mitp

Alistair Cockburn

Use Cases
effektiv erstellen

Das Fundament für gute Software-Entwicklung

Geschäftsprozesse modellieren mit Use Cases

Die Regeln für Use Cases sicher beherrschen

Kapitel 2

Use Cases als Übereinkunft zum Verhalten

Das zur Diskussion stehende System ist ein Mechanismus, der eine Übereinkunft zwischen verschiedenen Stakeholdern ausführt. Use Cases bilden den Teil der Übereinkunft, der das Verhalten betrifft. Jeder Satz in einem Use Case beschreibt eine Aktion, die ein bestimmtes Stakeholder-Interesse wahrt oder erweitert. Ein Satz kann eine Interaktion zwischen zwei Akteuren beschreiben oder die interne Systemreaktion, die zur Wahrung der Stakeholder-Interessen erforderlich ist.

Sehen wir uns Use Cases zunächst unter dem Aspekt an, dass sie Interaktionen von Akteuren zeigen, die ein Ziel verfolgen. Danach erweitern wir die Besprechung und behandeln Use Cases als Übereinkunft von Stakeholdern, die Interessen haben. Nennen wir den ersten Teil das konzeptuelle Modell der **Akteure und Ziele** und den zweiten das konzeptuelle Modell der **Stakeholder und Interessen**.

2.1 Interaktionen von Akteuren, die ein Ziel verfolgen

2.1.1 Akteure haben Ziele

Stellen Sie sich einen Angestellten vor, der Serviceanforderungen am Telefon entgegennimmt (der Angestellte ist Primärakteur in Abbildung 2.1). Im Fall eines Anrufs hat der Angestellte ein Ziel: Der Computer soll die Anforderung registrieren und die Bearbeitung auslösen.

Das System hat die Aufgabe, die Serviceanforderung aus unserem Beispiel zu registrieren und die Bearbeitung auszulösen. (Es hat sogar die Aufgabe, die Interessen *aller* Stakeholder zu wahren. Der Angestellte und Primärakteur ist nur einer von ihnen. Im Moment wollen wir uns aber lediglich auf die Systemaufgabe konzentrieren, dem Primärakteur einen Dienst zu erweisen.)

Um die Aufgabe durchzuführen, setzt das System Teilziele. Es kann einige davon intern ausführen, braucht aber die Hilfe eines *unterstützenden Akteurs*, um andere auszuführen. Dieser unterstützende Akteur kann ein Subsystem für den Ausdruck sein oder auch eine andere Organisation, etwa eine Partnerfirma oder eine Regierungsbehörde.

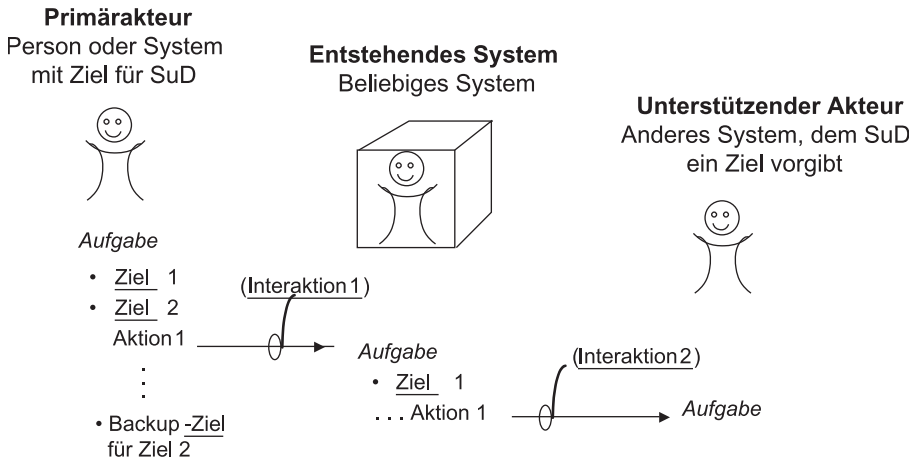

Abb. 2.1: Akteur mit Ziel ruft anderen Akteur auf, um Aufgaben zu erledigen

Der unterstützende Akteur führt die gewünschte Aufgabe in der Regel durch und liefert dem zur Diskussion stehenden System (System under discussion = SuD) ein Teilziel. Das SuD interagiert mit externen Akteuren. Es erreicht seine Teilziele in einer bestimmten Reihenfolge, bis seine Aufgabe schließlich erledigt ist und der gewünschte Service erbracht wurde.

Die Ausgabe des erwünschten Dienstes ist das oberste Ziel, das über Teilziele erreicht wird. Teilziele können unbegrenzt in weitere nachrangige Teilziele zerlegt werden. Wenn wir die Aufgaben der Akteure zergliedern, ist die Anzahl der aufzulistenden nachrangigen Teilziele nach oben offen. Der irische Satiriker und Dichter Jonathan Swift hat das in *On Poetry, A Rhapsody* so beschrieben (allerdings nicht im Zusammenhang mit Use Cases):

> *So, naturalists observe, a flea*
>
> *Hath smaller fleas that on him prey*
>
> *And these have smaller still to bite 'em*
>
> *And so proceed ad infinitum.*

Das schwierigste Stück Arbeit beim Verfassen von Use Cases ist vielleicht, die Flöhe auf den Flöhen unter Kontrolle zu halten – die nachrangigen Teilziele im Schreibprozess. Mehr zu diesem Punkt finden Sie in den Kapiteln 5, 7 und 20.

Das Konzeptmodell der Akteure und Ziele ist praktisch, denn es passt genauso gut auf Geschäfts- wie auf Computersysteme. Akteure können Einzelpersonen, Organisationen oder Computer sein. Es lassen sich gemischte Systeme beschreiben, die aus Personen, Unternehmen und Computern zusammengesetzt sind. Oder wir können ein Software-System beschreiben, das durch ein anderes Computersystem

gesteuert wird, das wiederum einen Menschen als unterstützenden Akteur aufruft oder eine Organisation, die ein Computersystem oder eine Einzelperson aufruft. Es handelt sich um ein nützliches und allgemeines Modell.

2.1.2 Ziele können verfehlt werden

Was soll der Angestellte tun, der einen Kunden am Telefon hat, wenn der Computer mitten in der Aufnahme der Kundenanforderung abstürzt? Wenn das System die erwünschte Aufgabe nicht erfüllen kann, muss sich der Angestellte ein *Backup*-Ziel überlegen und in diesem Fall wohl Papier und Bleistift hernehmen. Der Angestellte hat immer noch die Hauptverantwortung bei der Arbeit und muss einen Plan in petto haben, wenn das System seine Aufgabe nicht erledigen kann.

Genauso gut kann das System bei einem seiner Teilziele einen Fehlschlag erleiden. Es ist möglich, dass der Primärakteur die falschen Daten eingegeben hat, dass ein interner Fehler vorliegt oder dass der unterstützende Akteur den gewünschten Service nicht erbringen kann. Wie soll sich das System dann verhalten? Hier beginnen die Verhaltensanforderungen eines SuD spannend zu werden.

In manchen Fällen kann das System den Fehler korrigieren und den Normalablauf seines Verhaltens wieder aufnehmen. Manchmal muss es von seinem Ziel Abstand nehmen. Wenn Sie versuchen, von Ihrem Bankautomat Geld abzuheben und Ihr Limit überschreiten, wird das Ziel des Geldabhebens fehlschlagen. Es wird auch dann fehlschlagen, wenn der Bankautomat seine Verbindung mit dem Netzwerk-Computer verliert. Wenn Sie lediglich eine falsche Codenummer eingeben, gibt das System Ihnen eine erneute Gelegenheit, die Nummer richtig einzutippen.

Use Cases geben unter anderem deshalb so gute Beschreibungen des Systemverhaltens und der funktionalen Anforderungen im Allgemeinen ab, weil sie den Schwerpunkt auf Zielverfehlungen und die Reaktion auf Fehler legen. Wer sich schon einmal mit der funktionalen Dekomposition und Datenflüssen beschäftigt hat, weiß dies als einen der wichtigsten Vorzüge der Use Cases zu schätzen.

2.1.3 Interaktionen sind mehrstufig

Die einfachste Interaktion ist das Senden einer Nachricht. Sagt man beim Betreten eines Saals »Hallo Jean«, dann ist das eine einfache Interaktion. Die entsprechende einfache Interaktion in der prozeduralen Programmierung ist der Funktionsaufruf, etwa `print(value)`. In der objektorientierten Programmierung wäre die Entsprechung ein Objekt, das an ein anderes Objekt eine Nachricht sendet: `objectA->print(value)`.

Eine *Nachrichtensequenz* oder ein *Szenario* ist eine mehrstufige Interaktion. Wenn ich in einen Getränkeautomat eine 5-$-Note für ein Getränk eingebe, das 80 Cent kostet und aufgefordert werde, den korrekten Betrag einzuwerfen, sieht die Interaktion mit dem Automat so aus.

1. Ich führe den Geldschein ein.
2. Ich drücke auf »Cola«.
3. Der Automat gibt an: »Exakter Betrag erforderlich«.
4. Ich drücke auf die Rückgabetaste.
5. Der Automat gibt den eingezahlten Betrag in Münzen zurück.
6. Ich nehme die Münzen und gehe.

Wir können eine Sequenz so zusammenfassen, als ob es sich um einen Einzelschritt handelte (»Ich versuchte, eine Cola aus dem Automat herauszulassen, hätte aber dafür den genauen Betrag gebraucht.«) und den zusammenfassenden Schritt in eine größere Sequenz stellen:

1. Ich ging zu meiner Bank, um Geld abzuheben.
2. Ich versuchte, eine Cola aus dem Automat herauszulassen, hätte aber dafür den genauen Betrag gebraucht.
3. Ich lief zur Cafeteria und habe dort eine gekauft.

Auf diese Weise können Interaktionen genau wie Ziele zusammengefasst oder zerlegt werden. Jeder Schritt eines Szenarios erfasst ein Ziel und kann somit zu einem eigenständigen Use Case ausgearbeitet werden. Wie man sieht, gibt es bei den Interaktionen Flöhe mit Flöhen, wie das schon bei den Zielen der Fall war.

Erfreulicherweise können wir das Systemverhalten auf einer sehr hohen Ebene darstellen und Ziele und Interaktionen zusammenfassen. Zerlegt man sie Schritt für Schritt, kann das Systemverhalten so präzise spezifiziert werden wie gewünscht. Ich nenne eine Reihe von Use Cases auch gerne eine *unendliche Geschichte*. Unsere Arbeit besteht darin, diese Geschichte so zu schreiben, dass der Leser sich in ihr bequem zurechtfindet.

Der aufmerksame Leser wird bemerkt haben, dass ich den Begriff *Sequenz* recht locker verwende. In vielen Fällen müssen die Interaktionen nicht in einer bestimmten Sequenz ablaufen. Um obiges Getränk für 80 Cent zu kaufen, kann ich drei Zwanzig-, ein Zehn- und zwei Fünf-Cent-Stücke einwerfen oder... (führen Sie die Liste selbst zu Ende). Welches Geldstück zuerst eingeworfen wird, ist ohne Belang.

Offiziell ist *Sequenz* nicht der korrekte Begriff. In der Mathematik heißt die korrekte Bezeichnung *partielle Ordnung*. *Sequenz* ist aber kürzer, trifft den Punkt und wird von den Autoren der Use Cases leichter verstanden. Fragt jemand nach parallel ablaufenden Nachrichten, fordern Sie ihn einfach auf, etwas darüber zu schreiben. Meiner Erfahrung nach braucht niemand viel Anleitung, um wunderbare und klare Beschreibungen zu liefern. Ich bleibe daher beim Begriff *Sequenz*. Ein Beispiel für einen komplexen Sequenzablauf finden Sie in Use Case 22.

Wenn Sie vorhaben, eine formale Sprache für Use Cases zu gestalten, geraten Sie an diesem Punkt vielleicht in Schwierigkeiten. Die meisten Sprachendesigner zwingen den Autor dazu, entweder alle potenziellen Anordnungen aufzulisten oder komplexe Notationen anzuwenden, die die beliebige Anordnung von Ereignissen zulassen. Aber wir schreiben Use Cases für Leser, nicht für Computer und können einfach folgendermaßen formulieren: »Der Käufer wirft 80 Cent mit Zwanzig-, Zehn- und Fünf-Cent-Stücken in beliebiger Reihenfolge ein.«

Mit Sequenzen lassen sich vergangene Interaktionen einfach beschreiben, weil die Vergangenheit bereits abgeschlossen und damit festgelegt ist. Zur Beschreibung von künftigen Interaktionen benötigen wir eine *Reihe möglicher Sequenzen*, mit einer Sequenz für jede mögliche künftige Konstellation. Wenn ich erzähle, wie ich gestern eine Gehaltserhöhung verlangte, tue ich das folgerndermaßen:

> »Ich hatte gestern eine ernste Auseinandersetzung mit meiner Chefin. Ich sagte: ‚…', sie sagte ‚…', ich sagte ‚…' usw.

Aber für ein künftiges Geschehen muss ich das anders formulieren:

> »Ich bin richtig nervös wegen der nächsten Auseinandersetzung mit meiner Chefin.«
>
> »Warum?«
>
> »Ich werde eine Gehaltserhöhung verlangen.«
>
> »Und wie?«
>
> »Na, zuerst werde ich sagen: ‚…'. Wenn sie darauf antwortet ‚…' sage ich ‚…'. Antwortet sie aber ‚…', dann versuche ich es mit ‚…' usw.

Ähnlich liegt der Fall, wenn wir jemandem erklären, wie er eine Cola kaufen soll:

> »Zuerst hältst Du das Geld bereit.«
>
> »Wenn Du das Kleingeld genau parat hast, wirfst Du es ein und drückst die Cola-Taste.«
>
> »Wenn nicht, wirf das Geld ein und pass auf, ob der Automat das Wechselgeld herausgibt. Wenn ja, …«

Zur Beschreibung einer künftigen Interaktion muss man mit unterschiedlichen Bedingungen arbeiten und Sequenzreihen bilden. Für jede von ihnen wird die Bedingung, der Sequenzablauf und das Resultat angegeben.

Wir können eine Sequenzreihe zu einer einzigen Aussage zusammenfassen. »Geh los und hol ein Getränk aus dem Automat« oder »Dann bittest Du Deine Chefin um eine Gehaltserhöhung«. Genau wie einzelne Sequenzen lassen sich solche Aussagen je nach Bedarf in kurze abstrakte Beschreibungen zusammenfassen oder in detaillierte Beschreibungen zerlegen.

Wir haben feststellen können, dass ein Use Case eine Reihe möglicher Szenarien enthält, mit denen ein Ziel erreicht werden soll. Zur Vervollständigung wollen wir hinzufügen, dass

- alle Interaktionen sich auf das gleiche Ziel eines einzigen Primärakteurs beziehen;
- ein Use Case mit dem auslösenden Ereignis (Trigger) beginnt und anhält, bis das Ziel erreicht oder aufgegeben wurde *und* das System seine Aufgaben hinsichtlich der Interaktion erledigt hat.

2.1.4 Ein Use Case als Sammlung von Szenarien

Der Primärakteur hat ein Ziel; das System sollte dem Primärakteur dabei helfen, das Ziel zu erreichen. Einige Szenarien zeigen, dass das Ziel erreicht wird; einige enden damit, dass es aufgegeben wird. Jedes Szenario enthält eine Schrittsequenz, die aufzeigt, wie Aktionen und Interaktionen ablaufen. Ein Use Case ist eine Sammlung all dieser Szenarien und zeigt alle Möglichkeiten für den Erfolg oder Fehlschlag beim Erreichen des Ziels.

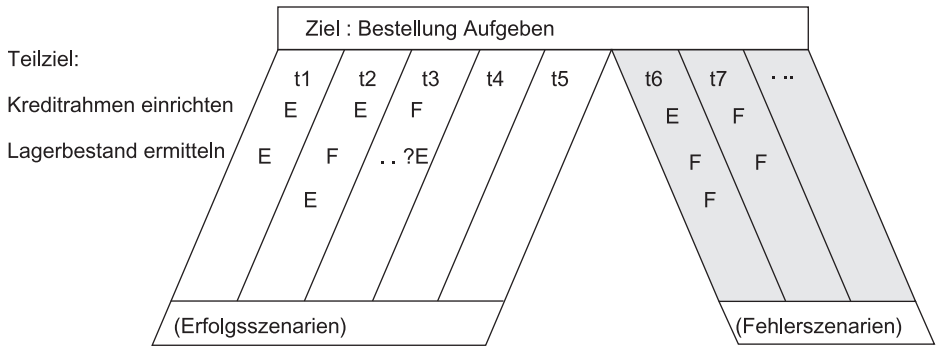

Abb. 2.2: Die gestreifte Hose: Szenarien, die mit einem Erfolg oder Fehlschlag enden.

Eine sinnvolle Metapher dafür ist das Bild der gestreiften Hose (Abbildung 2.2). Der Hosengürtel gibt das Ziel an, das alle Szenarien zusammenhält. Die beiden Beine sind jeweils die Szenarien, die mit einem Erfolg enden, und die Szenarien, die mit einem Fehlschlag enden. Ein Streifen entspricht einem Szenario, egal an welchem der beiden Beine er sich befindet. Den ersten Streifen am Bein für die Erfolgsfälle nennen wir *Standardablauf*. Die anderen Streifen sind weitere Szenarien, die erfolgreich enden – über alternative Erfolgspfade oder nach der Beseitigung eines zwischengelagerten Fehlschlags. Alle Streifen auf der Fehlerseite spielen Fehlschläge durch, die möglicherweise überwunden werden und dann schließlich mit einem Fehlerfall enden.

Nicht alle Szenarien werden von vorne bis hinten durchgestaltet. Das wäre keine gute Strategie, weil das Schreiben dann langwierig, redundant und schwer durchzuhalten wäre. Das Bild der gestreiften Hose erinnert uns ständig daran, dass jeder Use Case auf zwei Arten enden kann, dass das Ziel des Primärakteurs alle Szenarien verbindet und jedes Szenario in einfachen Worten beschreibt, wie ein Ziel erreicht oder verfehlt wird.

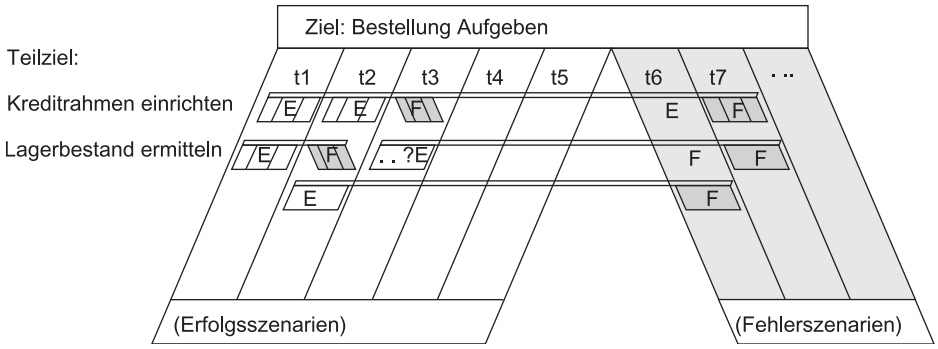

Abb. 2.3: Gestreifte Hose mit Teilzielen

Abbildung 2.3 fügt dem Bild der gestreiften Hose weitere Einzelheiten hinzu, die zeigen, wie sich ein Teil-Use-Case in den Use Case einfügt, der ihn aufruft. Ein Kunde möchte eine `Bestellung Aufgeben`. Eines der Teilzeile des Kunden heißt `Kreditrahmen Einrichten`. Das ist ein komplexes Teilziel, das mit einem Erfolg oder Fehlschlag enden kann: Es handelt sich um einen eigenständigen Use Case, den wir zu einem Einzelschritt zusammengefasst haben. Der Schritt `Kunde richtet Kreditrahmen ein` ist Gürtel einer weiteren Hose. In dem Streifen (dem Szenario), der diesen Schritt enthält, wird das Teilziel entweder erreicht oder nicht. In den Szenarien 1 und 2 unserer Abbildung wird das Teilziel erreicht. Szenarien 3 und 7 enden mit einem Fehlschlag. In Szenario 3 dagegen gelingt ein weiterer Versuch, das Teilziel `Kreditrahmen Einrichten` zu erreichen, und das Szenario endet mit einem Erfolg. In Szenario 7 schlägt auch der zweite Versuch fehl und der gesamte Use Case endet mit einem Fehlschlag von `Bestellung Aufgeben`.

Wir zeigen die kleinen Streifen für den Teil-Use-Case von Abbildung 2.3, um zu verdeutlichen, dass der äußere Use Case sich nicht darum kümmert, was beim Teil-Use-Case vor sich geht, damit der Endzustand erreicht wird. Er endet mit einem Erfolg oder nicht. Der äußere oder aufrufende Use Case baut einfach auf den Erfolg oder Fehlschlag des Schritts auf, nach dem der Teil-Use-Case benannt ist.

Wir erkennen anhand der Hosenmetapher folgende Prinzipien:

- Einige Szenarien enden mit einem Erfolg, andere mit einem Fehlschlag.
- Ein Use Case sammelt alle Erfolgs- und Fehlerszenarien.

- Jedes Szenario ist eine klare Beschreibung einer Reihe von Umständen mit einem einzigen Ergebnis.
- Die einzelnen Schritte bei einem Use Case sind die Szenarien (die Streifen der Hose) und bei einem Szenario die Teil-Use-Cases.
- Ein Schritt in einem Szenario kümmert sich nicht um die Streifen im Teil-Use-Case, sondern nur um Erfolg oder Fehlschlag.

Wir folgen diesen Prinzipien im gesamten Buch.

2.2 Die Übereinkunft zwischen Stakeholdern mit Eigeninteressen

Das Modell der Akteure und Ziele erklärt, wie man Sätze in einem Use Case formuliert, es sagt aber nichts über die Notwendigkeit aus, das innere Verhalten des zur Diskussion stehenden Systems zu beschreiben. Deshalb muss das Modell um den Gedanken erweitert werden, dass ein Use Case eine Übereinkunft zwischen Stakeholdern mit eigenen Interessen ist. Wir wollen diesen Gedanken das konzeptuelle Modell der *Stakeholder und Interessen* nennen. Der Modellteil, der sich mit Stakeholdern und Interessen beschäftigt, identifiziert, was Teil eines Use Case ist und was nicht.

Das SuD führt eine Übereinkunft aus, die zwischen den Stakeholdern getroffen wurde, wobei die Use Cases das Verhalten dieser Übereinkunft ausdetaillieren. Nicht alle Stakeholder sind während des Systembetriebs vor Ort. Normalerweise ist der Primärakteur präsent, aber auch das ist nicht immer der Fall. Die anderen Stakeholder sind nicht zugegen, man kann sie also als *Akteure hinter den Kulissen* bezeichnen. Das System agiert, um die Interessen dieser Akteure hinter der Bühne einzulösen. Zu den Aktionen gehören das Sammeln von Informationen, die Durchführung von Validierungen und die Aktualisierung der Protokolle (siehe Abbildung 2.4).

Ein Bankautomat muss Protokoll über alle Interaktionen führen, um die Interessen der Stakeholder im Streitfall zu wahren. Er zeichnet weitere Informationen auf, damit sie herausfinden können, wie weit eine Transaktion vor ihrem Fehlschlag ausgeführt wurde. Bankautomat und Bankensystem überprüfen vor der Auszahlung, ob der Kontoführer über eine ausreichende Deckung verfügt, um sicherzustellen, dass der Bankomat nicht mehr Geld ausgibt, als der Kunde tatsächlich auf der Bank hat.

Der Use Case erfasst in seiner Eigenschaft als Übereinkunft zum Verhalten das *gesamte* Verhalten, das die Wahrung der Stakeholder-Interessen betrifft, und *nur* dieses.

Zur sorgfältigen Ausarbeitung eines Use Case listen wir alle Stakeholder auf, benennen ihre Interessen an der Ausführung des Use Case und halten fest, was es für jeden einzelnen Stakeholder bedeutet, wenn ein Use Case mit einem Erfolg endet und was das System ihm garantieren soll. Anschließend verfassen wir die einzelnen Schritte des Use Case und vergewissern uns, dass ab dem Moment des auslösenden Ereignisses bis zum Ende des Use Case alle unterschiedlichen Interessen erfüllt werden. So erkennen wir, an welcher Stelle wir die Beschreibung beginnen und beenden und was wir in den Use Case aufnehmen oder was wir ausschließen.

Die meisten Menschen arbeiten beim Verfassen von Use Cases nicht so sorgfältig und kommen dennoch gut zurecht. Gute Autoren führen diese Übungen in Gedanken durch, wenn sie formlose Use Cases modellieren. Sie übersehen dabei einiges, haben aber genug Routine, um solche Auslassungen bei der Software-Entwicklung wettzumachen. Für manche Projekte genügt diese Vorgehensweise. Manchmal verursacht sie aber auch hohe Kosten. Vergleichen Sie dazu die Geschichte in Abschnitt 4.1, die zeigt, was geschieht, wenn einige Interessen übersehen werden.

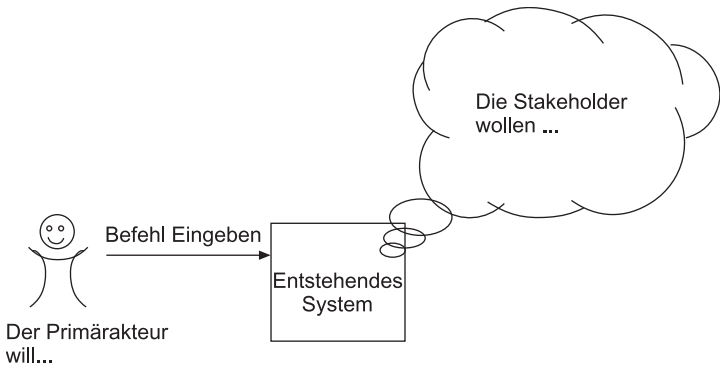

Abb. 2.4: Das SuD dient dem Primärakteur und schützt die Stakeholder hinter den Kulissen.

Zur Wahrung der Interessen der Stakeholder müssen wir drei Arten von Aktionen beschreiben:

- Interaktionen zweier Akteure (um dem Ziel näher zu kommen)
- Validierungen (um einen Stakeholder zu schützen)
- Interne Zustandswechsel (verursacht durch einen Stakeholder)

Das Stakeholder- und Interessenmodell ändert die Gesamtprozedur beim Verfassen eines Use Case nur geringfügig: Listen Sie Stakeholder und deren Interessen auf und verwenden Sie diese Liste als Prüfvorlage, um sich zu vergewissern, dass bei den Use-Case-Komponenten nichts vergessen wurde. Diese geringfügige Veränderung macht für die Qualität des Use Case einen gewaltigen Unterschied.

2.3 Das grafische Modell

Dieser Abschnitt ist für alle gedacht, die gerne abstrakte Modelle erstellen. Sie können ihn überspringen, wenn Sie nicht dazugehören.

Wie bereits festgestellt, beschreibt ein Use Case eine Übereinkunft bezüglich des Verhaltens zwischen Stakeholdern mit Eigeninteressen. Wir organisieren das Verhalten nach den operativen Zielen einer ausgewählten Reihe von Stakeholdern – denjenigen, die vom System erwarten, dass es etwas für sie *tut* - die wir als *Primärakteure* bezeichnen. Der Titel des Use Case ist das Ziel des Primärakteurs. Er enthält das gesamte Verhalten, das zur Beschreibung dieses Teils der Übereinkunft nötig ist.

Das System hat die Aufgabe, mit seinen Aktionen die vereinbarten Interessen der ausgewählten Stakeholder zu wahren. Eine Aktion kann einem dieser drei Typen angehören:

- eine Interaktion zweier Akteure, bei der Informationen in beide Richtungen weitergegeben werden;
- eine Validierung, die die Interessen eines der Stakeholder wahrt;
- ein interner Zustandswechsel, der ebenfalls ein Stakeholder-Interesse wahrt oder fördert.

Ein Szenario besteht aus Aktionsschritten. In einem *Erfolgsszenario* werden alle (durch Übereinkunft erzielten) Interessen der Stakeholder an der Dienstleistung erfüllt, deren Durchführung Aufgabe des Systems ist. In einem *Fehlerszenario* werden diese Interessen gemäß der Invarianten des Systems geschützt. Das Szenario endet, wenn alle Stakeholder-Interessen befriedigt oder gewahrt wurden.

Die drei Trigger, die die Ausgabe des Ziels anfordern, sind der Primärakteur, der eine Interaktion mit dem System auslöst, der Primärakteur, der einen Vermittler für die Auslösung der Interaktion einsetzt, und eine zeit- oder zustandsbasierte Initiierung.

Die Abbildungen 2.5 bis 2.8 zeigen das Modell der im vorliegenden Kapitel beschriebenen Use Cases bei Verwendung der Unified Modeling Language (UML).

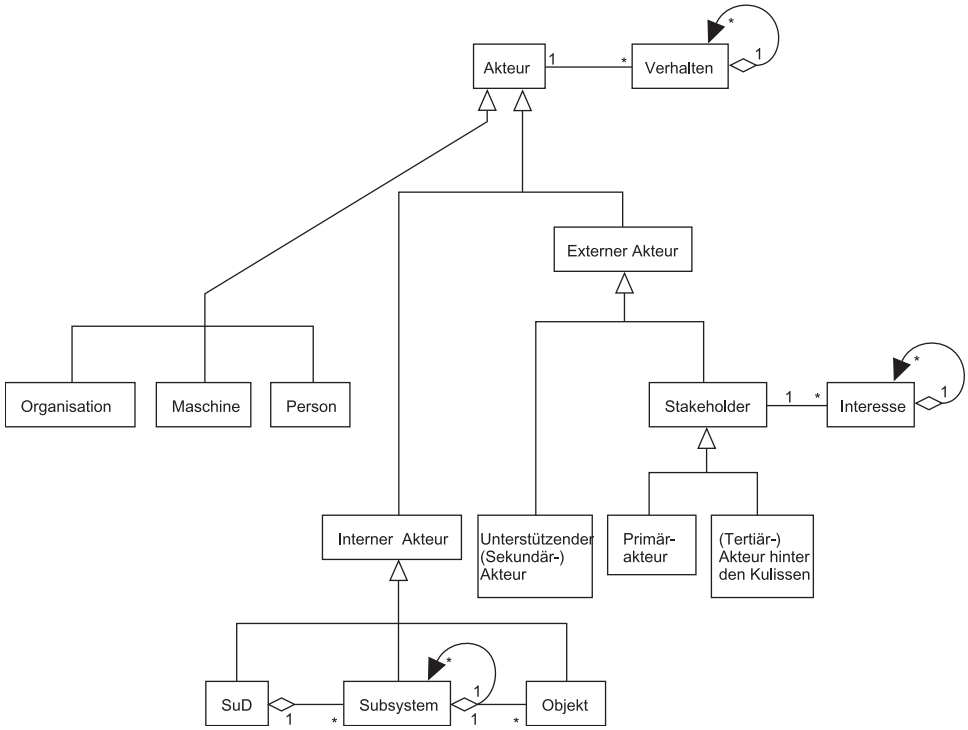

Abb. 2.5: *Akteure und Stakeholder:* Ein Stakeholder hat Interessen, ein Akteur ein Verhalten. Auch der Primärakteur ist Stakeholder.

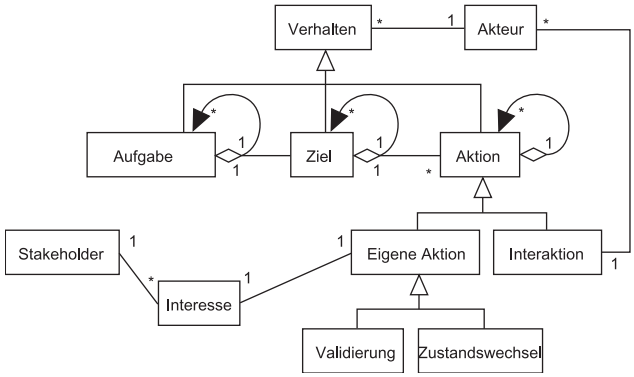

Abb. 2.6: *Verhalten:* Zielorientiertes Verhalten besteht aus Aufgaben, Zielen und Aktionen. Die eigenständigen Aktionen, die wir modellieren, fördern oder schützen die Stakeholder-Interessen. Interaktionen verbinden die Aktionen verschiedener Akteure.

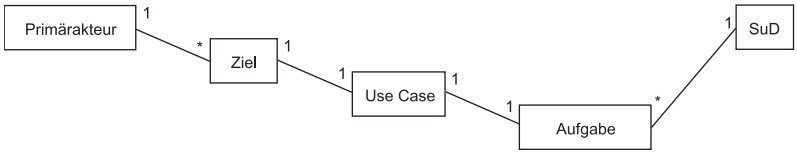

Abb. 2.7: *Ein Use Case als Aufruf einer Zuständigkeit:* Der Use Case erfasst das Ziel des Primärakteurs und ruft das SuD auf, das für die Aufgabe zuständig ist.

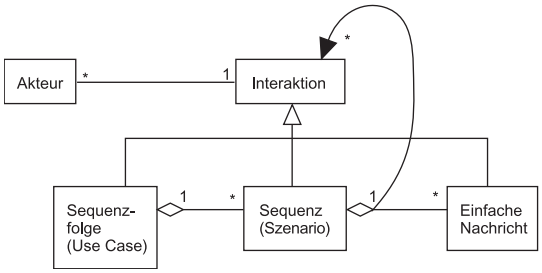

Abb. 2.8: *Zusammengesetzte Interaktionen:* N Akteure nehmen an einer Interaktion teil. Interaktionen lassen sich in Use Cases, Szenarien und einfache Nachrichten zerlegen. Auch hier verwenden wir der Einfachkeit halber den Begriff Sequenz.

Wir wollen nicht verschweigen, dass zum Debuggen dieses Modells ein jahrelanges Testen von Projekten mit einem Tool auf Modellgrundlage nötig wäre. Anders gesagt: Es verstecken sich darin vermutlich einige subtile Fehler. Wir führen es für all diejenigen auf, die es ausprobieren wollen und die vielleicht ein solches modellbasiertes Tool entwickeln möchten.