Einleitung

Woran liegt es, dass selbst neu entwickelte Anwendungen schon nach kurzer Zeit Eigenschaften von Legacy Code aufweisen? Wissen ist nicht ausreichend vorhanden oder verteilt, Komponenten sind komplex und inkonsistent, und vor allem: Es gibt zu wenige oder keine Tests. Dieses Buch zeigt klar auf, wie solche Fehlentwicklungen vermieden oder nachträglich behoben werden können. Mit vielen Diagrammen und Anwendungsbeispielen aus der Praxis führt es detailliert aus, wie Sie Refactoring und Redesign von Legacy Code effizient und sicher durchführen und eine agile Neuentwicklung architektonisch vorbereiten können. Für die nachhaltige Weiterbildung von ABAP-Entwicklungsteams beschreibt es zudem eine agile Coaching-Methode, die auf einem testorientierten Verbesserungsprozess aufsetzt. Kurz, dies ist ein Buch über agile Entwicklung, ausschließlich mit ABAP-Code!

Kontinuierliche Verbesserungen

Testautomatisierung spielt dabei eine fundamentale Rolle. Die Vergangenheit hat zwar gezeigt, dass große Anwendungen auch ohne ABAP-Unit-Tests entstehen können. Allerdings hat sich inzwischen der Innovationsdruck deutlich erhöht. Zum einen entwickeln sich die Technik und mit ihr auch die betriebswirtschaftlichen Möglichkeiten rasend weiter. Zum anderen versetzt die Cloud SAP-Kunden in die Lage, auf eine schleppende Versorgung mit Innovationen tatsächlich zu reagieren.

Bei immer kürzeren Auslieferungszyklen kann das manuelle Testen zeit- und kostentechnisch nicht mehr bewerkstelligt werden. Die schnellere und breitere Verfügbarkeit von Releases lässt den Softwareunternehmen zudem keinen Spielraum mehr für größere Nachbesserungen. Testautomatisierung schützt in diesem Kontext auch vor Regression bei kurzfristigen Korrekturen.

Diese Absicherung ist wichtig, weil stabile, technisch und betriebswirtschaftlich erfahrene Teams nicht mehr die Regel sind. Neben vermeintlichen Kosteneinsparungen ist es vor allem die Diskontinuität des Marktes, die von den Entwicklern mehr Flexibilität verlangt. Entwickler können sich aber nur dann schnell in neue Gebiete einarbeiten und bei Bedarf problemlos in alte zurückkehren, wenn sie eine saubere Codebasis vorfinden. Doch eine solche Codebasis ist weniger das Ergebnis umfangreicher Planung als vielmehr das Resultat kontinuierlicher Verbesserungen. Ohne Testautomatisierung sind solche *Mikroreleases* praktisch nicht durchführbar.

Agile Methoden

Kontinuierliche Verbesserungen von Code und Design sind aber auch wichtig, weil agile Projektmanagementmethoden wie *Scrum* diese implizit voraussetzen, wenn sie von gleichbleibender Entwicklungsgeschwindigkeit ausgehen. Wo immer diese geplanten und ungeplanten Verbesserungen aber nicht zur Entwicklungskultur gehören, bleiben der erhoffte Effizienzgewinn und auch die breite Akzeptanz aus.

Eine Entwicklungskultur wird maßgeblich dadurch geprägt, in welchem Umfang *Agile Software Engineering* (ASE) angewendet wird. Neben funktionaler Richtigkeit streben die unter diesem Begriff zusammengefassten Praktiken sauberen Code und ein nachhaltiges Management von Wissen an.

Eine agile Entwicklungskultur lässt sich allerdings weder anordnen noch einkaufen. Vielmehr kommt es darauf an, sich erfolgreich auf den Weg zu machen. Schulungen sind dabei wichtig, denn mit erweitertem Wissen fängt alles Neue an. Entscheidender ist allerdings, auf welchen Boden dieses Wissen in den jeweiligen Abteilungen und Firmen fällt. Entwicklungskultur ist etwas, wofür sich Product Owner, Scrum Master und Manager sowie Architekten und Qualitätsexperten gemeinsam verantwortlich fühlen müssen.

Ich lege deshalb viel Wert darauf, diese enge Verbindung zwischen Scrum und ASE zu erklären und mit praktischen Methoden zu untermauern.

Zielgruppe und Voraussetzungen

Mit diesem Buch können Sie als ABAP-Entwickler viel Wissenswertes über die Anwendung von ABAP Objects und ABAP Unit in der Praxis erfahren. Mit dem Fokus auf Entwicklungsprozessen verbessert dieses Buch Ihre Chancen, dieses Wissen gezielt zu Fähigkeiten auszubauen. Es ist definitiv kein Buch, um ABAP Objects zu erlernen. Vielmehr sollten Sie den Kurs BC401 nicht nur besucht, sondern seinen Inhalt schon professionell angewendet haben. Praktische Erfahrungen mit ABAP Unit sind ebenfalls hilfreich, können aber auch lesebegleitend erworben werden.

Die gezeigten Codebeispiele setzen alle auf ABAP 7.53 auf. Dieser Stand ist für das Verständnis aber in keiner Weise entscheidend. Die Herausforderung bei den Codebeispielen liegt eher in der Objektorientierung, weshalb ich viele um Diagramme und Schaubilder ergänzt habe.

Wenn Sie Entwickler einer anderen objektorientierten Programmiersprache sind, dürften Ihnen die ABAP-spezifischen Anweisungen anfangs neu, aber trotzdem verständlich sein. Da es mein Ansatz ist, technische Details frühzeitig in allgemeinverständlichen Hilfsmethoden zu kapseln, kommt Sprachunterschieden in diesem Buch nur eine untergeordnete Rolle zu. Zudem: Wir ABAP-Entwickler sind schon seit vielen Jahren in der Lage, Fachbücher mit Codebeispielen zu C++ oder Java mit Erfolg zu

lesen. Es gibt also keinen Grund, weshalb Ihnen das Gleiche nicht auch mit diesem ABAP-Buch gelingen sollte.

Für Scrum Master, Quality Engineers, Product Owner und Manager sollte schon ein wenig eigene Programmiererfahrung ausreichen, um den Verbesserungen von Code und Design folgen und somit meine strategischen und prozessorientierten Vorschläge verstehen und beurteilen zu können.

Aufbau des Buches

Mit **Kapitel 1**, »Einführung«, können Sie sich einen Überblick über die Ziele und Praktiken agiler Entwicklung verschaffen. Der Rest des Buches folgt dem Walking-Skeleton-Ansatz. Schon im ersten Teil lasse ich Sie an einem kompletten Prozess anhand eines Praxisbeispiels teilhaben. Konkret demonstriere ich einen testorientierten Verbesserungsprozess von Code und Design, der sowohl für die Neu- als auch für die Weiterentwicklung von zentraler Bedeutung ist.

Erst danach gehe ich mit weitestgehend unabhängigen Kapiteln thematisch in die Breite und in die Tiefe: im zweiten Teil des Buches in Richtung Design, im dritten Teil in Richtung Architektur, im vierten Teil in Richtung agile Methoden und im fünften Teil in Richtung Werkzeuge.

Sie können die Lesereihenfolge weitestgehend frei wählen. Ich empfehle Ihnen nur, mit dem ersten Teil zu beginnen, weil ich in ihm den komplexen Begriff der Testinfrastruktur praxisnah einführe und in den folgenden Teilen als bekannt voraussetze.

Teil I: Aufbau einer Testinfrastruktur

Kapitel 2, »Beispielanwendung für diesen Buchteil«, stellt kurz die reale Anwendung vor, die im ersten Teil des Buches um eine neue Funktionalität erweitert werden soll. Um Regression zu vermeiden, soll zuerst die bisherige Funktionalität mit ABAP-Unit-Tests abgesichert werden.

Kapitel 3, »Codebasierte Verbesserung eines Tests«, beschäftigt sich mit der Lesbarkeit von Testmethoden und damit, wie Dopplungen in Testmethoden vermieden werden können.

Kapitel 4, »Designbasierte Verbesserung des Tests«, führt diesen Verbesserungsprozess auf Ebene der Testklassen fort. Die Verbesserungen basieren dabei auf Prinzipien, Entwurfsmustern und Regeln für sauberen Code, wie sie Robert C. Martin (Clean Code, 2009) vorgestellt hat.

Kapitel 5, »Robuster Integrationstest«, beschäftigt sich mit der Unabhängigkeit und Wiederholbarkeit nicht isolierter Testmethoden.

Kapitel 6, »Minimierung von Abhängigkeiten«, stellt die objektorientierte Verschalung einer Legacy-API als effektives Mittel zur Entkopplung von Testcode und Produktcode vor. Daneben geht dieses Kapitel noch auf Factory-Klassen zur Erzeugung von Testdatenobjekten ein.

Kapitel 7, »Isolierter Komponententest«, gibt eine Einführung zu Testdoubles und demonstriert, dass auch für Legacy Code testgetriebene Weiterentwicklung möglich ist.

Kapitel 8, »Redesign mit Unit-Tests«, zeigt schließlich, wie ein Teil der Legacy-Anwendung mit bestehenden Tests neu implementiert werden kann.

Insgesamt entspricht der in diesem Teil vorgeschlagene Verbesserungsprozess einer lokalen Optimierung des Test- und Produktcodes einer Anwendung. Diese Optimierung orientiert sich stets daran, welchen Effizienzgewinn eine bestimmte Maßnahme kurzfristig einbringt.

Teil II: Testorientiertes ABAP-Design

In **Kapitel 9**, »Design von Methoden«, steht unter anderem die Signatur einer Methode im Fokus. Kompakte Methodensignaturen mit wenigen objektbasierten Parametern sind dabei das Ziel.

Kapitel 10, »Design von Klassen«, erklärt, warum eine Klasse einen hohen Zusammenhalt und eine geringe Kopplung an andere Klassen aufweisen sollte. Für die Erzeugung von Objekten sind Factory-Klassen und für die Verwendung dieser Objekte sind Interfaces von zentraler Bedeutung.

Kapitel 11, »Design von Paketen«, stellt die Kapselung von Klassen als eine Hauptaufgabe eines Pakets heraus. Das Ziel ist, dass andere Pakete nur von den Interfaces dieser Klassen abhängen.

Kapitel 12, »Testfälle«, führt in das Testdesign ein, also in die Auswahl von effektiven Testfällen. Die folgenden Kapitel fokussieren sich auf deren effiziente Umsetzung und Wartung.

In **Kapitel 13**, »Testdoubles«, und **Kapitel 14**, »Globale Testdoubles«, geht es darum, wie lokale bzw. globale Testdoubles erzeugt und in den Produktcode injiziert werden.

Kapitel 15, »Testklassen«, beschäftigt sich mit den Details von lokalen und globalen Testklassen.

In **Kapitel 16**, »Testdaten«, liegt das Augenmerk auf Testdatenobjekten. Mit ihnen können Testdaten auf lesbare und redundanzfreie Weise erzeugt und verwendet werden.

Kapitel 17, »Testinfrastruktur«, zeigt schließlich all diese testorientierten Hilfsklassen im Überblick. Strategische Überlegungen zur Testinfrastruktur runden diesen Teil ab.

Teil III: Agile Neuentwicklung

Teil III zeigt anhand einer realen Anwendung, wie die agilen Königsdisziplinen Walking Skeleton und akzeptanztestgetriebene Entwicklung bei einer Neuentwicklung zur Entfaltung kommen. Eine wichtige Voraussetzung dafür ist die frühzeitige Verfügbarkeit einer Testinfrastruktur.

Kapitel 18, »Planung und Vorarbeit«, beschreibt, wie diese Testinfrastruktur schon auf Basis eines übergeordneten Produktdesigns grob entworfen und angelegt werden kann.

Kapitel 19, »Testgetriebene Entwicklung«, simuliert den Beginn einer möglicherweise verteilten und schnell skalierenden Entwicklung.

Teil IV: Agile Methoden

Im Mittelpunkt des vierten Teils dieses Buches stehen Theorie und Praxis agiler Methoden und Praktiken.

Kapitel 20, »Scrum«, stellt das Framework vor, mit dem die meisten agilen Projekte heutzutage verwaltet werden. Scrum ist einfach und rückt neben der Entwicklung des Produkts auch die kontinuierliche Weiterentwicklung des Teams in den Mittelpunkt.

Kapitel 21, »Agile Software Engineering«, führt die wichtigsten agilen Praktiken zur Entwicklung von sauberem Code und Design ein. Dazu gehören unter anderem Refactoring, testgetriebene Entwicklung (TDD) und paarweise Programmierung (PP).

Kapitel 22, »Lean-Entwicklungsmodell«, beschreibt eine Form der Unternehmensführung, auf der sowohl Scrum als auch ASE in großem Maße aufbauen. Lean sieht unter anderem die Weiterentwicklung der Mitarbeiter als wichtige Voraussetzung für die Qualität der Produkte.

Ganz im Sinne von Lean kommt in **Kapitel 23**, »Entwicklung der Teams«, zuerst die ASE-Ausbildung zum Zuge. Erst danach gehe ich in **Kapitel 24**, »Entwicklung des Backlogs«, auf die agile Spezifikation der Produktanforderungen und in **Kapitel 25**, »Entwicklung des Produkts«, auf Strategien bei der Umsetzung dieser Anforderungen ein.

Teil V: Testorientierte ABAP-Werkzeuge

Kapitel 26, »ABAP Unit«, fasst noch einmal alles Wissenswerte zu ABAP Unit zusammen. Es stellt Verfahren vor, wie Funktionsgruppen und Programme mit Klassen entwickelt oder verbessert werden können. Eine Betrachtung dieser Entwicklungsobjekte darüber hinausgehend ist deshalb nicht nötig.

Kapitel 27, »ABAP Development Tools«, geht auf die Möglichkeiten von ABAP in Eclipse ein, die agile Entwicklung in effizienter Weise zu unterstützen.

Dagegen liegt der Schwerpunkt von **Kapitel 28**, »ABAP-Werkzeuge zur Testisolierung«, auf Frameworks zur Kontrolle von Testdaten und Testverhalten.

Informationskästen

In hervorgehobenen Informationskästen finden Sie in diesem Buch Inhalte, die wissenswert und hilfreich sind, aber etwas außerhalb der eigentlichen Erläuterung stehen. Damit Sie die Informationen in den Kästen sofort einordnen können, haben wir die Kästen mit Symbolen gekennzeichnet:

In Kästen, die mit diesem Symbol gekennzeichnet sind, finden Sie Informationen zu *weiterführenden Themen* oder wichtigen Inhalten, die Sie sich merken sollten.

Dieses Symbol weist Sie auf *Besonderheiten* hin, die Sie beachten sollten. Es *warnt* Sie außerdem vor häufig gemachten Fehlern oder Problemen, die auftreten können.

Mit diesem Symbol markierte Textstellen fassen wichtige thematische Zusammenhänge für Sie noch einmal *auf einen Blick* zusammen.

Einschränkungen

Dieses Buch konzentriert sich auf Agilität und Objektorientierung mit ABAP. Es klammert bewusst alle Möglichkeiten aus, die sich heute und in Zukunft für ABAP-Entwickler mit SAP HANA oder der SAP Cloud Platform zusätzlich ergeben. Dazu zählen zum einen Core Data Services Views (CDS Views), mit denen eine Hauptspeicherdatenbank in die Lage versetzt wird, auf einer bedingten Menge von Daten Berechnungen, Formatierungen usw. auszuführen. Zum anderen sind da ABAP Managed Database Procedures (AMDPs) zu nennen, also von ABAP verwaltete Datenbankprozeduren. Mit beiden Artefakten geht ein als *Code Pushdown* bezeichneter Paradigmenwechsel einher, nach dem Daten nicht mehr in die Anwendungsschicht geladen und dort verarbeitet werden. Der Performancevorteil kann beträchtlich sein. Durch die Verwendung dieser Artefakte nimmt der Umfang der mit ABAP Objects implementierten Anwendungsteile ab. Für die übrigen Anwendungsteile bleibt Objektorientierung genauso relevant wie für die große Menge an Bestandscode, der trotz Cloud noch lange im Einsatz bleiben wird.

Danksagung

Ich bedanke mich bei allen, die zur Entstehung dieses Buches über agile ABAP-Entwicklung beigetragen haben. Konkrete Verbesserungsvorschläge zum Manuskript, die Eingang in das vorliegende Buch gefunden haben, stammen von den SAP-Kolleginnen und -Kollegen Dr. Ole Krüger, Dr. Gerhard Dierkes, Markus Riepp, Frank Emminghaus, Knut Stargardt, Dr. Stefan Butscher, Michael Gutfleisch, Lars Euler, Jean-Philippe Lombardi, und Nena Raab.

In besonderem Maße möchte ich meinem SAP-Kollegen Dr. Ole Krüger Anerkennung zollen. Wer ein Buch alleine schreibt, sollte wenigstens einen so kompetenten und hilfsbereiten Vertrauten wie ihn an seiner Seite wissen.

Auf das Vertrauen der Führungskräfte kommt es ebenso an. Sie fördern die fachliche Weiterbildung über viele Jahre und schaffen die Freiräume, ohne die ein Fachbuch wie dieses nicht entstehen kann. Mein besonderer Dank geht dabei an Dr. Oliver Kroneisen, Vice President der Abteilung Financial Operations Development bei SAP, für die tatkräftige Unterstützung bei der Umsetzung dieses Buchprojekts.

Daneben möchte ich aber auch meine früheren Vorgesetzten Matthias Grabellus und Dr. Peter Neumayer als Wegbereiter anerkennen. Für die intensive Kooperation und Förderung bedanke ich mich sehr bei Dr. Jürgen Heymann, dem Leiter der Initiative Agile Software Engineering bei SAP. Ich freue mich, dass er das Vorwort zu diesem Buch beigesteuert hat.

Beim Rheinwerk Verlag danke ich besonders meiner Lektorin Janina Karrasch für die vielen wertvollen Verbesserungsvorschläge und das sichere Geleit. Aber auch die gewissenhafte Arbeit der Korrektorin Annette Lennartz, der Coverdesignerin Nadine Kohl und der Herstellung erwähne ich gerne.

Winfried Schwarzmann
Development Architect und Agile Software Engineering Coach
Financial Operations Development, SAP SE