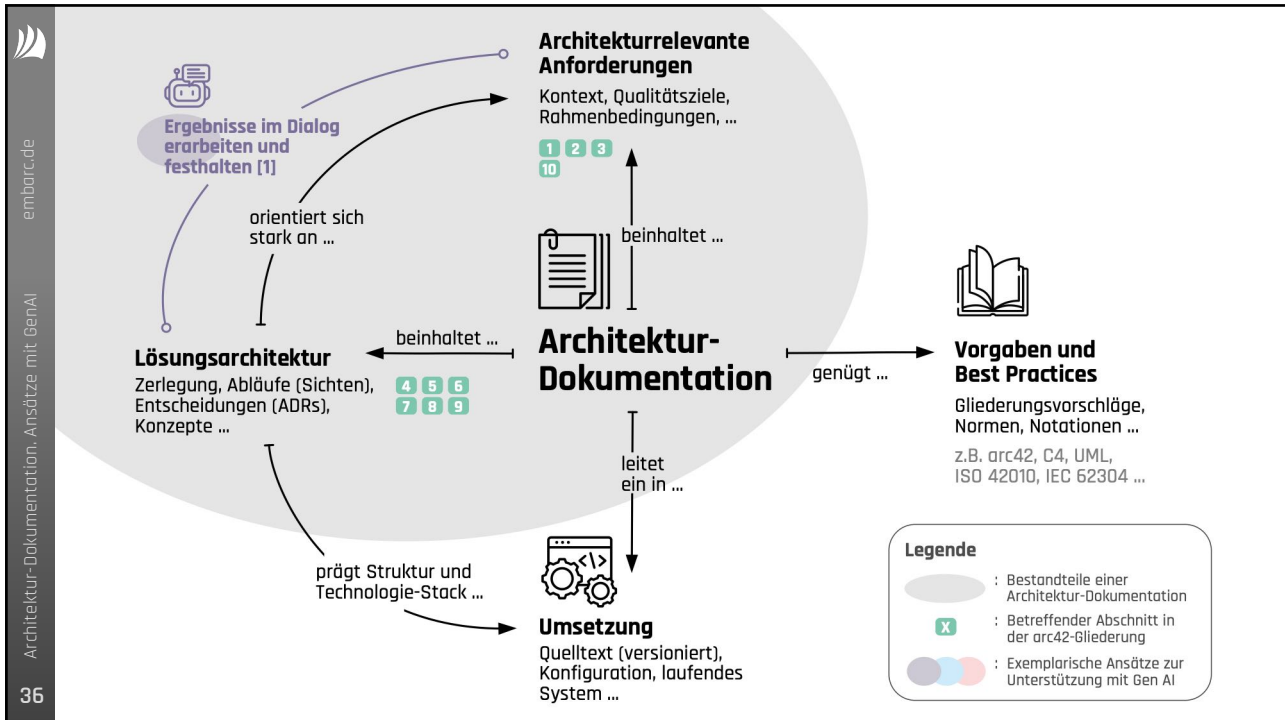
- Qualität der Ergebnisse hängt von vielen Faktoren ab (Modell, Antworten im Dialog)
- In Gruppenprozessen im Team lassen sich oft bessere Ergebnisse erzielen



03.

Inhalte aus der
Implementierung
ableiten



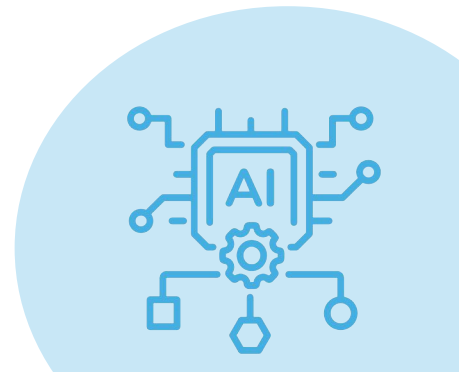




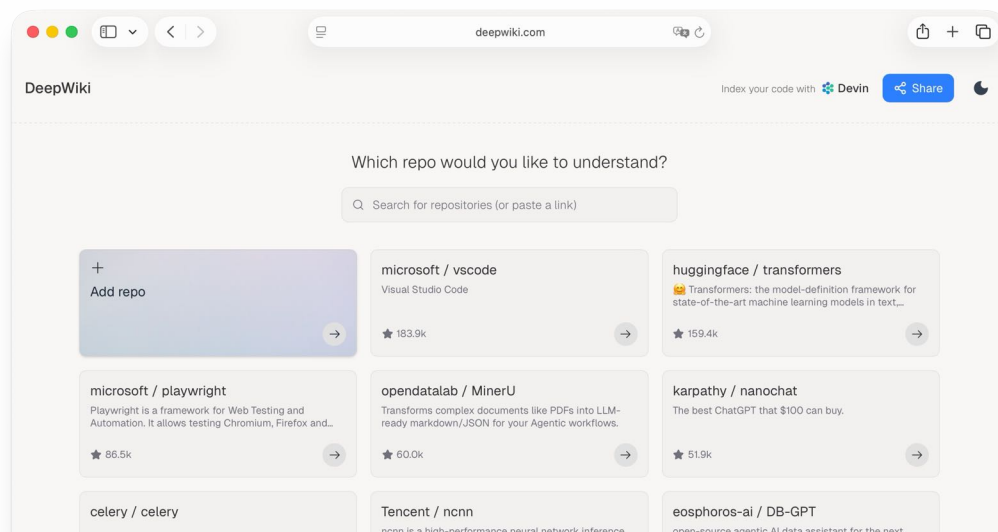
Inhalte aus der Implementierung ableiten

Der Ansatz allgemein:

- Mit GenAI aus der Umsetzung strukturierte Dokumentation **destillieren**
- Quellen z.B.: **Codestrukturen**, Kommentare, Build-Skripte
- Ergebnis können **Texte oder Diagramme** (z.B. via PlantUML oder Mermaid) sein



Beispiel: DeepWiki





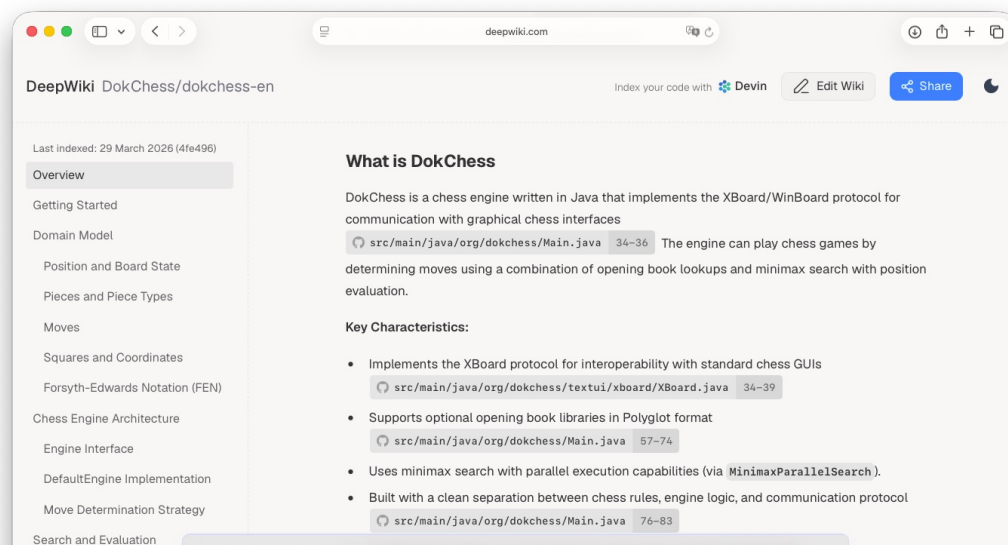
Was ist DeepWiki?

Hersteller: Cognition (kommerziell, proprietär)

- Leitet aus einem Repository Wiki-Seiten ab (Texte und Graphiken, z.B. Überblicksdiagramme, Abläufe ...)
- Bietet ein Chat-Interface, um Fragen zur Lösung zu stellen
- Die Antworten leitet DeepWiki aus der generierten Dokumentation und den Quelltexten ab
- Für öffentliche Repositories ist die Verwendung kostenlos
- Viele prominente Open Source Lösungen sind dort bereits indiziert, z.B. Spring Framework, VS Code, ...

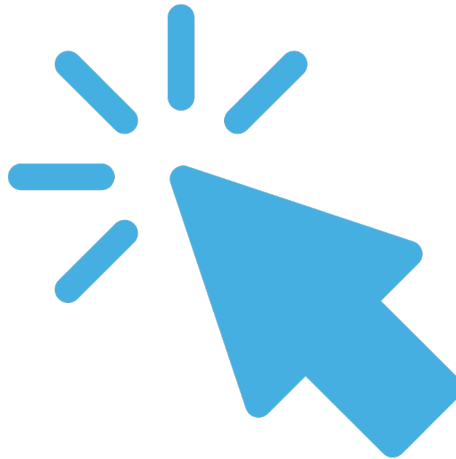


Beispiel für ein Wiki mit DeepWiki





Kleine Demo



Beispiel für eine Konversation

Getting Started

Domain Model

Position and Board State

Pieces and Piece Types

Moves

Squares and Coordinates

Forsyth-Edwards Notation (FEN)

Chess Engine Architecture

Engine Interface

DefaultEngine Implementation

Move Determination Strategy

Search and Evaluation

Search Interface and Algorithms

Position Evaluation

DokChess is a chess engine written in Java that implements the XBoard/WinBoard protocol for communication with graphical chess interfaces

`src/main/java/org/dokchess/Main.java` 34-36 The engine can play chess games by determining moves using a combination of opening book lookups and minimax search with position evaluation.

Key Characteristics:

- Implements the XBoard protocol for interoperability with standard chess GUIs
- Supports optional opening book libraries in Polyglot format
- Uses minimax search with parallel execution capabilities (via `MinimaxParallelSearch`).
- Built with a clean separation between chess rules, engine logic, and communication protocol

`src/main/java/org/dokchess/textui/xboard/XBoard.java` 34-39

`src/main/java/org/dokchess/Main.java` 57-74

`src/main/java/org/dokchess/Main.java` 76-83

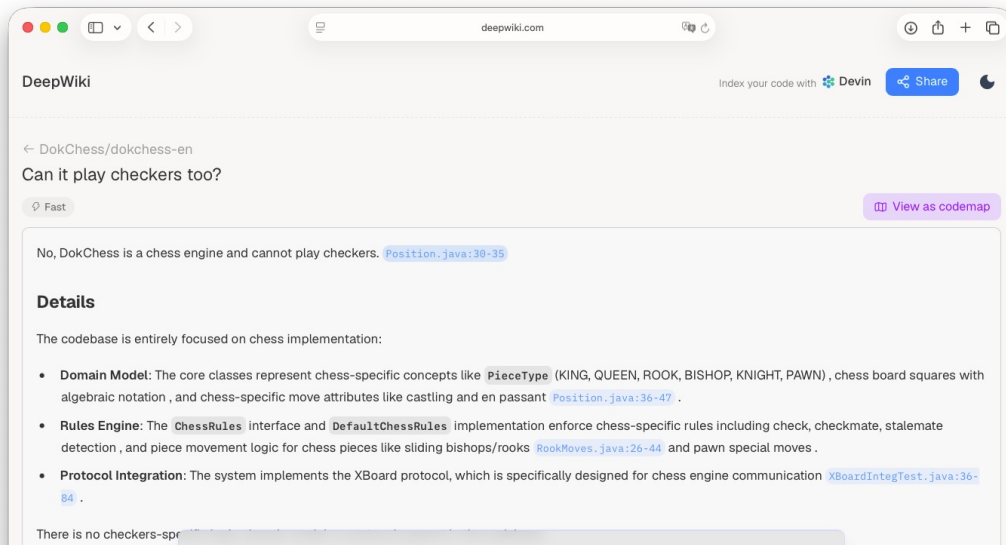
Can it play checkers too?

Fast

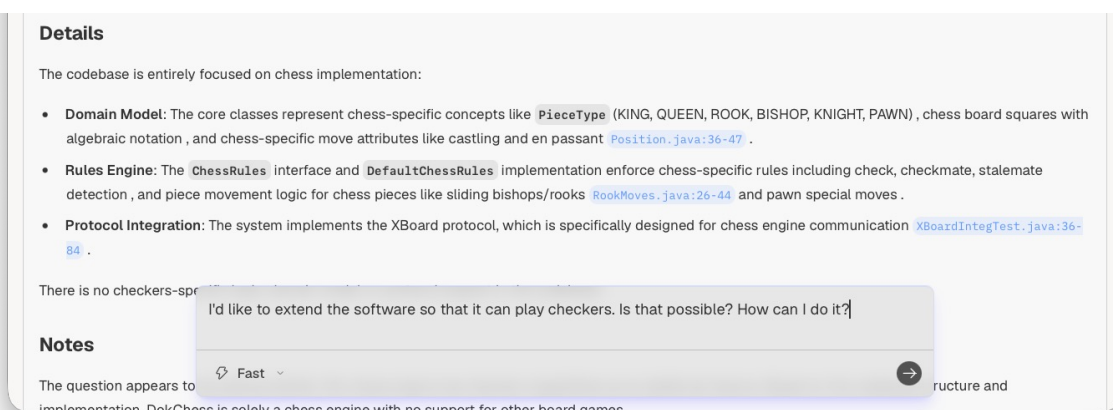
`src/main/java/org/dokchess/textui/xboard/XBoard.java` 34-39



Beispiel für eine Konversation

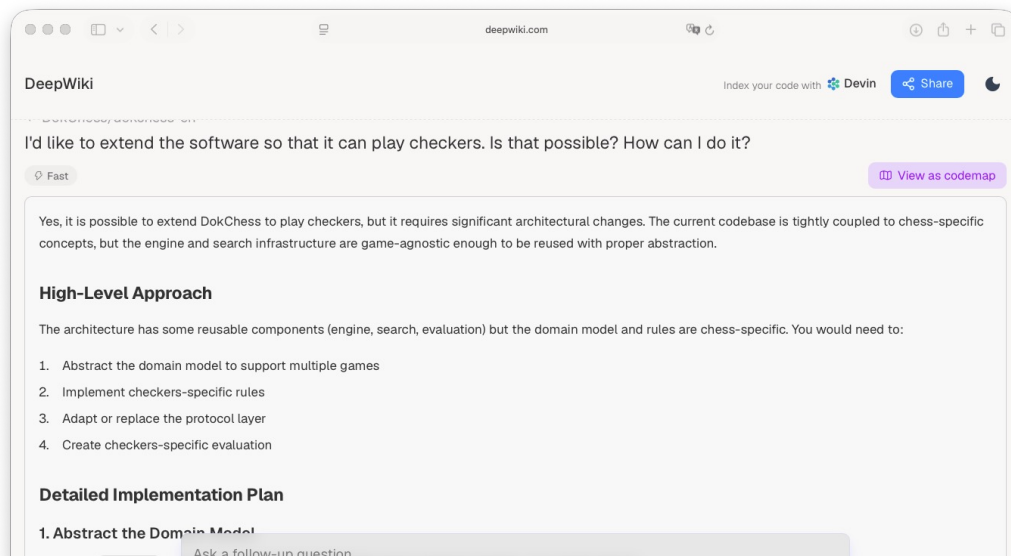


Beispiel für eine Konversation

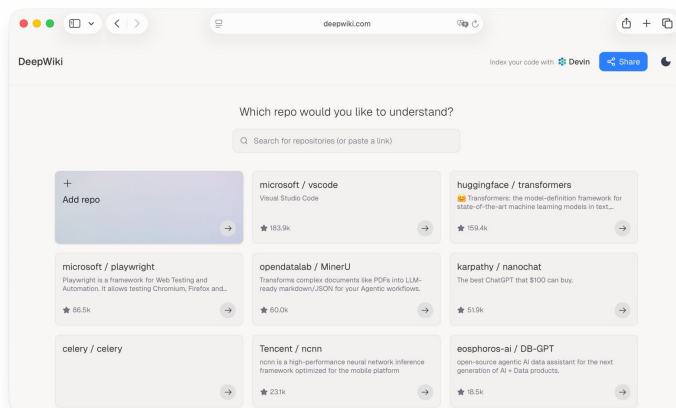




Beispiel für eine Konversation



Zum selbst ausprobieren ...



<https://deepwiki.com>



Mein Fazit zum Beispiel

Positiv

- Ergebnisse (Qualität der Texte) überzeugen
- Chat-Interface als möglicher neuer Zugang zur Dokumentation
- Perspektivisch: Vielleicht die Zukunft der Architekturdokumentation (?)



Probleme bzw. Offene Punkte

- Was passiert bei Änderungen in der Implementierung? Wie oft generiere ich neu? Bleibt meine Dokumentation stabil?
- Einsatz von SaaS um Quelltexte zu lesen in vielen Unternehmen und Organisationen kritisch.



Freie Alternativen („Nachbauten“), Auswahl



DeepWiki-Open

github.com/AsyncFuncAI/deepwiki-open



DeepWiki RS ("Litho")

github.com/sopaco/deepwiki-rs

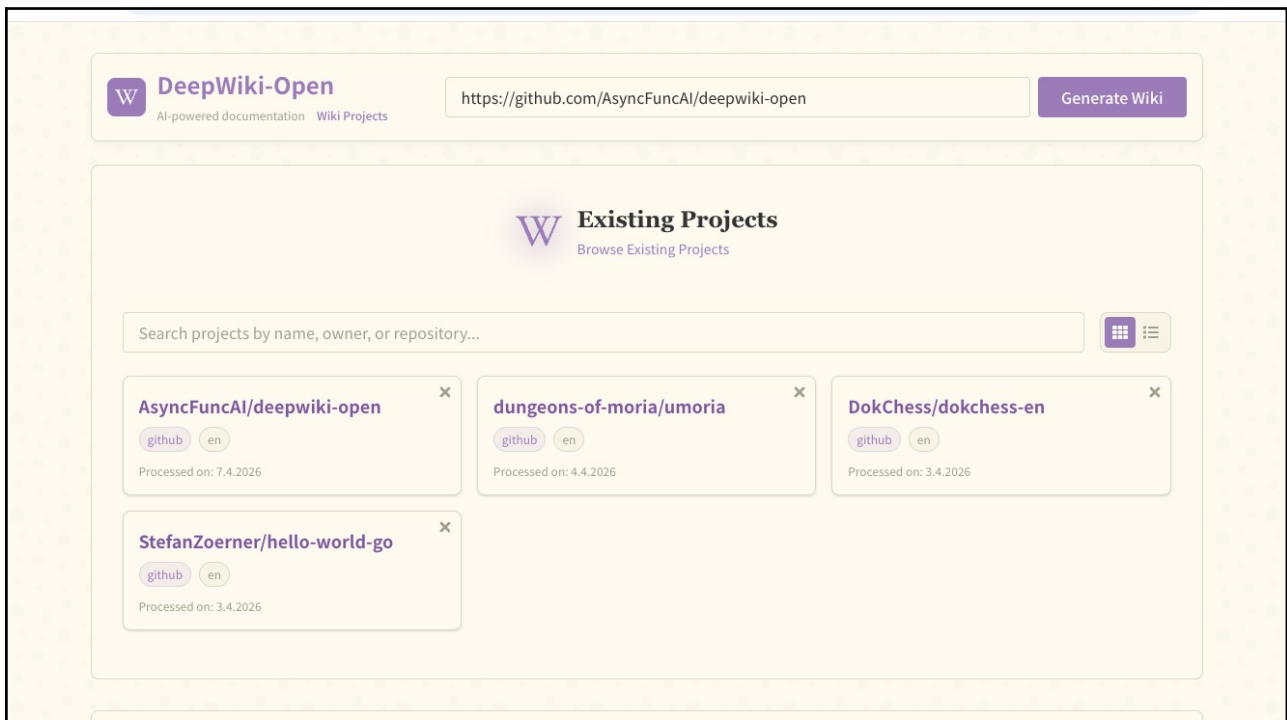
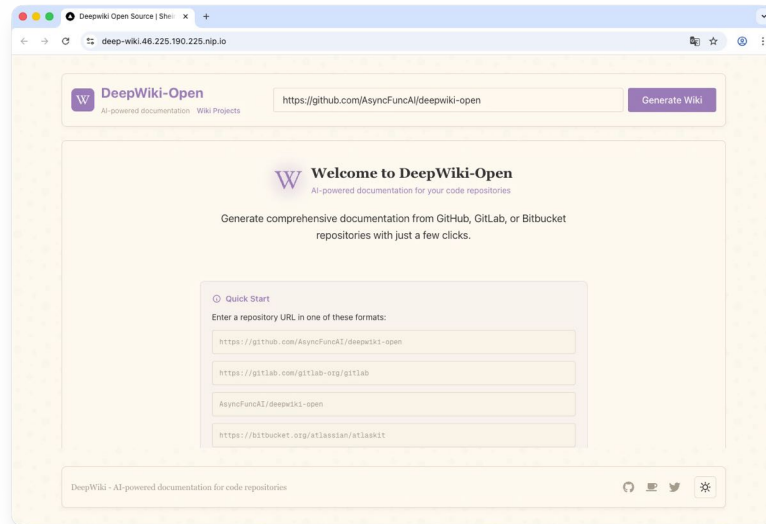


CodeWiki

github.com/FSoft-AI4Code/CodeWiki



DeepWiki-Open



Home

- Project Overview
- Historical Context
- System Architecture
 - Architecture Overview
 - Data Flow Diagram
 - Class Hierarchy
- Core Features
 - Monster Management System
 - Player Action System
 - Inventory System
- Data Management
 - Data Files Reference
 - Game State Management
- Frontend Components
 - UI Components

Monster Management System

Relevant source files

Monster Management System

Introduction

The Monster Management System in Umoria handles monster creation, behavior logic, and interactions with players and the environment. It maintains a centralized data structure for monster definitions, implements AI behaviors, and coordinates combat mechanics. This system is critical for game progression and maintains strict separation between monster data and gameplay logic.

Key responsibilities include:

- Monster spawning and population control
- AI state management (patrol, attack, flee)
- Combat resolution and damage calculation
- Special ability execution
- Death and experience distribution

This system interacts with the Player Management System and Item Management System through event-driven architecture patterns.

embarc.de

Architektur-Dokumentation: Ansätze mit GenAI

53

DeepWiki RS ("Litho")

Litho Book

litho_docs

Table of Contents

搜索文档内容...

1.Overview.md

2.Architecture.md

3.Workflow.md

4.Deep-Exploration

- Chess Rules Implementation.md
- Game State Management.md
- Move Evaluation Logic.md
- XBoard Communication.md

5.Boundary-Interfaces.md

__Litho_Summary_Brief___.md

__Litho_Summary_Detail___.md

1.Overview.md

System Context Overview

1. Project Introduction

Project Name and Description

Project Name: DokChess

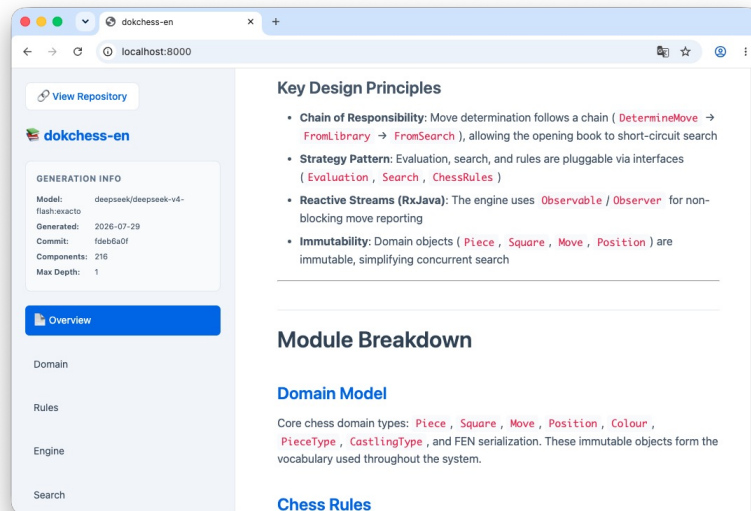
Description: DokChess is an open-source chess engine designed to facilitate gameplay and analysis through the XBoard protocol. It provides a robust backend service that manages game state, processes user commands, and evaluates moves, enabling both local and online chess play.

Core Functionality and Value

The core functionality of DokChess revolves around implementing a chess engine that adheres to the rules of chess while allowing for interaction through the XBoard protocol. The business value lies in providing a reliable and efficient chess engine that can be utilized by chess players and developers alike, enhancing the gaming experience and enabling the integration of chess functionalities into various applications.



CodeWiki



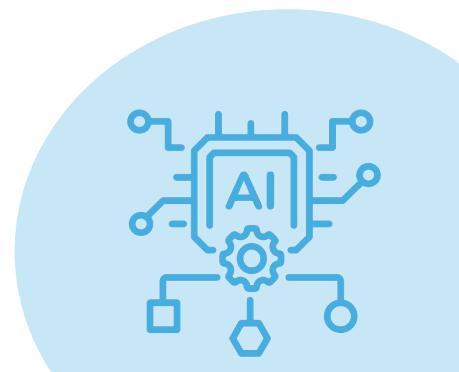
Fazit zum Ansatz allgemein

Inhalte aus der Implementierung ableiten

- Mit GenAI aus der Umsetzung strukturierte Dokumentation **destillieren**
- Quellen z.B.: **Codestrukturen**, Kommentare, Build-Skripte
- Ergebnis können **Texte oder Diagramme** sein

Denkbar z.B. für

- **Tech-Stack** inkl. der verwendeten Programmiersprachen und Frameworks
- **Bausteinsicht** (arc42, #5) mit Modulen, ihren Verantwortlichkeiten und Beziehungen, andere Sichten. **Laufzeitsicht** (arc42, #6)





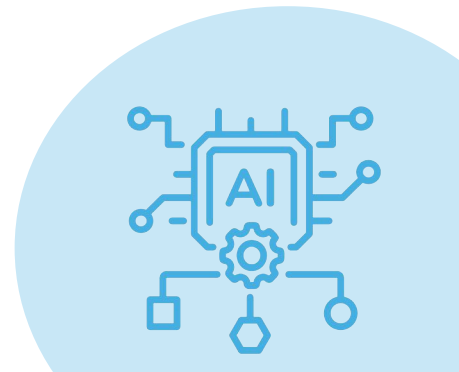
Diskussion des Ansatzes

Stärken aus meiner Sicht

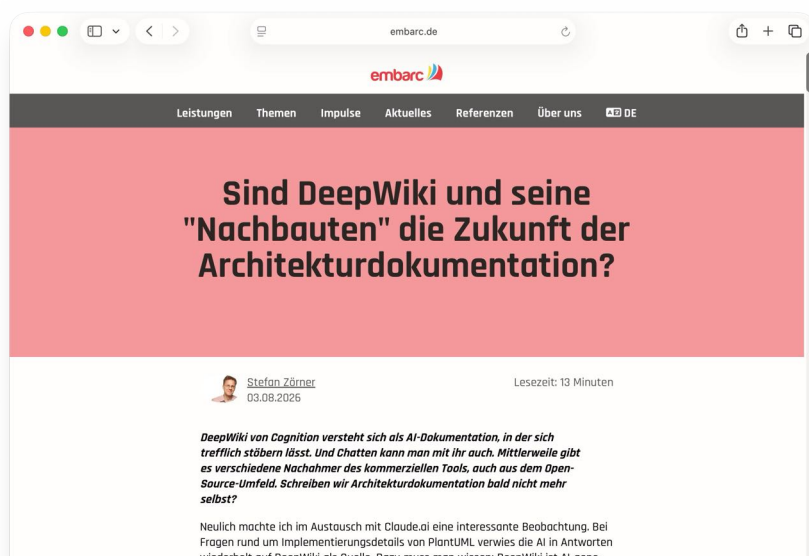
- Produziert (vergleichsweise) schnell vorzeigbare Ergebnisse, „Besser als Nichts“
- Oft ein guter Startpunkt

Kritisch zu sehen

- Für eine hohe Qualität vergleichsweise aufwändig (Dauer, Kosten)
- Wie arbeiten wir mit den Inhalten weiter? Speziell bei Änderungen im Quelltext

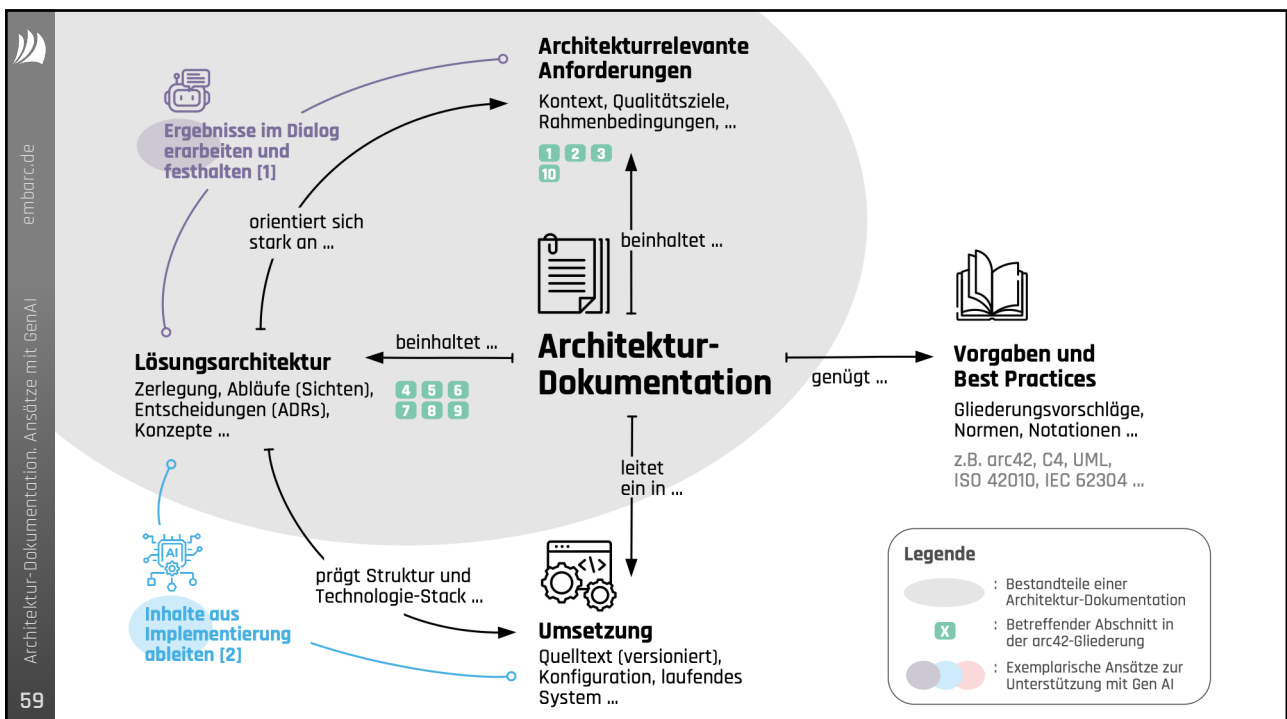
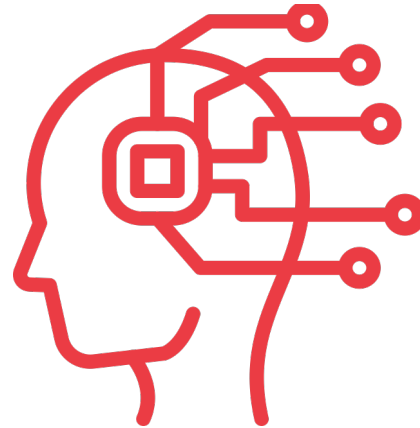


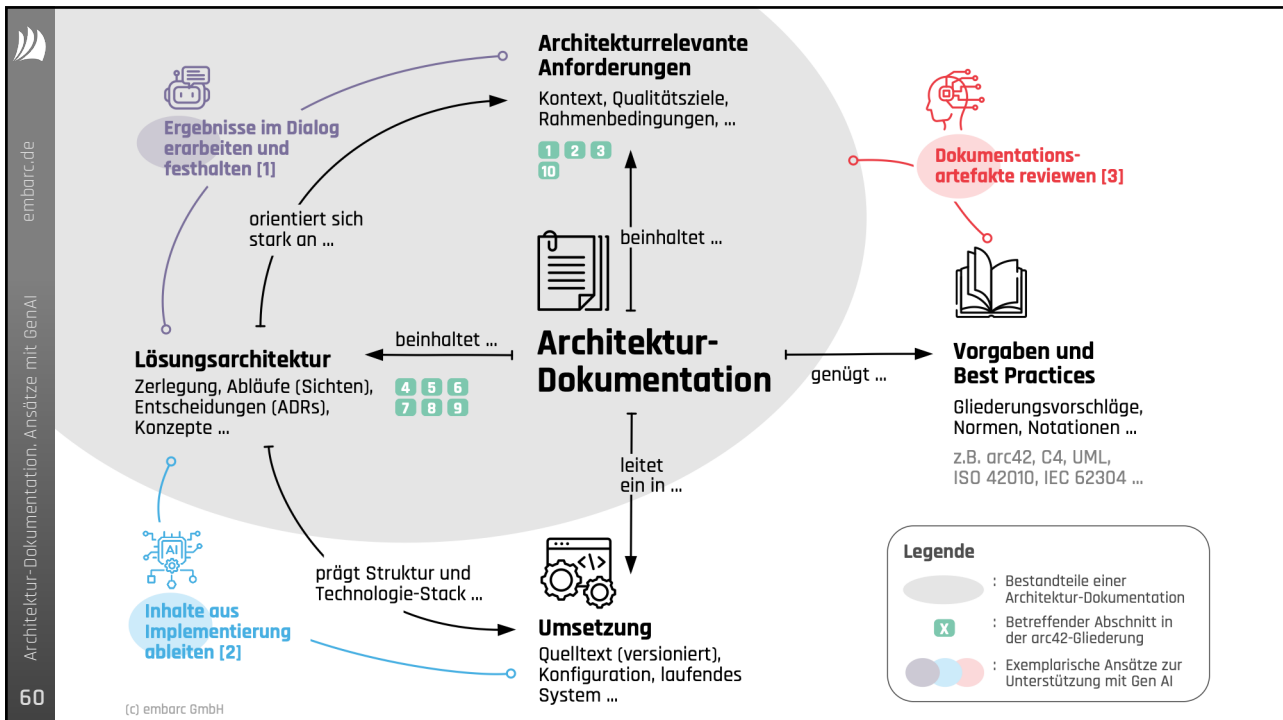
Blog-Beitrag zu DeepWiki und Alternativen



04.

Dokumentations- artefakte reviewen





Dokumentationsartefakte reviewen

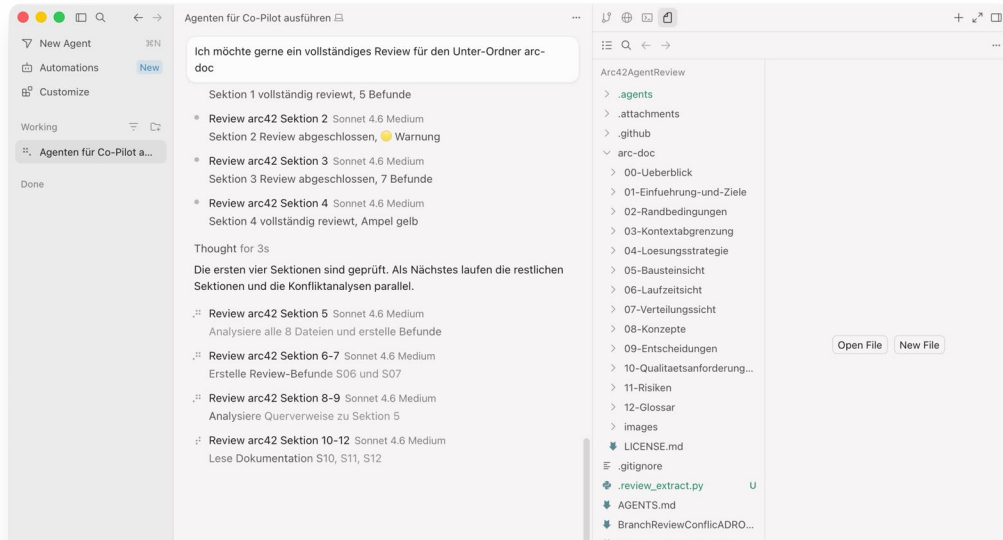
Der Ansatz allgemein:

- GenAI **begutachtet** systematisch eine bestehende Architekturdokumentation
- Überprüfung auf inhaltliche **Konsistenz**
- Einhaltung von **Vorgaben** und **Best Practices**





Beispiel: arc42agentic



Was ist arc42agentic?

Autor Matthias Nissen, frei verfügbar (Creative Commons)

- Agentensystem rund um arc42
- Review einer bestehenden Dokumentation (Markdown) gegen **Best Practices** von der arc42-Seite
- 12 spezialisierte **Agenten**, einer **pro Abschnitt** in arc42
- weitere **Konflikt-Agenten** prüfen Beziehungen zwischen Abschnitten (z. B. Qualitätsziele / Lösungsstrategie / Qualitätsszenarien)
- finden Widersprüche, die in Einzelkapiteln unsichtbar bleiben
- Läuft u.a. in in VS Code / GitHub Copilot

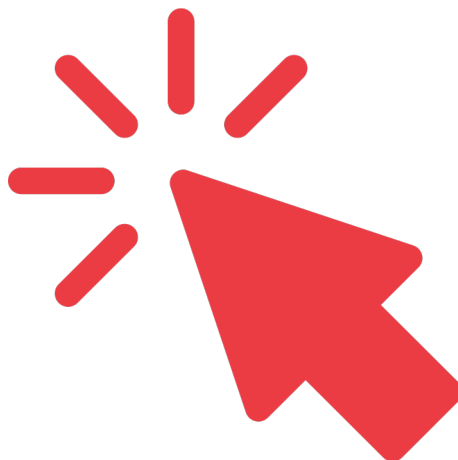


Prüf-Kriterien pro Abschnitt

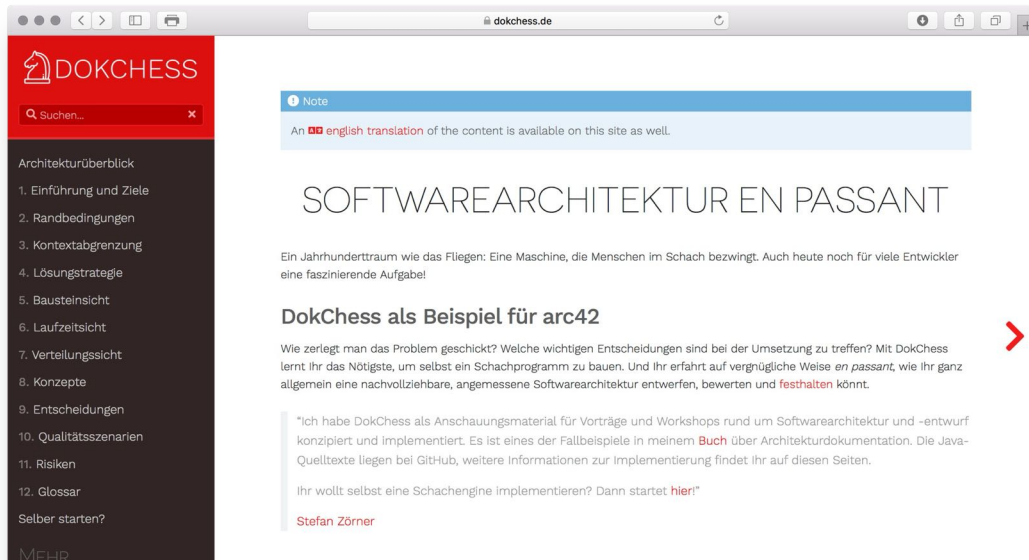
```
arc42agentic / .agents / arc42-review-s01-introduction.agent.md ↑ Top
Preview Code Blame 77 lines (54 loc) · 3.57 KB
16 ## Zu prüfende Dateien
22
23 ## Prüfkriterien
24
25 ### 1.1 Anforderungsüberblick (Requirements Overview)
26
27 **Formale Prüfung:**
28 - Existiert ein Abschnitt zum Anforderungsüberblick?
29 - Ist eine kurze textuelle Beschreibung oder tabellarische Darstellung vorhanden?
30 - Werden Verweise auf existierende Anforderungsdokumente gegeben?
31
32 **Inhaltliche Prüfung:**
33 - Werden die wesentlichen funktionalen Anforderungen und treibenden Kräfte beschrieben?
34 - Ist die Beschreibung kompakt und auf das Wesentliche fokussiert (nicht zu lang, nicht zu kurz)?
35 - Werden Business-Ziele des Systems hervorgehoben?
36 - Sind die Anforderungen gruppiert oder geclustert für bessere Übersicht?
37
38 ### 1.2 Qualitätsziele (Quality Goals)
39
40 **Formale Prüfung:**
41 - Existiert ein eigener Abschnitt für Qualitätsziele?
42 - Sind die Qualitätsziele als Tabelle oder Liste dargestellt?
43 - Sind maximal 3-5 Qualitätsziele aufgeführt?
44
45 **Inhaltliche Prüfung:**
46 - Handelt es sich um echte Architektur-Qualitätsziele (nicht Projektziele)?
47 - Sind die Qualitätsziele konkret und messbar (keine Buzzwords)?
48 - Werden konkrete Szenarien zu den Qualitätszielen angegeben?
49 - Sind die Qualitätsziele nach Priorität geordnet?
```



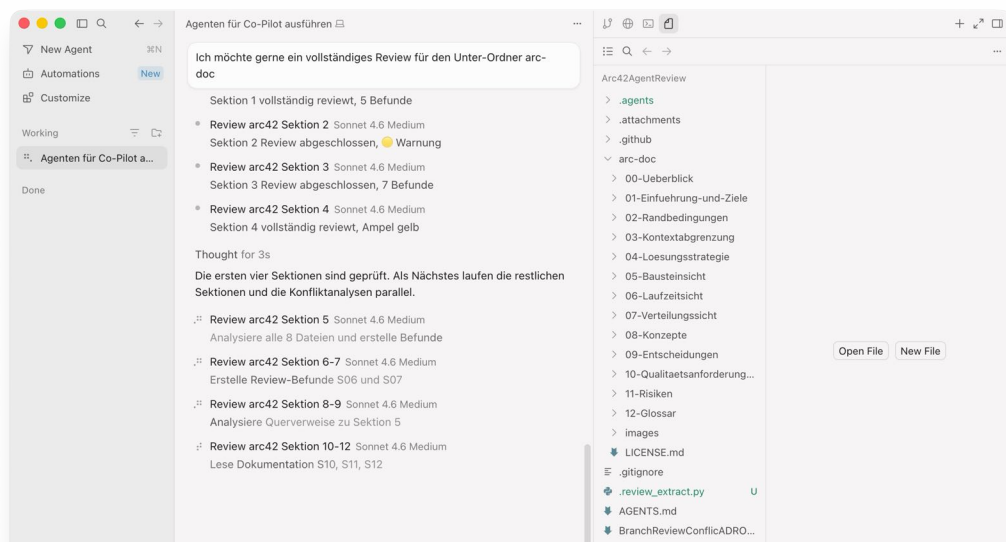
Kleine Demo



Beispiel-Input: dokchess.de



Prüfen z.B. in Cursor





Beispiel-Bericht (Zusammenfassung)

Zusammenfassung

Kategorie	Anzahl
● Kritische Befunde	4
● Warnungen / Empfehlungen	~52
● Hinweise	~43
Gesamt	~99

Handlungsempfehlungen

1. **Spielstärke-Qualitätsziel vereinheitlichen** — [04-01-Einstieg.md](#) an [01-02-Qualitätsziele.md](#) anpassen (KQS-01, S04-01).
2. **ADRs ins Deutsche übersetzen** — 09-01 und 09-02 (KCC-01, S09-01).
3. **Systemgrenze Eröffnungsbibliothek klären** — S3 und S5 konsistent modellieren (KKB-01).
4. **Mapping fachlich↔technisch ergänzen** — Tabelle in S3.2 (S03-04).
5. **Fehlende ADRs anlegen** — Minimax, Reactive Extensions, DI-Ansatz, Logging-Strategie (KSE-01-03, KKE-01/02).
6. **Sektion 11 vervollständigen** — Technische Schulden dokumentieren, Risikoübersicht mit Prioritäten (S11-01/02, KRQ-01-05).
7. **Blackbox-Beschreibungen bereinigen** — Implementierungsdetails aus S5.5, S5.7, S5.8 in Whitebox-Abschnitte verschieben (S05-02-06).



Konflikte im Detail

Sektionsübergreifende Konflikte

Konfliktdimension	Status	Befunde
Qualitätsstrang (S1 ↔ S4 ↔ S10)	●	Spielstärke-Ziel inkonsistent benannt; verwaiste Szenarien Z01/Z02, P01
Strategie ↔ Entscheidungen (S4 ↔ S9)	●	3 von 6 Schlüsselansätzen ohne ADR
Constraint-Compliance (S2 ↔ S4/S8/S9)	●	ADRs auf Englisch; Hardware-Nachweis unvollständig
Kontext ↔ Bausteine (S3 ↔ S5)	●	Systemgrenze Eröffnungsbibliothek widersprüchlich modelliert
Sichten-Konsistenz (S5 ↔ S6 ↔ S7)	●	Namensabweichungen, fehlende Schnittstellen in S5.2
Konzepte ↔ Entscheidungen (S8 ↔ S9)	●	2 Architekturentscheidungen in S8 statt S9
Risiken ↔ Qualität (S11 ↔ S1/S10)	●	3 von 5 Qualitätszielen ohne Risikoabdeckung



Beispiel-Befund (1)

Kritische Konflikte (Detail)

[KQS-01] Inkonsistentes Spielstärke-Qualitätsziel

Konflikttyp: K1 — Benennungswiderspruch

Schwere: ● Kritisch

Betroffene Dateien:

- [01-02-Qualitätsziele.md](#) — „Akzeptable Spielstärke (Funktionale Eignung)“
- [04-01-Einstieg.md](#) — „Attraktive Spielstärke (Attraktivität)“

Beschreibung: Adjektiv und ISO-25010-Kategorie weichen ab. „Attraktiv“ impliziert eine höhere Messlatte als „akzeptabel“; die Kategorie wechselt von Funktionale Eignung zu Attraktivität (Benutzbarkeit).

Lösungsvorschlag: In [04-01-Einstieg.md](#) Zeile 12 angleichen:

Akzeptable Spielstärke (Funktionale Eignung)



Beispiel-Befund (2)

[KCC-01] ADRs verletzen Dokumentationssprache Deutsch

Konflikttyp: K2 — Constraint-Verletzung

Schwere: ● Kritisch

Betroffene Dateien:

- [02-03-Konventionen.md](#) — Konvention: Dokumentation auf Deutsch
- [09-01-Anbindung.md](#), [09-02-Stellungsobjekte.md](#) — vollständig englische ADR-Texte (Status, Context, Decision, ...)

Lösungsvorschlag: Beide ADRs ins Deutsche übersetzen, Nygard-Struktur beibehalten.



Beispiel-Befund (3)

[KKB-01] Eröffnungsbibliothek: Fremdsystem vs. internes Subsystem

Konflikttyp: K1 — Systemgrenzen-Widerspruch

Schwere: ● Kritisch

Betroffene Dateien:

- [03-02-Technischer-Kontext.md](#) — „Polyglot Opening Book (Fremdsystem)“
- [05-05-Eroeffnung.md](#) — internes Subsystem mit `PolyglotOpeningBook`-Implementierung

Beschreibung: In S3 erscheint die Eröffnungsbibliothek als externes Fremdsystem; in S5 ist sie ein internes Subsystem. Die Systemgrenze zur Polyglot-Datei ist auf Ebene 1 nicht als Membran-Interaktionspunkt sichtbar.

Lösungsvorschlag: In S3 präzisieren, dass „Fremdsystem“ die externe **Datei** meint, nicht den Code-Baustein. In S5.1-Diagramm die Datei-Schnittstelle als externen Interaktionspunkt markieren.



Mein Fazit zum Beispiel

Positiv

- Ergebnisse sind gut nachvollziehbar und inhaltlich stimmig
- Gezielt einzelne Bereiche prüfbar, oder auch nur auf Konflikte
- Die Befunde sind inhaltlich tatsächlich gut.

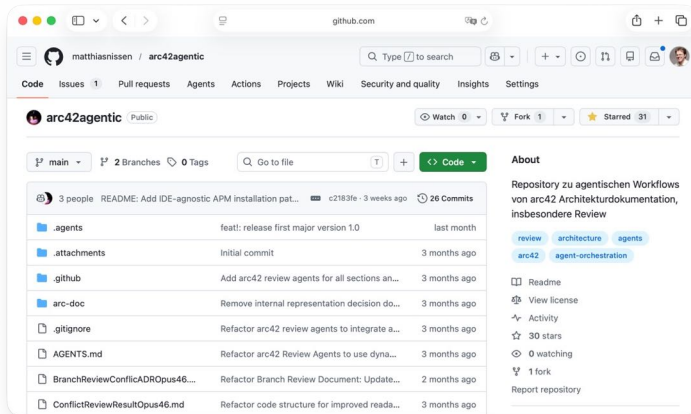
Probleme bzw. Offene Punkte



- Die Ergebnisse variieren je nach eingesetztem LLM stark.
- Dauer der Prüfung ist je Deployment sehr unterschiedlich, und kann durchaus in die Stunden gehen.



Zum selbst ausprobieren ...



<https://github.com/matthiasnissen/arc42agentic>



Fazit zum Ansatz allgemein

Der Ansatz allgemein:

- GenAI **begutachtet** systematisch eine bestehende Architekturdokumentation
- Überprüfung auf inhaltliche **Konsistenz**
- Einhaltung von **Vorgaben** und **Best Practices**

Denkbar z.B. für

- Unternehmenseigene Vorgaben
- Inhalte aller Art - vor allem aber Texte
- Unterstützung beim Reviewen und zeitnah bereits beim Anfertigen





Diskussion des Ansatzes

Stärken aus meiner Sicht

- Gut automatisierbar und in den Build-Prozess integrierbar
- Vergleichsweise leicht umzusetzen

Kritisch zu sehen

- Qualität hängt stark an den Kriterien
- Ergebnisse schwer reproduzierbar (heikel in formalen Prozessen)



05.

Fazit und weitere Informationen

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

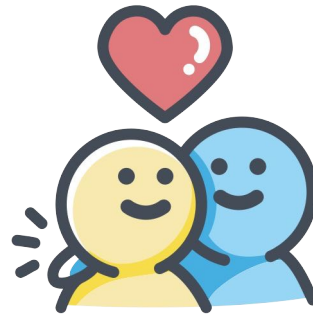
03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Fazit und weitere Informationen



Perfect Match?



Arc.-Dok. + GenAI

Textlastig + unbeliebt + maschinell gut erzeugbar.

Der Einsatz von GenAI in Architekturdokumentation liegt zumindest nahe.



embarc.de

Architektur-Dokumentation: Ansätze mit GenAI

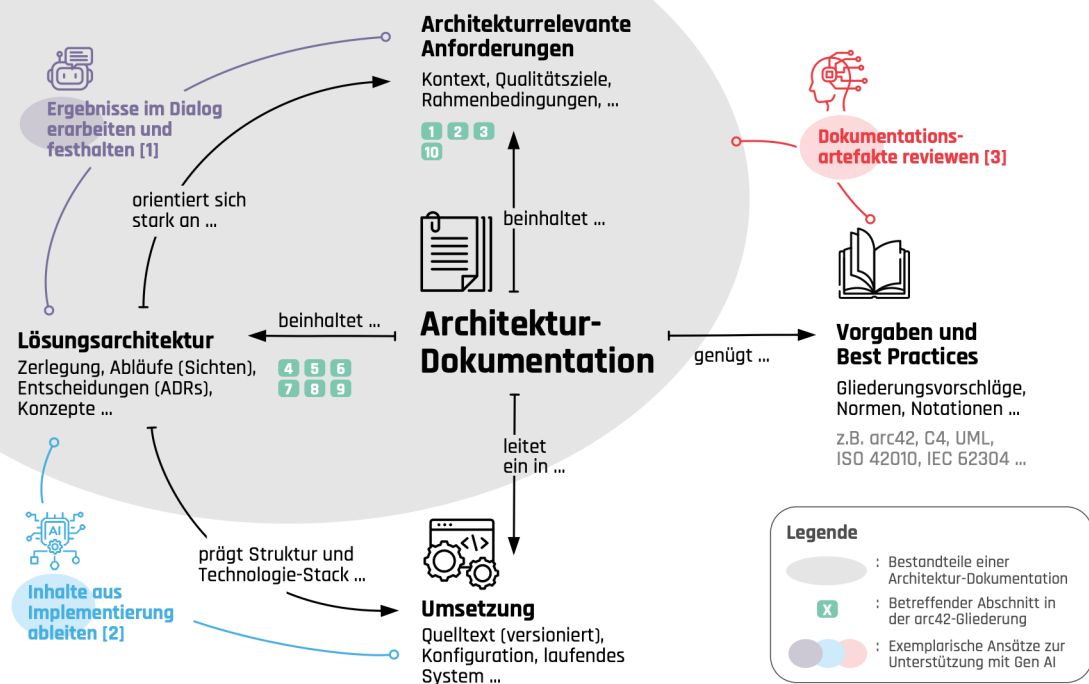
78



embarc.de

Architektur-Dokumentation: Ansätze mit GenAI

79



(c) embarc GmbH



TL;DR (Too Long; Didn't Read)

GenAI eröffnet für Architekturdokumentation **spannende Möglichkeiten**: als Sparringspartner im Dialog, zum Ableiten von Inhalten aus der Implementierung und zum Reviewen bestehender Artefakte.

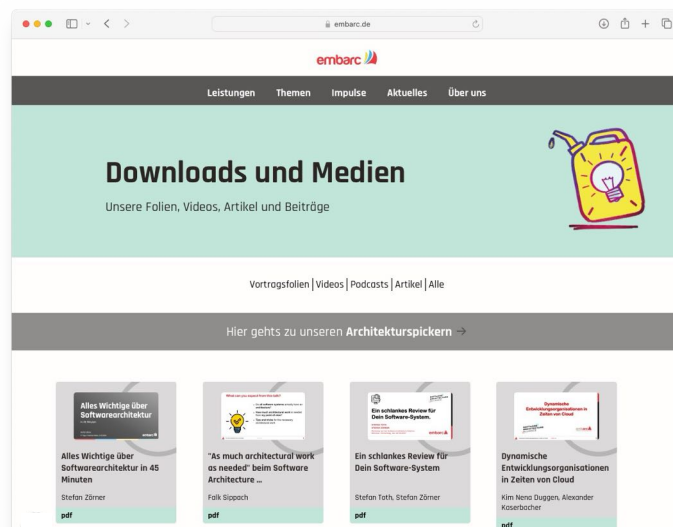
Die Ergebnisse überzeugen oft schon im **ersten Wurf**. Sie bleiben aber **Vorschläge**, die wir prüfen müssen. LLMs suggerieren leicht ein Fachwissen, das sie nicht haben.

Die mit den Zielen verbundenen Aufgaben bleiben bestehen, doch GenAI nimmt uns Routine ab und **senkt die Hürde**, überhaupt anzufangen.

Schreiben wir Architekturdokumentation bald nicht mehr selbst? Nein. **Aber wir schreiben sie seltener allein.**



Folien als PDF zum Download



➔ embarc.de/download/

Artikel zum Qualitätsziele-Beispiel

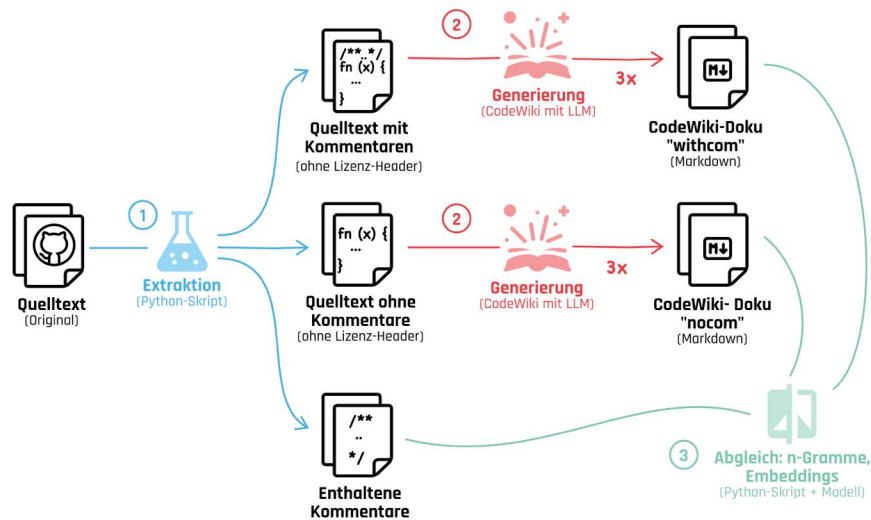


Weiterer Blog-Beitrag zu CodeWiki





Weiterer Blog-Beitrag zu CodeWiki



Kommentar-Fund in Dokumentation

The screenshot shows a code editor with a C++ file named 'rng.cpp'. The code includes a header file 'headers.h' and defines a test function. It uses a 'prime modulus multiplicative congruential generator (PMMLCG), also known as a Lehmer Grammer'. The documentation below the code, titled 'Overview', describes the 'random_number_generator' module, which provides a pseudo-random number generation implementation based on the same PMMLCG algorithm. The documentation references a 1988 paper by Stephen K. Park and Keith W. Miller.

```

rng.cpp
8  #include "headers.h"
9
10 // Define this to compile as a standalone test
11 // #define TEST_RNG
12
13 // This alg uses a prime modulus multiplicative congruential generator
14 // (PMMLCG), also known as a Lehmer Grammer, which satisfies the following
15 // properties
16 //
17 // (i) modulus: m - a large prime integer
18 // (ii) multiplier: a - an integer in the range 2, 3, ..., m - 1
  
```

Overview

The random_number_generator module provides a pseudo-random number generation implementation based on a prime modulus multiplicative congruential generator (PMMLCG), also known as a Lehmer generator. This implementation follows the algorithm described by Stephen K. Park and Keith W. Miller in their 1988 paper "Random Number Generators: Good ones are hard to find".

The module is designed to produce high-quality pseudo-random integers suitable for

Unser Line-Up

Vorträge, Workshops, neue Perspektiven und Plaudern
in lockerer Punsch-Atmosphäre am 7. Dezember 2026!



Ulrich Lehner
(LVM Versicherungen)



Tom Asel
(tangible concepts)



Roman Hecht
(Continental Reifen)



Niklas Trentmann



Falk Sippach



Michael Rabe
(Continental Reifen)



Stefan Zörner



Stefan Toth



Alle Infos & Anmeldung: embarc.de/architektur-punsch/



Vielen Dank.

Gerne Verlinken auf LinkedIn ...

linkedin.com/in/stefan-zoerner

