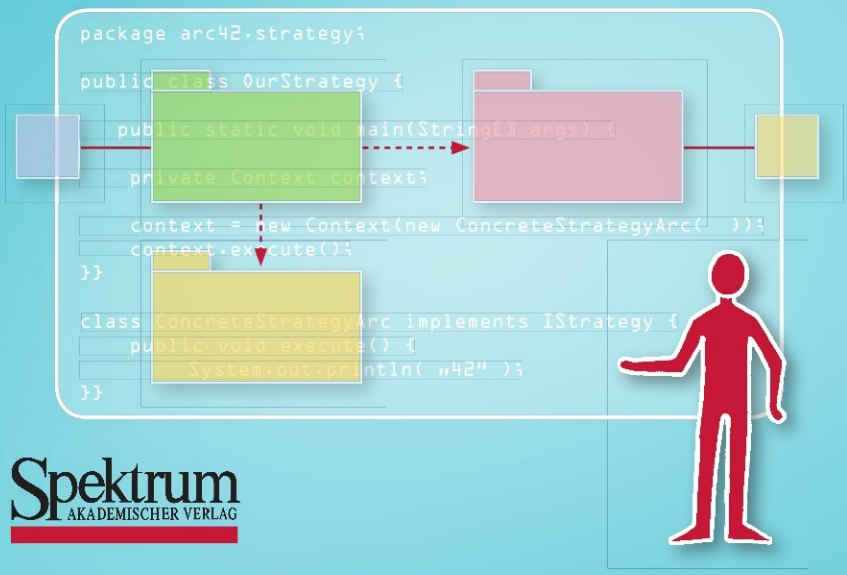
Gernot Starke Peter Hruschka

Software-Architektur kompakt

– angemessen und zielorientiert



Spektrum
AKADEMISCHER VERLAG

Software-Architektur kompakt

Werke der „kompakt-Reihe“ zu wichtigen Konzepten und Technologien der IT-Branche:

- ermöglichen einen raschen Einstieg,
- bieten einen fundierten Überblick,
- sind praxisorientiert, aktuell und immer ihren Preis wert.

Bisher erschienen:

- Heide Balzert
  UML kompakt, 2. Auflage
- Andreas Böhm / Elisabeth Felt
  e-commerce kompakt
- Christian Bunse / Antje von Knethen
  Vorgehensmodelle kompakt, 2. Auflage
- Holger Dörnemann / René Meyer
  Anforderungsmanagement kompakt
- Christof Ebert
  Outsourcing kompakt
- Christof Ebert
  Risikomanagement kompakt
- Karl Eilebrecht / Gernot Starke
  Patterns kompakt, 2. Auflage
- Andreas Essigkrug / Thomas Mey
  Rational Unified Process kompakt, 2. Auflage
- Peter Hruschka / Chris Rupp / Gernot Starke
  Agility kompakt, 2. Auflage
- Arne Koschel / Stefan Fischer / Gerhard Wagner
  J2EE/Java EE kompakt, 2. Auflage
- Michael Kuschke / Ludger Wölfel
  Web Services kompakt
- Torsten Langner
  C# kompakt
- Pascal Mangold
  IT-Projektmanagement kompakt, 3. Auflage
- Michael Richter / Markus Flückiger
  Usability Engineering kompakt
- Thilo Rottach / Sascha Groß
  XML kompakt: die wichtigsten Standards
- SOPHIST GROUP / Chris Rupp
  Systemanalyse kompakt, 2. Auflage
- Gernot Starke / Peter Hruschka
  Software-Architektur kompakt
- Ernst Tiemeyer
  IT-Controlling kompakt
- Ernst Tiemeyer
  IT-Servicemanagement kompakt
- Ralf Westphal
  .NET kompakt
- Ralf Westphal / Christian Weyer
  .NET 3.0 kompakt

Gernot Starke / Peter Hruschka

Software-Architektur kompakt

– angemessen und zielorientiert

Spektrum
AKADEMISCHER VERLAG

Autoren:
Dr. Gernot Starke
E-Mail: gs@gernotstarke.de

Dr. Peter Hruschka
E-Mail: hruschka@b-agile.de

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.

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

09 10 11 12 13 5 4 3 2 1

Planung und Lektorat: Dr. Andreas Rüdinger, Barbara Lühker
Herstellung und Satz: Crest Premedia Solutions (P) Ltd., Pune, Maharashtra, India
Umschlaggestaltung: SpieszDesign, Neu-Ulm

ISBN 978-3-8274-2093-0

Inhaltsverzeichnis

Warum Architektur?

Als Berater und Trainer lernen wir viele Projekte unterschiedlicher Branchen kennen. Im Jahre 2008 haben wir „40 Jahre Software Engineering“ gefeiert. 40 Jahre, seit einige führende Forscher und Entwickler bei der Internationalen Software-Engineering-Konferenz (ICSE) in Garmisch-Partenkirchen den Grundstock für systematische Softwareentwicklung als Ingenieursdisziplin legten und begannen, die Kunst der Softwareentwicklung schrittweise zum soliden Handwerk umzubauen.

Wir konnten in den vielen Jahren unserer Tätigkeiten tatsächlich einige Verbesserungen in der Softwareentwicklung beobachten. So beginnt zum Beispiel kaum jemand ein Projekt ohne vernünftige Planung und Schätzung. Projektmanagement nehmen wir – mit Ausnahme einiger oft politisch motivierter Monsterprojekte – als beherrschte Tätigkeit wahr. Auch implementieren und testen können wir mittlerweile ziemlich gut. Die Vielzahl der in Betrieb befindlichen Softwaresysteme belegt dies eindrucksvoll.

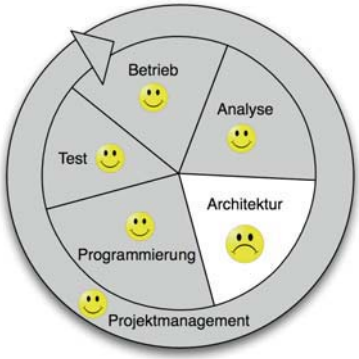
In den letzten 10–15 Jahren rückten die Themen Requirements Engineering, Businessanalyse und Systemanalyse in den Vordergrund. Viele Unternehmen haben eingesehen, dass ohne vernünftig aufbereitete Problemstellungen (= Requirements) fast zwangsläufig falsche oder unbrauchbare Lösungen entstehen. Deshalb haben zahlreiche Organisationen und Experten daran gearbeitet, das Berufsbild des Requirements Engineers zu stärken und die Analysetätigkeiten in Projekten sehr ernst zu nehmen (siehe beispielsweise [Cert-RE]).

Lediglich auf dem Gebiet zwischen Systemanalyse und Codierung beobachten wir noch viele weiße Flecken auf der IT-Landkarte. Fragen wir Unternehmen nach der Architekturdokumentation der in Betrieb befindlichen Systeme, so erhalten wir meist eine der drei folgenden Antworten:

1. „Architekturdokumentation? Dafür hatten wir keine Zeit!“
2. „Als wir damals mit dem Projekt begannen, gab es ein paar Unterlagen. Aber diese Dokumente sind längst veraltet.“
3. Am häufigsten treffen wir jedoch auf die Antwort: „Ursprünglich kannten alle Beteiligten die Struktur des Systems. Im Laufe der Zeit ging, durch zahlreiche Erweiterungen und diverse Personalwechsel, der Überblick verloren.“

Diesen (desolaten) Zustand möchten wir ändern. Wir halten im Zuge langlebiger Softwaresysteme eine stärkere Berücksichtigung der Ar-

chitektur für erfolgsentscheidend und nachhaltig sinnvoll. Das gilt auch für kleinere Systeme – nicht nur für ganz große!



Wir zeigen Ihnen in diesem Buch, wie Sie mit pragmatischen Ansätzen zur Architektur Ihre Systeme schneller und besser konstruieren können. Damit verbessern Sie die langfristige Wartbarkeit und Verständlichkeit und behalten auch nach Jahren noch den Überblick über die Strukturen und Zusammenhänge innerhalb Ihrer Systeme.

Bevor wir Ihnen das Berufsbild und die Verantwortung von Software-Architekten näher bringen und Ihnen aufzeigen, wie Sie angemessen und zielorientiert mit Software-Architekturen umgehen, beantworten wir drei Kernfragen:

1. Was ist Architektur?
2. Warum sollten Sie Architekturen dokumentieren?
3. Welche Eigenschaften besitzt eine angemessene Software-Architektur?

Was ist Architektur?

Die Architektur eines Systems beschreibt die Strukturen des Systems, dessen Bausteine, Schnittstellen und deren Zusammenspiel. Die Architektur besteht aus Plänen und enthält meistens Vorschriften oder Hinweise, wie das System erstellt oder zusammengebaut werden sollte.

Als Plan unterstützt die Architektur die Herstellung (hier: Programmierung und Test), die Weiterentwicklung und den Einsatz des Systems (hier: Betrieb).

Verwechseln Sie Architektur nicht mit dem fertigen System: Software-Architektur ist der *Plan* des Systems – nicht das System selbst. Software-Architektur besteht nicht nur aus Quellcode oder Unit-Tests (auch wenn manche Programmier-Freaks gerne Code über Alles stellen).

Daher ist es für Software-Architekten wichtig, sich mit der Beschreibung von Softwaresystemen zu beschäftigen. Sie müssen verständliche, redundanz- und widerspruchsfreie, effektive und effiziente Pläne Ihrer Systeme erstellen, auf deren Basis Sie selbst mit Entwicklungsteams dann laufende Software bauen.

Andere Disziplinen, wie Maschinenbau, Immobilienarchitektur und Elektrotechnik, haben ungeheuer von der Standardisierung ihrer jeweiligen Pläne profitiert. Heute kann jeder Ingenieur die Konstruktionszeichnungen seiner Berufskollegen verstehen und weiterentwickeln. In der Informatik sind wir von dieser Standardisierung noch weit entfernt – trotz *Unified Modeling Language* (UML, siehe [Rupp+05]) oder anderen Modellierungsansätzen.

In diesem Buch zeigen wir Ihnen eine sehr praxisnahe Art solcher Pläne, die auf dem frei verfügbaren arc42-Template basiert (siehe [arc42]).

Warum sollten Sie Architektur dokumentieren?

Softwaresysteme bleiben oftmals viele Jahre im Einsatz – und werden während dessen kontinuierlich weiterentwickelt. Dieses „Leben" von Software endet oft tragisch: Mangelnde Wartbarkeit und unzureichende Dokumentation machen jede kleine Änderung zu einem Vabanquespiel mit ungewissem Ausgang. Oftmals lassen sich selbst minimale Erweiterungen nur mit massivem finanziellen Aufwand bewältigen – bis irgendwann das Nachfolgeprojekt startet…

Sie als Software-Architekt haben es in der Hand, dieses „Verfaulen" von Software gründlich zu verhindern und Ihre Systeme langfristig wartbar, flexibel und verständlich zu konstruieren. Dafür müssen Sie auf die *innere Qualität* der Systeme achten und Ihre Architekturen *sauber* beschreiben (denken Sie an den vorigen Absatz: Architekturen sind *Beschreibungen* der Produkte).

Welche Eigenschaften hat eine angemessene Software-Architektur?

Wir haben eben eine „saubere“ Architektur gefordert. Was bedeutet denn nun „sauber“ für Software-Architekturen? Aus unserer Sicht bedeutet es vor allem:

- relevant
- effizient pflegbar
- sparsam
- verständlich und nachvollziehbar
- korrekt und aktuell
- prüfbar
- akzeptiert bei Lesern

Relevant

In die Architekturdokumentation gehören nur solche Informationen, die für die Architektur des Systems relevant sind. Architekturdokumentation ist keine Implementierungs- oder Quellcodedokumentation.

Die Entscheidung, welche konkrete Information für ein System relevant für die Beschreibung der Architektur ist, liegt zuerst beim Architekten. Dieser wird jedoch hoffentlich im Zuge der iterativen Architekturentwicklung andere Beteiligte dazu befragen.

Wir erachten folgende Informationen als relevant für Architekturdokumentation:

- systemspezifische Strukturen und Entscheidungen im Großen
- Entscheidungen oder Strukturen, die bedeutsam für Implementierung und spätere Änderungen sind
- unerwartete oder unkonventionelle Lösungen (hierzu zählt Vieles, was Entwickler als *cool* bezeichnen)
- querschnittliche oder wiederholt auftretende Lösungsmuster, Aspekte oder technische Konzepte

Effizient pflegbar

Aufgrund des hohen Pflegeaufwands, der mit umfangreicher Dokumentation grundsätzlich verbunden ist, müssen Sie den Pflegeaufwand Ihrer Architekturdokumentation mit Verständlichkeit, Genauigkeit und Aktualität balancieren. Folgende Faktoren beeinflussen die Pflegbarkeit von Dokumentation:

- Umfang (Menge): Kürzere Dokumente sind einfacher zu pflegen.

- Redundanz: Wiederholung kann helfen, Vor- und Rücksprünge innerhalb von Dokumenten zu vermeiden, erhöht aber den Aufwand zur Pflege.
- Standardisierte Strukturen (siehe Abschnitt „Verständlich und nachvollziehbar").
- Eingesetzte Werkzeuge: Eine in den gesamten Arbeitsprozess integrierte Werkzeugkette kann den Pflegeaufwand von Dokumentation signifikant verringern – bedarf jedoch möglicherweise eines hohen Initialisierungsaufwands, um die Werkzeugkette einzurichten.

Zwei wesentliche Ansätze verbessern die Pflegbarkeit von Dokumentation: Halten Sie zum einen etablierte oder standardisierte Gliederungsstrukturen ein – das erleichtert Ihnen, Ihren Lesern sowie anderen Autoren das Zurechtfinden innerhalb der Dokumentation. Zum anderen begrenzen Sie die Menge an Details, die Sie beschreiben. Duplizieren Sie nur in (seltenen!) Ausnahmefällen Informationen, insbesondere wenn diese aus anderen Quellen (Modellen, Dokumenten, Programmcode) leicht nachgelesen werden können. Anders formuliert: Verzichten Sie auf den Versuch der *Vollständigkeit-in-jeglicher-Hinsicht*. Weniger ist oftmals mehr.

Sparsam

Ihre Devise muss lauten: „So viel wie nötig, so wenig wie möglich." Die Beschreibung von Software- oder System-Architekturen sollte grundsätzlich mit einem Überblick beginnen und Details nur nach Bedarf aufzeigen. Seien Sie gezielt faul! Überlegen Sie sich, für wen Sie wie viel Information in die Architekturdokumentation packen müssen. Je dünner Ihre Architekturdokumentation ausfällt, umso effizienter ist sie pflegbar. Das soll aber kein Freibrief für das Weglassen von Informationen sein. Die Devise beginnt mit der Aussage: „So viel wie nötig"!

Verständlich und nachvollziehbar

Verständlichkeit und Nachvollziehbarkeit von Dokumentation vereinfachen neben der Erstellung des Systems auch dessen Test, Betrieb und Weiterentwicklung – mit positiven Folgen für Kosten und Risiken des Systems.

Navigierbarkeit und Auffindbarkeit von Informationen sind die besten Freunde von Verständlichkeit. Leser müssen Informationen schnell und zuverlässig finden können. Das können Sie als Autor bei-

spielsweise durch standardisierte Gliederungen erreichen: Dadurch wissen Leser bereits vorab, wo sich bestimmte Informationseinheiten innerhalb der Dokumentation befinden (vorausgesetzt natürlich, Ihre Leser kennen die Gliederung). Zusätzlich hilft ein Stichwortverzeichnis, unterstützt durch ein Projekt- oder Systemglossar, bei der Suche nach spezifischen Begriffen.

Ein praktischer Tipp: Lassen Sie die Verständlichkeit und Nachvollziehbarkeit Ihrer Dokumentation von Ihren Lesern bewerten. Fragen Sie Ihre Leser konkret nach Verbesserungsvorschlägen und lassen sich die Kapitel, Abschnitte oder Seiten zeigen, die Ihre Leser als schwer verständlich einstufen.

Korrekt und aktuell

Jegliche Information innerhalb der Architekturdokumentation muss korrekt sein, sodass sich sämtliche Beteiligten (beispielsweise System- und Software-Architekten, Entwickler, Tester, Manager) darauf verlassen können. Die Notwendigkeit der *Aktualität* bedeutet, dass Sie Architekturdokumentation synchron mit der Weiterentwicklung des Systems pflegen und aktualisieren müssen – deswegen auch die Forderung nach *effizienter Pflegbarkeit*. Führen Sie grundsätzlich eine Änderungshistorie der Dokumentation, um Änderungen am System mit angemessenem Aufwand nachvollziehen zu können.

Bei der Aktualität müssen Sie besonders darauf achten, dass die Strukturen und Abhängigkeiten im Quellcode konsistent zur Dokumentation bleiben. Geben Sie hierauf genau acht, denn in zeitkritischen Projektphasen geht diese Konsistenz andernfalls leicht verloren.

Korrektheit impliziert einen angemessenen Grad an Vollständigkeit, der den Informationsbedarf der unterschiedlichen Stakeholder deckt. Verwechseln Sie aber Korrektheit nicht mit Ausgiebigkeit: Sie müssen die relevanten Dinge beschreiben, nicht möglichst viele Dinge.

Prüfbar

Zur frühzeitigen Identifikation möglicher Risiken oder Probleme sollte die Dokumentation von Software-Architekturen die Prüfbarkeit der gewählten Lösung hinsichtlich der an das System gestellten Anforderungen unterstützen. Die unterschiedlichen Architektursichten ermöglichen durch Walkthroughs die Prüfung funktionaler und nichtfunktionaler Anforderungen anhand von Bewertungs- oder Prüfszenarien (diese Sichten lernen Sie in den nachfolgenden Kapiteln kennen).