Modernes Software-Engineering

Entwurf und Entwicklung von Softwareprodukten

Ian Sommerville

Pearson

EXTRAS ONLINE

Modernes Software-Engineering

Modernes Software-Engineering

Entwurf und Entwicklung von Softwareprodukten

Ian Sommerville

Pearson

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.dnb.de* abrufbar.

10 9 8 7 6 5 4 3 2 1

24 23 22 21 20

ISBN 978-3-86894-396-2 (Buch)
ISBN 978-3-86326-892-3 (E-Book)

Programmleitung: Birger Peil, bpeil@pearson.de
Übersetzung: Katharina Pieper, Berlin; Petra Alm, Saarbrücken
Fachlektorat und Korrektorat: Katharina Pieper, Berlin
Coverabbildung: Dmitrydesign, www.shutterstock.com
Herstellung: Claudia Bäurle, pburkart@pearson.de
Satz: Gerhard Alfes, mediaService, Siegen (www.mediaservice.tv)
Druck und Verarbeitung: DZS-Grafik d.d.o., Ljubljana

Printed in Slovenia

Inhaltsübersicht

Inhaltsverzeichnis

Vorwort

Softwareprodukte – wie eigenständige Programme, Webanwendungen und -dienste sowie mobile Anwendungen – haben unser tägliches Leben und unsere Arbeit verändert. Es gibt Zehntausende von Unternehmen, die Softwareprodukte entwickeln, und Hunderttausende von Software-Ingenieuren sind weltweit in der Softwareproduktentwicklung beschäftigt.
Entgegen der Meinung einiger Leute braucht man für die Entwicklung von Softwareprodukten viel mehr als nur Programmierkenntnisse. Daher habe ich dieses Buch geschrieben, um einige der Software-Engineering-Aktivitäten vorzustellen, die für die Herstellung zuverlässiger und sicherer Softwareprodukte wichtig sind.

An wen richtet sich das Buch?

Das Buch wurde für Studenten entworfen, die einen ersten Kurs im Bereich Software-Engineering belegen. Auch Personen, die darüber nachdenken, ein Produkt zu entwickeln, und die nicht viel Erfahrung im Software-Engineering haben, könnten es hilfreich finden.

Warum brauchen wir ein Buch über Software-Engineering, das sich auf Softwareprodukte konzentriert?

Die meisten Texte über Software-Engineering konzentrieren sich auf *projektbasiertes* Software-Engineering, bei dem ein Kunde eine Spezifikation vorgibt und die Software von einer anderen Firma entwickelt wird. Die Methoden und Techniken des Software-Engineerings, die sich für solche Großprojekte herausgebildet haben, sind jedoch nicht für die Entwicklung von Softwareprodukten geeignet.
Studierende finden es häufig schwierig, einen Bezug zu großen, kundenspezifischen Softwaresystemen herzustellen. Ich denke, dass es für Studenten leichter ist, Software-Engineering-Techniken zu verstehen, wenn sie sich auf die Art von Software beziehen, die Studierende ständig benutzen. Außerdem sind viele Techniken der Produktentwicklung für Studentenprojekte relevanter als projektorientierte Techniken.

Ist dies eine neue Ausgabe Ihres anderen Lehrbuchs über Software-Engineering?

Nein, dieses Buch verfolgt einen völlig anderen Ansatz und verwendet, abgesehen von ein paar Diagrammen, kein Material aus der 10. Auflage des Lehrbuchs *Software Engineering*.

Welche Themen werden in dem Buch behandelt?

In zehn Kapiteln werden die Themen Softwareprodukte, agiles Software-Engineering, Features, Szenarios und User-Storys, Softwarearchitektur, cloudbasierte Software, Microservice-Architektur, Sicherheit und Datenschutz, zuverlässige Programmierung, Testen sowie DevOps und Codemanagement behandelt.
Ich habe das Buch so gestaltet, dass es sich für einen einsemestrigen Software-Engineering-Kurs eignet.

Wie unterscheidet sich dieses Buch von anderen einführenden Texten zum Software-Engineering?

Wie ich schon sagte liegt der Schwerpunkt auf *Produkten* anstelle auf *Projekten*. Ich behandele Techniken, die die meisten anderen SE-Texte nicht abdecken, wie z.B. Personas und Szenarios, Cloud-Computing, Microservices, Sicherheit und DevOps. Da die Produktinnovation nicht aus der universitären Forschung stammt, wird keine Forschungsliteratur zitiert bzw. Referenzen darauf angegeben, und das Buch ist in einem informellen Stil geschrieben.

Welche Vorkenntnisse muss ich mitbringen, um aus dem Buch einen Nutzen zu ziehen?

Ich gehe davon aus, dass Sie über Programmiererfahrung in einer modernen objektorientierten Programmiersprache wie Java oder Python verfügen und dass Sie mit bewährten Programmiermethoden vertraut sind, z.B. mit der Verwendung aussagekräftiger Namen. Außerdem sollten Sie grundlegende Konzepte der Datenverarbeitung wie Objekte, Klassen und Datenbanken verstehen. Die Programmbeispiele in diesem Buch sind in Python geschrieben, aber sie sind für jeden mit Programmiererfahrung verständlich.

Welches zusätzliche Material ist verfügbar, um Dozierende zu unterstützen?

1. Übungen sowie Testfragen für alle Kapitel
2. Vorschläge, wie Sie das Buch in einem einsemestrigen Software-Engineering-Kurs einsetzen können
3. Präsentationen für die Lehre (Keynote, PowerPoint und PDF)

EXTRAS ONLINE

Sie können auf dieses Material (auf Englisch) zugreifen unter: *https://iansommerville.com/engineering-software-products*
Zusätzliches Material (Lösungshinweise auf Deutsch und weitere Fragen und Antworten) ist auf der Website des Buches nach einer kostenlosen Registrierung:
https://www.pearson-studium.de/modernes-software-engineering.html

Wo kann ich mehr erfahren?

Ich habe ein paar Blogeinträge geschrieben, die für das Buch relevant sind. Diese bieten weitere Informationen über meine Gedanken zur Lehre von Software-Engineering und zu meiner Motivation, das Buch zu schreiben.

„Out with the UML (and other stuff too): reimagining introductory courses in software engineering“
https://iansommerville.com/systems-software-and-technology/static/2018/03/18/what-should-we-teach-in-software-engineering-courses

„Engineering Software Products“
https://iansommerville.com/systems-software-and-technology/static/2018/05/31/engineering-software-products

Danksagungen

Ich möchte allen Personen danken, die die erste Fassung dieses Buchs gelesen haben und mich mit hilfreichen Vorschlägen unterstützt haben:

Paul Eggert – UCLA Los Angeles
Jeffrey-Miller – Universität von South California
Harvey-Siy – Universität von Nebraska, Omaha
Edmund S. Yu – Syrakus-Universität
Gregory-Gay – Universität von South Carolina
Josh-Delinger – Towson-Universität
Rocky-Sklavin – Universität von Texas, San Antonio
Bingyang Wei – Midwestern State Universität

Danke auch an Adam Barker von der Universität St. Andrews, der mich beim Thema Container unterstützte, und an Rose Kernan, die die Produktion des Buchs leitete.

Danke – wie immer – an meine Familie für ihre Hilfe und Unterstützung, während ich das Buch schrieb. Besonderer Dank gilt meiner Tochter Jane, die den Text hervorragend korrekturgelesen und kommentiert hat. Sie war eine schonungslose Lektorin! Ihre Änderungsvorschläge haben die Qualität meiner Prosa erheblich verbessert.

Und schließlich ein besonderer Dank an unser neuestes Familienmitglied, meinen wundervollen Enkel Cillian, der während der Entstehungszeit dieses Buchs geboren wurde. Seine quirlige Persönlichkeit und sein ständiges Lächeln waren eine sehr willkommene Ablenkung von der manchmal mühsamen Arbeit des Schreibens und Editierens.

Ian Sommerville

Softwareprodukte

1

ÜBERBLICK

Dieses Buch stellt Techniken des Software-Engineerings vor, die bei der Entwicklung von Softwareprodukten eingesetzt werden. Softwareprodukte sind generische Softwaresysteme, die an Regierungen, Unternehmen und private Endverbraucher verkauft werden. Sie können dazu dienen, eine Geschäftsfunktion wie die Buchhaltung zu unterstützen; es können Produktivitätswerkzeuge wie Notizsysteme sein; es können aber auch Spiele oder persönliche Informationssysteme sein. Die Größe von Softwareprodukten reicht von Millionen von Codezeilen in großen Geschäftssystemen bis hin zu einigen Hundert Codezeilen in einer einfachen App für Mobiltelefone.
Wir alle verwenden Softwareprodukte jeden Tag auf unseren Rechnern, Tablets und Telefonen. Ich benutze ein Softwareprodukt – den Ulysses-Editor –, um dieses Buch zu schreiben. Um die endgültige Version zu formatieren, werde ich ein anderes Produkt zur Textbearbeitung – Microsoft Word – einsetzen und Dropbox werde ich verwenden, um die Dateien mit dem Verlag auszutauschen. Auf meinem Handy benutze ich Softwareprodukte (Apps) zum Lesen von E-Mails, zum Lesen und Schreiben von Tweets, zum Ansehen des Wetterberichts und so weiter.
Die Engineering-Techniken, die für die Produktentwicklung eingesetzt werden, sind aus den Software-Engineering-Techniken entstanden, die im 20. Jahrhundert entwickelt wurden, um die Erstellung kundenspezifischer Software zu unterstützen. Als das Software-Engineering in den 1970er Jahren als Disziplin aufkam, war praktisch jede professionelle Software eine „einmalige“ Individualsoftware. Unternehmen und Regierungen wollten ihre betrieblichen Prozesse automatisieren und legten fest, was ihre Software leisten sollte. Die Software wurde dann von einem hauseigenen Entwicklerteam oder einem externen Softwareunternehmen programmiert.
Beispiele für kundenspezifische Software, die um diese Zeit herum entwickelt wurde, sind:

- das Flugverkehrsmanagementsystem der US-Bundesbehörde für Luftfahrt;
- Buchhaltungssysteme für alle großen Banken;
- Abrechnungssysteme für Versorgungsunternehmen wie Strom- und Gasversorger;
- militärische Führungs- und Kontrollsysteme.

Diese Individualsysteme wurden im Rahmen von Softwareprojekten erstellt, wobei das Softwaresystem auf einer Reihe von Softwareanforderungen basierte. Der Vertrag zwischen dem Softwarekunden und der Softwareentwicklungsfirma enthielt ein Anforderungsdokument, welches eine Spezifikation der zu liefernden Software darstellte. Die Kunden definierten ihre Anforderungen und arbeiteten mit dem Entwicklerteam zusammen, um die Funktionalität der Software und ihre kritischen Eigenschaften im Detail festzulegen.
Dieser projektbezogene Ansatz dominierte die Softwarebranche mehr als 25 Jahre lang. Die Methoden und Techniken, die sich zur Unterstützung der projektbasierten Entwicklung herausgebildet hatten, definierten, was unter „Software-Engineering“ zu verstehen war. Die Grundannahme war, dass erfolgreiches Software-Engineering viel Vorarbeit erforderte, bevor mit dem Schreiben von Programmen begonnen werden konnte. So war es beispielsweise wichtig, Zeit für das „richtige“ Aufstellen der Anforderungen aufzuwenden und grafische Modelle der Software zu zeichnen. Diese Modelle wurden während des Softwareentwurfsprozesses erstellt und zur Dokumentation der Software eingesetzt.
Als jedoch immer mehr Unternehmen ihren Geschäftsbetrieb automatisierten, wurde klar, dass die meisten Firmen keine kundenspezifische Software benötigten. Sie konn-

ten generische Softwareprodukte verwenden, die für allgemeine Geschäftsaufgaben entworfen wurden. Die Softwareproduktbranche entwickelte sich weiter, um diesem Bedürfnis gerecht zu werden. Projektbasierte Software-Engineering-Techniken wurden an die Entwicklung von Softwareprodukten angepasst.
Projektbasierte Techniken sind für die Produktentwicklung aufgrund der fundamentalen Unterschiede zwischen projektbezogenem und produktbezogenem Software-Engineering nicht geeignet. Diese Unterschiede sind in ▶ Abbildung 1.1 und ▶ Abbildung 1.2 dargestellt.

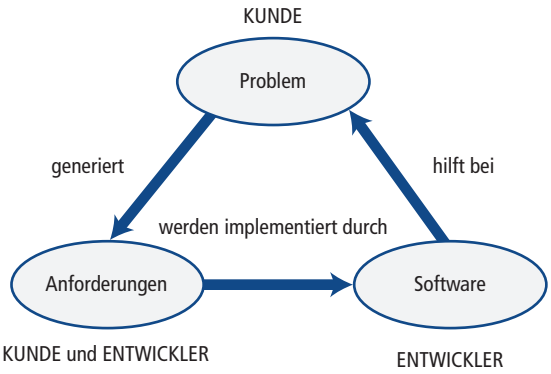

Abbildung 1.1: Projektbasiertes Software-Engineering

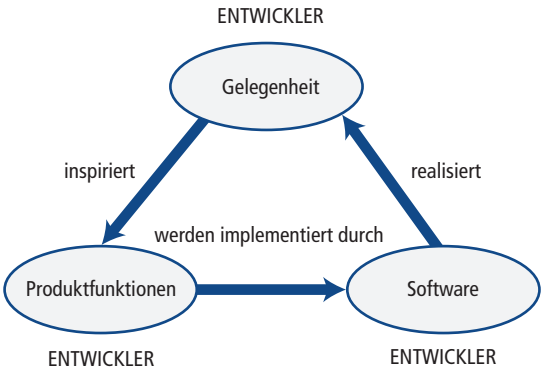

Abbildung 1.2: Produktbasiertes Software-Engineering

Softwareprojekte beziehen einen externen Auftraggeber oder Kunden mit ein, der über die Funktionalität des Systems entscheidet und einen rechtsverbindlichen Vertrag mit der Softwareentwicklungsfirma abschließt. Die Softwareanforderungen, die die zu implementierende Software spezifizieren, werden auf Basis der Problemstellung des Kunden und der aktuellen Prozesse erstellt. Ändert sich der Geschäftsbetrieb, dann muss sich die unterstützende Software ändern. Das Unternehmen, das die Software nutzt, entscheidet darüber und zahlt für die Änderungen. Software hat oft eine lange Lebensdauer und die Kosten für Änderungen an großen Systemen nach der Auslieferung übersteigen in der Regel die eigentlichen Softwareentwicklungskosten.

Softwareprodukte werden dagegen anders spezifiziert und entwickelt. Es gibt keinen externen Kunden, der Anforderungen erstellt, welche definieren, was die Software tun muss. Der Softwareentwickler entscheidet über die Eigenschaften des Produkts, wann neue Versionen zur Verfügung gestellt werden, auf welchen Plattformen die Software implementiert wird usw. Natürlich werden die Bedürfnisse potenzieller Kunden der Software berücksichtigt, aber die Käufer können nicht darauf bestehen, dass die Software bestimmte Funktionen oder Attribute enthält. Das Unternehmen, das die Software entwickelt, entscheidet, wann Änderungen an der Software vorgenommen werden und wann diese an die Benutzer weitergegeben werden.
Da die Entwicklungskosten auf einen viel größeren Kundenkreis verteilt sind, ist produktbasierte Software in der Regel für den einzelnen Kunden günstiger als Individualsoftware. Allerdings müssen die Softwarekäufer ihre Arbeitsweise an die Software anpassen, da sie nicht im Hinblick auf ihre speziellen Bedürfnisse entwickelt wurde. Und da die Kontrolle über Änderungen beim Entwickler und nicht beim Benutzer liegt, besteht die Gefahr, dass der Entwickler die Unterstützung der Software einstellt. Dann müssen die Kunden ein alternatives Produkt finden.
Der Ausgangspunkt für die Entwicklung eines Softwareprodukts ist eine Gelegenheit, die ein Unternehmen erkannt hat, um ein wettbewerbsfähiges kommerzielles Produkt zu entwickeln. Dies kann eine originelle Idee sein, wie z. B. die Idee von Airbnb, Unterkünfte zu teilen; eine Verbesserung gegenüber bestehenden Systemen, wie z. B. einem cloudbasierten Buchhaltungssystem; oder eine Generalisierung eines Systems, das für einen bestimmten Kunden entwickelt wurde, wie z. B. ein Asset-Management-System.
Da der Produktentwickler dafür verantwortlich ist, die Gelegenheit zu erkennen, kann er auch entscheiden, welche Funktionen in das Softwareprodukt aufgenommen werden. Diese Merkmale sollen für potenzielle Kunden ansprechend sein, sodass die Software eine Überlebenschance am Markt hat.
Neben den in ▶ Abbildung 1.1 und ▶ Abbildung 1.2 dargestellten Differenzen gibt es zwei weitere wichtige Unterschiede zwischen projektbezogenem und produktbezogenem Software-Engineering:

1. Softwareproduktfirmen können entscheiden, wann sie ihr Produkt verändern oder vom Markt nehmen. Verkauft sich ein Produkt nicht gut, so kann das Unternehmen die Kosten senken, indem es dessen Entwicklung einstellt. Kundenspezifische Software, die in einem Softwareprojekt entwickelt wird, hat in der Regel eine lange Lebensdauer und muss während der gesamten Zeit unterstützt werden. Der Kunde bezahlt für den Support und entscheidet, wann und ob dieser endet.

2. Bei den meisten Produkten ist es entscheidend, das Produkt schnell zum Kunden zu bringen. Es kann passieren, dass exzellente Produkte scheitern, weil ein qualitativ schlechteres Produkt zuerst auf den Markt kommt und die Kunden dieses Produkt kaufen. Üblicherweise zögern die Käufer, zu einem anderen Produkt zu wechseln, wenn sie erst einmal Zeit und Geld in ihre anfängliche Wahl investiert haben.

Die schnelle Markteinführung des Produkts ist für alle Arten von Produkten wichtig, von kleinen mobilen Anwendungen bis hin zu unternehmensweit eingesetzten Produkten wie Microsoft Word. Das bedeutet, dass Engineering-Techniken, die auf schnelle Softwareentwicklung ausgerichtet sind (agile Methoden), in der Produktentwicklung universell eingesetzt werden. Ich komme auf agile Methoden und ihre Rolle in der Produktentwicklung in ▶ Kapitel 2 zu sprechen.

Wenn Sie etwas über Softwareprodukte lesen, stoßen Sie möglicherweise auf zwei weitere Begriffe: „Softwareproduktlinien“ und „Plattformen“ (▶ Tabelle 1.1). Softwareproduktlinien sind Systeme, die so konzipiert sind, dass sie an die spezifischen Bedürfnisse der Kunden angepasst werden können, indem Teile des Quellcodes geändert werden. Plattformen bieten eine Reihe von Funktionen, mit denen eine neue Funktionalität erstellt werden kann. Allerdings müssen Sie immer innerhalb der von den Plattformanbietern festgelegten Einschränkungen arbeiten.

Technologie	Beschreibung
Softwareproduktlinie	Eine Reihe von Softwareprodukten, die einen gemeinsamen Kern haben. Jedes Element der Produktlinie beinhaltet kundenspezifische Anpassungen und Ergänzungen. Softwareproduktlinien können verwendet werden, um ein Individualsystem für einen Kunden mit speziellen Anforderungen zu implementieren, die von einem generischen Produkt nicht erfüllt werden können. So kann beispielsweise ein Unternehmen, das Kommunikationssoftware für Rettungsdienste bereitstellt, über eine Softwareproduktlinie verfügen, bei der das Kernprodukt grundlegende Kommunikationsdienste wie Empfang und Protokollierung von Anrufen, Einleitung einer Notfallmaßnahme, Weitergabe von Informationen an Rettungsfahrzeuge usw. umfasst. Jeder Kunde könnte jedoch andere Funkgeräte verwenden und die Fahrzeuge könnten unterschiedlich ausgestattet sein. Das Kernprodukt muss auf jeden Kunden so angepasst werden, dass es mit den jeweils vorhandenen Geräten zusammenarbeitet.
Plattform	Ein Softwareprodukt (oder Software-/Hardwareprodukt), das Funktionalität beinhaltet, sodass neue Anwendungen darauf aufgebaut werden können. Ein Beispiel für eine Plattform, die Sie wahrscheinlich nutzen, ist Facebook. Es bietet eine umfangreiche Palette von Produktfunktionen, unterstützt aber auch die Erstellung von „Facebook-Apps“. Diese fügen neue Funktionen hinzu, die von einem Unternehmen oder einer Facebook-Interessengruppe genutzt werden können.

Tabelle 1.1: Softwareproduktlinien und Plattformen

Als die ersten Softwareprodukte ausgeliefert wurden, kamen sie auf einer CD und wurden von den Kunden auf ihren Rechnern installiert. Die Software lief auf diesen Computern, auf dem auch die Benutzerdaten gespeichert wurden. Es gab keinerlei Kommunikation zwischen den Computern der Benutzer und den Rechnern des Anbieters. Heute können Kunden die Produkte entweder von einem App-Store oder von der Website des Anbieters herunterladen.
Einige Produkte basieren noch immer auf einem eigenständigen Ausführungsmodell, bei dem alle Berechnungen auf den Computern der Produktinhaber durchgeführt werden. Durch die allgegenwärtige Hochgeschwindigkeitsvernetzung stehen jetzt jedoch alternative Ausführungsmodelle zur Verfügung. In diesen Modellen fungieren die Rechner der Produktinhaber als Client, die Ausführung und Datenspeicherung erfolgt teilweise oder vollständig auf den Servern des Herstellers (▶ Abbildung 1.3).

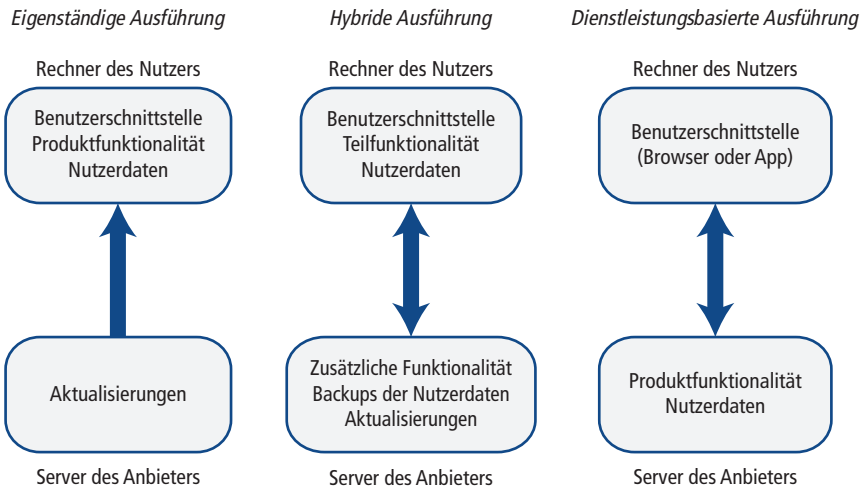

Abbildung 1.3: Software-Ausführungsmodelle

Es gibt zwei Alternativen zu eigenständigen Softwareprodukten:

1. *Hybride Produkte:* Einige Funktionen sind auf dem Rechner des Benutzers und andere auf den Servern des Produktanbieters implementiert, auf die über das Internet zugegriffen wird. Viele Telefon-Apps sind hybride Produkte, bei denen die rechenintensive Verarbeitung auf entfernte Server ausgelagert wird.

2. *Dienstleistungsbasierte Produkte:* Der Zugriff auf Anwendungen erfolgt über das Internet von einem Webbrowser oder einer App aus. Eventuell gibt es einige lokale Verarbeitungen mit Javascript, doch der Großteil der Berechnungen wird auf entfernten Servern durchgeführt. Immer mehr Softwareproduktfirmen stellen von Produkten auf Dienstleistungen um, weil dies die Produktpflege vereinfacht und neue Geschäftsmodelle wie Pay-as-you-go möglich macht. Ich behandele serviceorientierte Systeme in den ▶Kapiteln 5 und 6.

Das Hauptmerkmal der Produktentwicklung ist, wie schon gesagt, dass es keinen externen Kunden gibt, der Softwareanforderungen generiert und der für die Software bezahlt. Dies gilt auch für einige andere Arten der Softwareentwicklung:

1. *Studentische Projekte:* Im Rahmen einer Informatik- oder Ingenieurveranstaltung bekommen die Studierenden die Aufgabe, in Gruppen zusammenzuarbeiten, um Software zu entwickeln. Die Gruppe ist verantwortlich für die Entscheidung über die Merkmale des Systems und wie die Zusammenarbeit organisiert wird, um diese Merkmale zu implementieren.

2. *Forschungssoftware:* Die Software wird von einem Forscherteam zur Unterstützung ihrer Arbeit entwickelt. So hängt die Klimaforschung beispielsweise von groß angelegten Klimamodellen ab, die von Forschern entworfen und in Software realisiert werden. In kleinerem Umfang kann eine Ingenieurgruppe Software entwickeln, um die Eigenschaften des von ihnen verwendeten Materials zu modellieren.

3 *Interne Werkzeugentwicklung:* Ein Softwareentwicklerteam entscheidet, dass es einige spezielle Werkzeuge benötigt, die seine Arbeit unterstützen. Sie spezifizieren und implementieren diese Werkzeuge als „interne" Produkte.

Sie können die hier von mir erläuterten Techniken der Produktentwicklung für jede Art von Softwareentwicklung einsetzen, die nicht von externen Kundenanforderungen gesteuert ist.

Es existiert die verbreitete Meinung, dass die Entwicklung von Softwareprodukten schlicht eine Form fortgeschrittener Programmierung sei und dass traditionelles Software-Engineering dafür irrelevant sei. Alles, was Sie wissen müssten, wäre, wie man eine Programmiersprache sowie die Frameworks und Bibliotheken für diese Sprache verwendet. Das ist jedoch ein Irrglaube und ich habe dieses Buch geschrieben, um zu erklären, welche Aktivitäten abgesehen vom Programmieren meiner Meinung nach für die Entwicklung hochwertiger Softwareprodukte unerlässlich sind.

Wenn Ihr Produkt ein Erfolg werden soll, dürfen Sie nicht nur über die Programmierung nachdenken. Sie müssen versuchen zu verstehen, was Ihre Kunden benötigen und wie potenzielle Benutzer mit Ihrer Software arbeiten können. Sie müssen die Gesamtstruktur Ihrer Software (Softwarearchitektur) gestalten und sich mit Technologien wie Cloud-Computing und Security-Engineering auskennen. Sie müssen professionelle Techniken zur Verifizierung und zum Testen Ihrer Software einsetzen sowie Systeme zur Versionsverwaltung, um den Überblick über eine sich ändernde Codebasis zu behalten.

Außerdem müssen Sie sich mit dem Geschäftsszenario für Ihr Produkt auseinandersetzen. Sie müssen Ihr Produkt verkaufen, wenn Sie überleben wollen. Ein Geschäftsszenario zu erstellen beinhaltet vielleicht eine Marktforschung, eine Analyse der Wettbewerber sowie ein Verständnis darüber, wie Ihre Zielkunden leben und arbeiten. In diesem Buch geht es jedoch um die Technik, nicht um die wirtschaftlichen Aspekte, daher behandele ich hier keine geschäftlichen und kommerziellen Themen.

1.1 Die Produktvision

Ihr Ausgangspunkt für die Entwicklung eines Produkts sollte eine informelle „Produktvision" sein. Eine Produktvision ist eine einfache und prägnante Aussage, die den Kern des zu entwickelnden Produkts definiert. Die Produktvision erklärt, wie sich das Produkt von anderen Konkurrenzprodukten unterscheidet. Sie dient als Grundlage für die Entwicklung einer detaillierteren Beschreibung der Funktionen und Attribute des Produkts. Wenn neue Funktionen vorgeschlagen werden, sollten Sie diese mit der Vision vergleichen, um sicherzustellen, dass die Neuerungen dazu passen.

Die Produktvision sollte drei grundlegende Fragen beantworten:

1 *Was* für ein Produkt wollen Sie entwickeln? Was unterscheidet dieses Produkt von Konkurrenzprodukten?

2 *Wer* sind die Zielpersonen und Kunden für das Produkt?

3 *Warum* sollten Kunden dieses Produkt kaufen?

Die Notwendigkeit der ersten Frage liegt auf der Hand – bevor Sie beginnen, müssen Sie wissen, was Ihr Ziel ist. Die anderen Fragen betreffen die Marktfähigkeit des Produkts. Die meisten Produkte sind für Kunden außerhalb des Entwicklerteams

bestimmt. Sie müssen deren Hintergrund verstehen, um ein rentables Produkt zu schaffen, das diese Kunden attraktiv finden und kaufen wollen.
Wenn Sie im Web nach „Produktvision“ suchen, finden Sie mehrere Varianten dieser Fragen sowie Vorlagen zur Darstellung der Produktvision. Jede dieser Vorlagen kann verwendet werden. Die Vorlage, die mir gefällt, stammt aus dem Buch *Crossing the Chasm* von Geoffrey Moore.[1] Moore schlägt vor, einen strukturierten Ansatz zu verwenden, um die Produktvision basierend auf Schlüsselwörtern zu schreiben:

- FÜR (Zielkunde)
- DER/DIE (Aussage zum Bedürfnis oder der Gelegenheit)
- IST DER/DIE/DAS (Produktname) EIN (Produktkategorie)
- DER/DIE/DAS (Hauptvorteil, zwingender Kaufgrund)
- ANDERS ALS (wichtigster Wettbewerber)
- UNSER PRODUKT (Aussage zur wichtigsten Differenzierung)

Joel Spolsky gibt in seinem Blog „Joel on Software“ ein Beispiel für ein Produkt, das mit dieser Visionsvorlage beschrieben wird:[2]

> *FÜR die Marketing- und Vertriebsabteilungen eines mittelständischen Unternehmens, DIE grundlegende CRM-Funktionen benötigen, IST DER CRM-Innovator EIN webbasierter Service, DER Absatzverfolgung, Leadgenerierung und Funktionen zur Unterstützung der Vertriebsmitarbeiter bietet, der die Kundenbeziehungen an kritischen Berührungspunkten verbessert. ANDERS ALS andere Dienstleistungen oder Softwareprodukte bietet UNSER PRODUKT einen sehr leistungsfähigen Service zu moderaten Preisen.*

Wie Sie sehen, beantwortet diese Vision die Schlüsselfragen, die ich oben identifiziert habe:

1. *Was?* – Ein webbasierter Service, der Funktionen zur Vertriebsverfolgung, Leadgenerierung und Unterstützung von Vertriebsmitarbeitern bietet. Die Informationen können verwendet werden, um die Beziehungen zu den Kunden zu verbessern.

2. *Wer?* – Das Produkt richtet sich an mittelständische Unternehmen, die eine einheitliche Software für ihr Customer-Relationship-Management benötigen.

3. *Warum?* – Die wichtigste Produktunterscheidung besteht darin, dass es einen leistungsfähigen Service zu moderaten Preisen bietet, also billiger als alternative Produkte sein wird.

Um Softwareproduktvisionen ranken sich viele Mythen. Bei erfolgreichen Softwareprodukten für Endkunden präsentieren die Medien gerne Visionen so, als ob sie aus einem „Heureka-Moment“ heraus geboren wären, als die Unternehmensgründer eine „Hammeridee“ hatten, die die Welt verändert. Diese Sichtweise vereinfacht den Aufwand und das Experimentieren stark, die beim Herausbilden einer Produktidee anfallen. Produktvisionen für erfolgreiche Produkte entstehen in der Regel aus einer Menge Arbeit und vielen Diskussionen. Eine erste Idee wird schrittweise verfeinert, indem

1 Geoffrey Moore, *Crossing the Chasm: Marketing and selling technology products to mainstream customers*. Capstone Trade Press, 1998.
2 J. Spolsky, Product Vision, 2002; *http://www.joelonsoftware.com/articles/JimHighsmithonProductVisi.html*.

mehr Informationen gesammelt werden und das Entwicklerteam über die Durchführbarkeit der Produktumsetzung diskutiert. Verschiedene Informationsquellen tragen zur Produktvision bei (▶ Tabelle 1.2).

Informationsquelle	Erläuterung
Branchenerfahrung	Die Produktentwickler arbeiten eventuell in einem bestimmten Bereich (z. B. Marketing und Vertrieb) und wissen, welche Softwareunterstützung sie benötigen. Sie könnten über die Mängel in der von ihnen verwendeten Software frustriert sein und die Möglichkeiten für ein besseres System sehen.
Produkterfahrung	Anwender bestehender Software (z. B. Textverarbeitungssoftware) sehen vielleicht einfachere und bessere Möglichkeiten, vergleichbare Funktionen bereitzustellen, und schlagen ein neues System vor, das diese implementiert. Neue Produkte könnten die Vorteile der neuesten technologischen Entwicklungen wie Sprachschnittstellen nutzen.
Kundenerfahrung	Die Softwareentwickler könnten umfangreiche Gespräche mit potenziellen Kunden des Produkts führen, um zu verstehen, mit welchen Problemen diese konfrontiert sind; welche Einschränkungen, wie z. B. Interoperabilität, ihre Flexibilität beim Kauf neuer Software begrenzen; und welche entscheidende Merkmale die benötigte Software haben sollte.
Prototypen und „Herumspielen"	Entwickler haben vielleicht eine Idee für eine Software, müssen aber ein besseres Verständnis für die Idee selbst entwickeln sowie dafür, was dazugehört, um diese Idee zu einem Produkt werden zu lassen. Sie könnten ein Prototypsystem als Versuch entwickeln und dieses nutzen, um mit Ideen und Variationen „herumzuspielen".

Tabelle 1.2: Informationsquellen zur Entwicklung einer Produktvision

1.1.1 Ein Beispiel für eine Vision

Die Leser dieses Buchs haben als Studenten oder Schüler möglicherweise Lernplattformen (auch virtuelle Lernumgebungen genannt, *Virtual Learning Environment*, VLE) wie Moodle verwendet. Lehrer setzen diese Lernplattformen zum Austeilen von Unterrichtsmaterialien und Aufgaben ein. Die Schüler oder Studenten können die Materialien herunterladen und bearbeitete Aufgaben hochladen. Obwohl der Name darauf hindeutet, dass sich Lernplattformen auf das Lernen konzentrieren, sind sie in Wirklichkeit darauf ausgerichtet, die Verwaltung des Lernens und nicht das Lernen selbst zu unterstützen. Sie bieten einige Funktionen für die Lernenden, aber sie sind keine offenen Lernumgebungen, die auf die Bedürfnisse eines bestimmten Lehrers zugeschnitten und angepasst werden können.

Vor einigen Jahren habe ich an der Entwicklung einer digitalen Umgebung zur Lernförderung gearbeitet. Dieses Produkt war nicht einfach eine weitere Lernumgebung, sondern sollte den Lernprozess flexibel unterstützen. Unser Team sah sich bestehende Lernplattformen an und sprach mit Lehrern und Schülern, die diese verwendeten. Wir besuchten verschiedene Schultypen, von Kindergärten bis hin zu Hochschulen, um zu untersuchen, wie sie Lernumgebungen nutzten und wie Lehrer außerhalb dieser

Umgebungen mit Software experimentierten. Wir führten ausführliche Gespräche mit Lehrern darüber, was sie mit einer digitalen Lernumgebung erreichen möchten. Schließlich kamen wir zu der in ▶ Tabelle 1.3 dargestellten Vision.

FÜR Lehrer und Pädagogen, *DIE* eine Möglichkeit benötigen, Schülern bei der Nutzung webbasierter Lernressourcen und -anwendungen zu helfen, *IST DAS* iLearn-System *EINE* offene Lernumgebung, *DIE* es Lehrern ermöglicht, die von Klassen und Schülern verwendeten Ressourcen für diese Schüler und Klassen leicht selbst zu konfigurieren. *ANDERS ALS* virtuelle Lernumgebungen wie Moodle liegt der Schwerpunkt von iLearn auf dem Lernprozess und nicht auf der Verwaltung von Materialien, Aufgaben und Kursarbeiten. *UNSER PRODUKT* erlaubt es Lehrern, fach- und altersspezifische Umgebungen für ihre Schüler zu erstellen, wobei sie alle geeigneten webbasierten Ressourcen wie Videos, Simulationen und schriftliche Materialien verwenden können.
Schulen und Universitäten sind die Zielkunden für das *iLearn-System*, welches das Lernerlebnis der Schüler zu relativ niedrigen Kosten deutlich verbessern wird. Es wird Lernanalysen sammeln und verarbeiten, die die Kosten für die Fortschrittskontrolle und Berichterstattung senken werden.

Tabelle 1.3: Eine ausformulierte Vision für das iLearn-System

Im Bildungsbereich sind die Lehrer und Schüler, die Lernumgebungen nutzen, nicht für den Kauf der Software verantwortlich. Der Käufer ist eine Schule, eine Universität oder ein Ausbildungszentrum. Die Person, die für den Kauf zuständig ist, muss die Vorteile für die Organisation kennen. Daher haben wir den letzten Absatz der ausformulierten Vision in ▶ Tabelle 1.3 hinzugefügt, um zu verdeutlichen, dass es sowohl Vorteile für die Institution als auch für die einzelnen Lernenden gibt.

1.2 Softwareproduktmanagement

Softwareproduktmanagement ist eine Geschäftstätigkeit, die sich auf die Softwareprodukte konzentriert, die von dem Unternehmen entwickelt und vertrieben werden. Produktmanager übernehmen die Gesamtverantwortung für das Produkt und sind an Planung, Entwicklung und Marketing beteiligt. Sie bilden die Schnittstelle zwischen dem Softwareentwicklerteam, der weiteren Organisation sowie den Kunden des Produkts. Die Produktmanager sollten vollwertige Mitglieder des Entwicklerteams sein, damit sie über Geschäfts- und Kundenanforderungen mit den Softwareentwicklern kommunizieren können.

Softwareproduktmanager sind in allen Phasen des Produktlebenszyklus involviert – von der ersten Konzeption über die Visionsentwicklung und Implementierung bis hin zum Marketing. Zuletzt treffen sie die Entscheidung darüber, wann das Produkt vom Markt genommen werden sollte. In mittleren und großen Softwareunternehmen ist die Rolle des Produktmanagers eventuell als eine eigene Stelle ausgelegt; in kleineren Softwareunternehmen wird diese Rolle wahrscheinlich mit anderen technischen oder betrieblichen Rollen kombiniert sein.

Die Aufgabe des Produktmanagers ist es, den Blick mehr nach außen auf die bestehenden und potenziellen Kunden des Produkts zu richten als sich auf die zu entwickelnde Software zu konzentrieren. Es passiert dem Entwicklerteam allzu leicht, sich in den Details von „coolen Features“ der Software zu verfangen, die den meisten Kunden wahrscheinlich egal sind. Damit ein Produkt erfolgreich ist, muss der Produktmanager sicherstellen, dass das Entwicklerteam Funktionen implementiert, die dem Kunden

einen echten Mehrwert bieten, und nicht nur Funktionen, die technisch interessant sind.
In einem Blogeintrag erklärt Martin Eriksson,[3] dass sich Produktmanager mit Fragen zum Geschäftsbetrieb, zur Technologie und zum Nutzererlebnis (*User Experience*, UX) befassen müssen. In ▶ Abbildung 1.4, die auf der Grundlage von Erikssons Diagramm erstellt ist, werden diese vielfältigen Belange veranschaulicht.

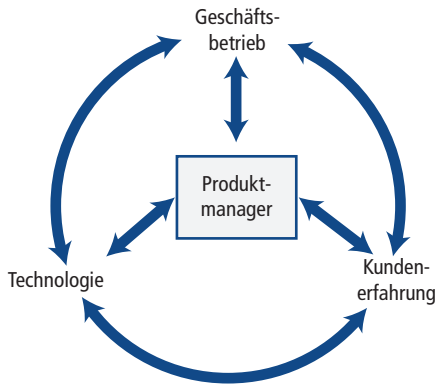

Abbildung 1.4: Belange des Produktmanagements

Produktmanager müssen Generalisten mit sowohl technischen als auch kommunikativen Fähigkeiten sein. Geschäfts-, Technologie- und Kundenfragen sind voneinander abhängig und die Produktmanager müssen diese allesamt berücksichtigen:

1. *Geschäftsanforderungen:* Produktmanager müssen sicherstellen, dass die zu entwickelnde Software die Geschäftsziele und -vorstellungen sowohl der Softwareproduktfirma als auch ihrer Kunden erfüllt. Sie müssen die Anliegen und Bedürfnisse der Kunden und des Entwicklerteams an die Leiter des Produktgeschäfts kommunizieren. Sie arbeiten mit Führungskräften und Marketingmitarbeitern zusammen, um einen Releaseplan für das Produkt zu erstellen.

2. *Technologische Beschränkungen:* Produktmanager müssen Entwickler auf Technologiefragen aufmerksam machen, die für Kunden wichtig sind. Diese können den Zeitplan, die Kosten und die Funktionalität des zu entwickelnden Produkts beeinflussen.

3. *Kundenerfahrung:* Produktmanager sollten in regelmäßigem Austausch mit den Kunden stehen, um zu verstehen, was diese von einem Produkt erwarten, um welche Arten von Benutzern es sich handelt und wie deren Hintergrund aussieht, und wie das Produkt verwendet werden kann. Ihre Erfahrung mit den Potenzialen der Kunden ist ein entscheidender Faktor für die Entwicklung des Designs der Benutzeroberfläche. Produktmanager könnten auch Kunden in Alpha- und Betatests des Produkts einbeziehen.

Da der Fokus in diesem Buch mehr auf den technischen Aspekten liegt, gehe ich nicht im Detail auf die geschäftliche Rolle von Produktmanagern oder deren Rolle in Berei-

3 Basierend auf M. Eriksson, „What, exactly, is a Product Manager?“; *http://mindtheproduct.com/2011/10/what-exactly-is-a-product-manager* (2011).

chen wie Marktforschung und Finanzplanung ein. Stattdessen konzentriere ich mich auf deren Interaktionen mit dem Entwicklerteam. Produktmanager können in sieben Schlüsselbereichen mit dem Entwicklerteam interagieren (▶ Abbildung 1.5).

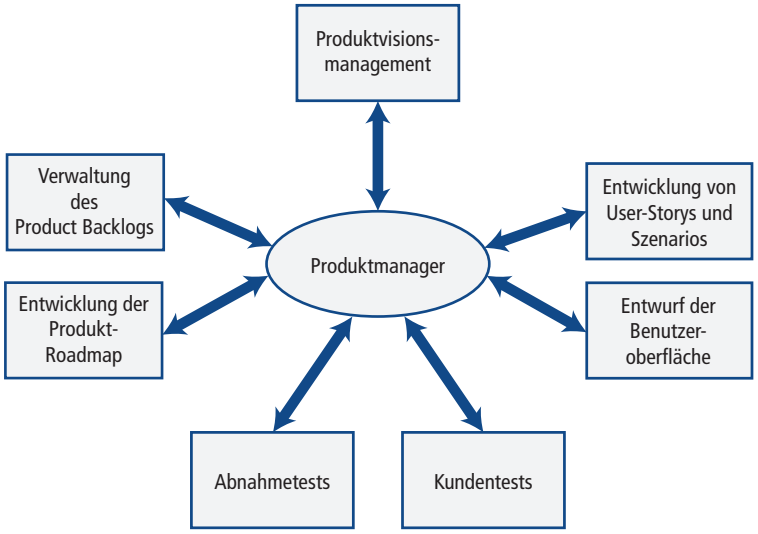

Abbildung 1.5: Technische Interaktionen der Produktmanager

1.2.1 Produktvisionsmanagement

Einige Autoren sagen, dass der Produktmanager für die Entwicklung der Produktvision verantwortlich sein sollte. Große Unternehmen mögen diesen Ansatz zwar verfolgen, aber in kleinen Softwareunternehmen ist er oft nicht praktikabel. In Start-ups ist die Quelle der Produktvision häufig eine originelle Idee der Unternehmensgründer. Diese Vision wird oft schon lange entwickelt, bevor jemand an die Ernennung eines Produktmanagers denkt.

Natürlich ist es für Produktmanager sinnvoll, bei der Entwicklung der Produktvision die Führung zu übernehmen. Sie sollten in der Lage sein, Markt- und Kundeninformationen in den Prozess einzubringen. Ich denke jedoch, dass alle Teammitglieder in die Entwicklung der Vision einbezogen werden sollten, damit jeder hinter dem steht, auf was man sich schließlich einigt. Wenn das Team die Vision „besitzt“, ist es wahrscheinlicher, dass alle bei der Arbeit an einem Strang ziehen, um diese Vision zu verwirklichen.

Eine Schlüsselrolle von Produktmanagern ist die Verwaltung der Produktvision. Während des Entwicklungsprozesses werden zwangsläufig Änderungen von Personen sowohl innerhalb als auch außerhalb des Entwicklerteams vorgeschlagen. Produktmanager müssen diese Änderungen bewerten und mit der Produktvision abgleichen. Sie müssen prüfen, ob die Änderungen nicht im Widerspruch zu den Ideen stehen, die in der Produktvision verankert sind. Die Produktmanager müssen auch sicherstellen, dass es keine „Visionsdrift“ gibt, das heißt, dass die Vision schrittweise erweitert wird und damit breiter und weniger fokussiert wird.

1.2.2 Entwicklung der Produkt-Roadmap

Eine Produkt-Roadmap ist ein Plan für die Entwicklung, das Release und die Vermarktung des Softwareprodukts. Es werden wichtige Produktziele und Meilensteine festgelegt, wie z.B. die Fertigstellung kritischer Funktionen, die Fertigstellung der ersten Version für die Anwendertests und so weiter. Dazu gehören Termine, an denen diese Meilensteine erreicht werden sollten, und Erfolgskriterien, die bei der Beurteilung helfen, ob die Projektziele erreicht wurden. Die Roadmap sollte einen Releaseplan enthalten, der zeigt, wann verschiedene Versionen der Software verfügbar sein werden und welche Hauptmerkmale jedes Release mit sich bringen wird.
Die Entwicklung der Produkt-Roadmap sollte zwar vom Produktmanager geleitet werden, muss aber auch das Entwicklerteam sowie Unternehmensleitung und Marketingmitarbeiter einbeziehen. Je nach Produktart sind eventuell wichtige Fristen einzuhalten, wenn das Produkt erfolgreich sein soll. Beispielsweise müssen viele Großunternehmen gegen Ende ihres Geschäftsjahres Entscheidungen zur Beschaffung treffen. Möchten Sie ein neues Produkt an solche Unternehmen verkaufen, dann muss es vor diesem Termin verfügbar sein.

1.2.3 Entwicklung von User-Storys und Szenarios

User-Storys und Szenarios werden häufig verwendet, um eine Produktvision zu verfeinern und um Merkmale des Produkts zu identifizieren. Es sind natürlichsprachige Beschreibungen davon, was Benutzer vielleicht mit einem Produkt machen wollen. Anhand dieser Informationen kann das Team entscheiden, welche Funktionen aufgenommen werden müssen und was diese Funktionen leisten sollen. Ich behandele User-Storys und Szenarios in ▶Kapitel 3.
Die Aufgabe des Produktmanagers ist es, die existierenden und potenziellen Kunden des Produkts zu verstehen. Produktmanager sollten daher die Ausarbeitung von Benutzerszenarios und User-Storys leiten, die auf der Kenntnis des Geschäftsfelds des Kunden basieren. Produktmanager sollten auch Szenarios und Storys, die von anderen Teammitgliedern vorgeschlagen wurden, an die Kunden weiterleiten, um zu überprüfen, ob sie mit der tatsächlichen Arbeitsweise der Zielgruppe des Produkts übereinstimmen.

1.2.4 Verwaltung des Product Backlogs

In der Produktentwicklung ist es wichtig, dass der Prozess durch ein „Product Backlog“ gesteuert wird. Ein Product Backlog ist eine To-do-Liste, die festlegt, was getan werden muss, um die Produktentwicklung abzuschließen. Während des Entwicklungsprozesses wird das Backlog schrittweise ergänzt und verfeinert. In ▶Kapitel 2 erläutere ich, wie Product Backlogs in der Scrum-Methode verwendet werden.
Der Produktmanager spielt eine entscheidende Rolle bei der Auswahl der Backlog-Einträge, die bei der Entwicklung vorrangig behandelt werden sollten. Produktmanager helfen auch, umfangreiche Backlog-Einträge wie „Automatisches Speichern implementieren“ bei jeder Projektiteration weiter zu verfeinern. Werden Änderungsvorschläge gemacht, so obliegt es dem Produktmanager zu entscheiden, ob das Product Backlog neu sortiert werden sollte, um die vorgeschlagenen Änderungen zu priorisieren.

1.2.5 Abnahmetests

Bei Akzeptanz- oder Abnahmetests wird verifiziert, dass ein Software-Release die in der Produkt-Roadmap festgelegten Ziele erfüllt und dass das Produkt effizient und zuverlässig ist. Produktmanager sollten an der Entwicklung von Tests der Produkteigenschaften beteiligt sein, die widerspiegeln, wie Kunden das Produkt verwenden. Sie können Nutzungsszenarios durcharbeiten, um zu überprüfen, ob das Produkt für die Freigabe an Kunden bereit ist.
Akzeptanztests werden während der Produktentwicklung verfeinert und die Produkte müssen diese Tests bestehen, bevor sie an die Kunden ausgeliefert werden.

1.2.6 Kundentests

Beim Kundentest wird eine Version eines Produkts an bestehende und potenzielle Kunden ausgegeben, die dann Feedback über die Produktmerkmale, die Benutzerfreundlichkeit und die Anpassung des Produkts an ihr Unternehmen abgeben. Produktmanager sind an der Auswahl der Kunden beteiligt, die daran interessiert sein könnten, an Kundentests teilzunehmen, und arbeiten mit diesen dabei zusammen. Sie müssen sicherstellen, dass der Kunde das Produkt nutzen kann und dass durch die Ausführung der Kundentests hilfreiche Informationen für das Entwicklerteam gesammelt werden.

1.2.7 Entwurf der Benutzeroberfläche

Die Benutzeroberfläche (*User Interface*, UI) eines Produkts ist für die kommerzielle Akzeptanz eines Softwareprodukts entscheidend. Technisch hervorragende Produkte werden wahrscheinlich nicht kommerziell erfolgreich sein, wenn die Anwender sie als zu kompliziert empfinden oder wenn deren Benutzeroberfläche nicht kompatibel ist mit anderer Software, mit der sie arbeiten. Das UI-Design ist eine Herausforderung für kleine Entwicklerteams, da die meisten Anwender weniger technisch versiert sind als Softwareentwickler. Für Entwickler ist es oft schwierig, sich die Probleme vorzustellen, die Benutzer mit einem Softwareprodukt haben könnten.
Produktmanager sollten wissen, was die Benutzer einschränken könnte, und gegenüber dem Entwicklerteam als Ersatznutzer fungieren. Sie sollten die UI-Features während ihrer Entwicklung bewerten, um zu überprüfen, dass diese Funktionen nicht unnötig komplex sind oder die Benutzer dadurch zu einer unnatürlichen Arbeitsweise gezwungen werden. Produktmanager könnten organisieren, dass potenzielle Benutzer die Software ausprobieren, ihre Benutzeroberfläche kommentieren und bei der Gestaltung von Fehlermeldungen und einem Hilfesystem assistieren.

1.3 Erstellen von Prototypen des Produkts

Bei der Erstellung eines Produktprototyps wird eine frühe Version eines Produkts entwickelt, um Ideen zu testen und um sich und die Geldgeber des Unternehmens davon zu überzeugen, dass das Produkt echtes Marktpotenzial hat. Sie benutzen einen Produktprototyp, um die Machbarkeit dessen zu prüfen, was Sie damit vorhaben, und um Ihre Software potenziellen Kunden und Geldgebern vorzuführen. Prototypen können

Sie auch dabei unterstützen, eine Vorstellung davon zu gewinnen, wie Sie die endgültige Version Ihres Produkts organisieren und strukturieren können.
Auch wenn Sie eine inspirierende Produktvision schreiben, kann es sein, dass Ihre potenziellen Nutzer sich nur dann richtig auf Ihr Produkt einlassen, wenn sie eine lauffähige Version Ihrer Software sehen. Dann können die Anwender aufzeigen, was ihnen gefällt und was nicht, und neue Features vorschlagen. Risikokapitalgeber, an die Sie sich vielleicht wegen einer Finanzierung wenden, bestehen in der Regel darauf, einen Produktprototyp zu sehen, bevor sie sich verpflichten, ein Start-up-Unternehmen zu unterstützen. Der Prototyp spielt eine entscheidende Rolle dabei, Investoren davon zu überzeugen, dass Ihr Produkt kommerzielles Potenzial hat.
Ein Prototyp kann auch dazu beitragen, grundlegende Softwarekomponenten oder -dienste zu identifizieren und Technologien zu testen. Möglicherweise stellen Sie fest, dass die Technologie, die Sie verwenden wollten, unzureichend ist und dass Sie Ihre Ideen zur Implementierung der Software überarbeiten müssen. Beispielsweise könnten Sie herausfinden, dass das Design, das Sie für den Prototyp gewählt haben, nicht die erwartete Belastung des Systems aushält, sodass Sie die gesamte Produktarchitektur neu gestalten müssen.
Der Bau eines Prototyps sollte das Erste sein, was Sie bei der Entwicklung eines Softwareprodukts tun. Ihr Ziel sollte es sein, eine funktionierende Version Ihrer Software zu haben, die verwendet werden kann, um deren wichtigste Funktionen zu demonstrieren. Ein kurzer Entwicklungszyklus ist entscheidend; Sie sollten anstreben, in vier bis sechs Wochen ein vorzeigbares System zum Laufen zu bringen. Natürlich dürfen Sie es dabei an der ein oder anderen Stelle nicht so genau nehmen: So könnten Sie beispielsweise Probleme wie Zuverlässigkeit und Performanz ignorieren und mit einer rudimentären Benutzeroberfläche arbeiten.
Manchmal ist das Prototyping ein zweistufiger Prozess:

1. *Demonstration der Machbarkeit:* Sie erstellen ein ausführbares System, das die neuen Ideen in Ihrem Produkt demonstriert. Die Ziele in dieser Phase sind zum einen zu sehen, ob Ihre Ideen tatsächlich funktionieren, und zum anderen den Geldgebern und der Unternehmensleitung zu zeigen, dass Ihre Produkteigenschaften besser sind als die der Wettbewerber.

2. *Kundenvorführung:* Ausgehend von einem bestehenden Prototyp, der zur Demonstration der Machbarkeit erstellt wurde, erweitern Sie diesen mit Ihren Ideen für kundenspezifische Funktionen und deren möglicher Realisation. Bevor Sie einen Kundenprototyp entwickeln, müssen Sie einige Anwenderstudien durchführen und eine klare Vorstellung von Ihren potenziellen Anwendern und Nutzungsszenarios haben. In ▶Kapitel 3 erkläre ich, wie man Benutzerpersonas und Anwendungsszenarios entwickelt.

Bei der Entwicklung eines Prototyps sollten Sie stets Technologien verwenden, mit denen Sie sich auskennen, damit Sie keine Zeit damit verschwenden, eine neue Sprache oder ein neues Framework zu lernen. Es muss keine robuste Softwarearchitektur entworfen werden. Sie können Sicherheitsfunktionen und Prüfcode weglassen, mit denen die Zuverlässigkeit der Software gewährleistet werden soll. Ich empfehle jedoch, bei Prototypen immer automatisiertes Testen und Codemanagement zu verwenden. Diese Themen werden in den ▶Kapiteln 9 und 10 behandelt.
Falls Sie Software entwickeln, ohne dafür einen externen Kunden zu haben (z.B. Software für eine Forschungsgruppe), dann kann es sein, dass Sie tatsächlich nur ein Pro-

totypsystem benötigen. Während Ihr Verständnis des Problems reift, können Sie den Prototyp weiterentwickeln und verfeinern. Sobald es jedoch externe Benutzer Ihrer Software gibt, sollten Sie Ihren Prototyp immer als ein „Wegwerfsystem“ betrachten. Es ist unvermeidlich, dass Sie Kompromisse eingehen und Abkürzungen nehmen, um die Entwicklung zu beschleunigen – dies führt zu Prototypen, die sich immer schwerer ändern und weiterentwickeln lassen, wenn neue Funktionen integriert werden sollen. Nahezu unmöglich könnte es werden, Sicherheitsaspekte und Zuverlässigkeit nachträglich einzufügen.

Zusammenfassung

- Softwareprodukte sind Softwaresysteme mit allgemeinen Funktionen, die vermutlich für einen breiten Kundenkreis von Nutzen sind.
- Im Bereich des produktbasierten Software-Engineerings ist dasselbe Unternehmen sowohl für die Entscheidung über die Merkmale, die Teil des Produkts sein sollen, als auch für die Entscheidung über die Implementierung dieser Merkmale verantwortlich.
- Softwareprodukte können als eigenständige Produkte, die auf den Computern des Kunden laufen, als Hybridprodukte oder als dienstleistungsbasierte Produkte ausgeliefert werden. Bei Hybridprodukten sind einige Funktionen lokal implementiert und auf andere wird über das Internet zugegriffen. Bei dienstleistungsbasierten Produkten sind alle Funktionen per Fernzugriff zugänglich.
- Eine Produktvision beschreibt kurz und bündig, was zu entwickeln ist, wer die Zielkunden für das Produkt sind und warum Kunden das von Ihnen entwickelte Produkt kaufen sollten.
- Branchenerfahrung, Produkterfahrung, Kundenerfahrung und ein experimenteller Softwareprototyp können allesamt zur Entwicklung der Produktvision beitragen.
- Zu den Hauptaufgaben der Produktmanager gehören die Verantwortung für die Produktvision, die Entwicklung einer Produkt-Roadmap, die Erstellung von User-Storys und Szenarios, die Verwaltung des Product Backlogs, die Durchführung von Kunden- und Akzeptanztests sowie die Gestaltung der Benutzeroberfläche.
- Produktmanager arbeiten an der Schnittstelle zwischen dem Unternehmen, dem Softwareentwicklerteam und den Produktkunden. Sie erleichtern die Kommunikation zwischen diesen Gruppen.
- Sie sollten immer einen Produktprototyp entwickeln, um Ihre eigenen Ideen zu verfeinern und potenziellen Kunden die geplanten Produkteigenschaften vorzuführen.

Ergänzende Literatur

„What is Product Line Engineering?": Dieser Artikel und die beiden verlinkten Artikel geben einen Überblick über das Software-Engineering für Produktlinien und heben die Unterschiede zwischen der Entwicklung von Produktlinien und der von Softwareprodukten hervor (Biglever Software, 2013).
http://www.productlineengineering.com/overview/what-is-ple.html

„Building Software Products vs Platforms": Dieser Blogeintrag erklärt kurz die Unterschiede zwischen einem Softwareprodukt und einer Softwareplattform (B. Algaze, 2016).
https://blog.frogslayer.com/building-software-products-vs-platforms/

„Product Vision": Dies ist ein älterer Artikel, der aber eine ausgezeichnete Zusammenfassung darüber gibt, was mit einer Produktvision gemeint ist und warum sie wichtig ist (J. Spolsky, 2002).
http://www.joelonsoftware.com/articles/JimHighsmithonProductVisi.html

Agile Product Management with Scrum: Ich vermeide es im Allgemeinen, Bücher über Produktmanagement zu empfehlen, da sie für die meisten Leser zu detailliert sind. Ein Blick in dieses Buch lohnt sich jedoch wegen seines Fokus auf Software und deren Integration mit der agilen Scrum-Methode, die ich in ▶Kapitel 2 behandele. Es ist ein kurzes Buch, das eine prägnante Einführung in das Produktmanagement enthält und die Erstellung einer Produktvision diskutiert (R. Pichler, 2010, Addison-Wesley).

Der Blog des Autors enthält ebenfalls Artikel zum Thema Produktmanagement.
http://www.romanpichler.com/blog/romans-product-management-framework/

„What, Exactly, is a Product Manager?": Dieser ausgezeichnete Blogeintrag erklärt, warum es wichtig ist, dass Produktmanager an der Schnittstelle von Geschäftswelt, Technologie und Anwendern arbeiten (M. Eriksson, 2011).
http://www.mindtheproduct.com/2011/10/what-exactly-is-a-product-manager/

Präsentationen, Videos und Links

https://iansommerville.com/engineering-software-products/static/presentations-videos-and-links/software-products

Lösungshinweise

Übungen

1. Beschreiben Sie kurz die grundlegenden Unterschiede zwischen projektbezogener und produktbezogener Softwareentwicklung.
2. Worin bestehen drei wichtige Unterschiede zwischen Softwareprodukten und Softwareproduktlinien?
3. Identifizieren Sie anhand der exemplarischen Produktvision für das iLearn-System das WAS, WER und WARUM für dieses Softwareprodukt.
4. Warum müssen Softwareproduktmanager Generalisten mit unterschiedlichen Fähigkeiten sein und nicht nur technische Spezialisten?
5. Sie sind Softwareproduktmanager für ein Unternehmen, das Lernsoftwareprodukte auf der Grundlage wissenschaftlicher Simulationen entwickelt. Erläutern Sie, warum es wichtig ist, eine Produkt-Roadmap zu entwickeln, damit die endgültigen Produktfreigaben in den ersten drei Monaten des Jahres verfügbar sind.
6. Warum sollten Sie einen Prototyp implementieren, bevor Sie mit der Entwicklung eines neuen Softwareprodukts beginnen?

Agiles Software-Engineering

2

ÜBERBLICK