1 Einleitung

Software ist allgegenwärtig. Nahezu jedes komplexere Produkt ist heute softwaregesteuert und auch viele Dienstleistungen stützen sich auf Softwaresysteme. Software und Softwarequalität sind daher ein entscheidender Wettbewerbsfaktor. Ein Unternehmen, das neue oder bessere Software in kürzerer Zeit in sein Produkt integrieren bzw. auf den Markt bringen kann (Time-to-Market), ist seinen Mitbewerbern überlegen.

Agile Entwicklungsmodelle versprechen eine schnellere »Time-to-Market« bei gleichzeitig besserer Ausrichtung an den Kundenanforderungen und nicht zuletzt bessere Softwarequalität. So erstaunt es nicht, dass heute in Unternehmen zunehmend agile Methoden eingesetzt werden – auch in großen, internationalen Projekten und in Produktentwicklungseinheiten großer Konzerne, quer durch alle Branchen. In den meisten Fällen bedeutet dies den Umstieg von der Entwicklung nach V-Modell auf die agile Entwicklung mit Scrum[1].

Die Umstellung auf »agil« und das nachhaltige produktive agile Arbeiten sind jedoch nicht ganz einfach. Insbesondere dann, wenn mehr als nur ein Team davon betroffen ist. Jedes Teammitglied, das Projektmanagement, aber auch das Management in der Linienorganisation muss teils gravierende Änderungen gewohnter Abläufe und Arbeitsweisen vollziehen. Dabei sind Softwaretest und Softwarequalitätssicherung ganz entscheidend daran beteiligt, ob ein Team, eine Softwareabteilung oder ein ganzes Unternehmen agiles Entwickeln langfristig erfolgreich beherrscht und so die erhofften Vorteile nachhaltig realisieren kann.

Zu den populären agilen Entwicklungsmethoden gibt es eine Fülle auch deutschsprachiger Literatur. Einige empfehlenswerte Einführungen, z.B. zu Scrum, finden sich im Literaturverzeichnis dieses Buches. In der Regel wird das Thema »agile Softwareentwicklung« in diesen

1. Umfrage [URL: Testpraxis] und Studie [URL: StatusQuoAgile].

Büchern aus Sicht des Entwicklers und Programmierers betrachtet. Demgemäß stehen agile Programmiertechniken und agiles Projektmanagement im Vordergrund. Wenn das Thema Testen erwähnt wird, geht es meistens um Unit Test und zugehörige Unit-Test-Werkzeuge, also im Wesentlichen um den Entwicklertest. Tatsächlich kommt dem Testen in der agilen Entwicklung aber eine sehr große und erfolgskritische Bedeutung zu und Unit Tests alleine sind nicht ausreichend.

Dieses Buch möchte diese Lücke schließen, indem es agile Softwareentwicklung aus der Perspektive des Testens und des Softwarequalitätsmanagements betrachtet und aufzeigt, wie »agiles Testen« funktioniert, wo »traditionelle« Testtechniken auch im agilen Umfeld weiter benötigt werden und wie diese in das agile Vorgehen eingebettet werden.

1.1 Zielgruppen

Verstehen, wie Testen in agilen Projekten funktioniert.

Das Buch richtet sich zum einen an Leser, die in das Thema agile Entwicklung erst einsteigen, weil sie künftig in einem agilen Projekt arbeiten werden oder weil sie Scrum in ihrem Projekt oder Team einführen wollen oder gerade eingeführt haben:

- Entwicklungsleiter, Projektmanager, Testmanager und Qualitätsmanager erhalten Hinweise und Tipps, wie Qualitätssicherung und Testen ihren Beitrag dazu leisten können, das Potenzial agiler Vorgehensweisen voll zu entfalten.
- Professionelle (Certified) Tester und Experten für Softwarequalität erfahren, wie sie in agilen Teams erfolgreich mitarbeiten und ihre spezielle Expertise optimal einbringen können. Sie lernen auch, wo sie ihre aus klassischen Projekten gewohnte Arbeitsweise umstellen oder anpassen müssen.

Wissen über (automatisiertes) Testen und agiles Qualitätsmanagement erweitern

Ebenso angesprochen werden aber auch Leser, die bereits in agilen Teams arbeiten und eigene »agile« Erfahrungen sammeln konnten und die ihr Wissen über Testen und Qualitätssicherung erweitern wollen, um die Produktivität und Entwicklungsqualität in ihrem Team weiter zu erhöhen:

- Product Owner, Scrum Master, Qualitätsverantwortliche und Mitarbeiter mit Führungsfunktion erfahren in kompakter Form, wie systematisches, hoch automatisiertes Testen funktioniert und welchen Beitrag Softwaretester im agilen Team leisten können, um kontinuierlich, zuverlässig und umfassend Feedback über die Qualität der entwickelten Software zu liefern.

- Programmierer, Tester und andere Mitglieder eines agilen Teams erfahren, wie sie hoch automatisiertes Testen realisieren können, und zwar nicht nur im Unit Test, sondern auch im Integrations- und im Systemtest.

Das Buch beinhaltet viele praxisorientierte Beispiele und Übungsfragen, sodass es auch als Lehrbuch und zum Selbststudium geeignet ist.

1.2 Zum Inhalt

Kapitel 2

Kapitel 2 gibt einen knappen, vergleichenden Überblick über die derzeit populärsten agilen Vorgehensmodelle Scrum und Kanban. Leser mit Managementaufgaben, die ihr Projekt oder ihre Unternehmenseinheit auf »agil« umstellen wollen, erhalten hier eine Zusammenfassung der wichtigsten agilen Managementinstrumente. Dem gegenübergestellt wird das Vorgehen in Projekten, die sich an klassischen Vorgehensmodellen orientieren. Dies vermittelt einen Eindruck über die Veränderungen, die mit der Einführung agiler Ansätze einhergehen. Leser, die Scrum und Kanban schon kennen, können dieses Kapitel überspringen.

Kapitel 3

Kapitel 3 zeigt auf, welche leichtgewichtigen Planungs- und Steuerungsinstrumente in Scrum anstelle des klassischen Projektplans zum Einsatz kommen. Denn »agil« zu arbeiten, bedeutet keineswegs »planlos« zu arbeiten. Auch Kapitel 3 richtet sich primär an Leser, die auf agile Entwicklung umsteigen. Die Erläuterungen und Hinweise, welchen Beitrag die jeweiligen Planungsinstrumente zur »konstruktiven Qualitätssicherung« und damit zur Fehlervermeidung liefern, sind jedoch auch für Leser mit agiler Projekterfahrung wertvoll.

Kapitel 4

Kapitel 4 behandelt das Thema Unit Tests und »Test First«. Es erklärt, was Unit Tests leisten und wie Unit Tests automatisiert werden. Systemtester, Fachtester oder Projektbeteiligte ohne oder mit geringer Erfahrung im Unit Test finden hier Grundlagen über die Techniken und Werkzeuge im entwicklungsnahen Test, die ihnen helfen, enger mit Programmierern und Unit-Testern zusammenzuarbeiten. Programmierer und Tester mit Erfahrung im Unit Test erhalten hilfreiche Tipps, um ihre Unit Tests zu verbessern. Ausgehend von diesen Grundlagen wird Test First (testgetriebene Entwicklung) vorgestellt und die hohe Bedeutung dieser Praktik für agile Projekte erklärt.

Kapitel 5

Kapitel 5 erklärt Integrationstests und »Continuous Integration«. Auch Programmierer, die ihren Code intensiv mit Unit Tests prüfen, vernachlässigen dabei oft Testfälle, die Integrationsaspekte überprüfen. Daher werden in diesem Kapitel zunächst wichtige Grundlagen

über Softwareintegration und Integrationstests vermittelt. Anschließend wird die Continuous-Integration-Technik vorgestellt und erläutert, wie ein Continuous-Integration-Prozess im Projekt eingeführt und angewendet wird.

Kapitel 6 Kapitel 6 erörtert Systemtests und »Test nonstop«. Aufbauend auf den Grundlagen über Systemtests werden wichtige Techniken für manuelle und automatisierte System- und Akzeptanztests im agilen Umfeld erläutert. Anschließend wird aufgezeigt, wie auch Systemtests effizient automatisiert und in den Continuous-Integration-Prozess des Teams eingebunden werden können. Kapitel 6 ist dabei nicht nur für Systemtester und Fachtester gedacht, sondern auch für Programmierer, die besser verstehen wollen, welche Testaufgaben jenseits des entwicklungsnahen Tests im agilen Team gemeinsam zu bewältigen sind.

Kapitel 7 Kapitel 7 stellt klassisches und agiles Verständnis von Qualitätsmanagement und Qualitätssicherung gegenüber und erläutert die in Scrum »eingebauten« Praktiken zur vorbeugenden, konstruktiven Qualitätssicherung. Der Leser erhält Hinweise und Tipps, wie Qualitätsmanagement »agiler« realisiert werden kann und wie Mitarbeiter aus Qualitätssicherungsabteilungen, Qualitätsmanagementbeauftragte und andere Qualitätssicherungsspezialisten ihr Know-how in agilen Projekten einbringen und so einen wertvollen Beitrag für ein agiles Team leisten können.

Kapitel 8 Kapitel 8 präsentiert mehrere Fallstudien aus Industrie, Onlinehandel und Unternehmen der Softwarebranche. Diese spiegeln Erfahrungen und Lessons Learned wider, die die Interviewpartner bei der Einführung und Anwendung agiler Vorgehensweisen in ihrem jeweiligen Unternehmen gesammelt haben.

Kapitelstruktur Die Kapitel 2, 3, 7 und 8 erörtern Prozess- und Managementthemen und sprechen demgemäß den an Managementaspekten interessierten Leser an. Die Kapitel 4, 5 und 6 erörtern das (automatisierte) »agile Testen« auf den verschiedenen Teststufen und sprechen den (auch) technisch interessierten Leser an. Dabei werden die Ziele und Unterschiede von Unit Tests, Integrations- und Systemtests ausführlich angesprochen. Denn wie bereits erwähnt, wird Testen leider in vielen agilen Projekten oft mit Unit Tests gleichgesetzt. Abbildung 1–1 illustriert die Kapitelstruktur nochmals:

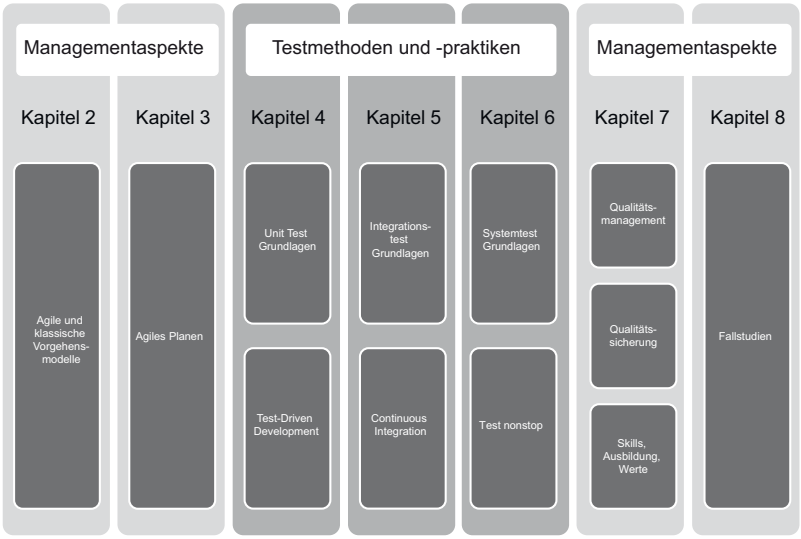

Abb. 1–1
Kapitelstruktur

Cross-Referenz zum ISTQB®-Syllabus

Das Buch deckt den Stoff des *ISTQB® Certified Tester – Foundation Level Extension Syllabus Agile Tester* (Version 2014) ab. Die Gliederung des Buches folgt jedoch nicht der Gliederung des Lehrplans. Auch werden viele Aspekte, die beim Testen in agilen Projekten eine Rolle spielen, im Buch über den ISTQB®-Lehrplan hinausgehend oder zusätzlich behandelt.

Die folgende Cross-Referenz-Tabelle erleichtert es, gezielt den ISTQB®-Lehrstoff anhand der im Lehrplan genannten Lernziele nachlesen zu können:

Tab. 1–1
Cross-Referenz zum ISTQB®-Syllabus

Lernziele nach Certified Tester – Foundation Level Extension Syllabus, 2014	**Kapitel und Abschnitte in diesem Buch**
1. Agile Softwareentwicklung	
1.1 Die Grundlagen der agilen Softwareentwicklung	
FA-1.1.1 Das Grundkonzept der agilen Softwareentwicklung basierend auf dem Agilen Manifest beschreiben können.	2
FA-1.1.2 Die Vorteile des Whole-Team Approach (interdisziplinäres, selbstorganisiertes Team) verstehen.	2.1, 2.4
FA-1.1.3 Den Nutzen von frühen und häufigen Rückmeldungen verstehen.	3.2, 3.4 4.2, 4.4, 4.5 5.5 6.1, 6.2, 6.4, 6.6, 6.8 7.1, 7.2, 7.2.3, 7.5, 7.5.1, 7.6, 7.6.1, 7.7

→

Lernziele nach Certified Tester – Foundation Level Extension Syllabus, 2014	Kapitel und Abschnitte in diesem Buch
1. Agile Softwareentwicklung	
1.2 Aspekte agiler Ansätze	
FA-1.2.1 Die Ansätze der agilen Softwareentwicklung nennen können.	2
FA-1.2.2 User Stories schreiben in Zusammenarbeit mit Entwicklern und Vertretern des Fachbereichs.	3.3 6.1, 6.3.1, 6.3.3, 6.4, 6.8.2 7.6.2
FA-1.2.3 Verstehen, wie in agilen Projekten Retrospektiven als Mechanismus zur Prozessverbesserung genutzt werden.	2.4 3.6 4.5 5.5 6.2 7.2.3, 7.5.1
FA-1.2.4 Die Anwendung und den Zweck von Continuous Integration (der kontinuierlichen Integration) verstehen.	5, 5.5 6.9
FA-1.2.5 Die Unterschiede zwischen Iterations- und Releaseplanung kennen und wissen, wie sich ein Tester gewinnbringend in jede dieser Aktivitäten einbringt.	3.3, 3.4, 3.5, 3.8 6.9
2. Grundlegende Prinzipien, Praktiken und Prozesse des agilen Testens	
2.1 Die Unterschiede zwischen traditionellen und agilen Ansätzen im Test	
FA-2.1.1 Die Unterschiede der Testaktivitäten zwischen agilen und nicht agilen Projekten benennen und erläutern können.	2, 2.4 3.5, 3.7, 3.7.4 4.5 5.6 6.10
FA-2.1.2 Beschreiben können, wie Entwicklungs- und Testaktivitäten in einem agilen Projekt umgesetzt werden.	4 5 6
FA-2.1.3 Die Bedeutung von unabhängigem Test in agilen Projekten darlegen können.	6.8, 6.8.1 7.6.1, 7.7
2.2 Der Status des Testens in agilen Projekten	
FA-2.2.1 Erläutern können, welches Mindestmaß an Arbeitsergebnissen sinnvoll ist, um den Testfortschritt und die Produktqualität in agilen Projekten sichtbar zu machen.	3.7, 3.7.1, 3.7.2 4.5 5.6 6.10 7.5.1

→

Lernziele nach Certified Tester – Foundation Level Extension Syllabus, 2014	Kapitel und Abschnitte in diesem Buch
2. Grundlegende Prinzipien, Praktiken und Prozesse des agilen Testens	
2.2 Der Status des Testens in agilen Projekten	
FA-2.2.2 Damit vertraut sein, dass sich die Tests über mehrere Iterationen hinweg kontinuierlich weiter entwickeln und daher auch erklären können, warum zum Beherrschen der Risiken im Regressionstest Testautomatisierung wichtig ist.	4.5 5.6 6.2, 6.8, 6.10
2.3 Rolle und Fähigkeiten eines Testers in einem agilen Team	
FA-2.3.1 Verstehen, über welche Fähigkeiten (bzgl. Menschen, Domainwissen und Testen) ein Tester in agilen Teams verfügen muss.	7.7
FA-2.3.2 Wissen, was die Rolle eines Testers in einem agilen Team ist.	2, 2.1 3.7, 3.7.2, 3.7.4 4.2.2 4.5 5.6 6.10 7.5, 7.6, 7.7
3. Methoden, Techniken und Werkzeuge des agilen Testens	
3.1 Agile Testmethoden	
FA-3.1.1 Die Konzepte der testgetriebenen Entwicklung, der abnahmetestgetriebenen Entwicklung und der verhaltensgetriebenen Entwicklung nennen können.	3.3 4.2 5.6 6.1, 6.3.3, 6.5, 6.6, 6.7
FA-3.1.2 Die Konzepte der Testpyramide nennen können.	3.7.3
FA-3.1.3 Die Testquadranten und ihre Beziehungen zu Teststufen und Testarten zusammenfassen.	3.7.3
FA-3.1.4 Für ein vorgegebenes agiles Projekt die Rolle eines Testers in einem Scrum-Team übernehmen.	2, 2.1 3.7, 3.7.2, 3.7.4 4.2.2, 4.5 5.6 6.10 7.5, 7.6, 7.7
3.2 Qualitätsrisiken beurteilen und Testaufwände schätzen	
FA-3.2.1 Qualitätsrisiken in einem agilen Projekt einschätzen.	3.7, 3.7.2 4.1.2, 4.1.4, 4.5 5.5.3 6.2, 6.3.3, 6.8 7.3.1, 7.6.1

→

Lernziele nach Certified Tester – Foundation Level Extension Syllabus, 2014	Kapitel und Abschnitte in diesem Buch
3. Methoden, Techniken und Werkzeuge des agilen Testens	
3.2 Qualitätsrisiken beurteilen und Testaufwände schätzen	
FA-3.2.2 Testaufwand auf Basis des Iterationsinhalts und der Qualitätsrisiken schätzen.	■ 3.7 ■ 4.1.2 ■ 5.2, 5.2.2 ■ 6.2, 6.8, 6.10
3.3 Techniken in agilen Projekten	
FA-3.3.1 Die relevanten Informationen interpretieren können, um Testaktivitäten zu unterstützen.	■ 3.1, 3.2, 3.3 ■ 6.6 ■ 7.3.1
FA-3.3.2 Den Fachbereichsvertretern erklären können, wie testbare Abnahmekriterien zu definieren sind.	■ 3.3 ■ 6.1, 6.3, 6.3.3, 6.4.2, 6.6, 6.7
FA-3.3.3 Für eine vorgegebene User Story abnahmetest-getriebene Testfälle (ATDD) schreiben können.	■ 3.3 ■ 6.3, 6.3.3, 6.4.2, 6.4.3
FA-3.3.4 Auf Basis von vorgegebenen User Stories mithilfe von Blackbox-Testentwurfsverfahren funktionale und nicht funktionale Testfälle schreiben können.	■ 4, 4.1, 4.2 ■ 5, 5.1, 5.2 ■ 6, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7
FA-3.3.5 Explorative Tests durchführen können, um das Testen eines agilen Projekts zu unterstützen.	■ 6.3, 6.3.1, 6.3.2
3.4 Werkzeuge in agilen Projekten	
FA-3.4.1 Verschiedene für Tester verfügbare Werkzeuge gemäß ihrem Zweck und den Aktivitäten in agilen Projekten kennen.	■ 3.7.2, 3.7.4 ■ 4.3, 4.4, 4.5 ■ 5.3.1, 5.5, 5.6 ■ 6.4, 6.5, 6.7

Fallbeispiel, Checkfragen und Übungen

Viele, wenn nicht die meisten agilen Ideen, Techniken und Praktiken sind, wenn man den Ausführungen in der entsprechenden Literatur folgt, einfach nachzuvollziehen. Auch die Ideen, Hinweise und Tipps der folgenden Kapitel werden dem Leser vielleicht zunächst einfach erscheinen. Die Knackpunkte stellen sich erst in der Praxis bei der Umsetzung heraus. Das Buch geht auf diese Hürden ein, und damit der Leser praktisch nachvollziehen und »erleben« kann, wo die Herausforderungen stecken, sind folgende Elemente im Text zu finden:

- Ein durchgängiges Fallbeispiel, anhand dessen die jeweils vorgestellten Methoden und Techniken veranschaulicht werden.
- Checkfragen und Übungen, anhand derer der Leser am Ende eines Kapitels den besprochenen Stoff rekapitulieren kann, aber auch

seine Situation und sein Agieren im eigenen Projekt kritisch hinterfragen kann.

1.3 Fallbeispiel

Dem Fallbeispiel des Buches liegt folgendes fiktives Szenario zugrunde: Die Firma »eHome-Tools« entwickelt Systeme zur Hausautomation. Das Funktionsprinzip solcher Systeme ist folgendes:

Fallbeispiel eHome-Controller

- **Aktoren:**
  Lampen und andere elektrische Verbraucher werden mit elektronischen Schaltern verbunden (sog. Aktoren). Jeder Aktor ist (per Kabel- oder Funkverbindung) an einen Kommunikationsbus angeschlossen und über diesen »fernsteuerbar«.
- **Sensoren:**
  An den Bus können zusätzlich elektronische Temperaturfühler, Windmesser, Feuchtigkeitssensoren usw. angekoppelt werden, aber auch einfache Kontaktsensoren, die z.B. geöffnete Fenster erkennen und melden.
- **Bus:**
  Schaltkommandos an die Aktoren, aber auch Statusmeldungen der Aktoren und Messwerte der Sensoren werden in Form von sogenannten Telegrammen über den Bus von und zum Controller übertragen.
- **Controller:**
  Der Controller sendet Schaltkommandos an die Aktoren (z.B. »Licht Küche ein«) und empfängt Statusmeldungen der Sensoren (z.B. »Temperatur Küche 20 Grad«) und Aktoren (z.B. »Licht Küche eingeschaltet«). Er ist in der Lage, ereignisgesteuert (also abhängig von eingehenden Meldungen) oder auch zeitgesteuert Folgeaktionen auszulösen (z.B. »20:00 Uhr → Rollo Küche schließen«).
- **Bedienoberfläche:**
  Der Controller bietet den Bewohnern des eHome auch eine geeignete Bedienoberfläche. Diese visualisiert den aktuellen Status des eHome und ermöglicht es den Bewohnern, Befehle (z.B. »Licht Küche aus«) per »Mausklick« an die Hauselektrik zu senden.

»eHome-Tools« steht mit seinen Produkten im harten Wettbewerb zu einer Vielzahl von Anbietern. Um in diesem Wettbewerb zu bestehen, wird beschlossen, eine neue Controller-Software zu entwickeln. Allen Beteiligten ist klar, dass Schnelligkeit ein wesentlicher Erfolgsfaktor für

das Vorhaben ist. Denn immer mehr Interessenten und Kunden fragen »eHome-Tools« nach einer Bediensoftware, die auf Smartphones und anderen mobilen Geräten läuft. Auch die Offenheit und Erweiterbarkeit des Systems für Geräte von Fremdherstellern ist enorm wichtig, um Marktanteile hinzuzugewinnen. Wenn das neue System Geräte konkurrierender Hersteller steuern kann, rechnet man sich Chancen aus, auch Kunden dieser Hersteller z. B. beim Ausbau ihrer Systeme für eigene Geräte begeistern zu können. Dazu muss man nicht nur möglichst schnell eine möglichst breite Palette von Wettbewerbs-Hardware unterstützen, sondern auch künftig in der Lage sein, neu am Markt auftauchende Gerätemodelle kurzfristig zu unterstützen.

Daher wird entschieden, den Controller »agil« zu entwickeln und monatlich eine verbesserte, neue Version des Controllers herauszubringen, die mehr Geräte und weitere Protokolle unterstützt.

1.4 Webseite

Die im Buch enthaltenen Codebeispiele sind auf der Webseite zum Buch unter [URL: SWT-knowledge] veröffentlicht und herunterladbar. Der Leser kann diese Beispiele so auf seinem eigenen Rechner nachvollziehen und mit eigenen Testfällen experimentieren.

Auch die Übungsfragen sind dort veröffentlicht und kommentierbar. Über eine rege Onlinediskussion im Leserkreis über mögliche Lösungsalternativen freue ich mich.

Trotz der hervorragenden Arbeit des Verlags und der Reviewer und mehrerer Korrekturläufe sind Fehler im Text nicht auszuschließen. Notwendige Korrekturhinweise zum Buchtext werden ebenfalls auf der Webseite veröffentlicht werden.