

Abbildung 6.1 zeigt, wie UP-Artefakte (mit einer Betonung der Text-Use-Cases) ein Projekt beeinflussen. Allgemein formulierte Ziele und Use-Case-Diagramme bilden den Input für die Formulierung des Use-Case-Textes. Die Use Cases können ihrerseits viele andere Artefakte der Analyse, des Entwurfs, der Implementierung, des Projektmanagements und der Tests beeinflussen.

6.1 Beispiel

Informell sind Use Cases *Textgeschichten*, die beschreiben, wie ein Akteur ein System verwendet, um bestimmte **Ziele** zu erreichen. Hier ist ein Beispiel für einen Use Case in *Kurzform*:

Process Sale: Ein Kunde kommt mit Artikeln, die er kaufen möchte, an eine Kasse. Der Kassierer verwendet das POS-System, um alle gekauften Artikel zu registrieren. Das System zeigt eine laufende Summe und Einzelheiten über die Positionen an. Der Kunde gibt Zahlungsinformation ein, die das System überprüft und registriert. Das System aktualisiert den Lagerbestand. Der Kunde erhält von dem System einen Kaufbeleg und geht dann mit den Artikeln weg.

Beachten Sie, dass *Use Cases keine Diagramme sind, sondern aus Text bestehen*. Use-Case-Anfänger machen oft den Fehler, sich auf die zweitrangigen UML-Use-Case-Diagramme statt auf den wichtigen Use-Case-Text zu konzentrieren.

→ **Siehe auch:** Abschnitt 6.17, UML anwenden: Use-Case-Diagramme

Use Cases müssen häufig detaillierter oder strukturierter sein als dieses Beispiel, aber im Wesentlichen geht es darum, funktionale Anforderungen zu entdecken und aufzuzeichnen. Zu diesem Zweck werden Geschichten geschrieben, die beschreiben, wie ein System benutzt wird, um Ziele – anders ausgedrückt: *Anwendungsfälle* – eines Benutzers zu erfüllen. Das Konzept sollte eigentlich nicht schwer zu verstehen sein, obwohl es häufig schwierig ist festzustellen, was gebraucht wird, und es in einer brauchbaren Form festzuhalten.

6.2 Definition: Was sind Akteure, Szenarios und Use Cases?

Zunächst einige informelle Definitionen: Ein *Akteur* (engl. *actor*) ist eine Komponente mit einem Verhalten, wie beispielsweise eine Person (identifiziert durch eine Rolle), ein Computersystem oder eine Organisation – beispielsweise ein Kassierer.

Ein *Szenario* ist eine spezielle Folge von Aktionen und Interaktionen zwischen Akteuren und dem System; es wird auch als *Use-Case-Instanz* bezeichnet. Ein Szenario ist eine bestimmte Geschichte über den Gebrauch des Systems oder ein konkreter Pfad durch den Use Case; beispielsweise das Szenario eines erfolgreichen Kaufs von Artikeln mit Bargeld oder das Szenario eines gescheiterten Kaufs von Artikeln, weil eine Zahlung per Kredit abgewiesen wurde.

Informell ist ein *Use Case* also eine Sammlung verwandter Erfolgs- und Misserfolgsszenarios, die beschreiben, wie ein Akteur ein System benutzt, um ein Ziel zu erreichen. Hier ist ein Beispiel für einen Use Case mit verschiedenen Szenarios in einem *informellen Format*:

Handle Returns (Retouren bearbeiten)

Standardszenario: Ein Kunde kommt mit Artikeln, die er zurückgeben möchte, an eine Kasse. Der Kassierer verwendet das POS-System, um jeden zurückgegebenen Artikel zu registrieren ...

Alternative Szenarios: Wenn der Kunde per Kredit bezahlt hat und die Rückzahlungstransaktion auf sein Kreditkonto abgelehnt wird, den Kunden informieren und ihm Bargeld auszahlen.

Kapitel 6

Wenn die Artikelbezeichnung in dem System nicht gefunden wird, den Kassierer benachrichtigen und eine manuelle Eingabe der Artikelnummer empfehlen (vielleicht ist sie beschädigt).

Wenn das System mit dem externen Buchhaltungssystem nicht kommunizieren kann, ...

Nachdem jetzt Szenarios (Use-Case-Instanzen) definiert worden sind, ist eine andere, aber ähnliche Definition von Use Case, die in dem RUP verwendet wird, sinnvoller:

Ein Satz von Use-Case-Instanzen, wobei eine Instanz eine Folge von Aktionen ist, die ein System ausführt und die zu einem beobachtbaren Ergebnis führen, das für einen bestimmten Akteur einen Wert hat [RUP].

6.3 Use Cases und das Use Case Model

UP definiert das *Use Case Model* innerhalb der *Requirements*-Disziplin. Es besteht hauptsächlich aus einem Satz aller schriftlichen Use Cases; es ist ein Modell der Funktionalität und der Umgebung des Systems.

Use Cases sind Textdokumente, keine Diagramme, und die Use-Case-Modellierung besteht hauptsächlich darin, Text zu schreiben, und nicht, Diagramme zu zeichnen.

Das Use Case Model ist nicht das einzige Anforderungsartefakt im UP. Außerdem gibt es die Supplementary Specification, das Glossary, die Vision und die Business Rules. Sie sind für die Anforderungsanalyse alle nützlich, aber an diesem Punkt von untergeordneter Bedeutung.

➔ **Siehe auch:** Kapitel 7, *Andere Anforderungen*

Das Use Case Model kann optional ein UML-Use-Case-Diagramm enthalten, um die Namen der Use Cases und Akteure und ihre Beziehungen zu zeigen. Dies ergibt ein hübsches *Kontextdiagramm* eines Systems und seiner Umgebung. Außerdem bietet es eine schnelle Übersicht über die Namen der Use Cases.

➔ **Siehe auch:** Abschnitt 6.17, *UML anwenden: Use-Case-Diagramm*

Use Cases haben nichts mit Objektorientierung zu tun; wir führen keine OO-Analyse durch, wenn wir Use Cases schreiben. Dies ist kein Problem – Use Cases sind breit anwendbar, wodurch ihre Nützlichkeit verbessert wird. Davon abgesehen sind Use Cases der Schlüsselinput für die Anforderungsanalyse im klassischen OOA/D.

6.4 Motivation: Warum Use Cases verwenden?

Wir haben diverse Ziele, die wir mit Hilfe von Computern erfüllen möchten, angefangen von der Registrierung von Verkäufen, über das Spielen von Spielen bis hin zur Schätzung der künftigen Ausbeute von Ölquellen. Kluge Analytiker haben zahlreiche Methoden erfunden, um Ziele zu erfassen, doch die besten sind einfach und vertraut. Warum? Dadurch wird es – insbesondere für Kunden – leichter, zur Definition und Überprüfung der Ziele beizutragen. Dies verringert das Risiko, das Ziel zu verfehlten. Auch wenn sich dies vielleicht wie ein beiläufiger Kommentar anhört, ist es wichtig. Forscher haben komplexe Analysemethoden entwickelt, die nur sie allein verstehen, aber normale Geschäftleute ins Koma fallen lassen! Das Versäumnis, die Anwender an Softwareprojekten zu beteiligen, steht auf der Liste der Gründe für das Scheitern von Projekten fast ganz oben [Larman03], sodass alles, was dazu beiträgt, die Anwender zu beteiligen, wirklich wünschenswert ist.

Use Cases zu schreiben ist so einfach, dass auch Fachbereichsexperten oder andere Lieferanten von Anforderungen selbst Use Cases schreiben oder sich am Schreiben von Use Cases beteiligen können.

➔ **Siehe auch:** Abschnitt 6.19; *Motivation: Andere Vorteile von Use Cases? Anforderungen im Kontext*

Ein anderer Wert von Use Cases besteht darin, dass *sie die Anwenderziele und -perspektive betonen*. Wir fragen: »Wer benutzt das System, wie sehen die typischen Anwendungsszenarios des Anwenders aus und welche Ziele verfolgt er damit?« Diese Betrachtungsweise ist benutzerzentrierter, als einfach eine Liste von Systemeigenschaften abzufragen.

Über Use Cases ist viel geschrieben worden. Doch obwohl eine Beschäftigung damit lohnt, verdunkeln kreative Leute häufig eine einfache Idee durch diverse Verfeinerungen oder überkomplizierte Varianten. Ein Novize der Use-Case-Modellierung (oder ein schwerer Fall von Typ-A-Analytiker!) lässt sich normalerweise daran erkennen, dass er sich mehr für untergeordnete Probleme wie Use-Case-Diagramme, Use-Case-Beziehungen, Use-Case-Pakete usw. interessiert, als sich auf die schwierige Aufgabe zu konzentrieren, die Textgeschichten einfach *aufzuschreiben*.

Davon abgesehen liegt eine Stärke von Use Cases in der Fähigkeit, sich durch verschiedene Grade der Ausarbeitung und Formalität an verschiedene Probleme anzupassen.

6.5 Definition: Sind Use Cases funktionale Anforderungen?

Use Cases *sind* Anforderungen, hauptsächlich funktionale (auf das Verhalten bezogene) Anforderungen, die beschreiben, was das System tun wird. Im Hinblick auf die FURPS+-Anforderungstypen betonen sie das »F« (funktional oder auf das Verhalten bezogen), können aber auch für andere Typen verwendet werden, insbesondere, wenn diese anderen Typen eng mit einem Use Case verbunden sind. Im UP – und in vielen modernen Methoden – sind Use Cases der zentrale Mechanismus, der zur Aufdeckung und Definition von Anforderungen empfohlen wird.

➔ Siehe auch: Abschnitt 5.4 über FURPS+

Eine verwandte Sichtweise betrachtet einen Use Case als Definition eines *Contracts* (Vertrags) über das Verhalten eines Systems [Cockburn01].

Um es noch einmal klar zu sagen: Use Cases sind tatsächlich Anforderungen (obwohl nicht alle Anforderungen Use Cases sind). Einige Leute sehen Anforderungen nur als »Das System soll ... tun, sein oder haben«-Listen von Funktionen oder Eigenschaften. Dies ist nicht richtig! Eine Schlüsselidee der Arbeit mit Use Cases ist es (normalerweise), die Bedeutung oder Anwendung von detaillierten Eigenschaftenlisten alten Stils zu verringern und stattdessen Use Cases für die funktionalen Anforderungen zu schreiben. Mehr zu diesem Punkt folgt in einem späteren Abschnitt.

6.6 Definition: Welche drei Arten von Akteuren gibt es?

Ein *Akteur* ist eine Komponente mit einem Verhalten. Dazu zählt auch das so genannte *System under Discussion* (*SuD*, dt. das System, das gerade beschrieben wird), wenn es die Dienste anderer Systeme in Anspruch nimmt. (Dies war eine Verfeinerung und Verbesserung gegenüber anderen Definitionen von Akteur, einschließlich der Definitionen in frühen Versionen der UML und des UP [Cockburn97]. Ältere Definitionen schlossen das SuD als Akteur aus, wenn es die Dienste anderer Systeme in Anspruch nahm, und waren damit inkonsistent. Alle Entities können mehrere *Rollen* spielen; dies gilt auch für das SuD.) Primärakteure und unterstützende Akteure können in den Aktionsschritten des Use-Case-Textes vorkommen. Akteure sind Rollen, die nicht nur von Personen, sondern auch von Organisationen, Software oder Maschinen gespielt werden können. In Bezug auf das SuD gibt es drei Arten von externen Akteuren:

- Primärakteur* – (engl. *primary actor*) hat Anwenderziele, die durch Nutzung der Dienste des SuD, beispielsweise den Kassierer, erfüllt werden.
 - Warum identifizieren? Um Anwenderziele zu finden, die die Use Cases anstoßen.
- Unterstützender Akteur* – (engl. *supporting actor*) stellt einen Dienst (beispielsweise Informationen) für das SuD zur Verfügung, wie beispielsweise der automatisierte Payment Authorization Service. Oft ein Computersystem, könnte aber auch eine Organisation oder Person sein.
 - Warum identifizieren? Um externe Schnittstellen und Protokolle zu klären.
- Offstage-Akteur* – (engl. *offstage actor*) hat ein Interesse am Verhalten des Use Case, ist aber kein Primärakteur oder unterstützender Akteur, wie beispielsweise die Finanzbehörden.
 - Warum identifizieren? Um sicherzustellen, dass *alle* notwendigen Interessen identifiziert und befriedigt werden. Die Interessen eines Offstage-Akteurs sind manchmal nicht leicht auszumachen oder leicht zu übersehen, wenn diese Akteure nicht ausdrücklich erwähnt werden.

6.7 Notation: Welche drei gebräuchlichen Use-Case-Formate gibt es?

Use Cases können in verschiedenen Formaten und Graden der Formalität beschrieben werden:

- kurz* – (engl. *brief*) knappe Zusammenfassung in einem Absatz; wird normalerweise für den Standardablauf verwendet. Das vorangegangene *Process Sale*-Beispiel war kurz.
 - Wann? Während der frühen Anforderungsanalyse, um schnell ein Gefühl für das Thema und den Umfang zu gewinnen. Möglicherweise werden nur einige Minuten benötigt, um den Use Case zu schreiben.
- ➔ **Siehe auch:** Beispiel *Process Sale* in Abschnitt 6.1
- informell* – (engl. *casual*) ein informelles Absatzformat. Mehrere Absätze, die verschiedene Szenarios beschreiben. Das vorangegangene *Handle Returns*-Beispiel war informell.
 - Wann? Wie oben.
- ➔ **Siehe auch:** Beispiel *Handle Returns* in Abschnitt 6.2
- voll ausgearbeitet* – (engl. *fully dressed*) alle Schritte und Varianten werden ausführlich beschrieben; es gibt unterstützende Abschnitte wie beispielsweise Vorbedingungen und Erfolgsgarantien.
 - Wann? Nachdem viele Use Cases identifiziert und in einem kurzen Format beschrieben worden sind; dann werden auf dem ersten Anforderungsworkshop einige (etwa 10%) der architektonisch wichtigen und geschäftlich wichtigen Use Cases im Detail ausgearbeitet.
- ➔ **Siehe auch:** Abschnitt 6.21, Prozess: Wie werden Use Cases in iterativen Methoden verwendet?

Das folgende Beispiel zeigt einen voll ausgearbeiteten Use Case für unsere NextGen-Fallstudie.

6.8 Beispiel: Process Sale, voll ausgearbeitetes Format

Voll ausgearbeitete Use Cases zeigen mehr Einzelheiten und sind strukturiert; sie graben tiefer.

Bei der iterativen und evolutionären UP-Anforderungsanalyse würden etwa 10% der kritischen Use Cases mit dieser Methode auf dem ersten Anforderungsworkshop ausgearbeitet werden. Dann beginnen Entwurf und Programmierung mit den architektonisch wichtigsten Use Cases oder Szenarios aus diesen 10%.

Es gibt diverse Formatvorlagen für detaillierte Use Cases. Die seit den frühen 90er Jahren wahrscheinlich am meisten genutzte und beschriebene Formatvorlage wurde von Alistair Cockburn, dem Autor des populärsten Buches und Ansatzes zur Use-Case-Modellierung, entworfen. Die Vorlage ist unter alistair.cockburn.us im Web erhältlich. Das folgende Beispiel illustriert diesen Stil.

Zunächst die Vorlage:

Use-Case-Abschnitt	Kommentar
Use-Case-Name (engl. <i>Use Case Name</i>)	Nennen Sie das Objekt und dann das Verb: <i>Verkauf bearbeiten</i> . ¹
Umfang (engl. <i>Scope</i>)	Das System under Design (SuD), das zu entwerfende System
Ebene (engl. <i>Level</i>)	»Anwenderziel« oder »Subfunktion«
Primärakteur (engl. <i>Primary Actor</i>)	Nutzt einen Dienst des Systems
Stakeholder und Interessen (engl. <i>Stakeholders and Interests</i>)	Für wen ist dieser Use Case relevant und welches Interesse hat er daran?
Vorbedingungen (engl. <i>Preconditions</i>)	Was muss am Anfang gewährleistet <i>und</i> für den Leser mitteilenswert sein?
Erfolgsgarantie, auch Nachbedingungen (engl. <i>Success Guarantee</i>)	Was muss nach der erfolgreichen Fertigstellung gewährleistet <i>und</i> für den Leser mitteilenswert sein?
Standardablauf (engl. <i>Main Success Scenario</i>)	Das Szenario für einen typischen, unbedingten erfolgreichen Durchlauf durch den Use Case
Erweiterungen (engl. <i>Extensions</i>)	Alternative Szenarios für Erfolg oder Misserfolg
Spezielle Anforderungen (engl. <i>Special Requirements</i>)	Zum System gehörige, nichtfunktionale Anforderungen
Liste der Technik- und Datenvariationen (engl. <i>Technology and Data Variations List</i>)	Alternative I/O-Methoden und Datenformate
Häufigkeit des Auftretens (engl. <i>Frequency of Occurrence</i>)	Beeinflusst die Untersuchung, das Testen und die zeitliche Gestaltung der Implementierung
Verschiedenes (engl. <i>Miscellaneous</i>)	Beispielsweise offene Probleme

Standardablauf und *Erweiterungen* sind die beiden Hauptabschnitte.

1. A.d.U.: Hier empfiehlt die englische Version, mit einem Verb zu beginnen, z. B. *Process Sale*. Dies passt schlecht zur Struktur der deutschen Sprache. Deshalb ist es im Deutschen besser, das Objekt zuerst zu nennen.

Kapitel 6

Das folgende Beispiel basiert auf dieser Schablone.

Beachten Sie bitte, dass dies in diesem Buch das Hauptbeispiel für einen detaillierten Use Case ist. Es zeigt viele gebräuchliche Elemente und häufig vorkommende Probleme.

Dieser Use Case zeigt wahrscheinlich viel mehr, als Sie jemals über ein POS-System wissen wollten! Aber er entstammt einem echten Projekt für ein POS-System und zeigt, wie man mit Use Cases komplexe Anforderungen der Praxis erfassen und vielfältige Verzweigungsszenarios beschreiben kann.

Use Case UC1: Process Sale

Umfang: NextGen-POS-Anwendung

Ebene: Anwenderziel

Primärakteur: Kassierer

Stakeholder und Interessen:

- Kassierer:** Will genaue, schnelle Eingabe und keine Zahlungsfehler, da Fehlbeträge von seinem Gehalt abgezogen werden.
- Verkäufer:** Will, dass die Verkaufsprovisionen aktualisiert werden.
- Kunde:** Will mit minimaler Anstrengung kaufen und Dienste in Anspruch nehmen. Will leicht sichtbare Anzeige von eingegebenen Artikeln und Preisen. Will Kaufbeleg für den Fall einer Rückgabe.
- Unternehmen:** Will Transaktionen genau festhalten und Kundenbedürfnisse befriedigen. Will sicherstellen, dass die Zahlungsforderungen des Payment Authorization Service gespeichert werden. Will eine gewisse Fehlertoleranz, um Verkäufe auch dann erfassen zu können, wenn Dienstkomponenten (wie z. B. die Fernvalidierung von Kreditanfragen) nicht verfügbar sind. Will automatische und schnelle Aktualisierung von Buchhaltung und Lager.
- Manager:** Will schnell vorrangige Operationen ausführen und Kassiererprobleme leicht beheben können.
- Finanzbehörden:** Wollen auf jeden Kauf Steuer erheben. Möglicherweise sind mehrere Behörden beteiligt (Bund, Land, Kommune).
- Payment Authorization Service:** Will digitale Autorisierungsanforderungen im richtigen Format und mit dem richtigen Protokoll erhalten. Will exakt über seine Verbindlichkeiten dem Geschäft gegenüber Buch führen.

Vorbedingungen: Kassierer wird identifiziert und authentifiziert.

Nachbedingungen: Verkauf ist gespeichert. Steuern sind richtig berechnet. Buchhaltung und Lagerbestand wurden aktualisiert. Provisionen wurden gespeichert. Kaufbeleg wurde erstellt. Zahlungsgenehmigungen sind gespeichert.

Standardablauf:

1. Kunde kommt mit Waren und/oder Diensten, die er kaufen will, an die Kasse.
2. Kassierer beginnt neuen Verkauf.
3. Kassierer gibt Artikelbezeichnung ein.
4. System speichert Verkaufsposition und zeigt Artikelbeschreibung, Preis und laufende Summe an. Preis wird anhand von Preisstellungsregeln berechnet.

Kassierer wiederholt Schritte 3–4, bis er das Eingabeende anzeigt.

5. System zeigt Summe und berechnete Steuern an.
6. Kassierer teilt dem Kunden die Summe mit und bittet um Zahlung.

Use Case UC1: Process Sale

7. Kunde bezahlt und System bearbeitet Zahlung.
8. System protokolliert den abgeschlossenen Verkauf und sendet Verkaufs- und Zahlungsinformationen an das externe Abrechnungssystem (für Buchhaltung und Provisionsberechnung) und das Lagerverwaltungssystem (zur Aktualisierung der Lagerbestände).
9. System präsentiert Kaufbeleg.
10. Kunde geht mit Kaufbeleg und Waren (wenn zutreffend).

Erweiterungen (oder alternative Abläufe):

*a. Jederzeit, wenn Manager eine vorrangige Operation anfordert:

1. System geht in den Managermodus.
2. Manager oder Kassierer führt eine Operation im Managermodus aus, z. B. Änderung des Bargeldsaldos, Wiederaufnahme eines angehaltenen Verkaufs an einer anderen Kasse, Stornierung eines Verkaufs usw.
3. System geht in Kassierermodus zurück.

*b. Jederzeit, wenn das System ausfällt:

Um Wiederherstellung und korrekte Buchhaltung zu gewährleisten, sicherstellen, dass alle transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios wiederhergestellt werden können.

1. Kassierer startet System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustands an.
2. System rekonstruiert früheren Zustand.
 - 2a. System entdeckt Anomalien, die Wiederherstellung verhindern:
 1. System signalisiert Fehler zum Kassierer, speichert den Fehler und geht in einen sauberen Zustand.
 2. Kassierer beginnt neuen Verkauf.

1a. Kunde oder Manager zeigen an, einen unterbrochenen Verkauf fortzusetzen.

1. Kassierer führt Wiederaufnahmeoperation aus und gibt die ID ein, um den Verkauf abzurufen.
2. System zeigt den Zustand des wieder aufgenommenen Verkaufs mit Zwischensumme an.

2a. Verkauf wurde nicht gefunden.

1. System meldet Fehler an Kassierer.
2. Kassierer startet wahrscheinlich neuen Verkauf und gibt alle Artikel erneut ein.
3. Kassierer fährt mit dem Verkauf fort (indem er wahrscheinlich weitere Artikel eingibt oder die Zahlung handhabt).

2-4a. Kunde teilt dem Kassierer mit, dass er einen speziellen Steuerstatus hat (z. B. Steuerbefreiung als Rentner, Eingeborener usw.).

1. Kassierer verifiziert und gibt dann den Code für den Steuerstatus ein.
2. System speichert den Steuerstatus (den es zur Steuerberechnung benötigt)

3a. Ungültige Artikelnummer (nicht im System gefunden):

1. System signalisiert Fehler und weist Eingabe ab.
2. Kassierer reagiert auf den Fehler:

Use Case UC1: Process Sale

- 2a. Es gibt eine (von Menschen) lesbare Artikelnummer (z. B. einen numerischen UPC):
1. Kassierer gibt die Artikel-ID manuell ein.
 2. System zeigt Artikelbeschreibung und Preis.
- 2a. Ungültige Artikelnummer: System signalisiert Fehler.
Kassierer versucht andere Methode.
- 2b. Es gibt keine Artikelnummer, aber auf dem Schild steht ein Preis:
1. Kassierer bittet Manager, eine vorrangige Operation auszuführen.
 2. Manager führt vorrangige Operation aus.
 3. Kassierer zeigt eine manuelle Preiseingabe, gibt Preis ein und fordert Standardbesteuerung für diesen Betrag an (weil es keine Produktinformationen gibt, kann die Steuerberechnungskomponente die Steuern nicht anderweitig ableiten).
- 2c. Kassierer führt `Find Product Help` aus, um die richtige Artikelnummer und den Preis zu suchen.
- 2d. Andernfalls fragt der Kassierer einen Mitarbeiter nach der richtigen Artikelnummer oder dem Preis und gibt entweder die Artikelnummer oder den Preis manuell ein (siehe oben).
- 3b. Es gibt mehrere Artikel derselben Artikelgruppe, und Überwachung der Identität der Einzelartikel ist nicht wichtig (z. B. 5 Pakete Veggie-Burger):
1. Kassierer kann Artikelgruppenbezeichnung und Menge eingeben.
- 3c. Artikel erfordert manuelle Eingabe von Artikelgruppe und Preis (wie beispielsweise Blumen oder Artikel mit einer speziellen Preisauszeichnung):
1. Kassierer gibt speziellen Code für die manuelle Artikelgruppeneingabe und den Preis ein.
- 3-6a: Kunde bittet Kassierer, einen Artikel zu stornieren:
Dies ist nur zulässig, wenn der Artikelwert kleiner als das Stornolimit des Kassierers ist, sonst muss ein Manager eine vorrangige Operation ausführen.
1. Kassierer gibt Artikelnummer des zu stornierenden Artikels ein.
 2. System entfernt Artikel und zeigt aktualisierte laufende Summe an.
- 2a. Artikelpreis übersteigt Stornolimit des Kassierers:
1. System meldet Fehler und schlägt vorrangige Manageroperation vor.
 2. Kassierer fordert vorrangige Manageroperation an, führt sie aus und wiederholt Operation.
- 3-6b. Kunde bittet Kassierer, den Verkauf zu stornieren:
1. Kassierer storniert den Verkauf im System.
- 3-6c. Kassierer unterbricht den Verkauf:
1. System speichert Verkauf, sodass er an einer Kasse wiederhergestellt werden kann.
 2. System gibt einen Beleg über den unterbrochenen Verkauf aus, der die Positionen und eine Verkaufs-ID ausweist, mit der der Verkauf abgerufen und wieder aufgenommen werden kann.
- 4a. Der vom System genannte Artikelpreis wird nicht gewünscht (z. B. hat sich der Kunde über etwas beschwert und bekommt einen niedrigeren Preis angeboten):
1. Kassierer fordert von Manager Genehmigung an.
 2. Manager führt vorrangige Operation aus.
 3. Kassierer gibt manuell den vorrangigen Preis ein.
 4. System zeigt neuen Preis an.

Use Case UC1: Process Sale

5a. System entdeckt Fehler bei der Kommunikation mit externem Steuerberechnungsservice:

1. System startet den Dienst auf dem POS-Knoten erneut und fährt fort.

1a. System stellt fest, dass der Dienst nicht neu startet.

1. System signalisiert Fehler.

2. Kassierer kann die Steuer manuell berechnen und eingeben oder den Verkauf abbrechen.

5b. Kunde sagt, er bekomme einen Rabatt (z. B. Mitarbeiter, bevorzugter Kunde):

1. Kassierer signalisiert Rabattanforderung.

2. Kassierer gibt Kundenidentifikation ein.

3. System zeigt Rabattbetrag an, der auf Rabattregeln basiert.

5c. Kunde sagt, er habe Kredit auf seinem Konto, den er für den Verkauf nutzen möchte:

1. Kassierer signalisiert Kreditanforderung.

2. Kassierer gibt Kundenidentifikation ein.

3. System wendet Kredit bis Preis = 0 an und reduziert den verbleibenden Kredit.

6a. Kunde sagt, dass er bar zahlen möchte, hat aber nicht genug Bargeld:

1. Kassierer bittet um andere Zahlungsmethode.

1a. Kunde bittet Kassierer, den Verkauf abzubrechen.

Kassierer bricht den Verkauf im System ab.

7a. Barzahlung:

1. Kassierer gibt den gezahlten Geldbetrag ein.

2. System zeigt den fälligen Saldo an und gibt die Geldschublade frei.

3. Kassierer legt das gezahlte Bargeld in die Schublade und gibt den Saldo in bar an den Kunden heraus.

4. System registriert die Bargeldzahlung.

7b. Zahlung per Kreditkarte:

1. Kunde gibt seine Kreditkarteninformationen ein.

2. System zeigt die Zahlung zur Verifizierung an.

3. Kassierer bestätigt.

3a. Kassierer bricht den Zahlungsschritt ab:

1. System geht in den Artikeleingabemodus zurück.

4. System sendet Zahlungsgenehmigungsanforderung an das System des externen Payment-Authorization Service und fordert Zahlungsgenehmigung an.

4a. System entdeckt Fehler bei der Zusammenarbeit mit dem externen System:

1. System meldet Fehler an Kassierer.

2. Kassierer bittet Kunden um andere Zahlungsmethode.

5. System erhält Zahlungsgenehmigung, meldet die Genehmigung dem Kassierer und gibt die Geldschublade frei (um den unterschriebenen Kaufbeleg für die Kreditzahlung abzulegen).

5a. System empfängt Zahlungsablehnung:

1. System meldet Ablehnung an Kassierer.

2. Kassierer bittet Kunde um andere Zahlungsmethode.

Use Case UC1: Process Sale

- 5b. Timeout beim Warten auf eine Antwort.
 1. System meldet Timeout an Kassierer.
 2. Kassierer kann es erneut versuchen oder den Kunden um andere Zahlungsmethode bitten.
6. System registriert die Zahlung per Kreditkarte einschließlich der Zahlungsgenehmigung.
7. System präsentiert den Mechanismus für die Unterschrifteingabe bei Kreditkartenzahlungen.
8. Kassierer bittet Kunden um eine Unterschrift für die Zahlung per Kreditkarte. Kunde gibt seine Unterschrift ein.
9. Wenn die Unterschrift auf einem Papierkaufbeleg erfolgt, legt Kassierer den Kaufbeleg in die Geldschublade und schließt sie.
- 7c. Zahlung per Scheck ...
- 7d. Zahlung per Guthaben ...
- 7e. Kassierer bricht Zahlungsschritt ab:
 1. System geht in den Artikeleingabemodus zurück.
- 7f. Kunde präsentiert Gutscheine/Coupons:
 1. Vor der Zahlungsabwicklung registriert der Kassierer jeden Coupon, und System reduziert den Preis entsprechend. System registriert die gebrauchten Coupons aus buchhalterischen Gründen.
 - 1a. Der eingegebene Coupon gilt nicht für einen gekauften Artikel:
 1. System meldet Fehler an Kassierer.
- 9a. Es gibt Produktrabatte:
 1. System präsentiert für jeden rabattierten Artikel Rabattformular und Rabattbeleg.
- 9b. Kunde fordert einen Geschenkgutschein (ohne sichtbaren Preis) an:
 1. Kassierer fordert Geschenkgutschein an und System präsentiert ihn.
- 9c. Drucker hat kein Papier mehr.
 1. Wenn System den Defekt entdecken kann, meldet es das Problem.
 2. Kassierer legt neues Papier ein.
 3. Kassierer fordert einen weiteren Kaufbeleg an.

Spezielle Anforderungen:

- Touchscreen-UI für einen großen Flachbildschirm. Text muss aus einem Meter Entfernung lesbar sein.
- Antwortzeit bei der Kreditautorisierung innerhalb von 30 Sekunden in 90% der Fälle.
- Irgendwie brauchen wir eine robuste Wiederherstellung, wenn der Zugriff auf Ferndienste wie beispielsweise das Lagerverwaltungssystem scheitert.
- Internationalisierung der Sprache in der Textanzeige.
- Die Geschäftsregeln in den Schritten 3 und 7 sollen frei wählbar (engl. *pluggable*, dt. *einsteckbar*) sein.
- ...

Liste der Technik- und Datenvariationen:

- *a. Der Wechsel in den Managermodus zur Durchführung vorrangiger Manageroperationen erfolgt mit einer speziell für diesen Zweck bereitgestellten Karte, die durch einen Kartenleser gezogen wird, oder durch Eingabe eines Autorisierungscodes über die Tastatur.
- 3a. Eingabe der Artikelnummern mit einem Strichcodeleser (wenn ein Strichcode vorhanden ist) oder per Tastatur.

Use Case UC1: Process Sale

- 3b. Artikelnummern können wahlweise nach den folgenden Systemen codiert sein: UPC, EAN, JAN oder SKU.
- 7a. Eingabe der Kreditkarteninformationen über Kartenleser oder Tastatur.
- 7b. Unterschriften für Zahlungen per Kreditkarte erfolgen auf einem Papierkaufbeleg. Wir sagen aber vor-
aus, dass in zwei Jahren viele Kunden die Übernahme einer digitalen Unterschrift fordern werden.

Häufigkeit des Auftretens: Beinahe laufend.

Offene Fragen:

- Welche Varianten in den Steuergesetzen gibt es?
- Das Problem des Wiederanlaufs bei Ferndiensten untersuchen.
- Welche Anpassungen sind für verschiedene Geschäfte/Branchen erforderlich?
- Muss ein Kassierer seine Geldschublade mitnehmen, wenn er sich abmeldet?
- Kann der Kunde den Kartenleser direkt verwenden oder muss dies der Kassierer tun?

Dieser Use Case ist nicht erschöpfend, sondern dient nur zur Veranschaulichung (obwohl er auf den Anforde-
rungen an ein echtes POS-System basiert, das mit einem OO-Entwurf in Java entwickelt wurde). Dennoch
enthält er genügend Einzelheiten und Komplexität, um einen realistischen Eindruck von einem voll ausgear-
beiteten Use Case mit vielen Anforderungsdetails zu vermitteln. Dieses Beispiel eignet sich gut als Modell für
viele Use-Case-Probleme.

6.9 Was bedeuten die Abschnitte?

6.9.1 Vorspannelemente

Umfang

Der Umfang begrenzt das zu entwerfende System (oder die zu entwerfenden Systeme). Üblicherweise beschreibt
ein Use Case die Benutzung eines Systems (nur Software oder Hardware plus Software); dieser Fall wird auch als
System Use Case bezeichnet. Auf einer abstrakteren Ebene können Use Cases auch beschreiben, wie ein Geschäft
von seinen Kunden und Partnern »benutzt« wird. Eine solche Prozessbeschreibung auf Unternehmensebene
wird auch als *Business Use Case* bezeichnet. Sie liefert ein gutes Beispiel für die breite Anwendbarkeit von Use
Cases, wird aber in diesem einführenden Buch nicht behandelt.

Ebene

Cockburn klassifiziert Use Cases in seinem System nach Ebenen und unterscheidet unter anderem eine Anwen-
derzielebene und eine Subfunktionsebene. Ein Use Case der *Anwenderzielebene* ist die übliche Form, um Szena-
rios zu beschreiben, in denen ein Primärakteur das System benutzt, um im Zuge seiner Arbeit Ziele zu erfüllen;
ein solcher Use Case entspricht ungefähr einem *Elementary Business Process* (EBP, elementarer Geschäftsprozess)
beim Business Process Engineering (Gestaltung von Geschäftsabläufen). Ein Use Case der *Unterfunktionsebene*
beschreibt Teilschritte, die erforderlich sind, um ein Anwenderziel zu erreichen; er wird normalerweise erstellt,
um die Duplikierung gleichartiger Teilschritte (und des damit verbundenen Textes) zu vermeiden, die in mehreren
normalen Use Cases vorkommen; ein Beispiel ist der Subfunktions-Use-Case *Pay by Credit*, der von vielen
normalen Use Cases gemeinsam genutzt werden kann.

→ **Siehe auch:** Unterabschnitt 6.16.2, *Der EBP-Test*

Primärakteur

Der Hauptakteur, der Dienste des Systems abruft, um ein Ziel zu erfüllen.

Liste der Stakeholder und Interessen – wichtig!

Diese Liste ist wichtiger und praktischer, als es auf den ersten Blick scheinen mag. Sie enthält Hinweise auf die erforderlichen Funktionen des Systems und seine Grenzen. Um zu zitieren:

Das [System] erfüllt einen Vertrag zwischen Stakeholdern, wobei die Use Cases die Verhaltensweisen beschreiben, die durch diesen Vertrag abgedeckt sind ... Der Use Case, als Vertrag über das Verhalten, erfasst *die (alle) und nur die* Verhaltensweisen, die mit der Erfüllung der Interessen des Stakeholders zu tun haben [Cockburn01].

Dies beantwortet die Frage: Was sollte in dem Use Case stehen? Die Antwort lautet: Das, was die Interessen aller Stakeholder erfüllt. Mit den Stakeholdern und ihren Interessen zu beginnen, bevor der Rest des Use Case geschrieben wird, stellt uns außerdem eine Methode zur Verfügung, die uns erinnert, welche detaillierteren Verantwortlichkeiten das System haben sollte. Ein Beispiel: Hätte ich eine Verantwortlichkeit für die Bearbeitung der Verkäuferprovision identifiziert, wenn ich vorher nicht den Stakeholder »Verkäufer« und seine Interessen aufgeführt hätte? Hoffentlich irgendwann, aber möglicherweise hätte ich diesen Punkt bei der ersten Analyseitzung übersehen. Sich auf den Standpunkt der Stakeholder und der Interessen zu stellen, ist eine gründliche und methodische Prozedur, um alle erforderlichen Verhaltensweisen zu entdecken und festzuhalten.

Stakeholder und Interessen:

- Kassierer: Will genaue, schnelle Eingabe und keine Zahlungsfehler, da Fehlbeträge von seinem Gehalt abgezogen werden.
- Verkäufer: Will, dass die Verkaufsprovisionen aktualisiert werden.
- ...

Vorbedingungen und Nachbedingungen

Vorweg: Befassen Sie sich nur mit einer Vorbedingung oder Nachbedingung, wenn diese etwas Nicht-Offensichtliches oder Relevantes ausdrückt, um dem Leser ein besseres Verständnis zu vermitteln. Fügen Sie kein überflüssiges »Rauschen« in Anforderungsdokumente ein.

Vorbedingungen beschreiben, was *immer erfüllt sein muss*, bevor ein Szenario in dem Use Case begonnen werden kann. Vorbedingungen werden *nicht innerhalb* des Use Case getestet, sondern es handelt sich um Bedingungen, die als erfüllt angenommen werden. Üblicherweise impliziert eine Vorbedingung ein Szenario eines anderen Use Case, wie beispielsweise, dass die Anmeldung erfolgreich abgeschlossen wurde. Beachten Sie, dass es Bedingungen gibt, die erfüllt sein müssen, die aber nicht erwähnenswert sind, wie beispielsweise *Das System hat Strom*. Vorbedingungen drücken beachtenswerte Annahmen aus, auf die der Leser nach Ansicht des Use-Case-Autors achten sollte.

Nachbedingungen drücken aus, was nach der erfolgreichen Fertigstellung des Use Case – entweder nach einem Standardablauf oder einem alternativen Ablauf – erfüllt sein muss. Die Nachbedingung sollte die Anforderungen aller Stakeholder erfüllen.

Vorbedingungen: Kassierer wird identifiziert und authentifiziert.

Nachbedingungen: Verkauf ist gespeichert. Steuern sind richtig berechnet. Buchhaltung und Lagerbestand wurden aktualisiert. Provisionen wurden gespeichert. Kaufbeleg wurde erstellt. Zahlungsgenehmigungen sind gespeichert.

6.9.2 Standardablauf

Dieser Ablauf wird manchmal etwas blumig auch als »Haupterfolgsszenario« oder als »glücklicher Weg« bezeichnet. Er beschreibt den normalen, typischen Use-Case-Durchlauf, der die Interessen der Stakeholder erfüllt. Beachten Sie, dass dieser Ablauf häufig *keine* Bedingungen oder Verzweigungen enthält. Obwohl dies nicht falsch oder ungültig wäre, kann man die Auffassung vertreten, dass ein Use Case verständlicher und ausbaufähiger ist, wenn konsistent alle bedingten Abläufe in den Abschnitt *Erweiterungen* eingefügt werden.

Richtlinie

Fügen Sie alle bedingten und verzweigten Abläufe in den Abschnitt *Erweiterungen* ein.

Es gibt drei Arten von Schritten, die in dem Szenario festgehalten werden:

1. Interaktionen zwischen Akteuren. (Beachten Sie, dass das SuD selbst als Akteur betrachtet werden sollte, wenn es in einer Akteurrolle mit anderen Systemen zusammenarbeitet.)
2. Validierungen (normalerweise durch das System).
3. Zustandsänderungen durch das System (wenn es beispielsweise etwas speichert oder ändert).

Schritt eins eines Use Case passt nicht immer in diese Klassifikation, sondern zeigt das Trigger-Ereignis an, das das Szenario startet.

Es ist eine gebräuchliche Regel, die Namen von Akteuren großzuschreiben, um sie leichter identifizieren zu können. Beachten Sie auch, wie Wiederholungen angezeigt werden.

Standardablauf:

1. Kunde kommt mit Waren und/oder Diensten, die er kaufen will, an die Kasse.
2. Kassierer beginnt neuen Verkauf.
3. Kassierer gibt Artikelbezeichnung ein.
4. ...

Kassierer wiederholt Schritte 3–4, bis er das Eingabeende anzeigt.

1. ...

6.9.3 Erweiterungen (oder alternative Abläufe)

Erweiterungen sind wichtig und machen normalerweise den größeren Teil des Textes aus. Sie enthalten alle anderen Erfolgs- oder Misserfolgszenarios oder -zweige. Beachten Sie in dem voll ausgearbeiteten Beispiel, dass der Abschnitt *Erweiterungen* beträchtlich länger und komplexer als der Abschnitt *Standardablauf* ist; dies ist üblich.

Wenn Use Cases gründlich geschrieben werden, sollte die Kombination von Standardablauf und Erweiterungen »fast« alle Interessen der Stakeholder befriedigen. Es gibt jedoch die Einschränkung, dass einige Interessen am besten als nichtfunktionale Anforderungen in der Supplementary Specification und nicht in Use Cases erfasst werden. Beispielsweise gehört das Interesse des Kunden an einer sichtbaren Anzeige von Artikelbeschreibungen und Preisen in eine Usability-Anforderung.

Erweiterungsszenarios sind Zweige des Standardablaufs und können deshalb mit denselben Schrittnummern 1 ... N bezeichnet werden. Ein Beispiel: Bei Schritt 3 des Standardablaufs kann eine ungültige Artikelnummer vorkommen, weil entweder die Eingabe falsch war oder das System die Nummer nicht kennt. Eine Erweiterung mit der Nummer 3a identifiziert zunächst die Bedingung und dann die Reaktion. Alternative Erweiterungen von Schritt 3 haben die Nummern 3b usw.

- 3a. Ungültige Artikelnummer (nicht im System gefunden):
 - 1. System signalisiert Fehler und weist Eingabe ab.
- 3b. Es gibt mehrere Artikel derselben Artikelgruppe und Überwachung der Identität der Einzelartikel ist nicht wichtig (z. B. 5 Pakete Veggie-Burger):
 - 1. Kassierer kann Artikelgruppenbezeichnung und Menge eingeben.

Eine Erweiterung besteht aus zwei Teilen: der Bedingung und der Handhabung.

Richtlinie: Wenn möglich, sollten Sie die Bedingung als etwas beschreiben, das von dem System oder einem Akteur *entdeckt* werden kann. Vergleichen Sie dagegen die folgenden beiden Varianten:

- 5a. System entdeckt Fehler bei der Kommunikation mit externem Steuerberechnungsservice:
 - 5a. Externes Steuerberechnungssystem funktioniert nicht:

Der erste Stil ist vorzuziehen, weil dies etwas ist, was das System entdecken kann; die zweite Variante ist eine Schlussfolgerung.

Die Handhabung der Erweiterung kann in einem Schritt oder – wie im folgenden Beispiel – in einer Folge von Schritten beschrieben werden. Das Beispiel zeigt auch, mit welcher Notation angezeigt wird, dass eine Bedingung innerhalb eines Bereiches von Schritten auftreten kann:

- 3–6a: Kunde bittet Kassierer, einen Artikel zu stornieren:
 - 1. Kassierer gibt Artikelnummer des zu stornierenden Artikels ein.
 - 2. System entfernt Artikel und zeigt aktualisierte laufende Summe an.

Am Ende der Handhabung einer Erweiterung wird per Default der Standardablauf fortgesetzt, es sei denn, die Erweiterung zeigt etwas anderes an (beispielsweise den Stopp des Systems).

Manchmal ist ein Erweiterungspunkt ziemlich komplex (wie beispielsweise die *Zahlung per Kreditkarte*). Dies kann ein Grund dafür sein, die Erweiterung als separaten Use Case auszuarbeiten.

Dieses Erweiterungsbeispiel zeigt auch die Notation, um Misserfolge innerhalb von Erweiterungen anzuzeigen.

- 7b. Zahlung per Kreditkarte:
 - 1. Kunde gibt seine Kreditkarteninformationen ein.
 - ...
 - 4. System sendet Zahlungsgenehmigungsanforderung an das System des externen Payment-Authorization Service und fordert Zahlungsgenehmigung an.
- 4a. System entdeckt Fehler bei der Zusammenarbeit mit dem externen System:
 - 1. System meldet Fehler an Kassierer.
 - 2. Kassierer bittet Kunden um andere Zahlungsmethode.

Erweiterungsbedingungen, die für alle (oder wenigstens die meisten) Schritte relevant sein können. Können mit **a*, **b* nummeriert werden.

***b. Jederzeit, wenn das System ausfällt:**

Um Wiederherstellung und korrekte Buchhaltung zu gewährleisten, sicherstellen, dass alle transaktionsrelevanten Zustände und Ereignisse in jedem Schritt des Szenarios wiederhergestellt werden können.

1. Kassierer startet System neu, meldet sich an und fordert Wiederherstellung eines früheren Zustands an.
2. System rekonstruiert früheren Zustand.

Ein anderes Use-Case-Szenario ausführen

Manchmal verzweigt ein Use Case, um ein anderes Use-Case-Szenario auszuführen. Ein Beispiel: Die Geschichte *Find Product Help* (um Produktdetails, Artikelbeschreibung, Preis, ein Bild oder Video usw. zu finden) ist ein selbstständiger Use Case, der manchmal innerhalb von *Process Sale* ausgeführt wird (normalerweise, wenn die Artikelnummer nicht gefunden wird). In der Notation von Cockburn wird dieser zweite Use Case unterstrichen dargestellt:

3a. Ungültige Artikelnummer (nicht im System gefunden):

1. System signalisiert Fehler und weist Eingabe ab.
2. Kassierer reagiert auf den Fehler:
 - 2a. ...
 - 2c. Kassierer führt Find Product Help aus, um die richtige Artikelnummer und den Preis zu suchen.

Wenn die Use Cases mit einem Hyperlink-fähigen Werkzeug geschrieben werden (was der Normalfall sein sollte), dann kann der Text des unterstrichenen Use Case durch einen Klick auf seinen Namen angezeigt werden.

6.9.4 Spezielle Anforderungen

Nichtfunktionale Anforderungen, Qualitätsattribute oder Bedingungen, die an einen bestimmten Use Case gebunden sind, sollten mit dem Datensatz in dem Use Case festgehalten werden. Dazu zählen Eigenschaften wie die Performanz, Zuverlässigkeit und Usability sowie Entwurfsbedingungen (häufig I/O-Geräte betreffend), die gefordert wurden oder wahrscheinlich bestehen.

Spezielle Anforderungen:

- Touchscreen-UI für einen großen Flachbildschirm. Text muss aus einem Meter Entfernung lesbar sein.
- Antwortzeit bei der Kreditauftragserstellung innerhalb von 30 Sekunden in 90% der Fälle.
- Irgendwie brauchen wir eine robuste Wiederherstellung, wenn der Zugriff auf Ferndienste wie beispielsweise das Lagerverwaltungssystem scheitert.
- Internationalisierung der Sprache in der Textanzeige.
- Die Geschäftsregeln in den Schritten 3 und 7 sollen frei wählbar (engl. *pluggable*, dt. *einsteckbar*) sein.
- ...

Diese Anforderungen in dem Use Case zu notieren ist der klassische UP-Ratschlag. Sie sollten ihn vernünftigerweise befolgen, wenn Sie einen Use Case *erstmals* niederschreiben. Doch viele Praktiker halten es für nützlich, letztlich alle nichtfunktionalen Anforderungen in der Supplementary Specification zusammenzufassen und zu konsolidieren, um die Verwaltung der Inhalte sowie die Verständlichkeit und Lesbarkeit der Use Cases zu ver-

bessern, weil diese Anforderungen normalerweise während der gesamten Analyse der Architektur berücksichtigt werden müssen.

6.9.5 Liste der Technik- und Datenvariationen

Häufig gibt es technische Varianten, *wie* etwas, aber nicht was getan werden muss. Es ist wichtig, dies in dem Use Case festzuhalten. Beispielsweise kann ein Stakeholder bestimmte Anforderungen an die Input- oder Output-Verfahren stellen und fordern, dass das POS-System die Eingabe von Kreditkartendaten per Kartenleser und Tastatur ermöglichen muss. Beachten Sie, dass dies ein Beispiel für eine frühe Entwurfsentscheidung oder Bedingung ist; im Allgemeinen ist es methodisch besser, vorzeitige Entwurfsentscheidungen zu vermeiden, aber manchmal sind sie – gerade bei I/O-Verfahren – offensichtlich oder unvermeidbar.

Es ist außerdem erforderlich, Variationen der Datenformate zu verstehen – wie beispielsweise Artikelnummern, die in unterschiedlichen Strichcodestandards wie UPC oder EAN codiert sind.

Diese Liste ist der Ort, um derartige Varianten festzuhalten. Sie eignet sich dazu, Variationen von Datenformaten zu notieren, die in einem bestimmten Schritt erfasst werden.

Liste der Technik- und Datenvariationen:

- 3a. Eingabe der Artikelnummern mit Hilfe eines Strichcodelesers (wenn ein Strichcode vorhanden ist) oder per Tastatur.
- 3b. Artikelnummern können wahlweise nach den folgenden Systemen codiert sein: UPC, EAN, JAN oder SKU.
- 7a. Eingabe der Kreditkarteninformationen über Kartenleser oder Tastatur.
- 7b. Unterschriften für Zahlungen per Kreditkarte erfolgen auf einem Papierkaufbeleg. Wir sagen aber voraus, dass in zwei Jahren viele Kunden die Übernahme einer digitalen Unterschrift fordern werden.

Glückwunsch: Die Use Cases sind geschrieben und falsch (!)

Das NextGen-POS-Team schreibt einige Use Cases in mehreren kurzen Anforderungsworkshops parallel zu einer Reihe kurzer, zeitlich beschränkter Iterationen, in denen auch produktionsreifer Code programmiert und getestet wird. Das Team erweitert den Satz der Use Cases schrittweise und verfeinert und adaptiert ihn anhand des Feedbacks, das aus der frühen Programmier- und Testarbeit sowie durch Demos gewonnen wurde.

Dies ist ein gekonnter evolutionärer Entwicklungsprozess, kein Wasserfallprozess, dennoch wird noch eine Dosis »Anforderungsrealismus« benötigt. Schriftliche Spezifikationen und andere Modelle vermitteln die *Illusion* der Korrektheit, doch Modelle lügen (unabsichtlich). Erst der Code und die Tests enthüllen die Wahrheit über das, was wirklich gewollt ist und funktioniert.

Die Use Cases, UML-Diagramme usw. werden nicht perfekt sein – garantiert! Es werden wichtige Informationen fehlen, und sie werden falsche Aussagen enthalten. Die Lösung besteht nicht darin, eine Wasserfallhaltung einzunehmen und zu versuchen, die Anforderungen am Anfang in fast perfekter vollständiger Form festzuhalten – obwohl wir natürlich in der verfügbaren Zeit unser Bestes tun und die Best Practices der Anforderungsanalyse lernen und anwenden sollten. Aber es wird nie genug sein!

Dies soll keine Aufforderung sein, sich ohne Analyse und Modellierung direkt auf das Codieren zu stürzen. Es gibt einen Mittelweg zwischen der Wasserfallmethode und einer Ad-hoc-Programmierung: die iterative und evolutionäre Entwicklung. Bei diesem Ansatz werden die Use Cases und Modelle inkrementell durch frühes Programmieren und Testen verfeinert, verifiziert und geklärt.

Sie wissen, dass Sie auf dem falschen Weg sind, wenn das Team versucht, alle oder zumindest die meisten Use Cases ausführlich zu beschreiben, bevor es mit der ersten Entwicklungsiteration beginnt – oder das Gegenteil tut.

6.10 Notation: Gibt es andere Formate? Eine zweispaltige Variante

Einige bevorzugen das zweispaltige Format oder das Konversationsformat, das die Interaktion zwischen den Akteuren und dem System betont. Es wurde erstmals von Rebecca Wirfs-Brock in [Wirfs-Brock93] vorgeschlagen und wird auch von Constantine und Lockwood zur Unterstützung der Usability-Analyse und -Technik [CL99] propagiert. Hier ist derselbe Inhalt im zweispaltigen Format:

Use Case UC1: Process Sale	
Primärakteur: Kassierer	
... wie vorher ...	
Standardablauf:	
Aktion (oder Absicht) des Akteurs	Verantwortlichkeit des Systems
1. Kunde kommt mit Waren und/oder Diensten, die er kaufen will, an die Kasse.	
2. Kassierer beginnt neuen Verkauf.	
3. Kassierer gibt Artikelbezeichnung ein.	
	4. System speichert Verkaufsposition und zeigt Artikelbeschreibung, Preis und laufende Summe an. Preis wird anhand von Preisstellungsregeln berechnet.
<i>Kassierer wiederholt Schritte 3–4, bis er das Eingabeende anzeigt.</i>	5. System zeigt Summe und berechnete Steuern an.
6. Kassierer teilt dem Kunden die Summe mit und bittet um Zahlung.	
7. Kunde bezahlt.	
...	8. System bearbeitet Zahlung.
	9. System protokolliert den abgeschlossenen Verkauf und sendet Informationen an das externe Abrechnungssystem (für Buchhaltung und Provisionsabrechnung) und das Lagerverwaltungssystem (zur Aktualisierung der Lagerbestände). System präsentiert Kaufbeleg.
	...

6.10.1 Das beste Format?

Es gibt nicht *das* beste Format; einige ziehen den einspaltigen, andere den zweispaltigen Stil vor. Abschnitte können hinzugefügt und entfernt werden; Überschriften können geändert werden. Nichts davon ist besonders wichtig; der Schlüssel liegt darin, die Einzelheiten des Standardablaufs und seine Erweiterungen formell zu beschreiben. [Cockburn01] fasst viele nutzbare Formate zusammen.

Persönliche Erfahrung

Das Folgende ist meine Vorgehensweise, keine Empfehlung. Einige Jahre habe ich wegen der klaren optischen Trennung im Dialogverfahren das zweispaltige Format verwendet. Doch dann bin ich zu dem einspaligen Stil zurückgekehrt, weil dieser kompakter und leichter zu formatieren ist, und der geringe Wert eines optisch getrennten Dialogverfahrens diese Vorteile für mich nicht aufwiegt. Ich finde es immer noch einfach, visuell die verschiedenen Parteien im Dialogverfahren (Kunde, System, ...) zu identifizieren, wenn alle Parteien und die Reaktionen des Systems normalerweise in separaten Schritten beschrieben werden.

6.11 Richtlinie: Schreiben Sie in einem wesentlichen UI-freien Stil

6.11.1 Neu und verbessert! Fingerabdruckleser als Option

Auf einem Anforderungsworkshop sagt der Kassierer möglicherweise, eines seiner Ziele sei es, »sich anzumelden«. Wahrscheinlich dachte er dabei an ein GUI, ein Dialogfeld, eine Anwender-ID und ein Passwort. Dies ist ein Mechanismus, um ein Ziel zu erreichen, nicht das Ziel selbst. Durch eine aufwärts gerichtete Untersuchung der Zielhierarchie (»Was ist das Ziel dieses Ziels?«) kommt der Systemanalytiker zu einem Ziel, das von dem Mechanismus unabhängig ist: »mich selbst identifizieren und eine Authentifizierung erhalten« oder einem noch höheren Ziel: »Diebstahl verhindern ...«.

Dieser Prozess der Entdeckung des obersten Ziels kann den Blick für neue und bessere Lösungen öffnen. Ein Beispiel: Tastaturen und Mäuse mit biometrischen Lesern, normalerweise für einen Fingerabdruck, sind jetzt weit verbreitet und preiswert. Wenn das Ziel in einer »Identifikation und Authentisierung« besteht, warum sollte es nicht leicht und schnell mit einem biometrischen Leser auf der Tastatur realisiert werden? Doch eine angemessene Beantwortung dieser Frage umfasst auch eine Usability-Analyse. Sind Ihre Finger mit Fett beschmiert? Haben Sie Finger?

6.11.2 In einem essenziellen Stil schreiben

Diese Idee ist in verschiedenen Use-Case-Richtlinien folgendermaßen zusammengefasst worden: »Lassen Sie die Benutzerschnittstelle außen vor; konzentrieren Sie sich auf die Absicht.« [Cockburn01]. Ihre Begründung und die geeignete Notation sind von Larry Constantine im Kontext einer Erstellung besserer Benutzerschnittstellen (UIs) und des Usability Engineering [Constantine94, CL99] ausführlicher untersucht worden. Constantine bezeichnet den Schreibstil als *essenziell*, wenn er UI-Details weglässt und sich auf die wirkliche Benutzerabsicht konzentriert. (Der Terminus ist von »essenziellen Modellen« in *Essential Systems Analysis* [MP84] abgeleitet.)

Bei einem essenziellen Stil wird die Geschichte auf der Ebene der *Absichten* des Anwenders und der *Verantwortlichkeiten* des Systems statt auf der Ebene ihrer konkreten Aktionen erzählt. Sie bleibt frei von Einzelheiten über Verfahren und Mechanismen, insbesondere solchen, die sich auf das UI beziehen.

Richtlinie

Schreiben Sie Use Cases in einem essenziellen Stil; lassen Sie die Benutzerschnittstelle außen vor und konzentrieren Sie sich auf die Absichten der Akteure.

Bei allen vorangegangenen Beispiel-Use-Cases in diesem Kapitel, wie beispielsweise *Process Sale*, habe ich mich bemüht, einen möglichst essenziellen Stil zu verwenden.

6.11.3 Gegensätzliche Beispiele

Essenzieller Stil

Nehmen Sie an, dass der Use Case *Manage Users* eine Identifikation und Authentifizierung erfordert:

- ...
- 1. Administrator identifiziert sich.
- 2. System authentifiziert Identität.
- 3. ...

Es gibt zahlreiche Optionen, um diese Absichten und Verantwortlichkeiten zu realisieren: biometrische Leser, grafische Benutzerschnittstellen (GUIs) usw.

Konkreter Stil – Bei früher Anforderungsarbeit zu vermeiden

Im Gegensatz dazu zeigt das folgende Beispiel einen *konkreten Use-Case-Stil*. Bei diesem Stil sind bereits Entscheidungen über die Benutzerschnittstelle in den Use-Case-Text eingebettet. Der Text zeigt möglicherweise Screenshots des Fensters, beschreibt die Fensternavigation, die Manipulation von GUI-Komponenten usw. Ein Beispiel:

- ...
- 1. Administrator gibt ID und Passwort in Dialogfeld ein (siehe Bild 3).
- 2. System authentifiziert Administrator.
- 3. System zeigt das »Anwender bearbeiten«-Fenster an (siehe Bild 4).
- 4. ...

Diese konkreten Use Cases können nützlich sein, um die konkrete oder detaillierte GUI-Entwurfsarbeit in einem späteren Schritt zu unterstützen, aber sie sind während der frühen Analysen der Anforderungen unpassend. Während der frühen Anforderungsarbeit gilt: »Lassen Sie die Benutzerschnittstelle außen vor – konzentrieren Sie sich auf die Absichten.«

6.12 Richtlinie: Schreiben Sie knappe Use Cases

Schreiben Sie gerne zahlreiche Anforderungen? Ich glaube, eigentlich nicht. Schreiben Sie deshalb knappe Use Cases. Löschen Sie Wörter, die »Rauschen« verursachen. Selbst kleine Änderungen summieren sich, wie beispielsweise »System authentifiziert ...« statt »Das System authentifiziert ...«.

6.13 Richtlinie: Schreiben Sie Blackbox Use Cases

Blackbox Use Cases sind am weitesten verbreitet und eine empfohlene Art; sie beschreiben nicht die interne Funktion des Systems, nicht seine Komponenten und nicht seinen Entwurf, sondern die *Verantwortlichkeiten des Systems*. Dieses Thema ist ein übergreifender Aspekt des objektorientierten Denkens – Softwarekomponenten haben Verantwortlichkeiten und arbeiten mit anderen Komponenten zusammen, die ebenfalls Verantwortlichkeiten haben.

Indem Sie Systemverantwortlichkeiten durch Blackbox Use Cases beschreiben, können Sie festlegen, *was* das System tun muss (sein Verhalten oder funktionale Anforderungen), ohne festzulegen, *wie* es dies tun soll (den Entwurf). Tatsächlich werden Analyse und Entwurf manchmal als das *Was* im Gegensatz zum *Wie* gegenübergestellt. Dies ist ein wichtiger Aspekt einer guten Softwareentwicklung: Bei der Anforderungsanalyse sollten Sie keine Wie-Entscheidungen treffen, das System als Blackbox betrachten und nur sein externes Verhalten beschreiben. Später beim Entwurf erstellen Sie eine Lösung, die diese Spezifikation erfüllt.

Blackbox-Stil	Kein Blackbox-Stil
Das System registriert den Verkauf.	Das System speichert den Verkauf in einer Datenbank ... oder (noch schlimmer): Das System erzeugt einen SQL-INSERT-Befehl für den Verkauf ...

6.14 Richtlinie: Schreiben Sie aus der Perspektive der Akteure und ihrer Ziele

Das folgende Beispiel zeigt die RUP-Use-Case-Definition des Use-Case-Erfinders Ivar Jacobson:

Ein Satz von Use-Case-Instanzen, wobei jede Instanz eine Folge von Aktionen ist, die ein System ausführt und die ein beobachtbares Ergebnis produziert, das *für einen bestimmten Akteur von Wert* ist.

Der Ausdruck »*beobachtbares Ergebnis, das für einen bestimmten Akteur von Wert ist*« ist ein schwieriges, aber wichtiges Konzept, das Jacobson als kritisch betrachtet, weil es zwei Einstellungen bei der Anforderungsanalyse betont:

- Konzentrieren Sie sich beim Schreiben der Anforderungen auf die Anwender oder Akteure eines Systems und fragen Sie, wie diese das System üblicherweise nutzen und welche Ziele sie dabei verfolgen.
- Konzentrieren Sie sich darauf, was der Akteur als wertvolles Ergebnis betrachtet.

Vielleicht scheint es selbstverständlich zu sein zu betonen, beobachtbare Werte für den Anwender zu erzeugen und sich auf die typischen Ziele der Anwender zu konzentrieren, aber die Softwareindustrie ist mit gescheiterten Projekten gepflastert, die nicht geliefert haben, was die Leute wirklich benötigten. Der alte Ansatz der Features- und Funktionenliste, um Anforderungen festzuhalten, kann zu einem negativen Ergebnis beitragen, weil er nicht dazu anleitet zu fragen, wer das Produkt verwendet und was einen Wert erzeugt.

→ *Siehe auch: Abschnitt 6.19, Motivation: Andere Vorteile von Use Cases? Anforderungen im Kontext*

6.15 Richtlinie: Wie man Use Cases findet

Use Cases werden definiert, um die Ziele der Primärakteure zu erfüllen. Deshalb läuft die grundlegende Prozedur folgendermaßen ab:

1. Grenzen Sie das System ab. Handelt es sich nur um eine Softwareanwendung? Bilden Hardware und Anwendung eine Einheit? Dies plus eine Person, die das System verwendet? Oder eine ganze Organisation?
2. Identifizieren Sie die Primärakteure – die Akteure, die die Dienste des Systems nutzen, um bestimmte Ziele zu erreichen.
3. Identifizieren Sie die Ziele jedes Primärakteurs.
4. Definieren Sie Use Cases, die Anwenderziele erfüllen; benennen Sie sie nach ihrem Ziel. Normalerweise entsprechen Use Cases auf Anwenderzeilebene den Anwenderzielen eins zu eins, doch es gibt wenigstens eine Ausnahme, die weiter unten behandelt wird.

Natürlich werden bei einer iterativen und evolutionären Entwicklung nicht alle Ziele oder Use Cases gleich am Anfang komplett oder korrekt identifiziert, sondern nach und nach entdeckt.

6.15.1 Schritt 1: Grenzen Sie das System ab

In dieser Fallstudie ist das POS-System selbst das zu entwerfende System; alles andere liegt jenseits der Systemgrenze, einschließlich des Kassierers, des Payment Authorization Service usw.

Wenn die Definition der Grenzen des zu entwerfenden Systems nicht klar ist, kann sie durch weitere Definition der Komponenten geklärt werden, die außerhalb des Systems liegen – die externen Primärakteure und die unterstützenden Akteure. Wenn die externen Akteure identifiziert worden sind, wird die Grenze deutlicher. Liegt beispielsweise die Verantwortlichkeit für die Zahlungsgenehmigung vollständig innerhalb der Systemgrenzen? Nein – es gibt einen externen Akteur, den Payment Authorization Service.

6.15.2 Schritte 2 und 3: Identifizieren Sie die Primärakteure und ihre Ziele

Es ist eine unnatürliche Einschränkung, die Primärakteure streng linear vor den Anwenderzielen zu identifizieren; in einem Anforderungsworkshop läuft ein Brainstorming ab, bei dem die Teilnehmer eine Mischung aus beidem produzieren. Manchmal enthüllen Ziele die Akteure oder umgekehrt.

Richtlinie: Ermitteln Sie zuerst die hauptsächlichen Akteure in einem Brainstorming, da diese den Rahmen für die weitere Untersuchung festlegen.

Gibt es Fragen, die helfen, Akteure und Ziele zu finden?

Zusätzlich zu den offenkundigen Primärakteuren und Zielen können die folgenden Fragen helfen, andere zu identifizieren, die übersehen worden sind:

- Wer startet und stoppt das System?
- Wer ist für die Systemadministration zuständig?
- Wer ist für das Anwender- und Sicherheitsmanagement zuständig?
- Ist die »Zeit« ein Akteur, weil das System auf zeitliche Ereignisse reagiert?
- Gibt es einen Überwachungsprozess, der das System bei einem Absturz neu startet?
- Wer evaluiert Systemaktivitäten oder Performanz?
- Wie werden Softwareupdates gehandhabt? Push- oder Pull-Aktualisierung?
- Wer bewertet Protokolle? Werden sie durch Fernabfragen abgerufen?
- Gibt es zusätzlich zu *menschlichen* Primärakteuren externe Software oder robotische Systeme, die die Dienste des Systems in Anspruch nehmen?
- Wer wird bei Fehlern oder Misserfolgen benachrichtigt?

Wie sollten Akteure und Ziele organisiert werden?

Es gibt wenigstens zwei Ansätze:

1. Wenn Sie die Ergebnisse entdecken, stellen Sie sie in einem Use-Case-Diagramm dar, wobei Sie die Ziele als Use Cases bezeichnen.
2. Stellen Sie zuerst eine Akteur-Ziele-Liste zusammen, begutachten und verfeinern Sie sie und erstellen Sie dann das Use-Case-Diagramm.

→ **Siehe auch:** Abschnitt 6.17, UML anwenden: Use-Case-Diagramme

Wenn Sie eine Liste der Akteursziele erstellen, gehört sie in einen Abschnitt des Vision-Artefakts.

Ein Beispiel:

Akteur	Ziel	Akteur	Ziel
Kassierer	Process Sales Process Rentals Handle Returns Cash in Cash out ...	System-administrator	Anwender hinzufügen Anwender ändern Anwender löschen Sicherheit verwalten Systemtabellen verwalten ...
Manager	Start up Shut down ...	Sales Activity System	Verkäufe und Performanzdaten analysieren
...

Das Sales Activity System ist eine Fernanwendung, die häufig Verkaufsdaten von jedem POS-Knoten in dem Netzwerk abrufen wird.

Warum wird nach Zielen der Akteure statt nach Use Cases gefragt?

Akteure haben Ziele und benutzen Anwendungen, um sie zu erreichen. Der Standpunkt der Use-Case-Modellierung besteht darin, diese Akteure und ihre Ziele zu finden und nach Lösungen zu suchen, die ein Ergebnis von Wert produzieren. Dies bedeutet für den Use-Case-Modellierer eine leichte Verschiebung des Schwerpunkts. Statt zu fragen: »Was sind die Aufgaben?«, beginnt er zu fragen: »Wer benutzt das System und welche Ziele verfolgt er damit?« Tatsächlich sollte der Name eines Use Case für ein Anwenderziel dieses Ziel ausdrücken, um diesen Standpunkt zu betonen – Ziel: Aufnahme oder Verarbeitung eines Verkaufs; Use Case: *Process Sale*.

Deshalb lautet das Schlüsselkonzept für die Untersuchung von Anforderungen und Use Cases:

Stellen Sie sich vor, wir sind zusammen auf einem Anforderungsworkshop. Wir könnten eine der beiden folgenden Fragen stellen:

- »Was tun Sie?« (grob gesagt, eine aufgabenorientierte Frage) oder
- »Was sind Ihre Ziele, deren Ergebnisse einen messbaren Wert haben?«

Ziehen Sie die zweite Frage vor.

Antworten auf die erste Frage geben wahrscheinlich eher aktuelle Lösungen und Prozeduren sowie die damit verbundenen Komplikationen wieder.

Antworten auf die zweite Frage, insbesondere in Kombination mit einer Untersuchung, die auf höhere Ziele in der Zielhierarchie aus ist (»Was ist das oberste Ziel?«), öffnen den Blick für neue und bessere Lösungen, konzentrieren sich auf die Schaffung eines geschäftlichen Mehrwerts und zielen auf den Kern dessen, was die Stakeholder von dem System erwarten.

Ist der Kassierer oder der Kunde der Primärakteur?

Warum ist der Kassierer und nicht der Kunde in dem Use Case *Process Sale* ein Primärakteur?

Die Antwort hängt von der Systemgrenze des zu entwerfenden Systems ab und für wen wir das System hauptsächlich entwerfen (siehe Abbildung 6.2). Wenn wir das Unternehmen oder den Kassendienst als ein zusammengefügtes System betrachten, *ist* der Kunde ein Primärakteur, der das Ziel hat, Waren oder Dienste zu bekommen und dann wegzugehen. Doch vom Standpunkt des POS-Systems aus (das wir für diese Fallstudie als Systemgrenze gewählt haben) dient das System den Zielen eines ausgebildeten Kassierers (und des Geschäfts), einen Verkauf an einen Kunden zu verarbeiten. Dabei gehen wir von einer traditionellen Kassenumgebung mit einem Kassierer aus, obwohl die Zahl der POS-Systeme zunimmt, die der Kunde selbst direkt bedienen kann.

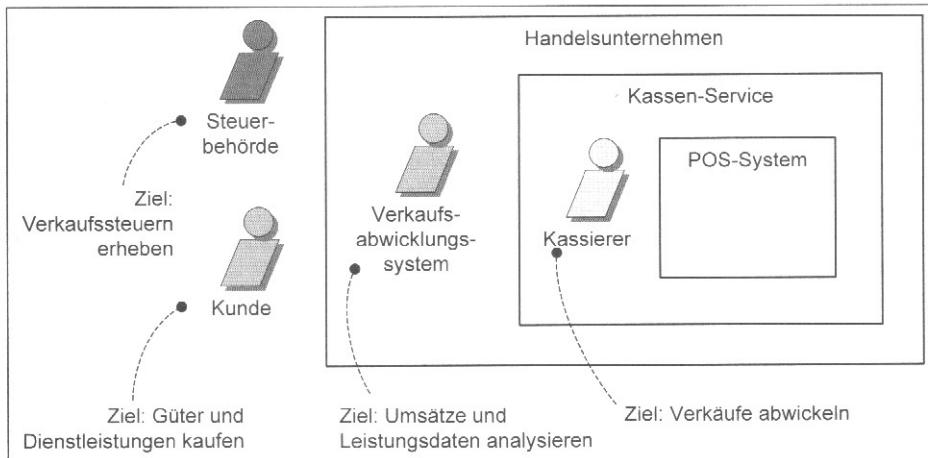


Abbildung 6.2: Primärakteure und Ziele bei verschiedenen Systemgrenzen

Der Kunde *ist* ein Akteur, aber im Kontext des NextGen-POS *kein* Primärakteur; stattdessen ist der Kassierer der Primärakteur, weil das System dafür konzipiert ist, hauptsächlich die Ziele eines ausgebildeten Kassierers zu erfüllen (einen Verkauf schnell abzuwickeln, Preise abzurufen usw.). Das System verfügt weder über ein UI noch über Funktionen, die gleichermaßen von dem Kunden oder dem Kassierer gebraucht werden könnten, sondern es ist darauf optimiert worden, die Bedürfnisse eines geschulten Kassierers zu erfüllen. Ein Kunde vor dem POS-Terminal könnte dieses nicht effektiv nutzen. Anders ausgedrückt: Es wurde für den Kassierer, nicht für den Kunden entworfen, und deshalb ist der Kassierer nicht einfach ein Proxy (Stellvertreter) für den Kunden.

Betrachten Sie dagegen eine Website zum Verkauf von Tickets, die identisch ist, egal ob ein Kunde sie direkt besucht oder ob ein Telefonagent sie bei einem Anruf eines Kunden besucht. In diesem Fall agiert der Agent einfach als Proxy (Stellvertreter) des Kunden – das System wurde nicht konzipiert, um spezielle Ziele des Agents zu erfüllen. Dann wäre es richtig, den Kunden und nicht den Telefonagenten als Primärakteur zu zeigen.

Andere Methoden, um Akteure und Ziele zu finden? Ereignisanalyse

Ein anderer Ansatz, um Analytiker beim Finden von Akteuren, Zielen und Use Cases zu unterstützen, besteht darin, externe Ereignisse zu identifizieren. Was sind sie, woher kommen sie und warum treten sie ein? Oft gehört eine Gruppe von Ereignissen zu demselben Use Case. Ein Beispiel:

Externes Ereignis	Von Akteur	Ziel/Use Case
Verkaufposition eingeben	Kassierer	Einen Verkauf abwickeln
Zahlung eingeben	Kassierer oder Kunde	Einen Verkauf abwickeln
...		

6.15.3 Schritt 4: Use Cases definieren

Im Allgemeinen sollten Sie einen Use Case für jedes Anwenderziel definieren. Benennen Sie den Use Case ähnlich wie das Anwenderziel – beispielsweise Ziel: Process Sale; Use Case: *Process Sale* (beides im Deutschen: *Verkauf abwickeln*)

Verwenden Sie ein Objekt und dann ein Verb.

Eine übliche Ausnahme zu der Regel, einen Use Case pro Ziel zu verwenden, besteht darin, separate CRUD-Ziele (Abkürzung für *create, retrieve, update, delete* – dt. *erstellen, abrufen, aktualisieren, löschen*) in einem CRUD-Use Case zusammenzufassen. Ein solcher Use Case wird idiomatisch als *Manage <X>* bezeichnet. Ein Beispiel: Die Ziele »Anwender editieren«, »Anwender löschen« usw. werden alle durch den Use Case *Manage Users* (*Anwender verwalten*) erfüllt.

6.16 Richtlinie: Welche Tests können dazu beitragen, nützliche Use Cases zu finden?

Welcher der folgenden Ausdrücke ist ein gültiger Use Case?

- Einen Liefervertrag aushandeln
- Rückgaben abwickeln
- Anmelden
- Eine Figur auf dem Spielfeld verschieben

Man könnte argumentieren, dass alle genannten Ausdrücke je nach Systemgrenze, Akteuren und Zielen Use Cases auf verschiedenen Ebenen beschreiben.

Doch statt allgemein zu fragen: »Was ist ein gültiger Use Case?«, lautet eine praktischere Frage: »Welche Ebene ist nützlich, um Use Cases für die Analyse der Anforderungen einer Anwendung auszudrücken?« Dafür gibt es mehrere Faustregeln, unter anderem:

- Der Boss-Test
- Der EBP-Test
- Der Größtentest

6.16.1 Der Boss-Test

Ihr Chef (engl. *boss*) fragt Sie: »Was haben Sie den ganzen Tag gemacht?« Sie antworten: »Anmeldung!« Ob Ihr Boss damit zufrieden ist?

Falls nicht, besteht der Use Case den Boss-Test nicht. Dies impliziert, dass der Use Case nicht viel damit zu tun hat, Ergebnisse mit einem messbaren Wert zu erzeugen. Es kann auf einer niedrigeren Zielebene ein Use Case sein, liegt aber nicht auf der Fokusebene, die in der Anforderungsanalyse angestrebt wird.

Dies bedeutet nicht, dass Use Cases, die den Boss-Test nicht bestehen, immer ignoriert werden sollten. Eine Authentifizierung des Anwenders mag zwar den Boss-Test nicht bestehen, kann aber wichtig und schwierig sein.

6.16.2 Der EBP-Test

Ein *Elementary Business Process* (EBP, dt. *elementarer Geschäftsprozess*) ist ein Konzept aus dem Bereich des Business Process Engineering. (EBP ist vergleichbar mit dem Terminus *User Task* (Anwenderaufgabe) aus dem Usability Engineering, obwohl die Bedeutung weniger strikt als in diesem Fachbereich ist.) EBP wird folgendermaßen definiert:

Eine Aufgabe, die von einer Person an einem Ort zu einem Zeitpunkt als Reaktion auf ein Geschäftsergebnis ausgeführt wird und die einen messbaren geschäftlichen Mehrwert schafft und die Daten in einem konsistenten Zustand lässt, beispielsweise Approve Credit (Kredit genehmigen) oder Price Order (Auftragspreis berechnen). [ursprüngliche Quelle unbekannt]

Konzentrieren Sie sich auf Use Cases, die EBPs beschreiben.

Der EBP-Test ähnelt dem Boss-Test besonders im Hinblick auf das Attribut des messbaren geschäftlichen Mehrwerts.

Die Definition darf nicht zu wörtlich genommen werden: Scheitert der Test, weil für einen EBP zwei Leute erforderlich sind oder wenn eine Person herumgehen muss? Wahrscheinlich nicht, aber die Definition grenzt die gemeinten Aufgaben brauchbar ab. Es geht nicht um einzelne kleine Schritte wie »eine Position löschen« oder »das Dokument drucken«, sondern der Standardablauf besteht wahrscheinlich aus fünf oder zehn Schritten. Er benötigt nicht Tage oder mehrere Sitzungen wie beispielsweise die Aufgabe »einen Liefervertrag aushandeln«, sondern es handelt sich um eine Aufgabe, die während einer einzelnen Sitzung erledigt wird. Ihre Dauer beträgt wahrscheinlich einige Minuten bis zu einer Stunde. Wie bei der Definition im UP soll die Aufgabe einen beobachtbaren oder messbaren geschäftlichen Mehrwert schaffen und das System und die Daten in einem stabilen und konsistenten Zustand lassen.

6.16.3 Der Größtentest

Ein Use Case besteht sehr selten aus einer einzelnen Aktion oder einem einzelnen Schritt, sondern enthält üblicherweise viele Schritte. Ein Use Case im voll ausgearbeiteten Format erfordert häufig drei bis zehn Textseiten. Häufig wird bei der Use-Case-Modellierung der Fehler gemacht, einen einzelnen Schritt innerhalb einer Folge verwandter Schritte selbst als Use Case zu definieren, indem beispielsweise ein separater Use Case namens *Enter an Item ID* definiert wird. Eine geringe Use-Case-Größe kann ein Indiz für einen solchen Fehler sein – der Use-Case-Name beschreibt dann fälschlicherweise einfach einen Schritt innerhalb einer längeren Folge von Schritten, und wenn Sie sich die Länge des voll ausgearbeiteten Textes vorstellen, wäre dieser Schritt außerordentlich kurz.

6.16.4 Beispiel: Die Tests anwenden

- Einen Liefervertrag aushandeln
 - Viel umfangreicher und länger als ein EBP. Könnte als *Business Use Case* statt als System Use Case modelliert werden.
- Retouren bearbeiten
 - Für den Boss OK. Sieht wie ein EBP aus. Größe ist gut.
- Anmelden
 - Boss ist nicht glücklich, wenn Sie dies den ganzen Tag lang tun!
- Eine Figur auf dem Spielfeld verschieben
 - Einzelter Schritt – fällt beim Größtentest durch.

6.16.5 Vernünftige Verstöße gegen die Tests

Die Mehrheit der identifizierten und analysierten Use Cases für eine Anwendung sollte die Tests bestehen. Doch es gibt einige gebräuchliche Ausnahmen.

Manchmal ist es nützlich, separate Use Cases zu schreiben, die auf einer Subfunktionsebene liegen und Teil-aufgaben oder Schritte innerhalb eines normalen Use Case auf EBP-Ebene repräsentieren. Ein Beispiel: Eine Teilaufgabe oder Erweiterung wie beispielsweise »Paying per Credit (Bezahlung per Kreditkarte)« kann in vielen grundlegenden Use Cases wiederholt werden. Falls dies der Fall ist, ist es wünschenswert, sie in einen separaten Use Case auszulagern, auch wenn sie den EBP-Test oder den Größtentest nicht wirklich bestehen, und sie mit mehreren Basis Use Cases zu verknüpfen, um eine Duplikierung des Textes zu vermeiden.

→ **Siehe auch:** Abschnitt 30.1, Die *Include*-Beziehung

7.4 NextGen-Beispiel: (Partielle) Supplementary Specification

Supplementary Specification

Revisionsverlauf

Version	Datum	Beschreibung	Autor
Inception-Entwurf	10. Jan 2031	Erster Entwurf. Muss hauptsächlich während der Elaboration verfeinert werden.	Craig Larman

Einführung

Dieses Dokument ist das Repozitorium aller NextGen-POS-Anforderungen, die nicht in den Use Cases erfasst sind.

Funktionalität

(Funktionalität, die vielen Use Cases gemeinsam ist)

Protokollierung und Fehlerhandhabung

Alle Fehler in einem persistenten Speicher protokollieren.

Pluggable Rules

An verschiedenen Szenariopunkten mehrerer (zu definierender) Use Cases die Fähigkeit unterstützen, die Funktionalität des Systems durch einen Satz willkürlicher Regeln anzupassen, die an diesem Punkt oder bei diesem Ereignis ausgeführt werden.

Sicherheit

Jede Nutzung erfordert eine Authentifizierung des Anwenders.

Usability

Human Factors

Der Kunde kann eine große Monitoranzeige des POS sehen. Deshalb:

- Text sollte aus einem Meter Entfernung leicht lesbar sein.
- Farben vermeiden, die von verbreiteten Formen der Farbenblindheit betroffen sind.

Geschwindigkeit, Einfachheit und fehlerfreie Verarbeitung sind bei der Verkaufsabwicklung von überragender Bedeutung, da der Käufer schnell gehen möchte und andernfalls die Käuferfahrung (und den Verkäufer) als weniger positiv erlebt.

Der Kassierer sieht häufig den Kunden oder die Artikel, nicht die Computeranzeige an. Deshalb sollten Signale und Warnungen auch akustisch und nicht nur grafisch angezeigt werden.

Zuverlässigkeit

Wiederanlauffähigkeit

Wenn externe Dienste (Payment Authorization, Buchhaltungssystem, ...) nicht verwendet werden können, sollte eine lokale Lösung versucht werden (beispielsweise speichern und weiterleiten), um einen Verkauf trotzdem abschließen zu können. Hier ist eine erheblich umfangreichere Analyse erforderlich ...

Performanz

Wie bereits unter *Human Factors* erwähnt worden ist, möchte der Käufer seinen Kauf sehr schnell abwickeln. Ein Engpass ist die externe Zahlungsgenehmigung. Unser Ziel: Autorisierung in weniger als einer Minute, in 90% aller Fälle.

Wartbarkeit

Anpassungsfähigkeit

Verschiedene Kunden des NextGen POS verfügen über separate Geschäftsregeln und haben unterschiedliche Verarbeitungsbedürfnisse bei der Abwicklung eines Verkaufs. Deshalb ist es an mehreren definierten Punkten des Szenarios (beispielsweise wenn ein neuer Verkauf eingeleitet wird, wenn eine neue Position hinzugefügt wird) möglich, eigene Geschäftsregeln (»pluggable rules«) einzufügen.

Konfigurierbarkeit

Verschiedene Kunden wünschen variable Netzwerkkonfigurationen für ihre POS-Systeme, wie beispielsweise dicke oder dünne Clients, zweischichtige oder N-schichtige physische Schichten usw. Außerdem möchten sie diese Konfigurationen ändern können, um sie an wechselnde Geschäfts- und Performanzanforderungen anzupassen. Deshalb wird das System in gewissem Maß konfigurierbar sein, um diesen Anforderungen gerecht zu werden. In diesem Bereich ist eine erheblich umfangreichere Analyse erforderlich, um die Bereiche, den Grad der Flexibilität und den erforderlichen Aufwand zu ermitteln, um die Konfiguration zu ermöglichen.

Implementierungsbedingungen

Die NextGen-Führung besteht auf einer Java-basierten Lösung und sagt voraus, dass dies langfristig die Portierbarkeit und Wartbarkeit verbessern und die Entwicklung erleichtern wird.

Gekaufte Komponenten

- Steuerrechner. Muss einbaufähige Komponenten für verschiedene Länder unterstützen.

Kostenlose Open-Source-Komponenten

In Allgemeinen empfehlen wir, in diesem Projekt möglichst viele kostenlose Java-basierte Open-Source-Komponenten zu verwenden.

Obwohl es zu früh ist, um Komponenten definitiv zu entwerfen und auszuwählen, schlagen wir die folgenden Komponenten als mögliche Kandidaten vor:

- JLog Logging Framework (Protokollierungs-Framework)
- ...

Interfaces

Beachtenswerte Hardware und Interfaces

- Touchscreen-Monitor (wird von Betriebssystemen als normaler Monitor behandelt; die Berührungsgerüste werden als Maus-Ereignisse interpretiert)
- Barcode-Laserscanner (diese sind normalerweise an eine spezielle Tastatur angeschlossen und der gescannte Input wird in der Software als Tastaturanschläge interpretiert)
- Belegdrucker
- Kredit/Debit-Kartenleser
- Unterschriftenleser (aber nicht in Release 1)

Softwareschnittstellen

Bei den meisten externen Systemen (Steuerrechner, Buchhaltung, Lagerbestand, ...), mit denen wir zusammenarbeiten, müssen wir fähig sein, verschiedene Systeme und deshalb verschiedene Schnittstellen zu bedienen.

Anwendungsspezifische Domain (Business) Rules

(Siehe das separate Business-Rules-Dokument für allgemeine Regeln.)

ID	Regel	Änderbarkeit	Quelle
RULE1	Käuferrabattregeln. Beispiele: Mitarbeiter – 20% Rabatt. Bevorzugter Kunde – 10% Rabatt. Rentner – 15% Rabatt.	Hoch. Jeder Händler verwendet andere Regeln.	Händlerpolitik.
RULE2	Verkaufsrabattregeln (auf Transaktionsebene). Gilt für die Summe vor Steuern. Beispiele: 10% Rabatt, wenn Summe größer als 100 USD. 5% Rabatt an jedem Montag. 10% Rabatt auf alle Verkäufe heute von 10:00 bis 15:00 Uhr.	Hoch. Jeder Händler verwendet andere Regeln, und sie können sich täglich oder stündlich ändern.	Händlerpolitik.
RULE3	Produktrabattregeln (auf Positionsebene). Beispiele: 50% Rabatt auf Tofu heute von 9:00 bis 10:00 Uhr. 10% Rabatt auf Traktoren diese Woche. Kaufen Sie zwei Veggie-Burger, erhalten Sie einen umsonst.	Hoch. Jeder Händler verwendet andere Regeln, und sie können sich täglich oder stündlich ändern.	Händlerpolitik.

Rechtliche Probleme

Wir empfehlen den Einsatz von Open-Source-Komponenten, wenn ihre Lizenzierung den Wiederverkauf von Produkten zulässt, die Open-Source-Software enthalten.

Alle Steuervorschriften müssen per Gesetz während der Verkäufe angewendet werden. Beachten Sie, dass sich die Gesetze häufig ändern können.

Informationen über relevante Bereiche

Preisstellung

Zusätzlich zu den Preisstellungsregeln, die in dem Abschnitt über die Domänenregeln beschrieben wurden, beachten Sie, dass die Produkte einen *Originalpreis* und optional einen *Dauertiefpreis* haben (können). Der Preis eines Produkts ist sein Dauertiefpreis vor der Anwendung weiterer Rabatte, falls vorhanden. Unternehmen speichern den Originalpreis, auch wenn es einen Dauertiefpreis gibt, aus buchhalterischen und steuerlichen Gründen.

Abwicklung von Zahlungen per Kreditkarte und Guthaben

Wenn eine Zahlung per Kreditkarte und Guthaben von einem Payment Authorization Service genehmigt wird, ist dieser für die Bezahlung des Verkäufers verantwortlich, nicht der Käufer. Folglich muss der Verkäufer in seiner Debitorenbuchhaltung für jede Zahlung die entsprechende Forderung an den Autorisierungsdienst registrieren. Normalerweise gleicht der Autorisierungsdienst nächtlich die Forderungen durch elektronische Überweisung des geschuldeten Gesamtbetrags des Tages (abzüglich einer geringen Transaktionsgebühr für seine Dienste) auf das Konto des Verkäufers aus.

Umsatzsteuer

Die Umsatzsteuerberechnung kann sehr komplex sein und sich regelmäßig als Reaktion auf gesetzliche Änderungen auf diversen Ebenen der Regierung ändern. Deshalb ist es ratsam, die Steuerberechnungen an eine entsprechende Software eines Drittanbieters zu delegieren (von denen es mehrere gibt). Steuerschulden können gegenüber der Kommune, dem Land oder dem Bund entstehen. Einige Artikel sind unbedingt steuerfrei, bei anderen kann die Steuer vom Status des Käufers oder Nutzers des Produkts abhängen (beispielsweise Rentner, Bauer oder Kind).