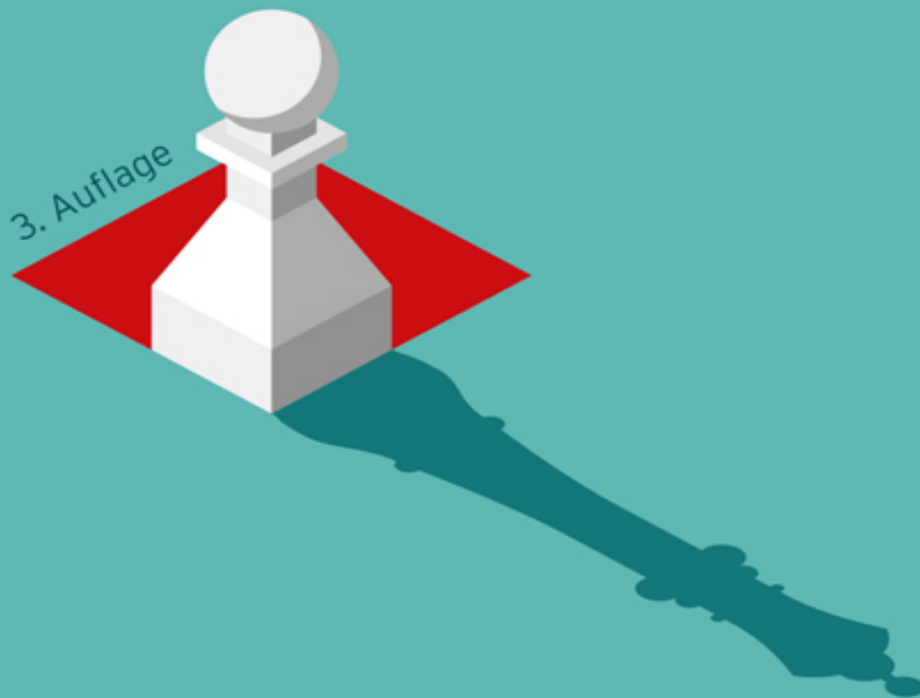


SOFTWARE- ARCHITEKTUREN

dokumentieren
und kommunizieren

Entwürfe, Entscheidungen und
Lösungen nachvollziehbar
und wirkungsvoll festhalten



Stefan ZÖRNER

HANSER

Mit einem Geleitwort von Gernot Starke

HANSER

Stefan Zörner

**Softwarearchitekturen
dokumentieren und kommunizieren**

Entwürfe, Entscheidungen und Lösungen nachvollziehbar und
wirkungsvoll festhalten

3., überarbeitete und erweiterte Auflage

Der Autor:

Stefan Zörner, Buchholz in der Nordheide

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2022 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Brigitte Bauer-Schiewek

Copy editing: Petra Kienle, Fürstfeldbruck

Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München

Umschlagrealisation: Max Kostopoulos,

unter Verwendung von Grafiken von © shutterstock.com/inimalGraphic

Foto des Autors: Jan Gentsch, Hamburg

Layout: Manuela Treindl, Fürth

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Print-ISBN: 978-3-446-46928-0

E-Book-ISBN: 978-3-446-47246-4

ePub-ISBN: 978-3-446-47293-8

Inhalt

Titelei

Impressum

Inhalt

Geleitwort zur 1. Auflage

Überblick: Dokumentationsmittel im Buch

1 Warum Softwarearchitekturen dokumentieren?

1.1 Montagmorgen

1.1.1 Fragen über Fragen

1.1.2 Wer fragt, bekommt Antworten ...

1.2 Voll unagil?

1.2.1 Agil vorgehen

1.2.2 Funktionierende Software vor umfassender Dokumentation

1.2.3 Dokumentation unterstützt Kommunikation

1.3 Wirkungsvolle Architekturdokumentation

1.3.1 Ziel 1: Architekturarbeit unterstützen

1.3.2 Ziel 2: Architektur nachvollziehbar und bewertbar machen

1.3.3 Ziel 3: Umsetzung und Weiterentwicklung leiten

1.3.4 Fremdwort Do|ku|men|ta|tion [...zion] [lat.]

1.4 Mission Statement für dieses Buch

1.5 Über dieses Buch

1.5.1 Für wen ich dieses Buch geschrieben habe

1.5.2 Wie dieses Buch aufgebaut ist

1.5.3 Wem ich danke schön sagen möchte

2 Was Softwarearchitektur ist und worauf sie aufbaut

2.1 Softwarearchitektur-Freischwimmer

2.1.1 Was ist Softwarearchitektur?

2.1.2 Wie entsteht Softwarearchitektur?

2.1.3 Softwarearchitekt/in (m/w/d) gesucht

2.1.4 Ein Architekturüberblick auf n Seiten, $n < 30$

2.2 Die Zielsetzung vermitteln

2.2.1 Jetzt kommt ein Karton!

2.2.2 Virtueller Produktkarton (Dokumentationsmittel)

2.2.3 Fallbeispiel: Schach-Engine „DokChess“

2.2.4 Tipps zum Erstellen von Produktkartons

2.2.5 Fallbeispiel: Schachplattform „immer-nur-schach.de“

2.3 Den Kontext abgrenzen

2.3.1 Systemkontext (Dokumentationsmittel)

2.3.2 Fallbeispiel: Systemkontext „immer-nur-schach.de“

[2.3.3 Tipps zur Erstellung des Systemkontextes](#)

[2.4 Im Rahmen bleiben](#)

[2.4.1 Warum Randbedingungen festhalten?](#)

[2.4.2 Randbedingungen \(Dokumentationsmittel\)](#)

[2.4.3 Fallbeispiel: Randbedingungen „immer-nur-schach.de“](#)

[2.4.4 Tipps zum Festhalten von Randbedingungen](#)

[2.5 Geforderte Qualitätsmerkmale](#)

[2.5.1 Was sind Qualitätsmerkmale?](#)

[2.5.2 Qualitätsziele \(Dokumentationsmittel\)](#)

[2.5.3 Fallbeispiel: Qualitätsziele „immer-nur-schach.de“](#)

[2.5.4 Fallbeispiel: Qualitätsziele „DokChess“](#)

[2.5.5 Qualitätsmerkmale genauer beschreiben](#)

[2.5.6 Qualitätsszenarien \(Dokumentationsmittel\)](#)

[2.5.7 Fallbeispiel: Qualitätsszenarien „immer-nur-schach.de“](#)

[2.5.8 Tipps zum Festhalten von Qualitätsszenarien](#)

[2.6 Weitere Einflüsse und Hilfsmittel](#)

[2.6.1 Stakeholder](#)

[2.6.2 Persona \(Dokumentationsmittel\)](#)

[2.6.3 Fallbeispiel: Persona „immer-nur-schach.de“](#)

[2.6.4 Risiken](#)

[2.6.5 Technische Risiken \(Dokumentationsmittel\)](#)

[2.6.6 Fallbeispiel: Technische Risiken „DokChess“](#)

[2.6.7 Glossar \(Dokumentationsmittel\)](#)

[3 Entscheidungen treffen und festhalten](#)

[3.1 Historisch gewachsen?](#)

[3.2 Architekturentscheidungen](#)

[3.2.1 Architekturentscheidung \(Dokumentationsmittel\)](#)

[3.2.2 Fallbeispiel: Spannende Fragen „DokChess“](#)

[3.2.3 Tipps zur Formulierung von Fragestellungen](#)

[3.2.4 Fallbeispiel: Fragestellungen „immer-nur-schach.de“](#)

[3.2.5 ADRs als eine alternative Strukturierung](#)

[3.3 Einflussfaktoren auf Entscheidungen](#)

[3.3.1 Den Überblick behalten](#)

[3.3.2 Kreuztabellen](#)

[3.3.3 Fallbeispiel: Einflüsse „immer-nur-schach.de“](#)

[3.3.4 Tipps zur Anfertigung von Kreuztabellen](#)

[3.4 Kompakte Darstellung der Lösungsstrategie](#)

[3.4.1 Softwarearchitektur auf einem Bierdeckel?](#)

[3.4.2 Lösungsstrategie \(Dokumentationsmittel\)](#)

[3.4.3 Fallbeispiel: Lösungsstrategie „DokChess“](#)

[3.4.4 Als Ergänzung: ein Überblicksbild](#)

[3.4.5 Eine Architekturbewertung auf dem Bierdeckel](#)

[4 Plädoyer für eine feste Gliederung](#)

[4.1 Aus Essener Feder](#)

[4.2 Vorteile einer festen Struktur](#)

[4.3 arc42 – Vorschlag für eine Gliederung](#)

[4.3.1 Was ist arc42?](#)

[4.3.2 Die Struktur der arc42-Vorlage](#)

[4.3.3 Wo funktioniert arc42 besonders gut?](#)

[4.3.4 arc42 in diesem Buch](#)

[**4.4 Alternativen zu arc42**](#)

[4.4.1 Standards zur Architekturbeschreibung](#)

[4.4.2 Vorgehensmodelle](#)

[4.4.3 Architektur-Frameworks](#)

[4.4.4 Fachliteratur als Inspiration?](#)

[**5 Sichten auf Softwarearchitektur**](#)

[**5.1 Strukturen entwerfen und festhalten**](#)

[5.1.1 Was ist was? Band 127: Unser Softwaresystem](#)

[5.1.2 Schritte der Zerlegung dokumentieren](#)

[5.1.3 Bausteinsicht \(Dokumentationsmittel\)](#)

[5.1.4 Fallbeispiel: Bausteinsicht „DokChess“ \(Ausschnitt\)](#)

[5.1.5 Komponenten: Babylonische Sprachverwirrung 2.0](#)

[5.1.6 Tipps zur Erstellung der Bausteinsicht](#)

[5.1.7 Interaktionspunkte beschreiben](#)

5.1.8 Schnittstellenbeschreibung (Dokumentationsmittel)

5.1.9 Fallbeispiel: Schnittstellen der Eröffnung in „DokChess“

5.2 Verschiedene Blickwinkel

5.2.1 Hat Mozart modelliert?

5.2.2 Fachliche Zerlegung vs. technische Zerlegung

5.2.3 Fallbeispiel: Bausteinsicht „immer-nur-schach.de“

5.3 Verhalten und Abläufe beschreiben

5.3.1 Abläufe in Entwurf und Dokumentation

5.3.2 Darstellungen für Abläufe

5.3.3 Laufzeitsicht (Dokumentationsmittel)

5.3.4 Fallbeispiel: Ablauf in DokChess

5.3.5 Fallbeispiel: Zustandsautomat XBoard (DokChess)

5.4 Die Dinge zum Einsatz bringen

5.4.1 Betriebsaspekte in der Architekturdokumentation

5.4.2 Darstellungen für Verteilung

5.4.3 Verteilungssicht (Dokumentationsmittel)

5.4.4 Fallbeispiel: „immer-nur-schach.de“

5.5 Alternative Vorschläge für Sichten

5.6 Muster kommunizieren

5.6.1 Muster in der Softwareentwicklung

5.6.2 Wann sollten Sie Muster dokumentieren (und wo)?

5.6.3 Einsatz von Mustern dokumentieren

5.6.4 Fallbeispiel: DokChess

6 Übergreifende Konzepte

6.1 Warum übergreifende Themen?

6.2 Themen und Lösungsoptionen

6.2.1 Mögliche Themen für übergreifende Konzepte

6.2.2 Typische Lösungsoptionen

6.3 Themenauswahl

6.3.1 Wie wählen Sie Themen für die Dokumentation aus?

6.3.2 Fallbeispiel: Übergreifende Themen „DokChess“

6.4 Eine Gliederungstechnik für Konzepte

6.4.1 Werkzeug: Warum? Was? Wie? Wohin noch?

6.4.2 Gliederung für ein Konzept

6.4.3 Informeller Text für den Architekturüberblick

6.5 Tipps zur Erstellung übergreifender Konzepte

7 Werkzeuge zur Dokumentation

7.1 Notationen passgenau wählen

7.2 Toolparade zur Architekturdokumentation

7.2.1 Erstellung und Pflege

7.2.2 Verwaltung von Inhalten

7.2.3 Kommunikation von Lösungen

7.3 Repository: UML vs. Wiki

7.3.1 Steht alles im Wiki?

7.3.2 Steht alles im UML-Tool?

7.3.3 UML-Tool + Wiki == Traumpaar?

7.4 Docs-as-Code als Trend

7.5 Wie auswählen?

8 Lightfäden für das Vorgehen zur Dokumentation

8.1 Während der Entwicklung dokumentieren

8.1.1 Zielgruppen Ihrer Dokumentation

8.1.2 Dokumentationsmittel und Dokumente

8.1.3 Womit anfangen?

8.1.4 Während der Arbeit: Kommunizieren und Pflegen

8.2 Der Softwaredetektiv: Bestehendes dokumentieren

8.2.1 Auslöser für Dokumentationsbedarf

8.2.2 Mögliche Szenarien und Ziele des Dokumentierens im Nachhinein

8.2.3 Sherlock Holmes vs. Die drei ???

8.2.4 Informationsquellen identifizieren

8.2.5 Dokumentationsmittel unter der Lupe

8.2.6 Exkurs: Werkzeuge zur Rekonstruktion

8.3 Variationen von „Ein System“

8.3.1 Dokumentation von Systemlandschaften

8.3.2 Dokumentation von Frameworks und Blue Prints

8.3.3 Große, heterogene Systeme und Microservices-Lösungen

8.4 Standpunkt: Mein minimaler Architekturüberblick

9 Architekturüberblick DokChess

9.1 Einführung und Ziele

9.1.1 Aufgabenstellung

9.1.2 Qualitätsziele

9.1.3 Stakeholder

9.2 Randbedingungen

9.2.1 Technische Randbedingungen

9.2.2 Organisatorische Randbedingungen

9.2.3 Konventionen

9.3 Kontextabgrenzung

9.3.1 Fachlicher Kontext

9.3.2 Technischer Kontext oder Verteilungskontext

9.4 Lösungsstrategie

9.4.1 Aufbau von DokChess

9.4.2 Spielstrategie

9.4.3 Die Anbindung

9.5 Bausteinsicht

9.5.1 Ebene 1: Gesamtsystem (Whitebox)

9.5.2 XBoard-Protokoll (Blackbox)

9.5.3 Spielregeln (Blackbox)

9.5.4 Engine (Blackbox)

9.5.5 Eröffnung (Blackbox)

9.5.6 Ebene 2: Engine (Whitebox)

9.5.7 Zugsuche (Blackbox)

9.5.8 Stellungsbewertung (Blackbox)

9.6 Laufzeitsicht

9.6.1 Zugermittlung Walkthrough

9.7 Verteilungssicht

9.7.1 Infrastruktur Windows

9.8 Querschnittliche Konzepte

9.8.1 Abhängigkeiten zwischen Modulen

9.8.2 Schach-Domänenmodell

9.8.3 Benutzungsoberfläche

9.8.4 Plausibilisierung und Validierung

9.8.5 Ausnahme- und Fehlerbehandlung

9.8.6 Logging, Protokollierung, Tracing

9.8.7 Testbarkeit

9.9 Entwurfsentscheidungen

9.9.1 Wie kommuniziert die Engine mit der Außenwelt?

9.9.2 Sind Stellungsobjekte veränderlich oder nicht?

9.10 Qualitätsanforderungen

[9.10.1 Qualitätsbaum](#)

[9.10.2 Qualitätsszenarien](#)

[9.11 Risiken und technische Schulden](#)

[9.11.1 Risiko: Anbindung an das Frontend schlägt fehl](#)

[9.11.2 Risiko: Aufwand der Implementierung zu hoch](#)

[9.11.3 Risiko: Erreichen der Spielstärke scheitert](#)

[9.12 Glossar](#)

[10 Stolpersteine der Architekturdokumentation](#)

[10.1 Probleme](#)

[10.2 Fiese Fallen ...](#)

[10.3 ... und wie man sie umgeht oder entschärft](#)

[10.4 Reviews von Architekturdokumentation](#)

[Glossar](#)

[Literaturverzeichnis](#)

Geleitwort zur 1. Auflage

Dokumentation – Unwort der IT?

Viele IT-Systeme gelten zu Recht als schlecht erweiterbar, schwer verständlich und ungemein komplex. Teilweise liegt das an ihrer mangelhaften Dokumentation, an fehlenden oder unklaren Erläuterungen. Bei anderen Systemen begegnet mir das Gegenteil: Hunderte von Dokumenten, ungeordnet auf Netzlaufwerken, ohne klaren Einstiegspunkt. Kein Wunder, dass Dokumentation als Unwort gilt.

Die meisten Teams, die ich in meinem IT-Leben begleitet habe, konnten gut bis sehr gut programmieren, viele haben ausgezeichnete technische Konzepte entwickelt und umgesetzt. Aber kaum eines dieser Teams konnte auch nur halbwegs ordentlich dokumentieren. Oft als lästige Nebensache verflucht, mit fadenscheinigen Argumenten auf „später“ verschoben oder von Anfang an aufs Abstellgleis verbannt: Dokumentation gilt in Projekten als uncool oder, schlimmer noch, als Strafarbeit: Doku – das sollen andere machen.

Hinter dieser weit verbreiteten, negativen Haltung steckt Unsicherheit: Kaum ein Entwickler, Architekt oder Projektleiter hat jemals gelernt, über Systeme zielorientiert, methodisch und

mit moderatem Aufwand zu kommunizieren – und Dokumentation ist schriftliche (d. h. persistente) Kommunikation.

Genau dafür stellt dieses Buch großartige, praxiserprobte und direkt umsetzbare Lösungen bereit: Sie erfahren, wie Sie mit einfachen Mitteln die Anforderungen an langfristige, lesbare und verständliche Dokumentation erfüllen können. Stefan Zörner erklärt Ihnen, wie Sie sowohl den großen Überblick als auch das notwendige kleine Detail für Ihre Leser sachgerecht aufbereiten und darstellen. Besonders freut mich natürlich, dass er etwas Werbung für unser (freies) arc42-Template macht :-)

Ein echtes Novum finden Sie in [Kapitel 6](#) über technische Konzepte: Überall heißt es in der Praxis: „Wir brauchen ein Konzept für *<schwieriges technisches Problem>*“ ... aber niemand erklärt uns, wie solche Konzepte denn genau aussehen sollen. Wir überlegen jedes Mal neu, in welcher Struktur wir unsere Lösungsideen darstellen und wie wir argumentieren sollen. Stefan eilt mit diesem Buch zu Hilfe: Er hat (unterstützt durch Uwe Vigneschow) aus den langjährigen Erfahrungen von Lernmethodikern und Hirnforschern genau die Hinweise extrahiert, die wir für verständliche, nachvollziehbare und klare Konzepte benötigen. (Neugierig geworden? Blättern Sie direkt mal zu [Abschnitt 6.4.1](#) und überfliegen das Vier-Quadranten-Modell.)

Aber damit nicht genug: Getreu dem Motto, dass Beispiele die besten Lehrmeister sind, hat Stefan ein wirklich cooles System entworfen, gebaut und für dieses Buch vorbildlich dokumentiert: Seine Schach-Engine DokChess illustriert, wie gute Dokumentation aussehen kann (und spielt außerdem noch ganz passabel Schach).

Ich wünsche Ihnen Freude mit diesem Buch. Als Reviewer durfte ich ja schon vor längerer Zeit frühe Versionen testlesen. Mehr als einmal haben mir Stefans Ratschläge in konkreten Projektsituationen seitdem geholfen.

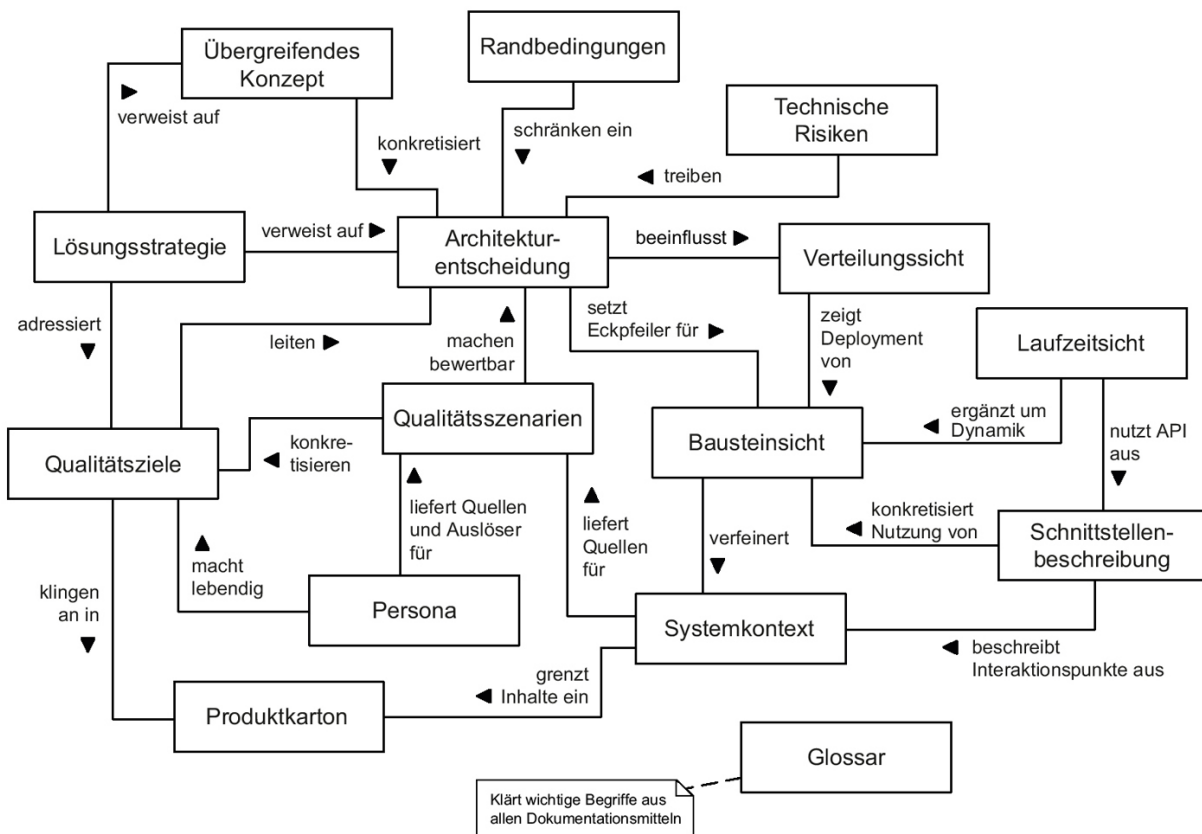
May the force of the proper word and diagram be with you.

Köln, im März 2012

Gernot Starke

Überblick: Dokumentationsmittel im Buch

Die Abbildung zeigt alle im Buch vorgestellten Dokumentationsmittel („Zutaten“) für Softwarearchitektur. Verbindungslinien visualisieren wichtige methodische Zusammenhänge. Die Pfeile an den Linien geben die Leserichtung für die Beschriftung an (Beispiel: Bausteinsicht verfeinert Systemkontext).



Dokumentationsmittel des Buchs mit wichtigen Zusammenhängen

Der Tabelle auf der nächsten Seite können Sie entnehmen, in welchem Abschnitt im Buch Sie den Steckbrief zum betreffenden Dokumentationsmittel finden.

Überblick über die Dokumentationsmittel

Dokumentationsmittel	Nutzen	Steckbrief
Architekturentscheidung	Nachvollziehbare Darstellung einer zentralen, risikoreichen Entscheidung	Abschnitt 3.2.1
Bausteinsicht	Visualisierung der	Abschnitt

	Struktur des Softwaresystems und wie die Teile voneinander abhängen	5.1.3
Glossar	Etablieren eines einheitlichen Wortschatzes im ganzen Vorhaben	Abschnitt 2.6.7
Laufzeitsicht	Visualisierung von dynamischen Strukturen und Verhalten, vor allem von Abläufen	Abschnitt 5.3.3
Lösungsstrategie	Stark verdichteter Architekturüberblick; Gegenüberstellung der wichtigsten Ziele und Lösungsansätze	Abschnitt 3.4.2
Persona	Archetypische Beschreibung einer Personengruppe und deren Ziele (Stakeholder)	Abschnitt 2.6.2
Produktkarton	Plakative Darstellung der wesentlichen Funktionen, Ziele und Merkmale des Systems	Abschnitt 2.2.2
Qualitätsszenarien	Konkretisierung von	Abschnitt

	Qualitätsanforderungen in kurzen, beispielhaften Sätzen	2.5.6
Qualitätsziele	Motivation der wichtigsten an das System gestellten Qualitätsanforderungen	Abschnitt 2.5.2
Randbedingungen	Sammlung der technischen bzw. organisatorischen Vorgaben, die beim Entwurf einzuhalten sind (oder waren)	Abschnitt 2.4.2
Schnittstellenbeschreibung	Detaillierte Beschreibung, wie ein Baustein Funktionalität bereitstellt (oder welche er benötigt)	Abschnitt 5.1.8
Systemkontext	Visualisierung der Fremdsysteme und Benutzer, mit denen das System interagiert	Abschnitt 2.3.1
Technische Risiken	Beschreibung der Risiken, die Einfluss auf die Softwarearchitektur haben (oder hatten)	Abschnitt 2.6.5
Übergreifendes Konzept	Darstellung eines übergreifenden	Abschnitt 6.4.2

	Themas, zur Vereinheitlichung im System oder zur Detaillierung eines Ansatzes	
Verteilungssicht	Visualisierung der Zielumgebung, der Inbetriebnahme und des Betriebs des Systems	<u>Abschnitt 5.4.3</u>

1 Warum Softwarearchitekturen dokumentieren?

„Historisch gewachsen.“

(dem neuen Teammitglied zum Gruß)

Dieses Kapitel motiviert das Thema und steckt die Aufgabe des Buchs ab. Insbesondere arbeitet es heraus, wo Architekturdokumentation unterstützt, d. h., welche Ziele Sie mit ihr verfolgen können. Am Ende des Kapitels erläutere ich Zielsetzung und Aufbau des Buchs.

1.1 Montagmorgen

Haben Sie schon einmal in einem Softwareentwicklungsprojekt gearbeitet, zu dem im weiteren Verlauf neue Teammitglieder hinzugestoßen sind? Oder waren Sie selbst schon einmal bei einem Vorhaben „der/die Neue“? Es ist also Montagmorgen und

ein neuer Mitarbeiter, im konkreten Fall vielleicht eine Entwicklerin, verstärkt das Team.

1.1.1 Fragen über Fragen

Neue im Team haben naturgemäß viele Fragen. Zu Beginn dreht es sich darum, überhaupt arbeitsfähig zu werden. Entsprechend sehen Fragen zum Beispiel so aus:

- Wo soll ich sitzen?
- Welche Tools brauche ich?
- Wie checke ich die Quelltexte aus und wie baue ich die Software?
- Warum sind bei mir die Tests rot?

Die Neuen durchlaufen bezüglich ihrer Fragen einen typischen Zyklus. Nachdem grundlegende Dinge geklärt sind, erkennen Sie an den Fragen, dass es nun in die Praxis geht, unsere Entwicklerin aus dem Beispiel also tatsächlich mitarbeiten will. Später am Tag, vielleicht auch erst am Dienstag oder Mittwoch, könnten ihre Fragen so aussehen:

- Ich finde mich nicht zurecht. Wie finde ich einen Einstieg?
- Diese Teile hier – wie arbeiten die zusammen? Habt ihr das irgendwo aufgemalt?
- Ich soll hier neue Funktionalität hinzufügen, wie stelle ich das an?
- Ich habe hier etwas Ähnliches gefunden, kann ich das wiederverwenden?

- Ich habe hier eine Kleinigkeit geändert. Warum sind jetzt plötzlich bei mir die Tests rot?

Nach ersten Erfolgen wird unser neues Teammitglied mutiger. Fragen drehen sich nicht mehr nur um das „Wie“ oder „Was“, sondern auch um das „Warum“. Jetzt wird auch hinterfragt:

- Diese Software, an der wir hier arbeiten, was macht die überhaupt?
- Warum benutzen wir eigentlich noch Java 5?
- Wieso habt ihr das so gemacht? Ist das nicht viel zu kompliziert?
- Würde man das nicht eigentlich so machen?

Die Beispielfragen sind zwar typisch, aber natürlich fiktiv. Es gibt jedoch eine große Kategorie von Softwarevorhaben, denen sich wunderbar auf die Finger schauen lässt: Open-Source-Projekte. Oft führen sie ihre Kommunikation öffentlich; jeder kann die Mailingliste der Entwicklerinnen und Entwickler abonnieren und ist quasi live dabei. [Bild 1.1](#) zeigt ein echtes Beispiel, den Absender habe ich unkenntlich gemacht.