

# Knigge für Software- architekten

Peter Hruschka und Gernot Starke





Peter Hruschka, Gernot Starke

# Knigge für Softwarearchitekten

schnell+kompakt

entwickler.press

Peter Hruschka, Gernot Starke  
Knigge für Softwarearchitekten  
schnell+kompakt  
ISBN: 978-3-86802-267-4

© 2012 entwickler.press  
ein Imprint der Software & Support Media GmbH

<http://www.entwickler-press.de>  
<http://www.software-support.biz>

Ihr Kontakt zum Verlag und Lektorat: [lektorat@entwickler-press.de](mailto:lektorat@entwickler-press.de)

Bibliografische Information Der Deutschen Bibliothek  
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Lektorat: Sebastian Burkart  
Korrektur: Frauke Pesch  
Satz: Dominique Kalbassi  
Umschlaggestaltung: Maria Rudi  
Belichtung, Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, Paderborn.

Alle Rechte, auch für Übersetzungen, sind vorbehalten. Reproduktion jeglicher Art (Fotokopie, Nachdruck, Mikrofilm, Erfassung auf elektronischen Datenträgern oder andere Verfahren) nur mit schriftlicher Genehmigung des Verlags. Jegliche Haftung für die Richtigkeit des gesamten Werks, kann, trotz sorgfältiger Prüfung durch Autor und Verlag, nicht übernommen werden. Die im Buch genannten Produkte, Warenzeichen und Firmennamen sind in der Regel durch deren Inhaber geschützt.

# Knigge?

Der klassische Knigge [1], Originaltitel „Über den Umgang mit Menschen“ beschreibt Umgangsformen unter Menschen, insbesondere die anzustrebenden „guten Manieren“: Sie sollen nicht mit vollem Mund bei Tisch sprechen, nicht die Finger ablecken, alten Damen über die Straße helfen und so weiter.

Damit machen Sie sich im täglichen Leben beliebt und können Eindruck schinden. Zur Berufslaufbahn „Softwarearchitekt“ hingegen schweigt die klassische Benimmliteratur.



Der Schriftsteller Freiherr Adolph Knigge  
(1752-1796)

## Erfolgsmuster

1. Der Proaktive	15
3. Der Vielsehende	27
4. Strukturierte Faulheit	37
6. Blick in den Rückspiegel	45
8. Der Multilinguist	57
11. Die Jongleuse	73
12. Der Vereinfachungskobold	81
14. Der technische Risikomanager	95
16. Die Lektorin	105
18. Der Entscheider	119
19. Die ständig Lernenden	127
20. Die Kommunikatorin	135
23. Der edle Ritter	159
24. Der Schmökerer	163

## Anti-Patterns

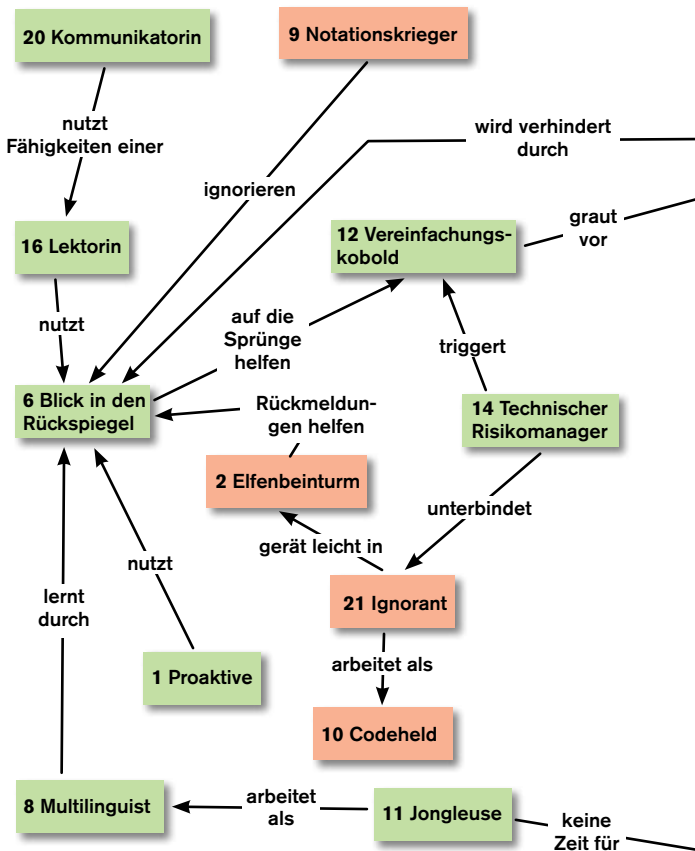
2. Elfenbeinturm	21
5. Der Diktator	41
7. Zu viel des Guten	51
9. Der Notationskrieger	63
10. Der Code-Held	67
13. Der Perfektionist	89
15. Der Prozessprediger	101
17. Der Verschätzer	113
21. Der Ignorant	141
22. Toolistan	149

## Weitere Goodies

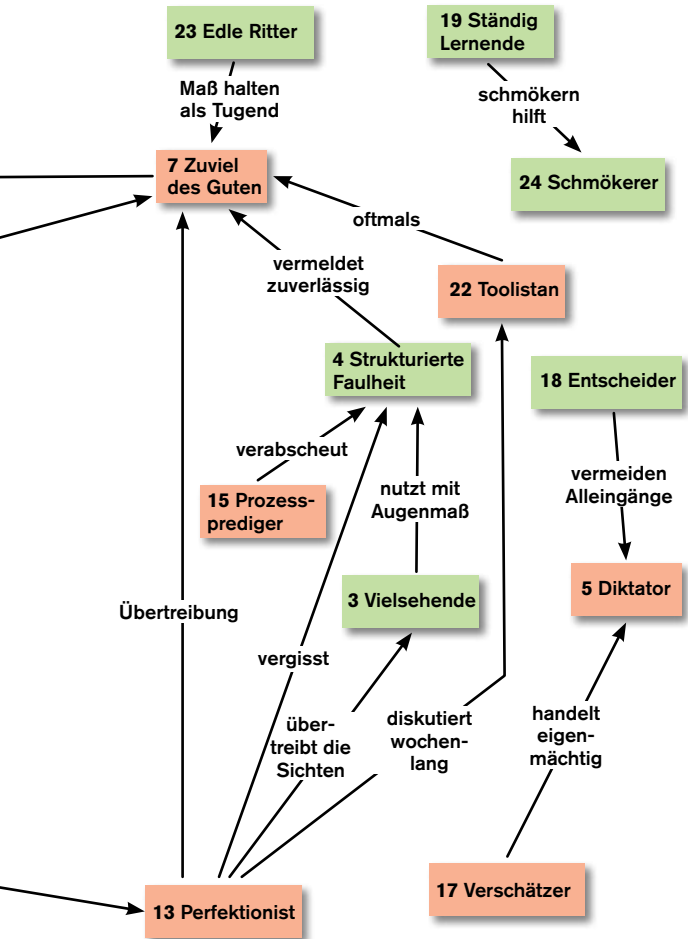
25. arc42 - The Elevator Pitch	171
26. Agile Modeling Guide	179
27. Eine Sache noch...	185
Über uns	191

# Knigge Pattern Language

© Peter Hruschka & Gernot Starke









# readme.first

In unserem Berufsleben durften wir Systeme in unterschiedlichen Branchen und bei vielen verschiedenen Kunden entwerfen und begleiten. Dazu gehörten Embedded Systems, Informationssysteme, Anlagen- und Produktionssteuerung, Web- und Data-Warehouse-Anwendungen, entwickelt und betrieben auf Mainframes, Client-/Server Clustern bis zu Standalone-Anwendungen. Dabei haben wir Licht und Schatten erlebt, sowohl hervorragend produktive als auch grausig schlechte Projekte, mit und teilweise auch ohne zugehörige Architekten. In diesem Buch stellen wir unsere Beobachtungen von SoftwarearchitektInnen und deren Verhalten in Form von „Mustern“ dar. Organisationsmuster für andere Bereiche der IT finden Sie in [2] und [3].

Wir gehen einen etwas anderen Weg als der alte Freiherr von Knigge, indem wir bewusst sowohl gutes wie auch schlechtes Verhalten von Softwarearchitekten vorstellen.

Sie lernen, wie man durch Erfolgsmuster bessere Systeme konstruiert und schlechte Architekturmanieren („Anti-Patterns“) vermeidet.

## **Gute Architekten bauen gute Systeme**

Wir sehen als wesentliches Merkmal guter Architekten, dass sie unter den jeweiligen Umständen die bestmöglichen Systeme konstruieren und deren Entwicklung begleiten. Systeme, die verständlich, langlebig, wartbar, funktional, performant und sicher sind. Systeme, die robust auf Fehler reagieren und ihre jeweiligen

Stakeholder positiv erstaunen, statt zu nerven. Kurz gesagt: Gute Architekten liefern hohe Qualität.

## **Gutes Verhalten macht gute Architekten**

Wir glauben fest daran, dass der Unterschied zwischen guten und schlechten Softwarearchitekten hauptsächlich in deren Verhalten begründet liegt, in ihrer Vorgehensweise oder Methodik.

Technologie, Frameworks oder Tools beeinflussen die Qualität von Lösungen erheblich weniger, obwohl Softwarearchitekten sie natürlich kennen und können müssen. Daher haben wir technische Themen in diesem Buch komplett ausgespart. Wir gehen (optimistisch, aber durch langjährige Beobachtung gestützt) davon aus, dass Softwarearchitekten ihr IT-technisches Handwerkszeug in der Regel ziemlich gut beherrschen.

### **HINWEIS**

In farbigen Textkästen geben wir Ihnen ernst gemeinte Ratschläge zur praktischen Umsetzung.

### **WAR STORY**

In manchen Kapiteln erzählen wir Ihnen beispielhafte Erlebnisse aus unseren Projekten – optisch gekennzeichnet wie dieser Absatz. Diese haben wir bei Bedarf mit unseren Namenskürzeln (PH und/oder GS) versehen.

## **Links und Literatur**

- [1] Wikipedia zu „Umgangsformen“: <http://de.wikipedia.org/wiki/Umgangsformen>, abgerufen 28. Februar 2010
- [2] Coplien, J.; Harrison, N.: „Organizational Patterns of Agile Software Development“. Prentice-Hall 2004
- [3] DeMarco, T.; Hruschka, P.; The Atlantic Systems Guild: „Adrenalin-Junkies und Formular-Zombies“. Hanser 2007

## Danksagung

Wir bedanken uns bei unseren zahlreichen Kunden, bei denen wir das Verhalten von Softwarearchitekten beobachten durften. Dort, im Dschungel der Projektpraxis, haben wir am eigenen Leib die hier geschilderten Muster und Verhaltensweisen erlebt und erlitten – danke dafür! Dank auch an die vielen Seminarteilnehmer, an denen wir unsere Thesen ausgetestet haben – und oftmals erstauntes, aber kopfnickendes Feedback auf unsere Interpretation der Aufgaben eines Softwarearchitekten bekommen haben.

Viele Mitglieder des iSAQB e. V., insbesondere Prof. Dr. Andreas Rausch und Prof. Dr. Arne Koschel haben uns mit intensiven, inhaltlichen Diskussionen rund um die Rolle von Softwarearchitekten unterstützt.

Weiterhin danken wir Claudia Fröhling und Sebastian Burkart vom Software & Support Media Team für ihren Optimismus und ihre Unterstützung bei diesem Buchprojekt.

Gernot: Uli, Lynn und Per: Ihr seid super, die beste Familie im Universum! Zeit mit euch ist immer zu kurz. Danke auch an meine kundigen Kollegen der innoQ (Christian Albrecht, Thomas Bandholtz, Philipp Ghadir, Martin Huber, Arnd Kleinbeck, Andreas Krüger, Till Schulte-Coerne, Christopher Stolle, Stefan Tilkov) für eure gründlichen (und manchmal gnadenlosen...) Reviews...

Peter: Mein besonderer Dank gilt meiner Traumfrau Monika, die ein weiteres Buchprojekt nicht nur toleriert, sondern durch Kommentare aus einer nicht-IT-Sicht wesentlich bereichert hat.



# Der Proaktive

Verantwortungsbewusste Softwarearchitekten gehen aktiv auf alle Projektbeteiligten zu, um Chancen und Risiken rechtzeitig zu erkennen und geeignete Maßnahmen einleiten zu können. Sie übernehmen die Initiative, starten notwendige Aktivitäten aus eigenem Antrieb und ohne explizite Aufforderung. Anstatt passiv oder reaktiv abzuwarten, bis jemand anderes mit einer ungelösten Aufgabe zu ihnen kommt, gehen Aktive diese Aufgaben selbstständig an.

In diesem Sinne ähnelt proaktives Verhalten dem erfolgreicher Unternehmer: Stets auf der Suche nach passenden, erfolgversprechenden Betätigungen.



Den negativen Gegenpol bezeichnen wir als Unterlasser oder reaktiv: Diese Menschen warten, bis ihnen jemand eine Aufgabe gibt. Reaktive werden frühestens nach Aufforderung tätig.

Sicherlich kommt proaktives Herangehen vielen Menschen und Rollen zugute. Innerhalb von IT-Projekten ist proaktives Herangehen bei Softwarearchitekten besonders wichtig. Sehen wir uns dazu einige Beispiele an.

### **Verbesserungsmöglichkeiten suchen**

Softwarearchitekten suchen ständig aktiv und an allen ihnen zugänglichen Stellen nach Verbesserungsmöglichkeiten – ohne explizite Aufforderung von außen. Sie schauen dabei deutlich über den Tellerrand ihres eigenen Arbeitsbereichs hinaus.

Konkret übernehmen Softwarearchitekten proaktiv Aufgaben in Anforderungsanalyse und -management, im Build- und Testmanagement sowie im Risikomanagement. Manchmal unternehmen sie Ausflüge in die Chefetagen, um den Managern die technische Lösung zu erklären oder Schwächen im Projektmanagement zu kompensieren. Als verantwortungsbewusster Softwarearchitekt müssen Sie (wiederum selbstständig und aus eigener Initiative) entscheiden, wann solche Ausflüge angemessen und notwendig sind, damit sie von Ihren Mitmenschen nicht als Einmischung empfunden werden. Hier tritt die Schwierigkeit bezüglich der Softskills zum ersten Mal auf. Die erwähnen wir in diesem Buch noch öfter.

### **Annahmen und Voraussetzungen klären**

Gute Softwarearchitekten klären von sich aus jegliche (ansonsten versteckte oder implizite) Annahmen oder Voraussetzungen auf. Entwurf und Implementierung der technischen Lösung sollten auf Tatsachen beruhen, nicht nur auf Vermutungen, Mutmaßungen und Betriebsblindheit.



## WAR STORY

Wir haben Pflichtenhefte und andere Anforderungsdokumente erhalten, in denen jede Menge *implizite* Annahmen *versteckt* waren. Insbesondere die Qualitätsanforderungen blieben oftmals unerwähnt. Architekturentscheidungen auf solcher Treibsandbasis sind gefährlich. Hätten wir uns in diesen Fällen passiv verhalten, wären die Unzulänglichkeiten wahrscheinlich erst im Betrieb aufgefallen. Wir haben stattdessen durch aktives Nachfragen bei verschiedenen Stakeholdern die Anforderungen ergänzt und *implizit* durch *explizit* ersetzt. Nachfragen ist immer besser als raten! (PH+GS)

## Auf andere zugehen

Proaktive Softwarearchitekten suchen von sich aus den regelmäßigen Kontakt zu anderen Stakeholdern im Projekt. Nicht, weil sie gerne grünen Tee trinken, sondern weil sie (richtig, aktiv!) Rückmeldung einholen und geben wollen. Genau das Gegenteil von „Abwarten und Tee trinken“: Initiativ Eindrücke und Meinungen der anderen erfragen, nach Hindernissen, erkannten Problemen oder Risiken suchen.

Gerne dürfen sie auch loben und sich loben lassen. Hierdurch können Softwarearchitekten eine Menge über ihre Lösungsansätze und deren Auswirkung auf die Projektrealität lernen. Gleichzeitig erhalten sie damit die Möglichkeit, ihre eigene Meinung zu Arbeitsergebnissen, Entscheidungen oder sonstigen Dingen im Projekt zu kommunizieren.

## HINWEIS

Je mehr Enthusiasmus Sie für Ihr System oder Projekt an den Tag legen, desto eher und lieber wird man Ihnen zuhören.

Sie sollten als Softwarearchitekt keinesfalls als Nörgler auftreten und jede Kleinigkeit bemäkeln. Rückmeldungen zum umständ-

lichen Bugtracking-Prozess mit Excel können Sie beispielsweise erst einmal für sich behalten, wenn Sie mit Ihren Auftraggebern und dem Team gerade an fundamentalen Architekturentscheidungen arbeiten.

### **Aufgaben selbst bestimmen**

Softwarearchitekten suchen aus eigener Initiative nach dem jeweils effektivsten (d. h. im Sinne der Zielerreichung optimalen) Einsatz der eigenen Zeit: Ob sie gerade Code schreiben, refaktorisieren oder testen sollen, ob sie Schnittstellen definieren oder Anforderungen klären sollen, ob sie Mitarbeiter coachen sollen oder ob die Dokumentation ein Update vertragen kann – das entscheiden sie proaktiv, ohne dass Projektleiter das erst vorgeben müssen.

### **Proaktiv ist die Ausnahme**

Falls Sie glauben, diese aktive Einstellung sei eine Selbstverständlichkeit, dann willkommen in Phantasia: Proaktives Handeln, ja selbst proaktives Denken, erleben wir in unserer Praxis eher als die Ausnahme denn als Regel. Es bedarf nämlich einer gehörigen Portion Mut und Courage, um sich über etablierte Konventionen hinwegzusetzen und sich um Dinge zu kümmern, die einen angeblich nichts angehen, die aber für den Erfolg von Projekten immens wichtig sind. Im schlimmsten Fall kann es passieren, dass Ihre Vorgesetzten Proaktivität als Einmischung verstehen und Ihr Verhalten als vorwitzig oder übertrieben ablehnen.

Wir möchten Sie zumindest verbal bei diesem Mut zur Aktion unterstützen: Langfristig wird sich für Sie aktives Herangehen an andere Projektbeteiligten, aktives Suchen nach Verbesserung und aktives Infragestellen zweifelhafter Konventionen lohnen – in Form höherer Zufriedenheit, besserer Projektergebnisse und dankbarer KollegInnen. Und dafür lohnt sich der Einsatz!

**HINWEIS**

Denken und handeln Sie wie ein Unternehmer. Gehen Sie aktiv auf Ihre Stakeholder zu und fordern benötigte Dinge ein oder geben interessante Dinge bekannt!

Gehen Sie Ihre Aufgaben aktiv an. Warten Sie nicht, bis Sie jemand auf offene Punkte hinweist. Sie selbst als Softwarearchitekt bestimmen, wann welche Aufgaben angemessen erledigt werden sollen!

Manchmal stecken hinter Zögern, Zaudern und Ängsten ihrer Stakeholder auch Erfahrungen, die diese Menschen haben, Sie selbst aber noch nicht.

**HINWEIS**

Seien Sie bereit, von Ihren Stakeholdern zu lernen. Akzeptieren Sie berechtigte Kritik und lehnen unberechtigte höflich, aber bestimmt ab.

**Verwandte Muster**

Der Proaktive nutzt den Blick in den Rückspiegel (6): Als proaktiver, vorausschauender Softwarearchitekt suchen Sie Risiken, *bevor* sie eintreten können. Bereits in frühen Entwurfs- oder Entwicklungsphasen begeben Sie sich auf die Suche nach Verbesserungsmöglichkeiten. Sie bewerten, bevor Sie jemand anders auf Probleme (= eingetretene Risiken) hingewiesen hat!

