Curriculum für

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Modul
ADOC

Architekturdokumentation

2025.1-RC1-DE-20251018

Inhaltsverzeichnis

Die Abkürzung "e. V." ist Teil des offiziellen Namens des iSAQB und steht für "eingetragener Verein", der seinen Status als juristische Person nach deutschem Recht beschreibt. Der Einfachheit halber wird iSAQB e. V. im Folgenden ohne die Verwendung dieser Abkürzung als iSAQB bezeichnet.

Lernziele im Überblick

- LZ 1-1: Nutzen und Ziele von Architekturdokumentation verstehen
- LZ 1-2: Bedeutung von Architekturdokumentation für unterschiedliche Stakeholder verstehen
- LZ 1-3: Architekturdokumentation gegen andere Arten von Dokumentation abgrenzen
- LZ 1-4: Passende Notationen für die Architekturdokumentation auswählen
- LZ 1-5: Dokumentation an verschiedene Entwicklungsvorgehen anpassen
- LZ 2-1: Architekturdokumentation erstellen
- LZ 2-2: Zielgruppen und Ziele der Architekturdokumentation ermitteln
- LZ 2-3: Passende Struktur für Architekturdokumentation entwerfen
- LZ 2-4: Passendes Vorgehen für die Architekturdokumentation definieren
- LZ 2-5: Anwendungsszenarien für GenAI in der Dokumentation identifizieren
- LZ 2-6: Passende Dokumentations-Templates kennen und kontextgerecht auswählen
- LZ 3-1: Einflussfaktoren und Randbedingungen dokumentieren
- LZ 3-2: Architekturentscheidungen dokumentieren
- LZ 3-3: Unterschiedliche Sichten zur Architekturdokumentation einsetzen
- LZ 3-4: Übergreifende Konzepte dokumentieren
- LZ 3-5: Konsistenz zwischen Teilen der Dokumentation sicherstellen
- LZ 4-1: Passende Werkzeuge für Dokumentationsarbeit auswählen
- LZ 4-2: Anforderungen an Dokumentationswerkzeuge und -artefakte definieren
- LZ 4-3: Docs-As-Code Ansatz kennen
- LZ 4-4: GenAI-Werkzeuge als Unterstützung für Architekturdokumentation anwenden
- LZ 5-1: Sicherstellen der Gebrauchstauglichkeit von Dokumentation
- LZ 5-2: Differenzieren verschiedener Arten der Begutachtung
- LZ 5-3: Differenzieren zwischen Reviews von Architektur und Dokumentation
- LZ 5-4: Ziele von Dokumentations-Reviews ermitteln und erklären
- LZ 5-5: Erstellen von Checklisten und Fragenkatalogen für Reviews
- LZ 5-6: Verschiedene Rollen bei Dokumentations-Reviews ausüben können
- LZ 5-7: Verbessern der Qualität von Dokumentation auf Basis von Review-Ergebnissen
- LZ 6-1: Vor- und Nachteile beispielhafter Architekturdokumentation erklären

Einführung: Allgemeines zum iSAQB Advanced Level

Was vermittelt ein Advanced Level Modul?

Das Modul kann unabhängig von einer CPSA-F-Zertifizierung besucht werden.

- Der iSAQB Advanced Level bietet eine modulare Ausbildung in drei Kompetenzbereichen mit flexibel gestaltbaren Ausbildungswegen. Er berücksichtigt individuelle Neigungen und Schwerpunkte.
- Die Zertifizierung erfolgt als Hausarbeit. Die Bewertung und mündliche Prüfung wird durch vom iSAQB benannte Experten vorgenommen.

Was können Absolventen des Advanced Level (CPSA-A)?

CPSA-A-Absolventen können:

- eigenständig und methodisch fundiert mittlere bis große IT-Systeme entwerfen
- in IT-Systemen mittlerer bis hoher Kritikalität technische und inhaltliche Verantwortung übernehmen
- Maßnahmen zur Erreichung von Qualitätsanforderungen konzeptionieren, entwerfen und dokumentieren sowie Entwicklungsteams bei der Umsetzung dieser Maßnahmen begleiten
- architekturrelevante Kommunikation in mittleren bis großen Entwicklungsteams steuern und durchführen

Voraussetzungen zur CPSA-A-Zertifizierung

- erfolgreiche Ausbildung und Zertifizierung zum Certified Professional for Software Architecture, Foundation Level® (CPSA-F)
- mindestens drei Jahre Vollzeit-Berufserfahrung in der IT-Branche; dabei Mitarbeit an Entwurf und Entwicklung von mindestens zwei unterschiedlichen IT-Systemen
    - Ausnahmen sind auf Antrag zulässig (etwa: Mitarbeit in Open-Source-Projekten)
- Aus- und Weiterbildung im Rahmen von iSAQB-Advanced-Level-Schulungen im Umfang von mindestens 70 Credit Points aus mindestens drei unterschiedlichen Kompetenzbereichen
- erfolgreiche Bearbeitung der CPSA-A-Zertifizierungsprüfung

Grundlegendes

Was vermittelt das Modul „ADOC“?

ADOC vermittelt die Fähigkeiten:

- die Dokumentation von Softwarearchitekturen selbständig zu konzipieren und zu erstellen
- bestehende Dokumentation bezüglich ihrer Eignung und Angemessenheit zu bewerten
- bestehende Dokumentation zu verbessern.

Teilnehmende lernen, die wesentlichen Aspekte von Softwarearchitekturen (u. a. Entscheidungen, Strukturen, Konzepte, Qualitätsanforderungen und Sichten) systematisch zu beschreiben. Dazu stellt ADOC grundlegende Beschreibungs- und Notationsmittel sowie mögliche Werkzeuge zur praktischen Umsetzung vor.

Struktur des Lehrplans und empfohlene zeitliche Aufteilung

Inhalt	Empfohlene Mindestdauer (min)	Übungszeit (min)
1. Grundbegriffe	70	20
2. Vorgehen bei Architekturdokumentation	90	60
3. Bestandteile von Architekturdokumentationen	90	90
4. Werkzeuge	90	30
5. Dokumentation bewerten	75	45
6. Beispiele für Dokumentation	60	0
Summe	475	245

Gesamtdauer: 720 min (12h)

Dauer, Didaktik und weitere Details

Die unten genannten Zeiten sind Empfehlungen. Die Dauer einer Schulung zum Modul ADOC sollte mindestens 2 Tage betragen, kann aber länger sein. Anbieter können sich durch Dauer, Didaktik, Art und Aufbau der Übungen sowie der detaillierten Kursgliederung voneinander unterscheiden. Insbesondere die Art der Beispiele und Übungen lässt der Lehrplan komplett offen.

Lizenzierte Schulungen zu ADOC tragen zur Zulassung zur abschließenden Advanced-Level-Zertifizierungsprüfung folgende Credit Points) bei:

Methodische Kompetenz:	20 Punkte
Technische Kompetenz:	0 Punkte
Kommunikative Kompetenz:	0 Punkte

Voraussetzungen

Teilnehmende sollten folgende Kenntnisse und/oder Erfahrung mitbringen:

- Grundlagen der Beschreibung von Architekturen mit Hilfe verschiedener Sichten, übergreifender Konzepte, Entwurfsentscheidungen, Randbedingungen etc., wie es im CPSA- F Foundation Level vermittelt wird.
- Wünschenswert sind eigene Erfahrungen in der Erstellung und Pflege technischer Dokumentation von Software, insbesondere der Architektur von Software- oder Software-nahen Systemen.

Hilfreich für das Verständnis einiger Konzepte sind darüber hinaus:

- Kenntnis typischer Herausforderungen bei Erstellung und Pflege von technischer Dokumentation:
  - Auswahl geeigneter Dokumentationsstrukturen, Notationen, Ergebnistypen (Stakeholderorientierung)
  - Behandlung großer Dokumentationen (insbesondere vorhandener oder veralteter Dokumentation)
  - Auswahl, Konfiguration und Einführung von Werkzeugketten (mit welchen Mitteln kann Dokumentation angemessen erstellt und gepflegt werden),
  - Versionierung von Dokumenten (Modelle und Texte)
  - Dokumentation in Teams, arbeitsteilige Erstellung und Pflege
  - Inhaltsbezogene und formale Reviews von Dokumentation.

Gliederung des Lehrplans

Die einzelnen Abschnitte des Lehrplans sind gemäß folgender Gliederung beschrieben:

- **Begriffe/Konzepte**: Wesentliche Kernbegriffe dieses Themas.
- **Unterrichts-/Übungszeit**: Legt die Unterrichts- und Übungszeit fest, die für dieses Thema bzw. dessen Übung in einer akkreditierten Schulung mindestens aufgewendet werden muss.
- **Lernziele**: Beschreibt die zu vermittelnden Inhalte inklusive ihrer Kernbegriffe und -konzepte.

Dieser Abschnitt skizziert damit auch die zu erwerbenden Kenntnisse in entsprechenden Schulungen.

1. Grundbegriffe von Architekturdokumentation

Dauer: 70 Min.	Übungszeit: ca. 20 Min.

1.1. Begriffe und Konzepte

Softwarearchitektur, Dokumentation, Stakeholder, Notation, Vorgehensmodell, Lösungsentwurf, UML.

1.2. Lernziele

LZ 1-1: Nutzen und Ziele von Architekturdokumentation verstehen

Softwarearchitekt:innen wissen, dass Architekturdokumentation kein Selbstzweck ist.

Sie können den Nutzen und verschiedene Ziele von Architekturdokumentation erklären, beispielsweise:

- Architekturdokumentation unterstützt beim Lösungsentwurf, und bei der Kommunikation der Lösung dem Team und anderen (z. B. Auftraggebern) gegenüber.
- Unterstützung der Entwicklung durch Vermittlung architektur- und entwicklungsrelevanter Entscheidungen, Fakten und Gründe.
- Schriftliche Kommunikation relevanter struktureller und konzeptioneller Sachverhalte
- Konservierung von Wissen über Entwicklung und Architektur

Sie können diese Ziele und den Nutzen von Dokumentation gegenüber Stakeholdern überzeugend vertreten.

LZ 1-2: Bedeutung von Architekturdokumentation für unterschiedliche Stakeholder verstehen

Softwarearchitekt:innen verstehen, dass unterschiedliche Stakeholder i. d. R. verschiedene Anforderungen an Architekturdokumentation haben, beispielsweise hinsichtlich:

- Arten von Informationen, etwa zu Schnittstellen, Bausteinen, querschnittlichen Themen, Entscheidungen, technischen Risiken, grundlegenden Lösungsansätzen o. ä.
- Darstellung solcher Informationen, etwa als (grafische oder textuelle) Modelle, als Grafiken oder Text
- Detaillierungsgrad, etwa: Überblick, abstrakte oder detaillierte Darstellung
- technische Repräsentation etwa mit oder ohne Schreib-/Änderungsmöglichkeiten, als PDF, HTML, Wiki, Textverarbeitung o. ä.
- Grad der Aktualität.

LZ 1-3: Architekturdokumentation gegen andere Arten von Dokumentation abgrenzen

Softwarearchitekt:innen verstehen, dass Architekturdokumentation Berührungspunkte und teilweise Überdeckung zu anderen Arten von Dokumentation besitzt, beispielsweise:

- Anforderungs-/Requirements-Dokumentation
- Implementierungsdokumentation
- Managementdokumentation
- Testdokumentation

- Betriebsdokumentation

LZ 1-4: Passende Notationen für die Architekturdokumentation auswählen

Softwarearchitekt:innen kennen verschiedene Notations- und Darstellungsmittel für Architekturinformationen, beispielsweise:

- Modellierungsmethoden, wie UML, C4 ([Brown (C4)]), ArchiMate, Business-Process Modeling Notation (BPMN), Entity-Relationship (ER) Diagramme, Ereignis-Prozess-Ketten (EPK)
- Textuelle Erläuterungen von Entscheidungen
- Canvas (e.g. Architecture-Communication-Canvas [ACC] oder Bounded-Context-Canvas [DDD-Canvas])

Sie wissen, dass formalere Notationen wie UML für Architekturdokumentation hilfreich sein können, aber nicht zwingend erforderlich sind.

LZ 1-5: Dokumentation an verschiedene Entwicklungsvorgehen anpassen

Softwarearchitekt:innen wissen, dass

- in eher formalen Entwicklungsvorhaben (etwa *safety-critical*) hohe Ansprüche an Architekturdokumentation gelten
- in eher leichtgewichtigen Entwicklungsprozessen (agilen Prozesse) Teams eine dem System und dessen Umfeld angemessene Art von Dokumentation wählen sollten
- je nach Vorgehen und Rahmenbedingungen des Vorhabens sehr unterschiedliche Dokumentationsarbeiten zu leisten sind

2. Vorgehen bei Architekturdokumentation

Dauer: 90 Min.	Übungszeit: 60 Min.

2.1. Begriffe und Konzepte

Vorgehensmodell, Rolle, Artefakt, iterativ/inkrementell, Lebenszyklus, agil.

2.2. Lernziele

LZ 2-1: Architekturdokumentation erstellen

Softwarearchitekt:innen können angemessene Architekturdokumentation für mittlere bis große IT-Systeme in beispielsweise folgenden Situationen erstellen:

- Erstellung neuer Systeme
- Weiterentwicklung bestehender Systeme mit vorhandener Dokumentation
- Nachdokumentation bestehender Systeme ohne vorhandene Dokumentation
- Dokumentation in Ausnahmesituationen (wenig Budget, wenig Zeit, wenig verfügbare Informationsquellen etc.)

Sie können Zielgruppen und Ziele der Dokumentation ermitteln (siehe LZ 2-2).

Details zu den notwendigen methodischen Mitteln finden sich in den Lernzielen LZ 3-1 bis LZ 3-5.

LZ 2-2: Zielgruppen und Ziele der Architekturdokumentation ermitteln

Softwarearchitekt:innen können die Zielgruppen („Stakeholder“) für Architekturdokumentation ermitteln (beispielsweise Management, Entwicklungsteam, fachliche Stakeholder, betriebliche Stakeholder, Test/QS).

Sie können zusammen mit diesen Stakeholdern konkrete Ziele der Dokumentation ermitteln.

LZ 2-3: Passende Struktur für Architekturdokumentation entwerfen

Softwarearchitekt:innen können für gegebene Situationen und Systeme eine passende Struktur für die Architekturdokumentation auswählen oder erarbeiten.

Als Basis dafür dienen Zielgruppen und Ziele (siehe LZ 2-2) sowie Templates (siehe LZ 2-5).

LZ 2-4: Passendes Vorgehen für die Architekturdokumentation definieren

Softwarearchitekt:innen finden für gegebene Situationen ein passendes Vorgehen zur Dokumentation. Sie können Stakeholdern und dem Entwicklungsteam die Vor- und Nachteile verschiedener Ansätze erläutern, etwa:

- Dokumentation entsteht **begleitend zur** Entwicklungsarbeit
- Dokumentation entsteht **vor** der Entwicklung/Implementierung
- Dokumentation entsteht **nach** der Entwicklung („Dokumentations-Sprint“)
- Dokumentation wird durch Personen **aus dem Entwicklungsteam** erstellt und gepflegt

- Dokumentation wird durch Personen **außerhalb des Entwicklungsteams** erstellt und gepflegt

Sie wissen, dass agile Entwicklung und Architekturdokumentation nicht im Widerspruch zueinander stehen, und können dies Stakeholdern gegenüber begründen.

LZ 2-5: Anwendungsszenarien für GenAI in der Dokumentation identifizieren

Softwarearchitekt:innen wissen, wo und wie der Einsatz von GenAI-Werkzeugen bei Erstellung, Pflege und Nutzung technischer Dokumentation Mehrwert bieten kann. Dies umfasst beispielsweise:

- Generieren von Dokumentation aus Code-Artefakten
- Identifizieren und sichern architekturrelevanter Anforderungen
- Abgleichen bestehender Dokumentation mit Quellcode
- KI-Modelle als Ersatz für statische Dokumentation einsetzen
- Informationen aus unstrukturierten oder schwach strukturierten Quellen (z. B. E-Mails, Chatverläufe, Tickets) extrahieren und mit bestehenden Inhalten verknüpfen

LZ 2-6: Passende Dokumentations-Templates kennen und kontextgerecht auswählen

Softwarearchitekt:innen kennen Beispiele von Architekturtemplates, etwa:

- arc42 ([arc42],[Starke+2023],[Starke+2016])
- C4 ([Brown (C4)], [Brown (Guidebook)])
- ISO/IEE 42010 ([ISO/IEC 42010:2022] „Recommended Practice for Architecture Description of Software-Intensive Systems“)

3. Bestandteile von Architekturdokumentationen

Dauer: 90 Min.	Übungszeit: 90 Min.

3.1. Begriffe und Konzepte

Sichten, Entscheidungen, übergreifende Konzepte und Themen, Schnittstellen, Strukturen, Randbedingungen, Risiken, Qualitätsziele.

3.2. Lernziele

LZ 3-1: Einflussfaktoren und Randbedingungen dokumentieren

Softwarearchitekt:innen können Faktoren, die Einfluss auf Softwarearchitektur haben, identifizieren und geeignet dokumentieren. Dazu zählen:

- Randbedingungen
- Qualitätsanforderungen und -ziele
- Technische Risiken

LZ 3-2: Architekturentscheidungen dokumentieren

Softwarearchitekt:innen können Architekturentscheidungen nachvollziehbar festhalten. Sie kennen beispielsweise *Architecture Decision Records* als eine feste Struktur für solche Entscheidungen (siehe [Nygard 2011], [ADR-GitHub])

LZ 3-3: Unterschiedliche Sichten zur Architekturdokumentation einsetzen

Softwarearchitekt:innen können verschiedene Architektursichten angemessen einsetzen und dokumentieren. Sie wissen, für welche Anwendungsfälle statische sowie dynamische Sichten am besten geeignet sind. Außerdem verstehen sie, wann welcher Detailgrad sinnvoll ist.

Sie kennen (und können anwenden) mindestens die folgenden Sichten:

- Bausteinsicht (auch: Komponentensicht) als statische Struktur
- Laufzeitsicht zur Beschreibung von Interaktionen, Reihenfolgen oder Zeitbedingungen.
- Verteilungs- oder Deploymentsicht zur Beschreibung technischer Infrastruktur und deren Zusammenhang zu den Quellcode-Bestandteilen (siehe Bausteinsicht) des Systems

Siehe auch [arc42], [Starke+2023] und [Zörner 2021].

LZ 3-4: Übergreifende Konzepte dokumentieren

Softwarearchitekt:innen können Konzepte, die über Modulgrenzen hinaus oder querschnittlich durch die Software gehen, erkennen und entsprechend dokumentieren.

LZ 3-5: Konsistenz zwischen Teilen der Dokumentation sicherstellen

Softwarearchitekt:innen sind in der Lage, Konsistenz zwischen einzelnen Teilen der Dokumentation herzustellen und zu bewahren.

4. Werkzeuge

Dauer: 90 min	Übungszeit: 30 min

4.1. Begriffe und Konzepte

Analoge und digitale Werkzeuge, Modellierungstools, Toolkette.

4.2. Lernziele

LZ 4-1: Passende Werkzeuge für Dokumentationsarbeit auswählen

Softwarearchitekt:innen können passenden Werkzeuge für die verschiedenen Aktivitäten auswählen:

- Erstellung und Pflege von Bestandteilen von Architekturdokumentation
- Verwaltung der Bestandteile
- Kommunikation von Inhalten mit Unterstützung der Bestandteile

Sie können:

- analoge und digitale Dokumentationswerkzeuge situations- und bedarfsgerecht einsetzen
- eine vollständige Werkzeugkette zur Architekturdokumentation anhand konkreter Anforderungen, Randbedingungen und weiterer Einflussfaktoren nachvollziehbar auswählen.

Sie kennen jeweilige Stärken, Schwächen und typische Herausforderungen beim Einsatz und der Integration verbreiteter Werkzeugenkategorien, beispielsweise:

- Wikis
- Modellierungswerkzeuge
- Zeichenprogramme
- Textverarbeitungen
- Versionsverwaltungen
- Sonstige (etwa: Blogs, Issue-Tracker etc.)

LZ 4-2: Anforderungen an Dokumentationswerkzeuge und -artefakte definieren

Softwarearchitekt:innen wissen, wie Werkzeuge bei verschiedenen Aufgaben rund um Architekturdokumentation unterstützen können.

Sie wissen, dass Dokumentation:

- ähnlich wie Quellcode versioniert und Release-fähig verwaltet werden sollte
- zu jedem definierten Stand der Software herstellbar oder generierbar sein sollte

LZ 4-3: Docs-As-Code Ansatz kennen

Softwarearchitekt:innen kennen den Ansatz *Documentation-As-Code*, bei dem die Architekturdokumentation ganz oder teilweise aus quellcode-ähnlichen Artefakten (z.B. Markdown oder AsciiDoc) erzeugt wird.

Sie kennen Werkzeuge, mit denen auch grafische Darstellungen aus textuellen Beschreibungen erzeugt werden können (etwa: [PlantUML])

LZ 4-4: GenAI-Werkzeuge als Unterstützung für Architekturdokumentation anwenden

Softwarearchitekt:innen kennen die Chancen und Risiken beim Einsatz von Generative AI (GenAI)-Werkzeugen im Rahmen der Dokumentation. Sie können solche Werkzeuge gezielt für einzelne Teilbereiche (etwa: architekturrelevante Anforderungen, Lösungsbeschreibung, Bewertung der Dokumentation) anwenden und sind in der Lage, die Ergebnisse abzusichern.

5. Dokumentation bewerten

Dauer: 75 Min.	Übungszeit: 45 Min.

5.1. Begriffe und Konzepte

Review, Checklisten, Fragenkataloge.

Die Lernziele in diesem Abschnitt beschäftigen sich mit der Überprüfung von Architekturdokumentation auf ihre Gebrauchstauglichkeit. Für den praktischen Einsatz von Dokumentation ist dies ein wesentlicher Erfolgsfaktor.

5.2. Lernziele

LZ 5-1: Sicherstellen der Gebrauchstauglichkeit von Dokumentation

Softwarearchitekt:innen erkennen die Überprüfung auf Gebrauchstauglichkeit als einen wesentlichen Erfolgsfaktor der Dokumentation.

Siehe insbesondere LZ 1-2: Bedeutung von Architekturdokumentation für unterschiedliche Stakeholder verstehen.

LZ 5-2: Differenzieren verschiedener Arten der Begutachtung

Softwarearchitekt:innen können zwischen inhaltlicher und formaler Begutachtung einer Architekturdokumentation differenzieren.

LZ 5-3: Differenzieren zwischen Reviews von Architektur und Dokumentation

Softwarearchitekt:innen können zwischen Review/Begutachtung einer Dokumentation und der Bewertung der Architektur oder des Systems differenzieren.

LZ 5-4: Ziele von Dokumentations-Reviews ermitteln und erklären

Softwarearchitekt:innen können mit den betroffenen Personen ("Stakeholdern") mögliche Ziele von Dokumentations-Reviews ermitteln und diese Ziele anderen Personen erklären.

LZ 5-5: Erstellen von Checklisten und Fragenkatalogen für Reviews

Softwarearchitekt:innen können Checklisten und Fragenkataloge für Reviews von Dokumentation erstellen.

LZ 5-6: Verschiedene Rollen bei Dokumentations-Reviews ausüben können

Softwarearchitekt:innen können im Rahmen von Dokumentations-Reviews in der Rolle von

- Autor:innen mit Einwänden angemessen umgehen,
- Gutachter:innen konstruktive Rückmeldungen an Autor:innen geben,
- Moderator:innen ein fachliches Review leiten.

LZ 5-7: Verbessern der Qualität von Dokumentation auf Basis von Review-Ergebnissen

Softwarearchitekt:innen können auf Basis ermittelter Review-Ergebnisse vorhandene Architekturdokumentation weiterentwickeln und pflegen und dabei die Qualität dieser Dokumentation systematisch verbessern.

6. Beispiele für Dokumentation von Softwarearchitekturen

Dauer: 60 Min.	Übungszeit: keine.

Innerhalb jeder akkreditierten Schulung muss mindestens ein Beispiel einer dokumentierten Softwarearchitektur vorgestellt werden.

Art und Ausprägung der vorgestellten Beispiele können von der Schulung bzw. den Interessen der Teilnehmenden abhängen und werden seitens iSAQB nicht vorgegeben.

6.1. Lernziele

LZ 6-1: Vor- und Nachteile beispielhafter Architekturdokumentation erklären

Softwarearchitekten:innen haben an mindestens einem Beispiel Architekturdokumentation kennen gelernt.

Sie können Vor- und Nachteile vorgestellter Beispiele erklären.

Siehe etwa [arc42-by-Example].

Referenzen

Dieser Abschnitt enthält Quellenangaben, die ganz oder teilweise im Curriculum referenziert werden.

A

- [ACC] arc42 and Contributors: The Architecture Communication Canvas, online: https://canvas.arc42.org
- [ADR-GitHub] Various Authors: ADR GitHub Organization. Online: https://adr.github.io/ .
- [arc42] Gernot Starke, Peter Hruschka and Contributors: arc42 - (open-source) template for architecture documentation and communication. https://arc42.org and https://docs.arc42.org
- [arc42-by-Example] Gernot Starke, Michael Simons, Stefan Zörner, Ralf D. Müller, and Hendrik Lösch: arc42 by Example: Software Architecture Documentation in Practice. E-Book: https://leanpub.com/arc42byexample

B

- [Bass 2021] L. Bass, P. Clements und R. Kazman: Software Architecture in Practice. 4. Edition, Addison-Wesley 2021.
- [DDD-Canvas] DDD-Crew: The Bounded Context Canvas, online https://github.com/ddd-crew/bounded-context-canvas.
- [Brown (C4)] S. Brown: The C4 model for visualising software architecture. Leanpub https://leanpub.com/visualising-software-architecture
- [Brown (Guidebook)] S. Brown: The software guidebook. A guide to documenting your software architecture. Leanpub https://leanpub.com/documenting-software-architecture

C

- [Clements+2010] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers et al.: Documenting Software Architectures – Views and Beyond. 2. Edition 2010, Addison Wesley.

H

- [Hargis+2004] G. Hargis et al.: Quality Technical Information: A Handbook for Writers and Editors. Prentice Hall, IBM Press, 2004.

I

- [ISO/IEC 42010:2022] ISO/IEC: 42010:2022: Software, systems and enterprise — Architecture description. Available on https://www.iso.org/standard/74393.html.

K

- [Kruchten 1995] P. Kruchten: Architectural Blueprints – The 4-1 View Model of Architecture. IEEE Software November 1995; 12(6), p. 42-50.

N

- [Nygard 2011] M. Nygard: Documenting Architecture Decisions. Blog Post 2011

https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions. See also https://adr.github.io/

P

- [PlantUML] PlantUML: Open-Source toolset and language definition to create diagrams from textual descriptions. Online: https://plantuml.com/ **R**
- [Read2023] J. Read: Communication Patterns: A Guide for Developers and Architects. O'Reilly Media, 2023.

S

- [Starke 2024] G. Starke: Effektive Softwarearchitekturen - Ein praktischer Leitfaden. 10. Auflage 2024, Carl Hanser Verlag, München.
- [Starke+2023] G. Starke, P. Hruschka: arc42 in Aktion. Carl-Hanser Verlag, 2. Auflage 2022.
- [Starke+2016] G. Starke, P. Hruschka: Communicating Software Architectures: lean, effective and painless documentation. Leanpub https://leanpub.com/arc42inpractice

U

- [UML] Universal Modeling Language, homepage: https://www.uml.org. Provides both the standard plus a collection of resources for further information.

Z

- [Zörner 2021] S. Zörner: Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten. 3. Auflage 2021, Carl-Hanser Verlag.