



Effektive Software- architekturen gernot STARKE

Ein praktischer Leitfaden



Ideal zur Vorbereitung auf die
iSAQB-Zertifizierung

HANSER

INOQ

HANSER

Gernot Starke

**Effektive
Softwarearchitekturen**

Ein praktischer Leitfaden

9., überarbeitete Auflage

Der Autor:

Dr. Gernot Starke, Köln

INNOQ Fellow

www.gernotstarke.de

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.

© 2020 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 für das Titelmotiv:

© [shutterstock.com/Rawpixel.com](https://www.shutterstock.com/Rawpixel.com) und Vector VA

Layout: Manuela Treindl, Fürth

Gesamtherstellung: Kösel, Krugzell

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

Print-ISBN: 978-3-446-46376-9

E-Book-ISBN: 978-3-446-46589-3

E-Pub-ISBN: 978-3-446-46690-6

Inhalt

Titelei

Impressum

Inhalt

Vorwort

Vorwort zur neunten Auflage

1 Einleitung

1.1 Softwarearchitekt(inn)en

1.2 Effektiv, agil und pragmatisch

1.3 Wer sollte dieses Buch lesen?

1.4 Wegweiser durch das Buch

1.5 Webseite zum Buch

1.6 Weiterführende Literatur

1.7 Danksagung

2 Architektur und Architekten

2.1 Was ist Softwarearchitektur?

2.2 Die Aufgaben von Softwarearchitekten

2.3 Wie entstehen Architekturen?

2.4 In welchem Kontext steht Architektur?

2.5 Weiterführende Literatur

3 Vorgehen bei der Architekturentwicklung

3.1 Informationen sammeln

3.2 Anforderungen klären

3.2.1 Was ist die Kernaufgabe des Systems?

3.2.2 Welche Kategorie von System?

3.2.3 Wesentliche Qualitätsanforderungen ermitteln

3.2.4 Relevante Stakeholder ermitteln

3.2.5 Fachlichen und technischen Kontext ermitteln

3.3 Einflussfaktoren und Randbedingungen ermitteln

3.4 Entwerfen und kommunizieren

3.5 Umsetzung begleiten

3.6 Lösungsstrategien entwickeln

3.7 Weiterführende Literatur

4 Entwurf: Grundlagen, Methoden und Muster

4.1 Grundlagen

4.1.1 Grundsätze des Entwurfs (Maxime)

4.1.2 Prinzipien

4.1.3 SOLID-Prinzipien des objektorientierten Entwurfs

4.1.3.1 Offen-Geschlossen-Prinzip

4.1.3.2 Liskov-Substitutionsprinzip (LSP)

[4.1.3.3 Interface Segregation Principle \(ISP\)](#)

[4.1.3.4 Dependency Inversion Principle \(DIP\)](#)

[4.2 Heuristiken](#)

[4.3 Entwurfsmethoden](#)

[4.3.1 Domain-Driven Design \(Entwurf nach Fachlichkeit\)](#)

[4.3.2 Quality-Driven Software Architecture](#)

[4.3.3 Top-down und Bottom-up](#)

[4.4 Schnittstellen entwerfen](#)

[4.4.1 Anforderungen an Schnittstellen](#)

[4.4.2 Worauf müssen Sie achten?](#)

[4.4.3 Tipps zum Entwurf von Schnittstellen](#)

[4.5 Architekturstile und -muster](#)

[4.5.1 Datenflussarchitekturstil](#)

[4.5.1.1 Architekturstil Batch-Sequentiell](#)

[4.5.1.2 Architekturstil Pipes und Filter](#)

[4.5.2 Datenzentrierter Architekturstil](#)

4.5.2.1 Repository

4.5.2.2 Blackboard

4.5.3 Hierarchische Architekturstile

4.5.3.1 Master-Slave

4.5.3.2 Schichten (Layer)

4.5.3.3 Architekturstil Ports-und-Adapter

4.5.4 Architekturstile verteilter Systeme

4.5.4.1 Client-Server

4.5.4.2 Command Query Responsibility Segregation

4.5.4.3 Broker

4.5.4.4 Peer-to-Peer

4.5.5 Ereignisbasierte Systeme – Event Systems

4.5.5.1 Ungepufferte Event-Kommunikation

4.5.5.2 Message- oder Event-Queue-Architekturen

4.5.5.3 Message-Service-Architekturen

4.5.6 Interaktionsorientierte Systeme

4.5.6.1 Model-View-Controller

4.5.6.2 Presentation Model

4.5.7 Weitere Architekturstile und -muster

4.6 Entwurfsmuster

4.6.1 Entwurf mit Mustern

4.6.2 Adapter

4.6.3 Beobachter (Observer)

4.6.4 Dekorierer (Decorator)

4.6.5 Stellvertreter (Proxy)

4.6.6 Fassade

4.6.7 Zustand (State)

4.7 Weiterführende Literatur

5 Kommunikation und Dokumentation von Architekturen

5.1 Architekten müssen kommunizieren und dokumentieren

5.2 Effektive Architekturdokumentation

[5.2.1 Anforderungen an Architekturdokumentation](#)

[5.2.2 Regeln für gute Architekturdokumentation](#)

[5.3 Typische Architekturdokumente](#)

[5.3.1 Zentrale Architekturbeschreibung](#)

[5.3.2 Architekturüberblick](#)

[5.3.3 Dokumentationsübersicht](#)

[5.3.4 Übersichtspräsentation der Architektur](#)

[5.3.5 Architekturtapete](#)

[5.4 Sichten](#)

[5.4.1 Sichten in der Softwarearchitektur](#)

[5.4.2 Vier Arten von Sichten](#)

[5.4.3 Entwurf der Sichten](#)

[5.5 Kontextabgrenzung](#)

[5.5.1 Elemente der Kontextabgrenzung](#)

[5.5.2 Notation der Kontextabgrenzung](#)

[5.5.3 Entwurf der Kontextabgrenzung](#)

5.6 Bausteinsicht

5.6.1 Elemente der Bausteinsicht

5.6.2 Notation der Bausteinsicht

5.6.3 Entwurf der Bausteinsicht

5.7 Laufzeitsicht

5.7.1 Elemente der Laufzeitsicht

5.7.2 Notation der Laufzeitsicht

5.7.3 Entwurf der Laufzeitsicht

5.8 Verteilungssicht

5.8.1 Elemente der Verteilungssicht

5.8.2 Notation der Verteilungssicht

5.8.3 Entwurf der Verteilungssicht

5.9 Dokumentation von Schnittstellen

5.10 Dokumentation technischer Konzepte

5.11 Werkzeuge zur Dokumentation

5.12 TOGAF zur Architekturdokumentation

5.13 Weiterführende Literatur

6 Modellierung für Softwarearchitekten

6.1 Modelle als Arbeitsmittel

6.1.1 Grafische oder textuelle Modellierung

6.2 UML 2 für Softwarearchitekten

6.2.1 Die Diagrammarten der UML 2

6.2.2 Die Bausteine von Architekturen

6.2.3 Schnittstellen

6.2.4 Die Bausteinsicht

6.2.5 Die Verteilungssicht

6.2.6 Die Laufzeitsicht

6.2.7 Darum UML

6.2.8 Darum nicht UML

6.3 Tipps zur Modellierung

6.4 Weiterführende Literatur

7 Technische Konzepte und typische Architekturaspekte

7.1 Persistenz

7.1.1 Motivation

7.1.2 Einflussfaktoren und Entscheidungskriterien

7.1.2.1 Art der zu speichernden Daten

7.1.2.2 Konsistenz und Verfügbarkeit (ACID, BASE oder CAP)

7.1.2.3 Zugriff und Navigation

7.1.2.4 Deployment und Betrieb

7.1.3 Lösungsmuster

7.1.3.1 Persistenzschicht

7.1.3.2 DAO: Eine Miniatur-Persistenzschicht

7.1.4 Bekannte Risiken und Probleme

7.1.5 Weitere Themen zu Persistenz

7.1.6 Zusammenhang zu anderen Aspekten

7.1.7 Praktische Vertiefung

7.1.8 Weiterführende Literatur

7.2 Geschäftsregeln

7.2.1 Motivation

7.2.2 Funktionsweise von Regelmaschinen

7.2.3 Kriterien pro & kontra Regelmaschinen

7.2.4 Mögliche Probleme

7.2.5 Weiterführende Literatur

7.3 Integration

7.3.1 Motivation

7.3.2 Typische Probleme

7.3.3 Lösungskonzepte

7.3.4 Entwurfsmuster zur Integration

7.3.5 Zusammenhang mit anderen Aspekten

7.3.6 Weiterführende Literatur

7.4 Verteilung

7.4.1 Motivation

7.4.2 Typische Probleme

7.4.3 Lösungskonzept

7.4.4 Konsequenzen und Risiken

7.4.5 Zusammenhang mit anderen Aspekten

7.4.6 Weiterführende Literatur

7.5 Kommunikation

7.5.1 Motivation

7.5.2 Entscheidungsalternativen

7.5.3 Grundbegriffe der Kommunikation

7.5.4 Weiterführende Literatur

7.6 Grafische Oberflächen (GUI)

7.6.1 Motivation

7.6.2 Einflussfaktoren und Entscheidungskriterien

7.6.3 GUI-relevante Architekturmuster

7.6.4 Struktur und Ergonomie von Benutzeroberflächen

7.6.5 Bekannte Risiken und Probleme

7.6.6 Zusammenhang zu anderen Aspekten

7.7 Geschäftsprozess-Management: Ablaufsteuerung im Großen

7.7.1 Workflow-Sprachen

7.7.2 Vorhersagbarkeit

7.7.3 Zweck der Ablaufsteuerung

7.7.4 Lösungsansätze

7.7.5 Integration von Workflow-Systemen

7.7.6 Mächtigkeit von WfMS

7.7.7 Weiterführende Literatur

7.8 Sicherheit

7.8.1 Motivation – Was ist IT-Sicherheit?

7.8.2 Sicherheitsziele

7.8.3 Lösungskonzepte

7.8.4 Security Engineering mit Patterns

7.8.5 Weiterführende Literatur

7.9 Protokollierung

7.9.1 Typische Probleme

7.9.2 Lösungskonzept

7.9.3 Zusammenhang mit anderen Aspekten

7.9.4 Weiterführende Literatur

7.10 Ausnahme- und Fehlerbehandlung

7.10.1 Motivation

7.10.2 Fehlerkategorien schaffen Klarheit

7.10.3 Muster zur Fehlerbehandlung

7.10.4 Mögliche Probleme

7.10.5 Zusammenhang mit anderen Aspekten

7.10.6 Weiterführende Literatur

7.11 Skalierbarkeit

7.11.1 Was bedeutet Skalierbarkeit?

7.11.2 Skalierungsstrategien

7.11.3 Elastizität

7.11.4 Scale-Up-Strategie

7.11.5 Vertikale Scale-Out-Strategie

7.11.6 Horizontale Scale-Out-Strategie

7.11.7 Der Strategiemix

7.11.8 Allgemeine Daumenregeln

7.11.9 CPU-Power

7.11.10 GPU-Power

7.11.11 RAIDs, SANs und andere Speichersysteme

7.11.12 Bussysteme für die Speicheranbindung

7.11.13 Geringere Bandbreite im Netz

7.12 Blockchain und dezentralisierte Architektur

7.12.1 Definition und Abgrenzung

7.12.2 Technische Grundlagen von Blockchains

7.12.3 Smart Contracts

7.12.4 Blockchains in nichtöffentlichen Kontexten

7.12.5 Architekturabwägungen

7.12.6 Architekturmuster

8 Analyse und Bewertung von Softwarearchitekturen

8.1 Qualitative Architekturbewertung

8.2 Quantitative Bewertung durch Metriken

8.3 Werkzeuge zur Bewertung

8.4 Weiterführende Literatur

9 Systematische Verbesserung und Evolution

9.1 Wege in den Abgrund

9.2 Systematisch verbessern

9.3 Bewährte Praktiken und Muster

9.4 Analyse: Probleme identifizieren

9.5 Evaluate: Probleme und Maßnahmen bewerten

9.6 Improve: Verbesserungsmaßnahmen planen und durchführen

9.6.1 Maxime für Verbesserungsprojekte

9.6.2 Kategorien von Verbesserungsmaßnahmen

9.7 Crosscutting: phasenübergreifende Praktiken

9.8 Mehr zu AIM⁴²

9.9 Weiterführende Literatur

10 Microservices

10.1 Was sind Microservices?

10.2 Warum Microservices?

10.3 Eigenschaften von Microservices

10.4 Microservices und die Organisation

10.5 Für welche Systeme eignen sich Microservices?

10.6 Herausforderungen bei Microservices

10.6.1 Überblick über viele Services behalten

10.6.2 Microservices effektiv entwickeln

10.6.3 Service Discovery

10.6.4 UI-Integration

10.6.5 Dezentralisierte Daten

10.6.6 Versionierung von Microservices

10.6.7 Laufzeitumgebungen und Infrastruktur verwalten

10.7 Beispiele für Microservices

10.8 Weiterführende Literatur

11 Enterprise-IT-Architektur

11.1 Wozu Architekturebenen?

11.2 Aufgaben von Enterprise-Architekten

11.2.1 Management der Infrastrukturkosten

11.2.2 Management des IS-Portfolios

11.2.3 Definition von Referenzarchitekturen

11.2.4 Weitere Aufgaben

11.3 Weiterführende Literatur

12 Beispiele von Softwarearchitekturen

12.1 Beispiel: Datenmigration im Finanzwesen

12.2 Beispiel: Kampagnenmanagement im CRM

13 Werkzeuge für Softwarearchitekten

13.1 Kategorien von Werkzeugen

13.2 Typische Auswahlkriterien

14 iSAQB Curriculum

14.1 Standardisierte Lehrpläne für Softwarearchitekten

14.1.1 Grundlagenausbildung und Zertifizierung Foundation-Level

14.1.2 Fortgeschrittene Aus- und Weiterbildung (Advanced-Level)

14.2 iSAQB-Foundation-Level-Lehrplan

14.2.1 Können, Wissen und Verstehen

14.2.2 Voraussetzungen und Abgrenzungen

14.2.3 Lernziele des iSAQB Foundation Level

14.2.4 Zertifizierung gemäß iSAQB

14.3 Beispielfragen zur Foundation Level Prüfung

15 Nachwort: Architektonien

15.1 In sechs Stationen um die (IT-)Welt

15.2 Ratschläge aus dem architektonischen Manifest

16 Literatur

Vorwort

*Haben Sie jemals einen dummen Fehler zweimal begangen?
– Willkommen in der realen Welt.
Haben Sie diesen Fehler hundertmal hintereinander gemacht?
– Willkommen in der Software-Entwicklung.*

Tom DeMarco,
in: „Warum ist Software so teuer?“

Wenn Sie sich für Baukunst interessieren, dann erkennen Sie sicherlich die „Handschrift“ berühmter Architekten wie Frank Lloyd Wright, Le Corbusier oder Mies van der Rohe immer wieder, egal, wo auf der Welt Sie auf Bauwerke dieser Meister stoßen. Die Funktionalität des Guggenheim Museums in New York oder des Opernhauses in Sydney gepaart mit deren Schönheit und Ästhetik sind unvergessliche Eindrücke. Das erwarten wir heute auch von unseren IT-Systemen: Funktionalität gepaart mit Stil!

Seit mehr als zwanzig Jahren versuche ich, Systementwicklern die Kunst des Architekturentwurfs nahezubringen. Die Erfahrung hat mich gelehrt, dass man jede Person, die mit gesundem Menschenverstand ausgestattet ist, zu einem guten Systemanalytiker ausbilden kann. Softwarearchitekten auszubilden, ist wesentlich schwieriger.

Früher waren viele unserer Systeme so einfach, dass einzelne Personen die Struktur leicht im Kopf behalten konnten. Heutzutage gehört mehr dazu, um die Struktur eines Systems zu beherrschen, die Auswirkungen von Technologieentscheidungen vorauszusehen und die Vielzahl von Hilfsmitteln wie Generatoren, Frameworks, Libraries und Entwicklungswerkzeuge kosteneffizient und zielführend einzusetzen.

Viele Jahre war ich davon überzeugt, dass nur Erfahrung in der Erstellung großer Systeme und selbst gemachte Fehler gute Architekten hervorbringen. Wir wussten einfach zu wenig über Wirkungen und Folgewirkungen von Designentscheidungen. In den letzten Jahren ist die Entwicklung von Architekturen mehr und mehr zur Ingenieursdisziplin herangereift.

Gernot Starke ist es gelungen, die Essenz dieser Disziplin auf den Punkt zu bringen. Die Tipps und Tricks, die er in diesem Buch zusammengetragen hat, vermitteln Ihnen eine Fülle von Praxiserfahrungen. Selbst wenn Sie zu den Veteranen der Branche gehören, werden Sie neben vielen Déjà-vu-Erlebnissen sicherlich noch die eine oder andere Perle entdecken. Wenn Sie gerade Ihre ersten Sporen als Architekt(in) verdienen, dann können Sie sich mit den Empfehlungen den einen oder anderen Holzweg ersparen.

Trotz aller Fortschritte in der IT bleiben Konstruktion und Ausgestaltung von Architekturen dauerhaft eine Domäne für kreative Gestaltungsarbeit von Menschen und Teams. Softwarearchitekt ist daher ein Beruf mit sicherer Zukunft!

Aachen, im September 2013

Peter Hruschka

Vorwort zur neunten Auflage

Seit der ersten Auflage 2002 sind 18 Jahre vergangen – in denen „Softwarearchitektur“ aus den Kinderschuhen zu einer volljährigen Disziplin der Softwareentwicklung reifen konnte.

Unsere Branche hat viel gelernt, Agilität und iterativ-inkrementelle Entwicklung sind endlich im Mainstream angekommen, schnelles Feedback eher Normalität denn Ausnahme, beispielsweise durch automatisiertes Testen.

Architekturaufgaben und technische Entscheidungen liegen oftmals beim Entwicklungsteam oder werden zumindest von mehreren Personen nach Rücksprache mit den Teams getroffen – und nicht mehr von Einzelpersonen „diktiert“.

Dafür sind allerdings die Anforderungen an unsere Systeme signifikant gestiegen: Benutzerzahlen von Online-Systemen gehen oft in die Millionen, Datendurchsatz oder -volumen messen wir in Tera oder Giga, nicht mehr in Kilo ...

Benutzungsschnittstellen müssen hochgradig ergonomisch sein und auf multiplen Endgeräten laufen – *zero-downtime* gehört (fast) zur Selbstverständlichkeit. Embedded- und Informationssysteme müssen oftmals sicherheitskritische Entscheidungen treffen. Ach ja – Entwicklungskosten für solche komplexen IT-Systeme sollen tendenziell sinken und neue Features am besten in kürzester Zeit zur Verfügung stehen. Für solche (krassen!) Herausforderungen benötigen Entwicklungsteams fundierte Architekturkompetenz – und genau dafür habe ich dieses Buch geschrieben.

Zur neunten Auflage: Seit der achten Auflage (2017) haben viele gründliche Leserinnen und Leser einige Dutzend kleine Fehler

gefunden, die in der vorliegenden Auflage (hoffentlich) behoben sind. Ich danke allen, die mich netterweise auf diese Dinge hingewiesen haben.

Die Aktualisierungen des iSAQB-CPSA-F-Lehrplans sind komplett eingeflossen, und das Kapitel über Microservices ist von den Altlasten der SOA befreit 😊.

Meinen Kollegen Dr. Lars Hupel konnte ich dazu gewinnen, einen kleinen (aber spannenden) Exkurs zum Thema „Blockchain“ beizusteuern.

Zum Buch gibt es eine (*responsive design*) Website, die ich mittels Github¹, Jekyll und Docker pflege (<http://esabuch.de>).

Möge dieses Buch helfen, bessere Software zu entwickeln!

Köln, Juni 2020

Gernot Starke

¹ Für Insider oder falls Sie Verbesserungsvorschläge haben:
<https://github.com/gernotstarke/esabuch.de-site>

1 Einleitung

*Wir bauen Software wie Kathedralen:
Zuerst bauen wir – dann beten wir.*

Gerhard Chroust

Bitte erlauben Sie mir, Sie mit einer etwas böartigen kleinen Geschichte zur weiteren Lektüre dieses Buchs zu motivieren.

Eine erfolgreiche Unternehmerin möchte sich ein Domizil errichten lassen. Enge Freunde raten ihr, ein Architekturbüro mit dem Entwurf zu betrauen und die Erstellung begleiten zu lassen. Nur so ließen sich die legendären Probleme beim Hausbau (ungeeignete Entwürfe, mangelnde Koordination, schlechte Ausführung, Pfusch bei Details, Kostenexplosion und Terminüberschreitung) vermeiden.

Um die für ihr Vorhaben geeigneten Architekten zu finden, beschließt sie, einigen namhaften Büros kleinere Testaufträge für Einfamilienhäuser zu erteilen. Natürlich verrät sie keinem der Kandidaten, dass diese Aufträge eigentlich Tests für das endgültige Unterfangen sind.

Nach einer entsprechenden Ausschreibung in einigen überregionalen Tageszeitungen trifft unsere Bauherrin folgende

Vorauswahl:

- Wasserfall-Architektur KG, Spezialisten für Gebäude und Unterfangen aller Art,
- V&V Architektur GmbH & Co. KG, Spezialisten für Regierungs-, Prunk- und Profanbauten,
- Extremarchitekten AG.

Alle Büros erhalten identische Vorgaben: Ihre Aufgabe besteht in Entwurf und Erstellung eines Einfamilienhauses (EFH). Weil unsere Unternehmerin jedoch sehr häufig, manchmal fast sprunghaft, ihre Wünsche und Anforderungen ändert, beschließt sie, die Flexibilität der Kandidaten auch in dieser Hinsicht zu testen.

Wasserfall-Architektur KG

Die Firma residiert im 35. Stock eines noblen Bürogebäudes. Dicke Teppiche und holzvertäfelte Wände zeugen vom veritablen Wohlstand der Firmeneigner.