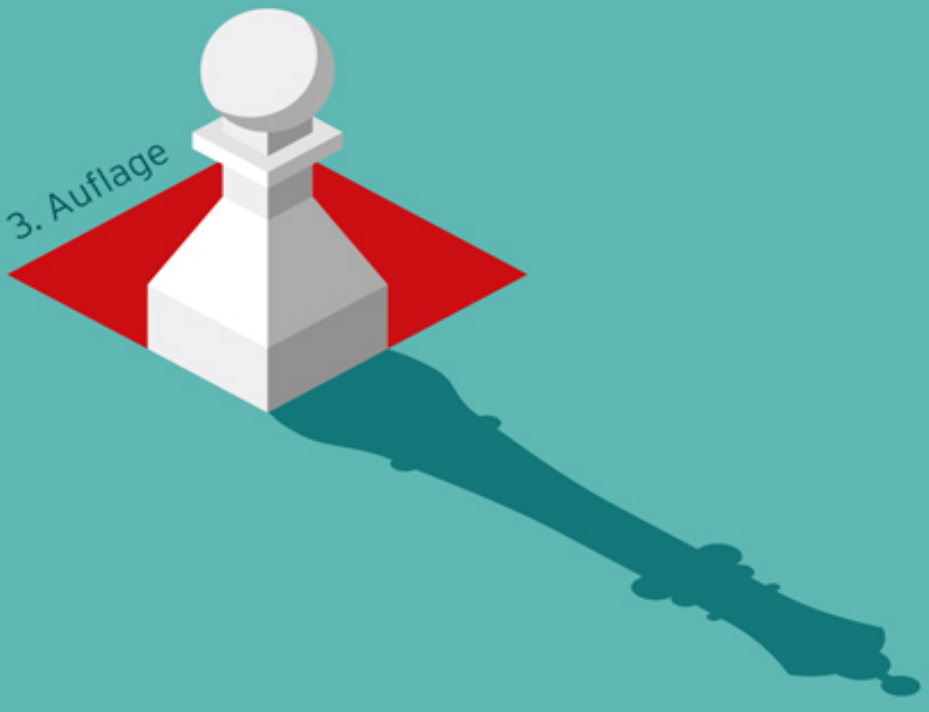
SOFTWARE-ARCHITEKTUREN dokumentieren und kommunizieren

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, Fürstenfeldbruck
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

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“

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

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 Vigenschow) 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.

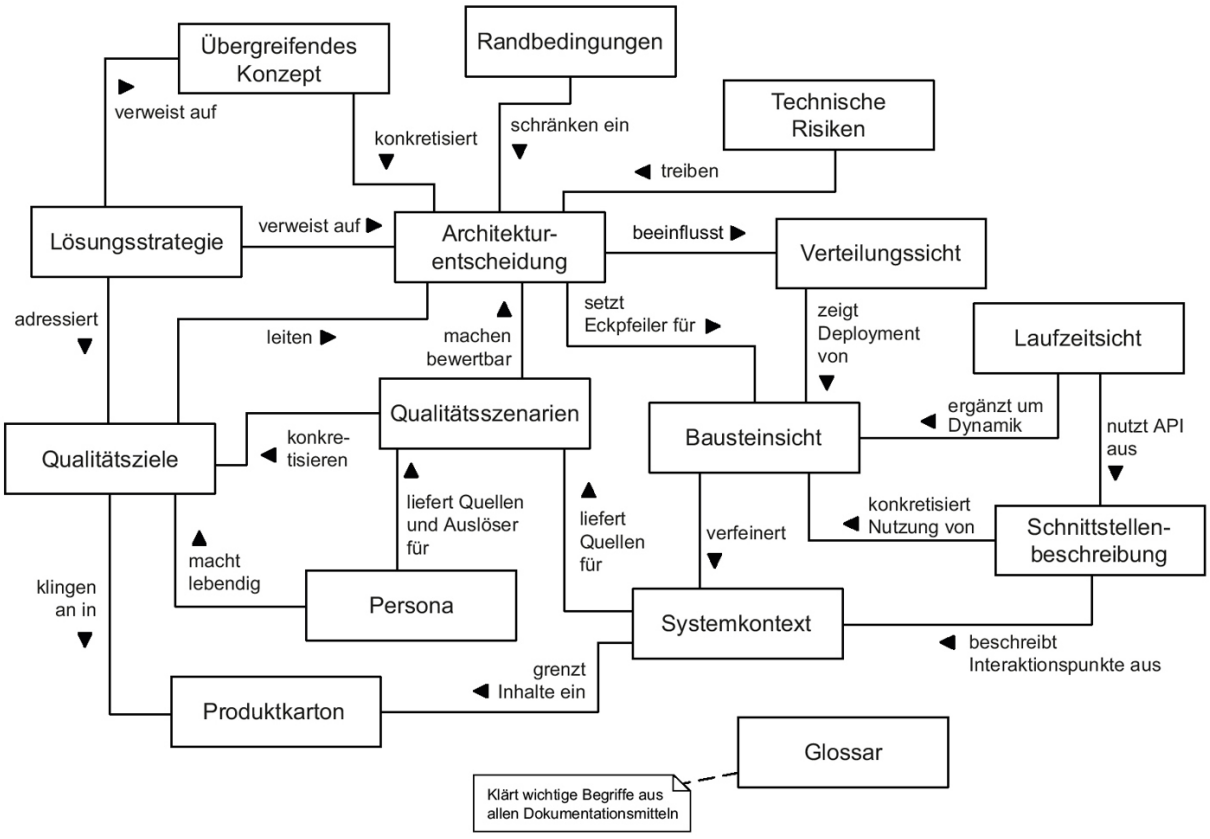
May the force of the proper word and diagram be with you.

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	Abschnitt 5.4.3

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.