Softwarearchitektur für Dummies

DAS VORWORT

Einleitung

Eine Anwendung ohne eine durchdachte Software-Architektur zu entwickeln, ist keine gute Idee. Wenn man ein Gebäude für einen bestimmten Zweck haben will, sagen wir, eine neue Schule, dann geht man ja auch zu einen Architekten, der schon Schulen realisiert hat. Er weiß, wie Schulen genutzt werden und woran man deshalb gleich zu Beginn denken muss, damit am Ende Schüler und Lehrer gern dort sind und mit Freude und Erfolg gelernt wird. Man würde nicht zum lokalen Bauunternehmer gehen, der zufällig die Schule in der Nachbarstadt gebaut hat und sagen: »Du kannst doch Schulen, bau uns auch eine.«

Der Software-Architekt oder die Software-Architektin[1] hat eine genauso spannende Aufgabe wie der Schul-Architekt: Er muss ein guter Techniker sein, damit seine Planungen effektiv implementiert werden können. Er muss für möglichst viele Aufgaben, Probleme und Herausforderungen passende Strategien und Software-Strukturen parat haben. Er muss für das einzelne Projekt strategisch und oft kreativ eine Auswahl treffen und einen Gesamtplan entwickeln. Und schließlich muss er auch noch ausführlich mit den beteiligten Menschen sprechen und ihnen genau zuhören: mit Benutzern, Entwicklern, Projektleitern, Teamleitern, Budgetverantwortlichen und vielen weiteren – denn sie alle haben Erwartungen und Ziele, die in Architektur, Design und Implementierung einfließen.

Klingt ein bisschen kompliziert? Kein Problem, denn Sie haben das richtige Buch in der Hand. Die Rückmeldungen von Studierenden in meinen Vorlesungen und Seminaren haben mich motiviert, immer sehr konkret, mit der Implementierung im Blick, vorzugehen. Vor allem ist es mir wichtig, Technik, Patterns und Strategien nebeneinander zu stellen und ihre Querbeziehungen aufzuzeigen. Denn dann wird es richtig spannend, weil man Alternativen abwägen und Entscheidungen treffen kann, um zu guten Designs und tragfähigen Architekturen zu kommen. Und aus meinen eigenen Software-Projekten in vielen verschiedenen Anwendungsbereichen bin ich sicher: Nichts davon ist »Theorie«, alles wird früher oder später in Ihre eigenen Architekturen einfließen.

Über dieses Buch

Software-Architektur versteht man erfahrungsgemäß am besten anhand von konkreten Aufgabenstellungen und den passenden Lösungen. Deshalb enthält dieses Buch:

- ✔ Software-Patterns für häufig auftauchende Aufgaben und Herausforderungen
- ✔ Ziele, Argumente und Strategien zur Auswahl von Patterns und zur Entscheidung zwischen alternativen Lösungen
- ✔ technische Hintergründe zu Architekturen und Entscheidungen

Sie brauchen das Buch nicht von vorn nach hinten durchzulesen. Jedes Kapitel behandelt diese Punkte abgeschlossen für einen bestimmten Aufgabenbereich. Je mehr Kapitel Sie

[1] Der Einfachheit halber benutze ich ab jetzt immer männliche Bezeichnungen. In meinen Seminaren merke ich aber aufgrund der durchdachten Beiträge: Es sollte viel mehr Software-Architektinnen geben!

lesen, desto mehr gewinnen Sie einen Rundumblick und desto mehr Werkzeuge haben Sie zur Verfügung. Aber Sie können natürlich dort beginnen, wo im aktuellen Projekt Fragen auftauchen, oder einfach bei dem Ansatz, der Sie gerade am meisten interessiert.

Konventionen in diesem Buch

Englische Fachbegriffe habe ich beibehalten, wenn es nicht gerade sehr gängige deutsche Entsprechungen gibt. Allerdings habe ich die Rechtschreibung angepasst, um das Lesen zu erleichtern: Nomen sind groß geschrieben und zusammengesetzte Begriffe stehen wie im Deutschen mit Bindestrich statt wie im Englischen in getrennten Wörtern. Beim ersten Auftreten ist ein Fachbegriff *kursiv* gesetzt, später nicht mehr.

Die meisten Konzepte lassen sich gut mit Beschreibungen und Illustrationen erläutern. Dort, wo es um technische Details geht und Quelltext notwendig wird, ist er in `Typewriter`-Font gesetzt.

Mit Ausnahme von Implementierungsbeispielen habe ich darauf verzichtet, einzelne Produkte oder Frameworks vorzustellen, weil das leicht wie eine Empfehlung oder Standardlösung wirkt und außerdem schnell veraltet. Die Produktauswahl ist ein wichtiger Teil der Architektur und speziell für jedes Projekt. Mit den Konzepten und Begriffen aus dem Text finden Sie aber leicht immer mehrere Kandidaten und können dann die besprochenen Auswahlkriterien anwenden. Die im Umfang größte Ausnahme hiervon ist Java: Die Plattform und das Ökosystem bieten viele kleine Beispiele unter einem Dach, und weil Sie wahrscheinlich Java schon begegnet sind, finden Sie sich leicht zurecht. Alle gezeigten Mechanismen gibt es analog aber auch in vielen anderen Plattformen.

Was Sie nicht lesen müssen

Die Kapitel sind so geschrieben, dass Sie sie unabhängig voneinander lesen können. Weitgehend können Sie auch bei einzelnen Abschnitten innerhalb des Kapitels einsteigen, wenn Ihnen eine Überschrift mit einem interessanten Begriff ins Auge springt. Allerdings enthält der Text keine Rückwärtsverweise. Damit Sie wissen, was Sie verpasst haben, sollten Sie wenigstens die vorherigen Überschriften im Kapitel kurz überfliegen. Die Stichworte »weiter oben« und »weiter unten« beziehen sich immer auf Abschnitte innerhalb desselben Kapitels. Wenn es zu einem Thema ausführlichere Erklärungen anderswo im Buch gibt, verweise ich auf das jeweilige Kapitel.

Außerdem habe ich zwei Formen von »Abkürzungen zum Ziel« eingebaut, wo die Vorstellung von Patterns oder Argumenten einmal ausführlicher sein muss, damit sie verständlich wird.

- ✔ Ein separater Abschnitt »Folgerungen für die Architektur« fasst rückblickend über mehrere Abschnitte zusammen, welche Punkte relevant sind, wenn Sie sich fragen, ob der vorgestellte Ansatz für Ihr Projekt sinnvoll sein kann.

- ✔ Am Ende eines längeren Abschnitts mit eher detaillierten Inhalten fasst ein »Bindfaden«-Symbol, das weiter unten noch gezeigt wird, die Haupteinsicht zusammen.

Törichte Annahmen über die Leser

Weil Sie gerade ein Buch über Software-Architektur anschauen, nehme ich einmal an, dass Sie auf irgendeine Weise mit der Entwicklung von Anwendungen zu tun haben, sei es als Entwickler, Designer, Projektleiter oder Architekt. Wahrscheinlich interessiert Sie mindestens einer der folgenden Punkte:

- ✔ Sie wollen Entscheidungen zum Aufbau einer Anwendung treffen.
- ✔ Sie wollen verstehen, ob eine bestimmte Architektur oder bestimmtes Framework für Ihr Projekt geeignet ist.
- ✔ Sie wollen Hypes und Marketing-Erzählungen rund um Software-Architekturen endlich durchschauen.
- ✔ Sie wollen als Entwickler so designen und implementieren, dass Sie bestimmte langfristige Ziele erreichen.
- ✔ Sie wollen als Entwickler eine vorgegebene Architektur verstehen, um sie richtig umzusetzen.
- ✔ Sie interessieren sich allgemein dafür, welche Herausforderungen man bei der Erstellung oder Weiterentwicklung einer Anwendung lösen muss.

An vielen Stellen versteht man Architektur, vor allem die Entscheidungen, nur, wenn man die Implementierung kurz durchdenkt. Und das tut der Text dann auch. Deshalb nehme ich auch an, dass Sie schon ein wenig Programmiererfahrung mitbringen. Praktische Anwendungen müssen es nicht sein, Praktika oder Übungen in fortgeschrittenen Vorlesungen an der Hochschule oder Universität reichen sicher aus.

Wie dieses Buch aufgebaut ist

Dieses Buch gliedert sich in fünf Teile, die allerdings nicht aufeinander aufbauen, sondern jeweils eigenständige Aspekte von Software-Architektur behandeln.

Teil I: Überblick

Zuerst geht es um einen kurzen Überblick über die Aufgaben von Software-Architektur und die Rolle des Software-Architekten. Die nicht-technischen Überlegungen hinter Architektur-Entscheidungen in Kapitel 2 führen in Seminaren immer zu den spannendsten Diskussionen: Wie entscheiden wir uns zwischen alternativen Lösungen, wenn keine eindeutig besser geeignet ist?

Teil II: Elemente von Architekturen

Der zweite Teil bringt Inhalte, die grundlegend für die später vorgestellten Software-Architekturen sind, aber eigentlich nicht zur Software-Architektur selbst gehören. Sie kommen deshalb normalerweise in separaten Vorlesungen oder Büchern vor. Damit Sie sich das sparen können, habe ich die relevanten Aspekte hier kurz zusammengefasst.

Einigen Raum nehmen technische Grundlagen ein, die bei vielen Architektur-Überlegungen unverzichtbar sind. Hier folge ich der Idee der *Mechanical Sympathy*, die ursprünglich von einem Rennfahrer formuliert wurde: Man fährt besser Auto, wenn man genau weiß, warum das Auto wie reagiert.

Teil III: Klassische Patterns und Stile

Der dritte Teil stellt klassische Software-Patterns für klassische Aufgabenstellungen vor. Eigentlich ein Muss für jeden Software-Architekten. Ein besonderer Fokus liegt auf der Motivation und Erläuterung der Herausforderungen und auf der Abwägung zwischen alternativen Lösungsmöglichkeiten.

Teil IV: Architekturen für die Cloud

Die Cloud bietet, vor allem durch die Virtualisierung von Servern, ganz neue Möglichkeiten für die Verteilung von Anwendungen auf verschiedene physische Rechner. Verteilte Systeme werden quasi zum Normalfall. Entsprechend brauchen Anwendungen aber auch Strukturen und Mechanismen, um zuverlässig mit den Unwägbarkeiten von verteilten Anwendungen umzugehen. Dazu gibt es verschiedene Architektur-Ansätze.

Teil V: Top-Ten

Die Informatik und die Software-Technik sind sehr schnelllebige Gebiete. Das birgt die Gefahr, dass sich Geschichte wiederholt: im Guten, wenn dieselben Aufgaben mehrmals und vielleicht immer besser gelöst werden. Aber auch im Schlechten, wenn dieselben Sackgassen immer wieder ausgelotet werden, in der Hoffnung, mit neuen Technologien endlich eine Lösung zu finden. Deshalb ist es auch für uns Software-Ingenieure wichtig zu wissen, wo wir herkommen: Was sind die grundlegenden Ideen und Meilensteine? Was war letztlich ein Hype, aber können wir daraus lernen?

Symbole, die in diesem Buch verwendet werden

Einige Absätze sind aus dem Textfluss hervorgehoben, damit sie gleich ins Auge springen.

Hinter dem Bindfaden-Symbol stehen Dinge, die Sie sich merken sollten. Das können Details sein, aber auch zusammenfassende Einsichten, die als Take-Home-Message noch mal eigenständig formuliert sind.

Hier stehen Hinweise, die das Verständnis erleichtern oder Querbeziehungen zu anderen Inhalten zeigen.

Das Warnschild zeigt Fallen an, in die man gedanklich leicht tappt, wenn man einen bestimmten Aspekt übersieht. Oft betrifft das scheinbare Abkürzungen, von denen man erst hinterher sieht, dass es eigentlich Sackgassen sind.

Hier geht es um technische Details, die für das konzeptuelle Verständnis nicht unbedingt notwendig sind.

Die gerade ausgeführten Konzepte brauchen dringend eine Erläuterung. Sie werden daher anhand eines längeren Beispiels nochmals ausführlicher beleuchtet.

Wie es weitergeht

Jetzt sind wir startklar! Sie wissen, was Sie im Buch erwartet und wo Sie was finden.

Sie müssen nur noch entscheiden, wo Sie anfangen wollen zu lesen: Wollen Sie den neuesten Hypes folgen? Dann empfehle ich eines der Kapitel aus Teil IV. Oder wollen Sie erst einmal schauen, was eigentlich wirklich hinter den »normalen« und scheinbar offensichtlichen Patterns wie Layers steckt? Dann fangen Sie mit Kapitel 7 an. Persönlich möchte ich Ihnen ans Herz legen, Kapitel 6 nicht zu lange zu verschieben, denn die Daten sind in vielen Fällen der eigentlich wertvolle Kern der Anwendung und bestimmen den langfristigen Erfolg stark mit.

Wie auch immer Sie sich entscheiden: Ich wünsche Ihnen viel Spaß und gute Erkenntnisse für Ihre nächsten Projekte!