Mahbouba Gharbi · Arne Koschel
Andreas Rausch · Gernot Starke

Basiswissen

für Software-architekten

Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture Foundation Level

dpunkt.verlag

iSAQB

4., überarbeitete und aktualisierte Auflage

Mahbouba Gharbi ist Geschäftsführerin und Chef-Architektin bei ITech Progress GmbH und iSAQB-Vorstandsvorsitzende, ist bekennender Softwarearchitektur-Fan, Autorin zahlreicher Fachartikel und häufige Sprecherin auf internationalen Konferenzen.

Prof. Dr. Arne Koschel ist Dozent an der Hochschule Hannover mit dem Schwerpunkt verteilte (Informations-)Systeme. Er hat langjährige industrielle Praxis in Entwicklung und Architektur verteilter Informationssysteme. Nebenberuflich berät und referiert er zu Themen wie SOA, Integration, Middleware, EDA und Cloud Computing. Er ist Active Board Member im iSAQB.

Prof. Dr. Andreas Rausch leitet den Lehrstuhl für Software Systems Engineering an der Technischen Universität Clausthal. Er war und ist in der industriellen Praxis als Berater und leitender Softwarearchitekt bei einer Reihe von großen verteilten Softwaresystemen tätig.

Dr. Gernot Starke, innoQ Fellow, arbeitet als Berater für methodische Softwarearchitektur, Technologiemanagement und Projektorganisation. Seit mehr als 15 Jahren gestaltet er die Architektur von Softwaresystemen unterschiedlicher Größe.

Mahbouba Gharbi · Arne Koschel · Andreas Rausch · Gernot Starke

Basiswissen für Softwarearchitekten

Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture – Foundation Level

4., überarbeitete und aktualisierte Auflage

dpunkt.verlag

Mahbouba Gharbi
m.gharbi@itech-progress.com

Arne Koschel
akoschel@acm.org

Andreas Rausch
andreas.rausch@tu-clausthal.de

Gernot Starke
gs@gernotstarke.de

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

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.

ISBN:
Print 978-3-86490-781-4
PDF 978-3-96910-012-7
ePub 978-3-96910-013-4
mobi 978-3-96910-014-1

4., überarbeitete und aktualisierte Auflage 2020

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

5 4 3 2 1 0

Vorwort zur 4. Auflage

Softwarearchitektur bildet – neben motivierten Teams und gutem Management – einen wichtigen Erfolgsfaktor von Softwareprojekten. Sie stellt im Sinne einer systematischen Konstruktion sicher, dass Qualitätsanforderungen wie beispielsweise Erweiterbarkeit, Flexibilität, Performance oder Time-to-Market erfüllt werden können.

Softwarearchitektinnen und Softwarearchitekten bringen die Kundenwünsche in Einklang mit den technischen Möglichkeiten und Randbedingungen. Sie sorgen für eine passende Struktur und das Zusammenspiel aller Systemkomponenten. Als Teamplayer arbeiten sie eng mit Softwareentwicklerinnen und Softwareentwicklern sowie anderen Projektbeteiligten zusammen.

Unser Buch »Basiswissen für Softwarearchitekten« orientiert sich am Lehrplan zum »Certified Professional for Software Architecture – Foundation Level« (CPSA-F) des International Software Architecture Qualification Board (iSAQB). Der iSAQB e.V. legt als internationales und offenes Gremium Standards für die Ausbildung, Prüfung und Zertifizierung von Softwarearchitekten fest.

Die 4. Auflage unseres Buches bietet eine Aktualisierung auf Basis des neuen CPSA-F-Lehrplans in der Version 5.1 vom Januar 2020. Bei der Überarbeitung des iSAQB-Lehrplans wurden einige Themen auf weitere Ausbildungsstufen verschoben und sind somit nicht mehr Teil des »Foundation Level«-Lehrplans. Diese Inhalte sind zwar weiterhin in unserem Buch zu finden, sie sind jedoch als »Exkurs« hervorgehoben. Wer das Buch nur zur Prüfungsvorbereitung nutzt, der kann diese Exkurse ignorieren. Des Weiteren wurde das Glossar aktualisiert. Die Leserinnen und Leser[1] können sich auch auf neue und verbesserte Prüfungsbeispielfragen freuen, die eine gezieltere Prüfungsvorbereitung ermöglichen.

Mit der Zertifizierung zum CPSA-F weisen Softwarearchitekten einen fundierten Wissens- und Kenntnisstand für die Konstruktion kleiner und mittlerer Systeme nach. Ausgehend von einer hinreichend detailliert beschriebenen Anfor-

1. Im weiteren Verlauf des Buches verwenden wir überwiegend die männliche Form und wollen damit Frauen sowie alle anderen Geschlechter selbstverständlich nicht ausschließen bzw. ausgrenzen.

derungsspezifikation können sie eine angemessene Softwarearchitektur entwerfen und dokumentieren. CPSA-F-Absolventen besitzen damit das Rüstzeug, um problembezogene Entwurfsentscheidungen auf der Basis ihrer vorab erworbenen Praxiserfahrung zu treffen.

Das Selbststudium des vorliegenden Buches ermöglicht die Vorbereitung auf diese Zertifizierungsprüfung – praktische Erfahrung in Entwurf und Entwicklung von Softwaresystemen, das Beherrschen einer höheren Programmiersprache sowie der Grundlagen von UML vorausgesetzt. Darüber hinaus empfehlen wir als Autoren grundsätzlich den Besuch entsprechender Präsenzveranstaltungen, weil der Erfahrungsaustausch mit anderen Experten nicht durch Lektüre zu ersetzen ist.

Wir als Autoren arbeiten, lehren und forschen seit vielen Jahren im Bereich des Software & Systems Engineering sowie zur Konstruktion mittlerer und großer IT-Systeme. Wir hoffen, einen Teil unserer Erfahrungen in diesem Buch für Sie als Leser angemessen aufbereitet zu haben.

Wir wünschen Ihnen viel Spaß beim Lesen sowie viel Erfolg bei Ihrer Schulungsmaßnahme und Prüfung zum CPSA-F.

Mahbouba Gharbi, Arne Koschel, Andreas Rausch, Gernot Starke
Ludwigshafen, Hannover, Clausthal-Zellerfeld, Köln, im Juni 2020

Inhaltsübersicht

Anhang 185

Inhaltsverzeichnis

1 Einleitung

Software ist allgegenwärtig. Dies gilt sowohl für kommerzielle Unternehmenssoftware als auch für nahezu alle anderen Bereiche des beruflichen, öffentlichen und privaten Alltags: Fliegen, Telefonieren, Überweisen, Autofahren – all das wäre ohne Software kaum noch möglich. In jedem Haushalt und in vielen Alltagsgegenständen, von der Waschmaschine bis zum Auto, werden softwaregesteuerte Bestandteile verwendet [BJ+06]. Software steht in der Regel nicht autark für sich, sondern ist in Geräte mit Hardware und Elektronik oder in Geschäftsprozesse, mit denen Unternehmen ihre Wertschöpfung erzielen, eingebettet [TTL00].

Der Nutzen und wirtschaftliche Erfolg von Unternehmen und Produkten wird zunehmend von Software und deren Qualität bestimmt (siehe [BM++96], [SV99], [TTL00]). Als Folge stehen Softwareingenieure und damit die Disziplin Software Engineering vor der Herausforderung, immer komplexere Anforderungen immer schneller und kostengünstiger bei gleichzeitig hoher Softwarequalität umzusetzen.

Die kontinuierliche Steigerung der Größe und Komplexität von softwareintensiven Systemen hat inzwischen dazu geführt, dass sie zu den komplexesten von Menschen geschaffenen künstlichen Systemen überhaupt zählen. Bestes Beispiel ist das Internet: ein auf Software basierendes weltumspannendes System. Inzwischen ist das Internet sogar auf der internationalen Raumstation ISS verfügbar und hat damit die Grenzen der Erde überschritten.

Nur ein strukturiertes und systematisches Herangehen kann dabei gesichert zum Erfolg führen. Trotz Anwendung etablierter Softwareentwicklungsmethoden bleibt die Anzahl der fehlgeschlagenen Softwareprojekte seit Jahren erschreckend hoch. Um dem entgegenzuwirken, versucht man in den frühen Phasen des Software Engineering bereits möglichst viele Fehler zu vermeiden bzw. dort zu identifizieren und auszumerzen. Zu diesen Phasen zählen insbesondere das Requirements Engineering sowie die Softwarearchitektur. Getreu den Worten von Ernst Denert, einem der Väter der methodischen Softwareentwicklung, wollen wir uns hier mit Softwarearchitektur beschäftigen, der »Königsdisziplin des Software Engineering« (zitiert aus dem Geleitwort von Ernst Denert in [Sie04]).

1.1 Softwarearchitektur als Disziplin im Software Engineering

Bereits in den 60er-Jahren wurden die Probleme mit Softwareprojekten unter dem Stichwort Softwarekrise bekannt. Vom 7. bis 11. Oktober 1968 fand im oberbayerischen Garmisch eine kleine Konferenz statt: Das Wissenschaftskomitee der NATO hatte 62 hochrangige Forscher und Praktiker von internationalem Ruf eingeladen, um unter dem Titel »Software Engineering« über die Zukunft der Softwareentwicklung nachzudenken. Heute gilt diese Konferenz als Geburtsstunde des Software Engineering [Dij72].

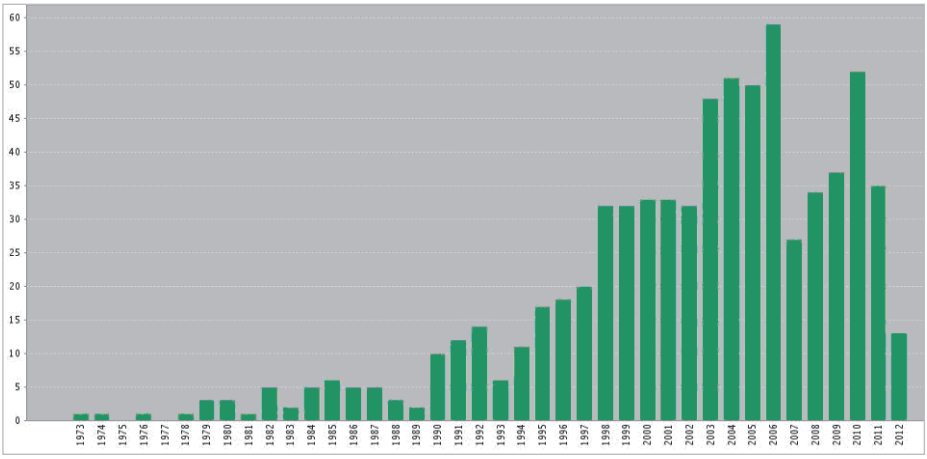

Abb. 1–1 *Veröffentlichungen zu Softwarearchitektur seit 1973 [Reu12]*

Im Vergleich zu traditionellen Ingenieurdisziplinen wie beispielsweise dem Bauwesen, das auf mehrere Tausend Jahre Erfahrung zurückblicken kann, ist Software Engineering mit dem Geburtsjahr 1968 noch sehr jung. So erscheint es auch nicht verwunderlich, dass dessen Teildisziplin Softwarearchitektur noch deutlich jünger ist. Abbildung 1–1 demonstriert dies deutlich: Das Web of Knowledge, eine der großen und renommierten Publikationsdatenbanken, verzeichnet erst ab den 90er-Jahren eine wachsende Anzahl von Publikationen zum Thema Softwarearchitektur [Reu12].

Betrachten wir hingegen die klassische Architektur im Bauwesen, so können wir auf eine bereits Jahrtausende währende Tradition zurückblicken. Ein wichtiger Vordenker war hier Marcus Vitruvius Pollio, ein römischer Architekt aus dem ersten Jahrhundert vor Christus. Er ist Autor des Werkes »De architectura«, das heute unter dem Titel »Ten Books on Architecture« bekannt ist [Vit60]. Vitruvius vertrat die These, dass gute Architektur durch eine kunstvolle Kombination der folgenden Elemente zu erreichen sei:

- **utilitas (Nützlichkeit):**
  Das Gebäude erfüllt seine Funktion.
- **firmitas (Festigkeit):**
  Das Gebäude ist stabil und langlebig.
- **venustas (Schönheit):**
  Das Gebäude ist ästhetisch gestaltet.

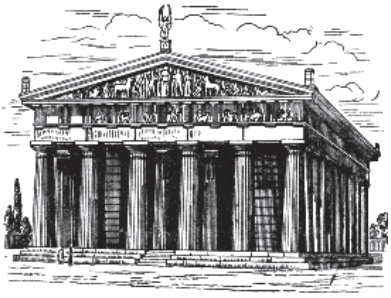

Abb. 1–2 *Architektur im alten Rom*

Diese These lässt sich direkt auf die Disziplin Softwarearchitektur übertragen. Ziel der Softwarearchitektur und damit Aufgabe eines Softwarearchitekten ist es, ein System zu konstruieren, das in einem kunstvoll ausgewogenen Dreiklang die drei folgenden Eigenschaften vereint:

- **utilitas (Nützlichkeit):**
  Die Software erfüllt die funktionalen und nicht funktionalen Anforderungen der Nutzer und Kunden.
- **firmitas (Festigkeit):**
  Die Software ist stabil im Hinblick auf die geforderten Qualitätseigenschaften, z.B. die Anzahl der gleichzeitig zu bedienenden Nutzer, und langlebig, da zukünftige Weiterentwicklungen möglich sind, ohne das System komplett neu bauen zu müssen.
- **venustas (Schönheit):**
  Die Software ist sowohl außen (gegenüber dem Nutzer) wohlstrukturiert, sodass sie intuitiv nutzbar ist, als auch innen (gegenüber demjenigen, der die Software pflegen und weiterentwickeln soll) wohlstrukturiert, sodass dieser die internen Strukturen der Software leicht verstehen und damit gut seinen Aufgaben nachkommen kann.

1.2 iSAQB – International Software Architecture Qualification Board

Softwarearchitektur ist eine sehr junge Disziplin, über deren genauen Umfang und ihre Ausgestaltung in der Informatik trotz vieler Publikationen immer noch viele unterschiedliche Meinungen kursieren. Aufgaben und Verantwortungsbereiche von Softwarearchitekten werden sehr unterschiedlich definiert und in vielen Softwareprojekten ständig neu verhandelt.

Für andere Disziplinen im Software Engineering hingegen, wie z. B. beim Projektmanagement, Requirements Engineering oder Testen, gibt es inzwischen einen deutlich ausgereifteren Wissenskanon. Dafür bieten unabhängige Organisationen Lehrpläne an, die klar beschreiben, welche Kenntnisse und Fähigkeiten eine entsprechende Ausbildung vermitteln soll (Testen: *www.istqb.org*, Requirements Engineering: *www.ireb.de*, Projektmanagement: *www.pmi.org*).

Vor diesem Hintergrund haben Anfang 2008 verschiedene Softwarearchitekturexperten aus Wirtschaft und Wissenschaft das »International Software Architecture Qualification Board« als eingetragenen Verein (iSAQB e.V., *www.isaqb.org*) gegründet. Dessen Ziel ist es, Standards für die Ausbildung und Zertifizierung von Softwarearchitekten zu definieren. Bewusst wird im iSAQB jegliche Hersteller- oder Produktorientierung vermieden. Zertifizierungen auf den unterschiedlichen Stufen Foundation Level, Advanced Level und Expert Level ermöglichen es Softwarearchitekten, sich den Stand ihrer Kenntnisse und Fähigkeiten durch ein anerkanntes Verfahren bescheinigen zu lassen (siehe Abb. 1–3).

Abb. 1–3 *iSAQB-Zertifizierungsstufen (www.isaqb.org)*

Von diesem standardisierten Lehr- und Ausbildungsplan profitieren sowohl etablierte als auch angehende Softwarearchitekten und ebenso Unternehmen oder auch entsprechende Aus- und Weiterbildungseinrichtungen, da er die eingangs geschilderte begriffliche Unsicherheit beseitigt. Nur auf Basis von präzisen Lehr- und Ausbildungsplänen kann eine Prüfung und Zertifizierung angehender Softwarearchitekten stattfinden und so letztlich ein qualitätsgesicherter Ausbildungsstand von Softwarearchitekten mit einem entsprechend akzeptierten Wissenskanon etabliert werden.

Die Zertifizierung zum **Certified Professional for Software Architecture** (CPSA) wird von unabhängigen Zertifizierungsstellen durchgeführt. Basis für die Zertifizierung zum CPSA (Foundation Level) ist ein anspruchsvoller, vom iSAQB in Einklang mit dem Lehrplan entwickelter, nicht öffentlicher Fragenkatalog, aus dem eine Teilmenge als Prüfungsfragen ausgewählt wird. Für die Zertifizierung zum Advanced Level werden neben der Erfordernis des Besuches von lizenzierten Schulungen bzw. der Anerkennung eines anderen, nicht durch den iSAQB definierten Zertifikats praktische Aufgaben gestellt. Der Expert Level befindet sich derzeit noch in Entwicklung.

Auf Basis dieses Lehrplans bieten verschiedene lizenzierte Schulungsveranstalter mehrtägige Kurse an, die Wissen in diesen Themengebieten auffrischen und vielfach deutlich vertiefen. Die Teilnahme an einem Kurs wird zwar nachdrücklich empfohlen, ist jedoch nicht Bedingung für die Prüfungsanmeldung zur Zertifizierung.

1.3 Certified Professional for Software Architecture – Foundation und Advanced Level

Der iSAQB hat inzwischen nicht nur die Zertifizierungsrichtlinien für den CPSA Foundation Level, sondern auch für den Advanced Level definiert.

Der Advanced Level ist modular aufgebaut und besteht aus einzelnen Schulungen, die sich jeweils einem bestimmten Schwerpunkt der Kompetenz eines IT-Professionals widmen:

- **Methodische Kompetenz:**
  Wissen und Fähigkeiten im Bereich des systematischen Vorgehens bei IT-Projekten, unabhängig von Technologien
- **Technische Kompetenz:**
  Wissen und Fähigkeiten im Bereich des Einsatzes von Technologien zur Lösung von Entwurfsaufgaben
- **Kommunikative Kompetenz:**
  Wissen und Fähigkeiten im Bereich der Kommunikation, Präsentation, Rhetorik und Moderation zur effektiven Wahrnehmung der Rolle im Softwareentwicklungsprozess

Voraussetzungen für den Advanced Level sind:

- Ausbildung und Zertifizierung zum CPSA-F (Foundation Level)
- Mindestens 3 Jahre Berufserfahrung in der IT-Branche
- Mitarbeit an Entwurf und Entwicklung von mindestens zwei verschiedenen IT-Systemen
- Für die Prüfung: mindestens 70 Credit Points aus allen drei Kompetenzbereichen (jeweils mindestens 10 Credit Points)

Die Prüfung besteht aus der Bearbeitung einer Prüfungsaufgabe in Eigenregie und der anschließenden Besprechung der Lösung mit zwei unabhängigen Prüfern in einem Interview.

Für den Foundation Level wurden die Bereiche, in denen ein Softwarearchitekt über fundiertes Wissen und Fähigkeiten verfügen sollte, im Rahmen eines öffentlich zugänglichen Lehrplans beschrieben [isaqb-lehrplan]. Danach soll angehenden Softwarearchitekten folgendes Spektrum an Inhalten vermittelt werden:

- der Begriff und die Bedeutung von Softwarearchitektur,
- die Aufgaben und Verantwortungsbereiche von Softwarearchitekten,
- die Rolle des Softwarearchitekten in Projekten,
- State-of-the-Art-Methoden und -Techniken zur Entwicklung von Softwarearchitekturen.

Im Mittelpunkt steht der Erwerb folgender Fähigkeiten:

- mit anderen Projektbeteiligten aus den Bereichen Anforderungsmanagement, Projektmanagement, Test und Entwicklung wesentliche Softwarearchitekturentscheidungen abzustimmen,
- Softwarearchitekturen auf Basis von Sichten, Architekturmustern und technischen Konzepten zu dokumentieren und kommunizieren,
- die wesentlichen Schritte beim Entwurf von Softwarearchitekturen zu verstehen und für kleine und mittlere Systeme selbstständig durchzuführen.

Die Schulung zum Foundation Level vermittelt das notwendige Wissen, um für kleine und mittlere Systeme ausgehend von einer hinreichend detailliert beschriebenen Anforderungsspezifikation eine dem Problem angemessene Softwarearchitektur zu entwerfen und zu dokumentieren. Diese kann dann als Implementierungsgrundlage bzw. -vorlage genutzt werden. Teilnehmer erhalten das Rüstzeug, um problembezogene Entwurfsentscheidungen auf der Basis ihrer vorab erworbenen Praxiserfahrung zu treffen.

Abbildung 1–4 zeigt die inhaltliche Struktur und die Gewichtung der einzelnen Bereiche des Lehrplans für den iSAQB Certified Professional for Software Architecture (CPSA), Foundation Level.