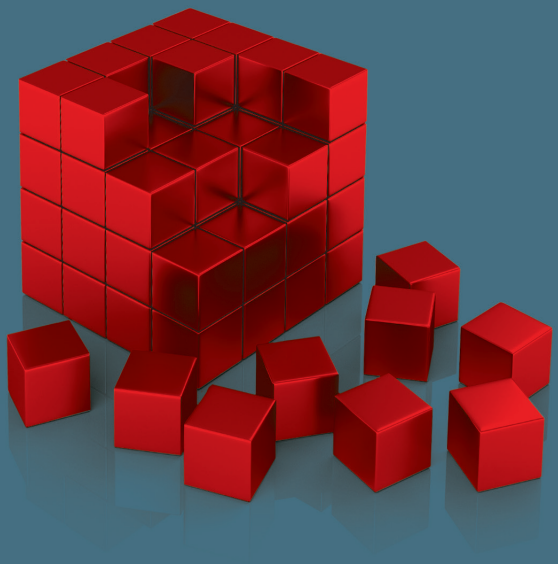
stefan TOTH

VORGEHENSMUSTER FÜR SOFTWARE-ARCHITEKTUR



KOMBINIERBARE PRAKTIKEN
IN ZEITEN VON AGILE UND LEAN

HANSER

Im Internet: www.swamuster.de

Toth

Vorgehensmuster für Softwarearchitektur

Stefan Toth

Vorgehensmuster für Softwarearchitektur

Kombinierbare Praktiken in Zeiten von Agile und Lean

HANSER

Der Autor:
Stefan Toth, Hamburg

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.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Petra Kienle, Fürstenfeldbruck
Herstellung: Irene Weilhart
Layout: Manuela Treindl, Fürth
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Stephan Rönigk
Druck und Bindung: Kösel, Krugzell
Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702
Printed in Germany

Print-ISBN: 978-3-446-43615-2
E-Book-ISBN: 978-3-446-43762-3

Inhalt

Geleitwort

Das Märchen vom agilen Architekten

„Heißt du etwa Rumpelstilzchen?“ –
„Das hat dir der Teufel gesagt, das hat dir der Teufel gesagt!“

(Kinder- und Hausmärchen der Brüder Grimm, 7. Auflage 1857)

Die schöne Müllerstochter, die aus Stroh Gold spinnen sollte, hat den Namen von Rumpelstilzchen nicht etwa geraten. Dazu hätte sie mehr als die drei bei den Gebrüdern Grimm beschriebenen Iterationen gebraucht. Sie hatte Wissen (nicht vom Teufel). Und als Ali Baba „Sesam, öffne Dich“ sprach, um in die Höhle mit unermesslichen Schätzen zu gelangen, hat er sich das auch nicht selbst ausgedacht. Er hat es sich abgeguckt von 40 Leuten, die schon mal drin waren in der Höhle. Er konnte auf deren Erfahrung zurückgreifen.

Der Schatz, um den es in diesem Buch von Stefan Toth geht, lässt sich verkürzt als Antwort auf folgende Frage beschreiben: Wie passt Softwarearchitekturmethodik zu einem zeitgemäßen Vorgehen? Oder besser noch: Wie können sie gemeinsam größeren Nutzen bringen?

Dass diese Frage viele bewegt, erlebe ich selbst regelmäßig in Workshops zu meinem Lieblingsthema Architekturdokumentation. Dort geht es darum, wie man Softwarearchitektur nachvollziehbar festhält und kommuniziert; in den Veranstaltungen drehen sich Fragen und Diskussionen regelmäßig darum, ob und wenn ja wie die gezeigten Zutaten zu einem agilen Vorgehen wie beispielsweise Scrum passen. Ganz allgemein können Sie das Interesse aber auch an den zahlreichen Blog- und Konferenzbeiträgen der letzten Jahre ablesen. Diese verknüpfen die Begriffe „agil“ (als griffigstes Wort für zeitgemäßes Vorgehen) und „Architektur“ mal mehr mal weniger pfiffig im Titel, etwa: „Jenseits des Elfenbeinturms – der agile Architekt“ oder „Architektur und agiles Vorgehen – ein Widerspruch?“. Und mehr noch ist es abzulesen an den vollen Sälen, wenn solche Vorträge stattfinden. Die Frage weckt Interesse. Gibt es gute Antworten?

Konferenzbeiträge – zumindest die, die ich gesehen habe – folgten in ihrem Ablauf häufig einem Schema: Zunächst werden die Begriffe „Agilität“ und „Architektur“ ausführlich definiert oder zumindest geklärt. Bei Agilität ist es dabei Folklore, das agile Manifest mit seinen berühmten vier Wertpaaren („Individuen und Interaktionen vor Prozessen und Werkzeugen“ etc.) auf eine Folie zu bannen. Dann wird der angebliche Widerspruch herausgearbeitet, der umso dramatischer ausfällt, je schwergewichtiger und klassischer das Verständnis von Softwarearchitektur, der zugrunde liegende Entwicklungsprozess und die damit verbundenen Artefakte in Notationen der 1990er-Jahre geschildert werden. Schließlich wird der Widerspruch durch sogenannte Best Practices aufgelöst („funktioniert doch super zusammen“).

Wolkige Tipps wie zum Beispiel: kein „Big Upfront Design“, auf die SOLID-Prinzipien achten, die Architektur iterativ und inkrementell entwickeln wie „den Rest“ auch ...

Die Zuhörer verlassen den Saal etwas enttäuscht. Alles richtig, gut und schön, aber wie genau machen wir das jetzt in unserem Projekt? Wo fangen wir an? Wenn schon kein Big Upfront Design, wie klein ist dann das richtige Small? Es liegt wohl auch, aber nicht nur am Format des Frontalvortrags und der oft kurzen Vortragsdauer (beliebt: 45 Minuten), dass die wirklich spannenden Fragen auf Konferenzen oft unbeantwortet bleiben. Mitunter fehlt es auch schlicht an ausreichenden praktischen Projekterfahrungen. Märchenstunde?

Für mich steht außer Zweifel, dass Stefan Toth die nötige Erfahrung besitzt. Er hat sehr unterschiedliche Projekte über einen längeren Zeitraum begleitet und zahlreiche einschlägige Workshops durchgeführt. Bei den Kunden wurde mal klassisch, mal agil, mal irgendwie dazwischen vorgegangen und auch die Branchen könnten unterschiedlicher kaum sein. Vom Finanzsektor bis zur Gaming-Plattform war alles dabei. Das Themenspektrum umfasste die methodische Softwarearchitektur vom Entwurf bis zur Bewertung von konkreten Architekturentscheidungen. So hat Stefan beispielsweise ein agiles Team begleitet und befähigt, regelmäßige Architekturbewertungen in ihren Entwicklungsprozess zu integrieren und eigenverantwortlich durchzuführen. Während viele bei Architekturbewertung sofort an schwergewichtige Methoden wie ATAM denken, wirkt hier nun ein schlankes, aber wirkungsvolles Set an Elementen, bei großer Akzeptanz im Team.

Das ist vielleicht auch schon die Grundidee des Buchs: Es gibt nicht den einen Weg für alle Projekte. Aber es gibt bewährte und schlanke Praktiken in Form von Puzzleteilen, die Nutzen stiften.

In einigen Projekten und Workshop-Situationen, eigentlich in zu wenigen, hatte ich als Kollege das Vergnügen, mit Stefan Toth zusammenzuarbeiten, und konnte wie die Mitarbeiter der Kunden an seinem Wissen und seinen Erfahrungen teilhaben. Und so freut es mich, dass Sie nun als Leser dieses Buchs ebenfalls davon profitieren können.

Denn Stefan Toth hat ein passendes Format zur Vermittlung seines Wissens und Könnens gewählt. Anders als es in einem knappen Vortrag möglich wäre, stellt er hier im Buch seine Ideen ausführlich dar und illustriert sie mit Beispielen. Gleichzeitig ist das Buch lebendig und kein langweiliger Schmöker. Stefan hat viel von seinem Witz in die Zitate und Antipatterns einfließen lassen, ohne dabei albern oder unsachlich zu werden. Die Idee, die einzelnen Zutaten als kombinierbare Muster darzustellen, macht die Inhalte nicht nur leichter erlernbar, sondern vor allem auch einzeln anwendbar. Das erleichtert den Start in Ihrem Projekt ungemein. Die einzelnen Zutaten sind trotzdem kein loses Schüttgut, sondern gut aufeinander abgestimmt und in ihrer Gesamtheit schlüssig. Ausdrucksstarke Visualisierungen – eine besondere Spezialität von Stefan – vermitteln komplizierte Inhalte gut erinnerbar und verknüpfen die einzelnen Muster.

Aus eigener Erfahrung kann ich sagen, dass die Erarbeitung und Aufbereitung von Inhalten in Form eines Buchs große Vorteile bietet (die hier auch ausgeschöpft wurden), aber auch einen nicht zu unterschätzenden Nachteil, zumindest verglichen mit Vorträgen oder einem Workshop. Es besteht die Gefahr, dass man als Autor weniger Feedback bekommt. Ich möchte Sie daher ermutigen, Erfahrungen, die Sie mit den dargestellten Praktiken machen konnten, zu teilen. Tauschen Sie sich aus, mit dem Autor und auch mit anderen Lesern.

Um zum Schluss noch mal auf Rumpelstilzchen zurückzukommen: In diesem Buch lernen Sie nicht, wie Sie aus Stroh Gold spinnen. Dafür viele andere Dinge, die Sie jetzt vermutlich

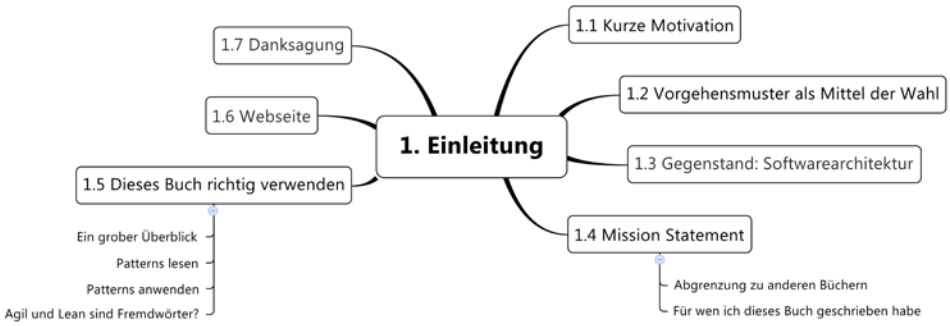
auch noch nicht können. Und es ist kein Märchen. Alles ist wahr. Wenn Sie mögen, schließen Sie das Buch nun kurz, sprechen mir nach: „Sesam, öffne Dich“, und schlagen es wieder auf. Und es tut sich tatsächlich ein reicher Schatz an Erfahrungswissen auf, der nur darauf wartet, Stück für Stück heraus in Ihr Projekt getragen zu werden. Mir bleibt nur noch, Ihnen viel Freude damit zu wünschen.

Stefan Zörner im Herbst 2013

1 Einleitung

Lesen Sie dieses Buch nicht. Legen Sie es weg. Jetzt.

Falls Sie dieser Empfehlung nicht gefolgt sind, haben Sie eine wichtige Voraussetzung für die erfolgreiche Lektüre bereits erfüllt: Sie glauben mir nicht alles, sondern denken selbst. Die anderen Voraussetzungen schaffe ich in diesem Kapitel. Ich bringe Ihnen zunächst die Form und das Thema des Buchs näher, erläutere grob den Aufbau und gebe einige (ernst gemeinte) Hinweise zum Umgang mit den Inhalten und Konzepten. Dazwischen erfahren Sie, ob dieses Buch etwas für Sie ist – das „Mission Statement" grenzt die Inhalte zu anderen Büchern ab und definiert die Zielgruppe.



1.1 Kurze Motivation

Ich habe dieses Buch erarbeitet, indem ich meine eigene Vorgehensweise angewandt habe. Dazu gehört, dass unwichtige Teile später bearbeitet werden und, wie es mit einer Timebox nun mal ist, eventuell hinten runterfallen.

So ist es mit der „kurzen Motivation" passiert, die Sie gerade lesen. Sie war ständig niedrig priorisiert und hat es am Schluss nicht geschafft. Sorry! Ich musste mich auf das Wesentliche konzentrieren: das, was Sie mitnehmen können, das, was Ihr Projekt bereichert, das, was Sie zum besseren Entwickler und Architekten macht. Da ist kein Platz für Motivation. Sie können es positiv sehen: Sie waren motiviert genug, dieses Buch zu kaufen oder zumindest es aufzuschlagen. Das ist doch was!

Das Einzige was ich Ihnen hier inhaltlich mitgeben kann, ist ein Zitat von Taiichi Ohno[1], das ich schon recht früh in diesen Unterabschnitt geworfen habe, weil es eine zentrale Idee des Buchs gut verkörpert: *„Es gibt so etwas wie Standardarbeit, aber Standards sollten permanent angepasst werden. Wenn Sie vom Standard als das Beste denken, was Sie leisten können, ist alles vorbei."* Er fährt fort, indem er sagt, wenn wir etwas als den *„bestmöglichen Weg"* etablieren, *„wird die Motivation für Kaizen [kontinuierliche iterative Verbesserung] verschwunden sein."* [Pop06][2].

Versuchen Sie in diesem Sinne, die Inhalte dieses Buchs als alleinstehende, aber kombinierbare Verbesserungsideen Ihrer Praxis zu verstehen. Sie kommen aus einem eher klassischen Projektkontext? Lassen Sie sich von leichtgewichtigeren Ideen inspirieren und verzahnen Sie die Architekturdisziplin effektiv mit der Entwicklung. Sie arbeiten in einem agilen Projekt? Experimenten Sie mit den vorgestellten Praktiken, um Architekturaufgaben effektiver im Team zu erledigen oder ein besseres Gefühl für die Architekturdisziplin zu bekommen. Versuchen Sie, Ihren Standardweg zur Architekturentwicklung zu hinterfragen, Schwächen zu erkennen und Stärken auszubauen. Egal, ob Sie nun eher klassisch oder eher agil unterwegs sind: Werden Sie mit Hilfe dieses Buchs ein bisschen besser. Ständig.

1.2 Vorgehensmuster als Mittel der Wahl

Um eine stückchenweise Verbesserung an Ihrer Architekturarbeit gut zu unterstützen, habe ich mich dafür entschieden, Patterns bzw. Muster zu beschreiben. Diese Form der Beschreibung auf Methodik- und Vorgehensebene einzusetzen, ist unüblich[3], ermöglicht es mir aber, gezielt auf Probleme in Softwareprojekten einzugehen. Mit der Zerlegung in zeitgemäße, problemorientierte Architekturpraktiken ist Architekturvorgehen weniger starr und weniger fordernd. Die Praktiken sind leichter erlernbar, einfacher auszuprobieren und generieren weniger Widerstand in der (Projekt-)Organisation. Statt eines aufwendigen „Tailoring" nehmen Sie sich einfach, was Sie brauchen.

Muster sorgen auch dafür, dass Lösungen wiederkehrender Probleme Namen bekommen. Selbst wenn Sie von der beschriebenen Musterlösung abweichen, müssen Sie Ihren Ansatz nicht von Grund auf neu erklären, sondern können den Unterschied zum bekannten Muster erläutern.

Trotz der Stückelung sind die beschriebenen Praktiken nicht voneinander isoliert. Die Muster verweisen aufeinander und können im Verbund eingesetzt werden – sie helfen so auch architektonisch risikoreichen Projekten (siehe Abschnitt 2.1.2). Insgesamt entsteht eine Architekturdisziplin, die schnelle Resultate liefert und Stück für Stück einführbar ist.

---

[1] Erfinder des Toyota Production Systems und Urvater von Lean

[2] Originalzitat auf Englisch: Taiichi Ohno: *„there is something called standard work, but standards should be changed constantly. Instead, if you think of the standard as the best you can do, it's all over."* Ohno goes on to say that if we establish something as the *„best possible way, the motivation for kaizen [continuous incremental improvement] will be gone."*

[3] Als ich mit diesem Buchprojekt begonnen habe, war mir kein einziges methodisch orientiertes Pattern-Buch bekannt, zwei ernstzunehmende Vertreter dieses Genres konnte ich jedoch mittlerweile in Erfahrung bringen: [Lef10][Els08]).

1.3 Gegenstand: Softwarearchitektur

Die Vorgehensmuster dieses Buchs sind nicht nur gut kombinierbar, sie prägen insgesamt auch eine zeitgemäße Vision von Softwarearchitektur aus. Bevor ich in Kapitel 2 inhaltlich in diese Vision einsteige, sei ein kurzer Blick auf die Disziplin an sich gestattet: „Was ist Softwarearchitektur?“

Wenn Sie diese Frage zehn Softwareentwicklern stellen, werden Sie neun bis zehn unterschiedliche Antworten erhalten. Und das liegt nicht unbedingt an Unwissenheit: Das Software Engineering Institute der Carnegie Mellon Universität sammelt Definitionen für Softwarearchitektur und hält derzeit bei knapp unter 200 [SEI13]. Eine recht einfache und meist konsensfähige Definition kommt von Martin Fowler:

> *„To me the term architecture conveys a notion of the core elements of the system, the pieces that are difficult to change. A foundation on which the rest must be built.“* [Fow04][4]

Obige Aussage ist deshalb sehr reizvoll, weil sie kein Set von zu erstellenden Artefakten vorgibt und keine Entscheidungsarten definiert, die immer architekturrelevant sind. Stattdessen wird eine klare Botschaft formuliert: Wenn es schwer änderbar ist, ist es Softwarearchitektur. Daraus lassen sich zwei wichtige Feststellungen ableiten:

1. **Softwarearchitektur ist wichtig**:
   „Schwere“ Änderbarkeit definiert sich darüber, dass Änderungen teuer, aufwendig oder qualitätsgefährdend sind. Entsprechende Entscheidungen bedrohen zentrale Rahmenbedingungen des Projekts (Projektbudget, Zeitplan oder Produktqualität). Eine Definition von Eoin Woods stellt diesen Aspekt zentral heraus: *„Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be canceled“* [Roz11][5]
2. **Unwichtige Fragestellungen verdienen keine Architekturaufwände**:
   Es gibt auch Entscheidungen, die leicht zurückzunehmen oder anzupassen sind. Die meisten Fragestellungen in der Projektpraxis fallen in diese Kategorie und sind damit *nicht* architekturrelevant. Architekturaufwände lohnen sich bei diesen Fragestellungen weniger und bremsen Ihr Projekt unnötig. George Fairbanks betont: *„You should pay as much attention to software architecture as it contributes risk to the overall project, since if there is little architecture risk, then optimizing it only helps little.“* [Fai10][6]

---

[4] Deutsch etwa: Für mich drückt Softwarearchitektur die Idee von Kernelementen des Systems aus, jene Teile, die schwer änderbar sind. Ein Fundament, auf dem der Rest aufbauen muss.

[5] Deutsch etwa: Softwarearchitektur ist die Menge der Entwurfsentscheidungen, welche, wenn falsch getroffen, Ihr Projekt scheitern lassen können.

[6] Deutsch etwa: Die Aufmerksamkeit, die Sie Softwarearchitektur entgegenbringen, sollte vom Risiko bestimmt werden, das von Architekturfragen ausgeht. Wenn wenig Architekturrisiko für das Gesamtprojekt besteht, hilft Architekturoptimierung auch wenig.

Softwarearchitektur vs. XY-Architektur

Es gibt viele Architekturdisziplinen und noch mehr, teilweise unternehmensspezifische, Namen dafür. In Anlehnung an [Woo08], möchte ich pragmatisch drei Ebenen definieren:

- **Unternehmensarchitektur** (Geschäftsarchitektur, strategische Architektur, Domänenarchitektur etc.)
- **Softwarearchitektur** (Applikationsarchitektur, Systemarchitektur, Lösungsarchitektur etc.)
- **Betriebsarchitektur** (technische Architektur, Technologiearchitektur, Integrationsarchitektur etc.)

Der Fokus dieses Buchs liegt auf *Softwarearchitektur*. Ich verwende im gesamten Buch die Begriffe „Softwarearchitektur“ und „Architektur“ synonym. Die beschriebenen Vorgehensmuster sind für die Erarbeitung einer Softwarelösung im Rahmen *eines* Projekts gedacht.

Nicht projektübergreifende Aspekte der Betriebsarchitektur lassen sich ebenfalls damit bearbeiten (und sind in der Projektpraxis oft mit Softwarearchitekturaufgaben vermischt). Arbeiten Sie übergeordnet auf strategischer Ebene, können Sie von einzelnen Ideen profitieren, müssen die Muster aber an ihren Kontext anpassen. ■

1.4 Mission Statement

Dieses Buch stellt praxiserprobte Praktiken vor, die Ihrem Projekt helfen, Herausforderungen der Softwarearchitektur zu meistern. Die Praktiken sind in Musterform beschrieben, um sie möglichst klar darzulegen, einfach verständlich zu machen und vor allem: sie häppchenweise erlern- und anwendbar zu machen. So ermöglicht dieses Buch auch eine skalierbare Methodik. Kleine und einfache Projekte können sich die nötigen Rosinen aus dem Sack von Mustern picken, größere komplexere Projekte können mehr Praktiken übernehmen.

Inhaltlich setzt sich dieses Buch folgende Ziele:

- Neue Ideen und gut funktionierende Praktiken aus modernen Vorgehensmodellen, in den Architekturwerkzeugkasten übertragen. Durch den Einsatz der enthaltenen Muster entsteht eine effektivere, zeitgemäße Architekturdisziplin.
- Architektur in Projekte integrierbar machen, die ein leichtgewichtiges Vorgehensmodell haben. Durch den Einsatz der enthaltenen Muster entsteht eine in das Projekt integrierte Architekturdisziplin, die so schlank wie möglich und so fundiert wie nötig arbeitet.
- Architekturarbeit dynamischer gestalten. Durch den Einsatz der enthaltenen Muster werden Projekte bis zu mehreren Teams in die Lage versetzt, dezentral und schnell zu tragfähigen