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 [TT+00].

Der Nutzen und wirtschaftliche Erfolg von Unternehmen und Produkten wird zunehmend von Software und deren Qualität bestimmt (siehe [BM00], [SV99], [TT+00]). 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].

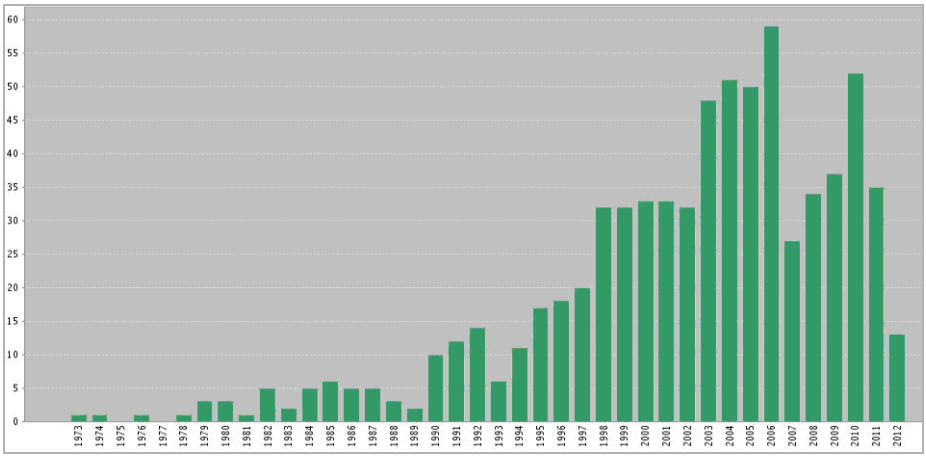

Abb. 1–1 *Veröffentlichungen zu Softwarearchitektur seit 1973 [Reu12]*

Im Vergleich zu traditionellen Ingenieursdisziplinen 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 nichtfunktionalen 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).

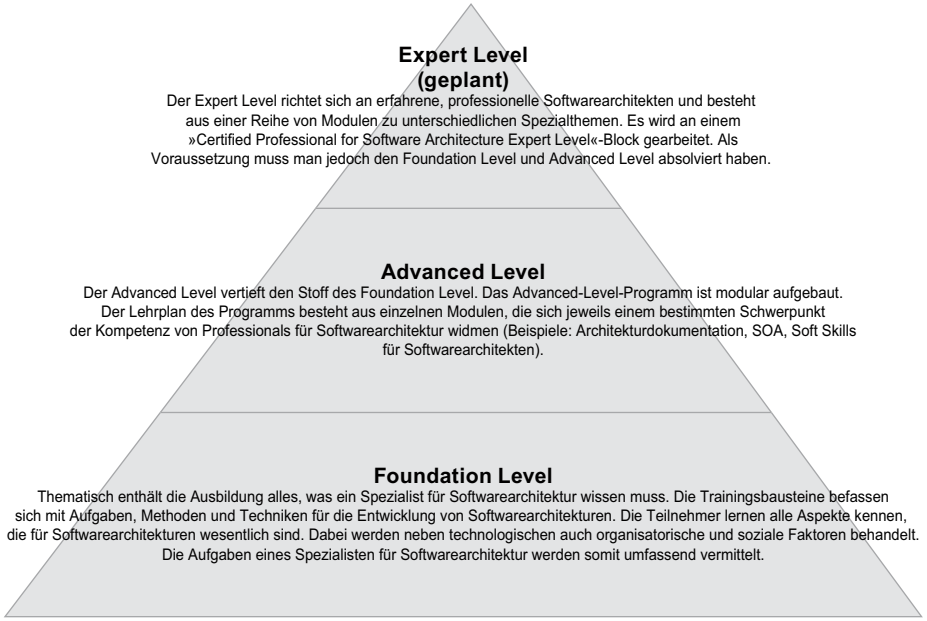

Abb. 1–3 *iSAQB-Zertifizierungsstufen (www.isaqb.org)*

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. Für den Expert Level ist zusätzlich eine mündliche Prüfung vorgesehen (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.

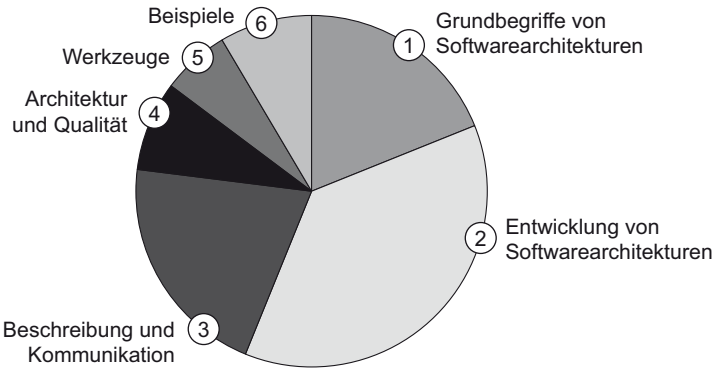

Abb. 1–4 *Struktur des iSAQB-Lehrplans für CPSA, Foundation Level*

Sie haben die Möglichkeit, sich bei verschiedenen unabhängigen Anbietern durch eine Prüfung gemäß dem iSAQB-Lehrplan zertifizieren zu lassen. Für die Zertifizierung setzen die Prüfungsanbieter standardisierte Prüfungsfragen ein, die der iSAQB erarbeitet hat.

Für die Prüfungen wird ein Multiple-Choice-Verfahren eingesetzt. Entsprechend objektiv ist das Prüfungsergebnis messbar.

Mit der Prüfung können Sie somit Ihr notwendiges Grundlagenwissen als Softwarearchitekt nachweisen. Natürlich müssen Sie dann später in der Anwendung zeigen, dass Sie Ihr Wissen auch praktisch und erfolgreich in konkreten Architekturen einzusetzen wissen.

1.4 Zielsetzung des Buches

Wir, die Autoren des Buches, haben gemeinsam mit anderen iSAQB-Mitgliedern am iSAQB-Lehrplan für den Certified Professional for Software Architecture, Foundation Level, gearbeitet. Im Rahmen dieser Zusammenarbeit ist auch die Idee zu diesem Buch entstanden. Dementsprechend verfolgen wir darin die zentrale Zielsetzung, kompakt und prägnant das notwendige Wissen für die CPSA-Prüfung, Foundation Level, und somit das Fundament für den Wissenskanon in der Disziplin Softwarearchitektur bereitzustellen. Das Buch ist somit die ideale Referenz für eine entsprechende Prüfungsvorbereitung. Sinnvollerweise empfehlen wir Ihnen ergänzend den Besuch entsprechender Schulungen, da dort das Lehrmaterial durch über dieses Buch hinausgehende praktische Beispiele von Softwarearchitekturen und persönliche Erfahrungen der jeweils Lehrenden ergänzt wird.

Da der iSAQB und somit auch das Buch primär auf methodische Fähigkeiten und Wissen fokussiert, gehören konkrete Implementierungstechnologien oder spezielle Werkzeuge explizit nicht zum standardisierten Lehrinhalt. Deshalb haben wir dieses Buch bewusst technologieneutral verfasst. Auch die von uns ver-

wendeten Notationen, wie z.B. die UML, sind nur exemplarisch zu verstehen. Ebenso ist es nicht Ziel des Buches, ein einzelnes konkretes Vorgehensmodell oder einen spezifischen Entwicklungsprozess darzustellen. Vielmehr werden von uns an vielen Stellen mehrere Beispiele z.B. für Notationen oder Vorgehensmodelle kurz vorgestellt.

In diesem Buch erklären wir vor allem wichtige Begriffe und Konzepte der Softwarearchitektur und stellen deren Bezug zu anderen Disziplinen dar. Darauf aufbauend führen wir die grundlegenden Techniken und Methoden für den Entwurf und die Entwicklung, die Beschreibung und Kommunikation sowie die Qualitätssicherung von Softwarearchitekturen ein. Schließlich betrachten wir die Rolle, Aufgaben, das Umfeld und die Arbeitsumgebung von Softwarearchitekten und deren Einbettung in die umfassende Organisations- und Projektstruktur.

1.5 Voraussetzungen

Entsprechend der oben genannten Zielsetzung setzt das vorliegende Buch – wie auch der iSAQB-Lehrplan – Erfahrung in der Softwareentwicklung voraus. Insbesondere gehören folgende Inhalte nicht zum Lehrplan und sind damit auch nicht Thema des Buches, obgleich sie zu den notwendigen Kompetenzen von Softwarearchitekten zählen:

- typischerweise mehrjährige praktische Erfahrung in der Softwareentwicklung, erworben durch Programmierung unterschiedlicher Projekte oder Systeme,
- vertiefte Kenntnisse und praktische Erfahrung mit mindestens einer höheren Programmiersprache,
- Grundlagen der Modellierung und Abstraktion sowie von UML, insbesondere der Klassen-, Paket-, Komponenten- und Sequenzdiagramme sowie deren Bezug zum Quellcode,
- praktische Erfahrung in technischer Dokumentation, insbesondere in der Dokumentation von Quellcode, Systementwürfen oder technischen Konzepten.

Darüber hinaus sind Kenntnisse und Erfahrung mit der Objektorientierung für das Verständnis einiger Konzepte hilfreich. Ebenfalls wünschenswert ist Erfahrung in der Konzeption und Implementierung verteilt ablaufender Anwendungen wie etwa Client-Server-Systeme oder Webanwendungen.

1.6 Leitfaden für den Leser

Der Aufbau dieses Buches orientiert sich vor allem an der Struktur und den Inhalten des iSAQB-Lehrplans Foundation Level nach Abbildung 1–4 bzw. [isaqb-lehrplan]:

- In Kapitel 2 beschreiben wir grundlegende Begriffe und Inhalte des Themenbereichs Softwarearchitektur, die in den folgenden Kapiteln aufgegriffen und vertieft werden. Beispielsweise wird dort der Begriff der »Sicht« auf ein Softwaresystem im Rahmen einer Softwarearchitektur eingeführt.
- Aspekte des praktischen Entwurfs von Softwarearchitekturen behandeln wir in Kapitel 3. Themen sind dort Varianten des Vorgehens bei der Architekturentwicklung, wichtige Architekturmuster wie Schichten, Pipes and Filters, Model-View-Controller, Entwurfsprinzipien wie Kopplung, Kohäsion, Trennung von Verantwortlichkeiten u.a.m.
- Inhalt des Kapitels 4 sind ausgewählte, praxisnahe Beschreibungsmittel sowie in der Praxis bewährte Richtlinien, die es Ihnen erlauben, Ihre Softwarearchitektur zu dokumentieren und zielgruppenorientiert anderen zu vermitteln. Das iSAQB-Sichtenmodell, Querschnittsaspekte in Softwarearchitekturen sowie praktisch bewährte Richtlinien für die Softwarearchitekturdokumentation sind Beispiele für den Inhalt des Kapitels.
- In Kapitel 5 werfen wir einen ersten Blick auf den Zusammenhang von Softwarearchitektur und Qualitätsfragestellungen. Wichtige Begriffe dieses Abschnitts sind u.a. bezogen auf Software: Qualität, Qualitätsmerkmale, ATAM (Architecture Tradeoff Analysis Method), Qualitätsbaum, Kompromisse (bei der Umsetzung von Qualitätsmerkmalen), qualitative Architekturbewertung und Risiken bzgl. der Erreichung von Qualitätsmerkmalen.
- Abschließend zeigen wir in Kapitel 6 eine Reihe von Beispielen für Unterstützungswerkzeuge des Softwarearchitekten wie z.B. solche zur Modellierung, Generierung oder Dokumentation.
- Einige beispielhafte Übungsfragen, ein Glossar sowie ein Quellenverzeichnis runden das Buch ab.

Speziell zur iSAQB-Prüfungsvorbereitung sollten Sie besonders die Kapitel 2–5 gründlich durcharbeiten und ergänzende Blicke in die anderen Abschnitte werfen. Ansonsten empfiehlt es sich, zumindest das Kapitel 2 komplett zu lesen und dann die Sie besonders interessierenden Themen zu vertiefen.

1.7 Zielpublikum

Als Zielpublikum dieses Buches sehen wir in erster Linie Zertifizierungsinteressierte, die es – ggf. neben Schulungen – als Vorbereitung zur Prüfung einsetzen wollen. Hinzu kommen Praktiker und Studierende, die die praktischen Grundbegriffe von Softwarearchitekturen kennenlernen wollen.

Interessant ist dieses Buch auch für Softwareprojektmanager, Softwareproduktmanager sowie Entscheider auf der mittleren Softwareentwicklungsebene als Einstiegsüberblick zum Thema Softwarearchitektur.

1.8 Danksagungen

Wir möchten uns an dieser Stelle beim iSAQB-Verein für seine unterstützende Mitwirkung bedanken, ganz besonders bei unseren iSAQB-Reviewern Stefan Zörner, Andreas Rothmann und Phillip Ghadir für ihre Arbeit. Frau Ingrid Schindler vom Lehrstuhl für Software Systems Engineering der Technischen Universität Clausthal und Mitarbeiter der ITech Progress haben uns bei der Erstellung der Abbildungen sehr unterstützt. Die Erstellung der 2. Auflage wäre ohne die unermüdliche Unterstützung von Christine O'Brien von ITech Progress nicht möglich gewesen.

Unserer Betreuerin seitens des dpunkt.verlags, Frau Christa Preisendanz, danken wir für ihre Geduld.

Zu guter Letzt möchten wir ganz besonders unseren Familien und Partnern danken, die an zahllosen Tagen die Zeit und Geduld aufgebracht haben, uns gemeinsam an diesem Buch arbeiten zu lassen.