Kapitel 10

Beispiel für die Entwicklung von adaptivem Code: Einführung

In diesem Kapitel werden Sie die folgenden Aufgaben durchführen:

- Kennenlernen des Teams, das die Beispielanwendung entwickelt und dabei adaptiven Code erstellt
- Beschreiben der Features des Produkts
- Erstellen einer ersten Version des Produkt-Backlogs für die Anwendung in Sprint 0

In den Kapiteln dieses Buchteils bauen Sie nach und nach eine funktionierende Anwendung auf. Dabei nutzen Sie Scrum und folgen den adaptiven Entwurfsprinzipien, die in diesem Buch beschrieben wurden. Dieser Teil fasst die bisherigen Inhalte zusammen und baut auf dieser Basis ein Gesamtbild auf. Wie bei den übrigen Kapiteln empfehle ich Ihnen auch hier, den zugehörigen Code[1] zu öffnen, während Sie das Kapitel durchlesen. Ohne den Inhalt dieser Kapitel fehlt der Kontext zum Verständnis des Quellcodes. Aber umgekehrt zeigen die Listings in den folgenden Kapiteln ohne die vollständige Microsoft Visual Studio-Projektmappe nur einen kleinen Ausschnitt des Gesamtbilds.

Das Format dieses Kapitels soll ein Szenario aus der Praxis widerspiegeln, aber aus Platzgründen und um die wesentlichen Punkte herauszuarbeiten, sind einige Kompromisse nötig. Dieses Kapitel enthält eine Einführung in ein fiktives Scrum-Team und skizziert das Produkt, das entwickelt werden soll.

Trey Research

Die Beispielanwendung wird von einem fiktiven Unternehmen namens Trey Research entwickelt. Das Unternehmen wirbt damit, dass es adaptiven Code schreibt, der Änderungen problemlos verkraftet.

1. In Anhang A, »Adaptive Tools«, finden Sie eine Anleitung, wie Sie auf den Code aller Beispielanwendungen zugreifen, die in diesem Buch beschrieben werden.

Das Team

Die Beispielanwendung wird mithilfe des Scrum-Prozesses entwickelt, daher braucht das Team Mitglieder, die die verschiedenen Scrum-Rollen übernehmen. Eine Übersicht über diese Rollen und den Scrum-Prozess finden Sie in Kapitel 1, »Einführung in Scrum«.

Das Team umfasst alle Rollen, die gebraucht werden, um die Anwendung von der ursprünglichen Idee bis zur Auslieferung voranzutreiben. Der Product Owner weiß, wie die Anwendung arbeiten soll, welche Features die höchste Priorität haben und welche den größten Gewinn für das Unternehmen bringen. Der Scrum Master konzentriert sich auf den Prozess, den das Team anwendet. Seine wichtigste Aufgabe ist, sicherzustellen, dass der Prozess für das Team funktioniert, dass dem Team keine Hindernisse in den Weg gelegt werden und dass der Product Owner über alle Probleme auf dem Laufenden bleibt, die während der Entwicklung von User Stories auftauchen. Das Entwicklungsteam von Trey Research besteht aus Entwicklern, die Stories implementieren, sowie einem Testanalysten, der Testfälle entwirft und prüft, ob die Stories die Kriterien für eine Auslieferung erfüllen.

Product Owner

Product Owner ist Petra. Sie ist eine erfahrene Geschäftsanalytikerin, die vor kurzem ihre Stelle bei Trey Research angetreten hat. Sie hat sich darauf spezialisiert, genau herauszufinden, was der Kunde will – eine sehr wertvolle Fähigkeit für einen Product Owner. Sie gibt bereitwillig zu, dass agile Prozesse etwas Neues für sie sind, freut sich aber über die Gelegenheit, mehr über ihre neue Rolle zu lernen.

Während des gesamten Entwicklungsprozesses kommuniziert Petra mit dem Unternehmen des Kunden, um genau herauszufinden, was sie wollen und aus welchen Gründen. Außerdem berechnet sie den Wert der verschiedenen Features für den Kunden, damit sie besser die Prioritäten für die Arbeit des Entwicklungsteams festlegen kann.

Scrum Master

Steve übernimmt zwei Rollen im Unternehmen. Er ist gleichzeitig Scrum Master und technischer Teamleiter. Diese Konstellation kommt häufiger vor, das Unternehmen möchte das allerdings in naher Zukunft ändern, damit sich Steve ganz auf seine bevorzugte Rolle als Scrum Master konzentrieren kann. Aus diesem Grund bemüht sich das Unternehmen gerade, einen erfahrenen Entwickler einzustellen, der Steve den Posten des technischen Teamleiters abnimmt.

In seiner Eigenschaft als Scrum Master stellt Steve sicher, dass das Team dem Scrum-Prozess folgt und dass die Teammitglieder mit der aktuellen Form des Prozesses zufrieden sind. Er schätzt sich als jemanden ein, der ehrlich und transparent mit jedem zugewiesenen Product Owner oder Kunden zusammenarbeitet und niemals Kennzahlen ändert oder unrealistische Termine nennt.

Obwohl Steve nur selten Zeit findet, selbst Code zu schreiben, nimmt er an Entwurfs-Meetings teil und versucht nach Möglichkeit, das Entwicklungsteam in die richtige Richtung zu lenken.

Entwickler

David und Dianne sind die beiden Softwareentwickler des Unternehmens. David ist noch Einsteiger, er ist vor einiger Zeit nach seinem Hochschulabschluss in das Unternehmen eingetreten. Dianne hat schon mehrere Jahre Berufserfahrung.

Steve hat David vor allem deswegen eingestellt, weil er sich ständig in den Bereichen Programmierverfahren und -techniken weiterbildet. David versucht, die neuesten Trends im Blick zu behalten und ist stets lernbegierig. Allerdings betrachtet er neue Techniken manchmal zu unkritisch als Universallösungen und wendet sie an, wo immer er kann. Das ist eine hervorragende Übungsmöglichkeit für David, aber sein Code ist oft ein Sumpf aus unnötiger Indirektion.

Dianne hat mehr Erfahrung als David, aber sie neigt dazu, der Flut neuer Technologien, die in den letzten Jahren populär wurden, ablehnend gegenüberzustehen. Sie hat einst genau dieselbe Phase durchlebt wie David jetzt und damit keine gute Erfahrung gemacht. Dianne bewirbt sich für die Stelle als technischer Teamleiter und ist entschlossen, ihre Fähigkeiten unter Beweis zu stellen und eine Beförderung zu ergattern. Zu diesem Zweck freut sie sich darauf, eng mit David zusammenzuarbeiten und ihm bei seiner Weiterentwicklung zu helfen.

Der Hype-Zyklus

Der Hype-Zyklus wurde von Gartner entwickelt, einem IT-Forschungs- und Consultingunternehmen. Er ist ein nützliches Werkzeug, um den Fortschritt neuer Techniken und Technologien zu beurteilen. Abbildung 10–1 zeigt seinen Ablauf.

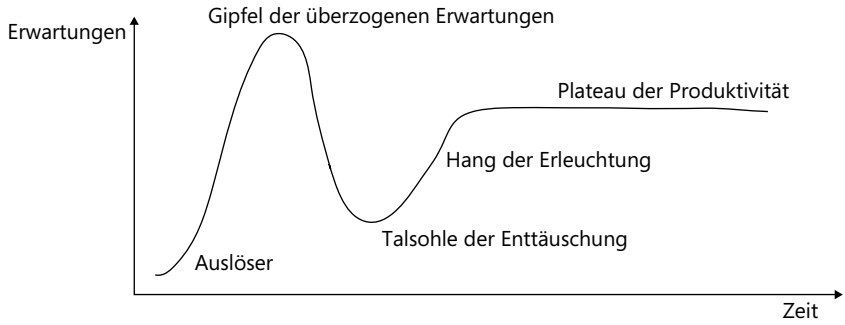

Abb. 10–1 Der Hype-Zyklus hilft, die Einstellung gegenüber neuen Techniken und Technologien zu erklären

→

Die x-Achse des Diagramms steht für den Zeitablauf und die y-Achse für Erwartungen. Anfangs gibt es einen Auslöser, der eine neue technologische Entdeckung oder eine nützliche neue Technik oder Prozedur sein kann. Bald danach steigen die Erwartungen sprunghaft an, bis sie den »Gipfel der überzogenen Erwartungen« erreichen. Ab diesem Punkt wird die Technologie oder Technik nicht mehr als so leistungsfähig eingeschätzt wie am Anfang. Daraufhin fallen die Erwartungen rapide in die »Talsohle der Enttäuschung«. Aber noch ist nicht alles verloren, denn der »Hang der Erleuchtung« führt langsam aufwärts bis zum »Plateau der Produktivität«. An diesem Punkt hat sich die Technologie oder Technik etabliert und es werden realistischere Erwartungen an sie gestellt.

Die Erwartungen von Entwicklern bezüglich bestimmter Programmiertechnologien oder -techniken befinden sich oft irgendwo in einem der Bereiche, die in diesem Diagramm beschrieben sind. Zum Beispiel nähert sich David bei Trey Research gerade dem Gipfel der überzogenen Erwartungen bezüglich Entwurfs-Patterns, SOLID-Prinzipien, Unit-Tests und Refaktorierung. Dianne arbeitet sich langsam aus der Talsohle der Enttäuschung heraus. Achten Sie bei den Sprint-Meetings, die in diesem Buchteil verfolgt werden, auf ihre Fragen und Antworten und überlegen Sie, welchen Einfluss ihre Position im Hype-Zyklus darauf hat.

Schließlich sollte noch erwähnt werden, dass Steve aufgrund seiner Erfahrung so weit vorangeschritten ist, dass er auf dem Plateau der Produktivität gelandet ist. Ein weiterer Vorteil dieser Entwicklung ist, dass Steve nicht mehr so anfällig für den Hype-Zyklus ist wie früher. Er ist vorsichtig, wenn ihm neue Techniken und Technologien als Heilsbringer für Produktivität, Qualität, Effizienz oder irgendeine andere Qualität angepriesen werden. Daher schwanken seine Erwartungen nicht so wild hin und her, wenn ihm eine neue Technik oder Technologie präsentiert wird.

Testanalyst

Tom ist der Testanalyst des Teams. Er konzentriert sich auf die Testautomatisierung. Er besaß schon immer die Fähigkeit, die Schwächen eines Softwareprodukts aufzudecken und die Benutzerzufriedenheit durch höhere Robustheit zu steigern. Er behandelt Features am liebsten als Blackboxes und interessiert sich weniger dafür, wie etwas implementiert ist, als dafür, ob es entsprechend den Spezifikationen (oder sogar darüber hinaus) funktioniert.

Tom hat das Gefühl, dass er im Team zu viel Arbeit erledigen muss, weil er ständig bereits implementierte Elemente zurückweist. Das bedeutet, dass er eine Story mindestens zweimal neu testen muss, bevor sie seine anspruchsvollen Standards erfüllt. Tom bemüht sich mit allen Kräften, dass Fehler möglichst selten bis auf den Computer eines Kunden gelangen und sie stattdessen schon früh durch seine automatisierten Tests abgefangen werden.

Das Produkt

Trey Research wurde von einem Kunden beauftragt, eine neue Online-Chat-Anwendung namens Proseware zu entwickeln. Dies ist eine webbasierte Chat-Anwendung, über die sich Leute aus der ganzen Welt miteinander unterhalten können. Während der ersten Besprechung mit dem Kunden hat Petra festgestellt, dass der Kunde sehr viele Ideen für das Projekt hat, aber erst einmal etwas Kleines zusammenbauen möchte, damit er besser darüber entscheiden kann, welche Richtung die Anwendung nehmen soll. Trey Research eignet sich gut für dieses Produkt, weil das Unternehmen einen inkrementellen Entwicklungsstil pflegt und die Fähigkeit besitzt, auf die manchmal sehr plötzlichen Änderungen an den Anforderungen zu reagieren.

Petra hat mit dem Kunden geredet, bevor das Projekt grünes Licht erhielt, daher kann sie das Team mit einigen Informationen darüber versorgen, was sie entwickeln sollen. Der Kunde möchte, dass Proseware von Trey Research gehostet wird, daher hat das Team freie Wahl bezüglich der Plattformen und Tools für das Projekt. Petra hat natürlich den Kunden befragt, wie viele Benutzer Proseware bedienen soll. Sie ist sich mit dem Kunden einig darüber, dass es ausreicht, wenn die Anwendung 20 Benutzer gleichzeitig unterstützt, und dies ist eine zentrale nichtfunktionelle Anforderung an Proseware.

Bevor das Team damit beginnt, irgendwelchen Code zu programmieren, beruft Petra ein Teammeeting ein, auf dem die Teammitglieder das Projekt diskutieren und ein anfängliches Backlog der User Stories zusammenstellen können. Sobald diese Stories vorliegen, kann das Team sich daran machen, innerhalb eines einzigen Sprints etwas Vorzeigbares zu erstellen.

Anfängliches Backlog

Der Kunde sendet Petra und dem Team eine Liste der gewünschten Features für Proseware. Dies geschieht in Form einer Prosabeschreibung und das Team hat sich für das Meeting das Ziel gesetzt, aus diesen Beschreibungen eine oder mehrere User Stories abzuleiten.

> *Wir sehen Proseware als die primäre Site für Menschen, die sich unterhalten wollen. Uns ist natürlich klar, dass Rom nicht an einem Tag erbaut wurde, daher haben wir unsere Zielsetzung auf etwas eingeschränkt, das so schnell wie möglich geliefert werden kann.*
>
> *Wir wollen zwar den Benutzern ermöglichen, sich gegenseitig Bilder oder Dateien zu schicken, die Kernfunktion ist aber die Textkommunikation. Alle, die sich in einem Raum treffen, sollten alle Nachrichten empfangen, die von irgendeinem anderen Anwesenden gesendet werden.*
>
> *Weil die Textseite des Chats so wichtig ist, müssen die Mitglieder eines Raums in der Lage sein, formatierten Text auszutauschen, etwas wie HTML. Natürlich sollte es nicht möglich sein, den Chat mit sinnlosen Bildern oder Videos zuzumüllen!*

Einige der Unterhaltungen in Proseware sollten von manchen Benutzern nicht bearbeitet werden dürfen, und manchen Benutzern sollte nur das Lesen erlaubt sein, aber nicht das Mitwirken.

Petra bringt diese kurze Beschreibung zum Meeting mit und das Team beginnt damit, die User Stories zu finden.

User Stories in Prosabeschreibungen finden

Manchmal verfassen Kunden Prosabeschreibungen ihrer Erwartungen, wie im letzten Abschnitt zu sehen. Aus diesen Beschreibungen muss das Team die User Stories, die implementiert werden müssen, extrahieren und ausformulieren. Das Trey Research-Team trifft sich zu einem Meeting, um die User Stories in der Beschreibung von Proseware zu finden, die der Kunde zur Verfügung gestellt hat.

PETRA: Okay, haben alle die Beschreibung gelesen? [Alle nicken, mit unterschiedlicher Begeisterung.]

STEVE: Da steht eine Menge drin!

DAVID: Nicht wirklich. Wir können das wahrscheinlich in einem einzigen Sprint erledigen. [Breites Grinsen von Steve.]

PETRA: Kann mir bitte jemand sagen, wie wir daraus eine Liste der User Stories bekommen sollen?

DIANNE: Zuerst sollten wir die Verben und Substantive raussuchen: *schicken, empfangen, formatieren, zumüllen* … *Raum*, *Unterhaltung*, *Benutzer*, *Mitglieder*, *Chat* …

STEVE: Stimmt, da steckt eine Menge drin.

PETRA: Als Benutzer will ich …, damit …

DIANNE: Sind sie nur *Benutzer*?

STEVE: Klingt als gäbe es mehr als eine Rolle.

DAVID: Sie nennen sie *Benutzer* und dann wieder *Mitglieder*.

TOM: Bist du denn kein *Mitwirkender* an einer Unterhaltung?

STEVE: Vielleicht, aber wir sollten ihre vertraute Sprache benutzen. Es sind einige Rollen im Spiel, wenn es darum geht, dass Unterhaltungen nicht geändert werden dürfen, oder nicht?

DIANNE: Ja, das klingt nach einer Art von Berechtigungssystem.

PETRA: Kann bitte jemand eine Story finden?

DAVID: Als Mitglied will ich formatierten Text verschicken, damit andere … *Zeug* … ansehen können?

STEVE: Sollen wir uns vorerst auf die Rollen und das Verhalten konzentrieren? Petra, du kannst etwas zum wirtschaftlichen Wert sagen, oder?

PETRA: Sicher.

DIANNE: Ja, aber sogar das sieht arg groß aus. Eher ein Epic als eine User Story.

PETRA: Entschuldigung, was ist ein *Epic* gleich wieder?

DAVID: Im Grund nur eine große Story. Dianne denkt, diese Story ist zu groß für einen einzelnen Sprint, aber ich weiß nicht, wie es kleiner sein könnte.

DIANNE: Naja, was ist einfacher als formatierter Text? Ignorieren wir den HTML-Teil, da versucht der Kunde uns nur eine Implementierung aufzuzwingen! Denken wir über *irgendeinen* formatierten Text nach: Was ist einfacher?

DAVID: Äh … *Un*formatierter Text?

STEVE: Auch als *reiner Text* bekannt.

DIANNE: Als Raummitglied will ich reine Textnachrichten an andere Raummitglieder schicken. [Petra schaut herum, ob jemand widersprechen will, aber alle scheinen sich einig zu sein.]

TOM: Hervorragend! Meine Damen und Herren, wir haben unsere erste User Story! [Alle außer Petra spenden eine spontane Runde Applaus.]

PETRA: Aber wollen sie nicht formatierten Text?

STEVE: Keine Sorge, das wird eine andere Story. Sie kommt später, nachdem der reine Text geliefert wurde.

PETRA: Okay. Als Raummitglied will ich formatierte Textnachrichten an andere Raummitglieder schicken.

STEVE: Damit haben wir zwei. Noch mehr?

DAVID: Ich will Räume bauen?

DIANNE: Ja. Ich denke, das macht diese Person zu einem Raumbesitzer, stimmt's? Als Raumbesitzer will ich Räume anlegen, um Unterhaltungen thematisch einzugrenzen?

STEVE: Gut.

TOM: Sie sagen außerdem, dass sie Bilder und Dateien schicken wollen.

PETRA: Ja, das ist eine weitere Story.

STEVE: Noch eine: Ich will Unterhaltungen aufbauen, die nur gelesen werden können.

DIANNE: Damit haben wir den letzten Satz abgedeckt. Ich denke, wir sind fertig.