

# 9

## Use-Case-Diagramm

---

Die Frage „**Was soll mein geplantes System eigentlich leisten?**“ sollte am Beginn jeder Systementwicklung stehen. Eine fundierte Beantwortung dieser Frage bewahrt Sie davor im Detail zu versinken, bevor Sie überhaupt wissen, was vom System überhaupt erwartet wird. Ein Use-Case-Diagramm (dt. Anwendungsfall-Diagramm) zeigt das externe Verhalten eines Systems aus der Sicht der Nutzer, indem es die Nutzer (im UML-Jargon „Akteure“ genannt), die Use-Cases und deren Beziehungen zueinander darstellt. Ein Nutzer kann eine Person, aber auch ein Nachbarsystem sein. Use-Cases bilden dabei die Reaktion des Systems auf Ereignisse seiner Umwelt ab und fassen dabei Teile der Systemdienstleistung zusammen.

Die externe  
Nutzersicht

## 9.1 Überblick

### 9.1.1 Die Use-Case-Analyse

System-  
dienstleistungen  
von außen  
betrachtet

Die im Rahmen einer Use-Case-Analyse erstellten Use-Case-Diagramme zeigen das nach außen sichtbare Verhalten eines Elements. Diese Elemente werden in der UML 2 formal als Gegenstand (engl. Subject) bezeichnet und stellen in der Regel komplette (Hardware- bzw. Software-) Systeme oder Subsysteme dar. Genauso gut können Sie damit aber auch das externe Verhalten von Klassen, Schnittstellen, Komponenten oder Knoten modellieren. Trotzdem verwenden wir im weiteren Verlauf des Kapitels den Begriff „System“ und zeigen durch Beispiele weitere mögliche Elemente, die externes Verhalten realisieren.<sup>1</sup>

Kapselung

Ein Use-Case (dt. Anwendungsfall) selbst kapselt eine in sich geschlossene Sammlung von Aktionen, die in einer spezifizierten Reihenfolge ablaufen. Er stellt somit seiner Umwelt eine Dienstleistung, sprich ein Verhalten, zur Verfügung. Denken Sie an einen Web-Browser. Dieses System bietet Ihnen als Dienstleistung (als Use-Case) die Möglichkeit, „eine Webseite anzuzeigen“ oder „die Webseite zu drucken“. Oder nehmen Sie als Beispiel für einen Knoten (Kapitel 8.2) einen Fileserver, der die Use-Cases „Verzeichnis anzeigen“, „Datei up- und download“ oder „Nutzer authentifizieren“ realisiert. Im Projekt sollten Sie das Verhalten eines Use-Cases mittels einer Use-Case-Beschreibung detaillieren. Da diese Use-Case-Beschreibung aber kein UML-Notationsmittel ist, sondern rein textuell erstellt wird, betrachten wir sie in diesem Kapitel nicht detaillierter. Informationen zu unterschiedlichen Schablonen für Use-Case-Beschreibungen finden Sie auf unseren Webseiten.

Knoten



8.2

Was statt wie

Sehen wir uns das Beispiel Web-Browser und den Use-Case „Webseite anzeigen“ näher an: Er umfasst den Auslöser (Initiator) des Use-Cases, die einzelnen Schritte (z.B. „Webadresse eingeben“ oder „Serveranfrage starten“, aber auch Sonder- und Fehlerfälle, z.B. die Eingabe einer unvollständigen Webadresse) und daran beteiligte externe Personen und Systeme, die so genannten Akteure. Ein Use-Case zeigt aber nicht, welche Klassen und Operationen an den Aktionen beteiligt sind. Er gibt auch keine Auskunft darüber, wie die Webseite für die Anzeige im Bildspeicher aufgebaut wird.

Akteure  
sind externe  
Kommunikations-  
partner

Der Akteur ist externer Kommunikationspartner des Use-Cases. Während des Ablaufs eines Use-Cases liefert oder empfängt der Akteur Signale bzw. Daten zum oder vom System, das den Use-Case realisiert. Typische Akteure unseres Web-Browsers sind der Internetsurfer bzw. das Betriebssystem als Schnittstelle für die Netzwerkübertragung oder zum Dateisystem.

Ein Use-Case-Ablauf ist ein zeitlich in sich abgeschlossener Vorgang mit einem Auslöser (Webseite angefordert) und einem Ergebnis (Webseite angezeigt). Ein Akteur initiiert einen Use-Case, der das Ergebnis entweder an den gleichen oder einen anderen Akteur liefert. Zur Ablaufzeit interagiert eine Instanz des Akteurs mit einer Instanz des Use-Cases.

<sup>1</sup> Realisieren ist hier im allgemeinen Sinn der Zuordnung zu verstehen und nicht in der engeren Auslegung als „Realization“-Beziehung (Kapitel 3).

Im Gegensatz zu vielen anderen UML-Diagrammen sind Use-Case-Diagramme – auch bedingt durch eine sehr begrenzte Anzahl an Notationselementen – eingängig und übersichtlich. Glücklicherweise wurde an der Trivialität von Use-Case-Diagrammen auch in der UML 2 nichts signifikantes verändert. Use-Case-Modellierung mag auf den ersten Blick trivial, wie „Malen-nach-Zahlen“, aussehen. In der Projektrealität trifft man häufig eine sehr einfache, skizzenhafte Verwendung von Use-Case-Diagrammen für erste Diskussionen mit den Stakeholdern an, die es Ihnen ermöglicht, Besprochenes in ein Bild zu fassen. Gerade im technischen Projektumfeld werden Use-Case-Diagramme aber auch etwas formaler zu einer ersten Kontextabgrenzung eingesetzt. Dort werden die einzelnen Ereignisse, die die Systemgrenze passieren, systematisch erhoben, notiert und ausgewertet. Erinnern Sie sich beim Einsatz von Use-Case-Diagrammen immer wieder daran, dass Sie ganz am Beginn einer Systementwicklung stehen und nicht jedes Detail in diesem Diagramm unterbringen müssen – die UML bietet Ihnen noch ein paar mehr Diagrammart an.

Malen nach Zahlen?

Ein Use-Case-Diagramm enthält die grafische Darstellung

Notationsmittel

- des Systems,
- der Use-Cases,
- der Akteure außerhalb des Systems und
- der Beziehungen zwischen Akteur und Use-Case, der Akteure untereinander oder Use-Cases untereinander.

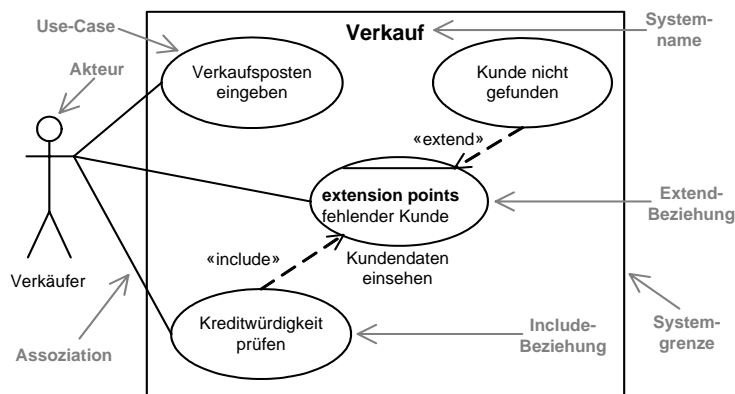


Abbildung 9.1: Ein Use-Case-Diagramm und seine Bestandteile

## 9.1.2 Ursprung von Use-Cases

Die Idee der Use-Cases, nämlich die Beschreibung des funktionalen Verhaltens eines Systems von außen gesehen, geht bereits auf die späten 70er und frühen 80er Jahre zurück [MPa84]. Populär und letztendlich in die UML eingeführt wurden sie durch Ivar Jacobson, der die Use-Cases als eine Haupttechnik in der Systemanalyse nutzte [Jac92].

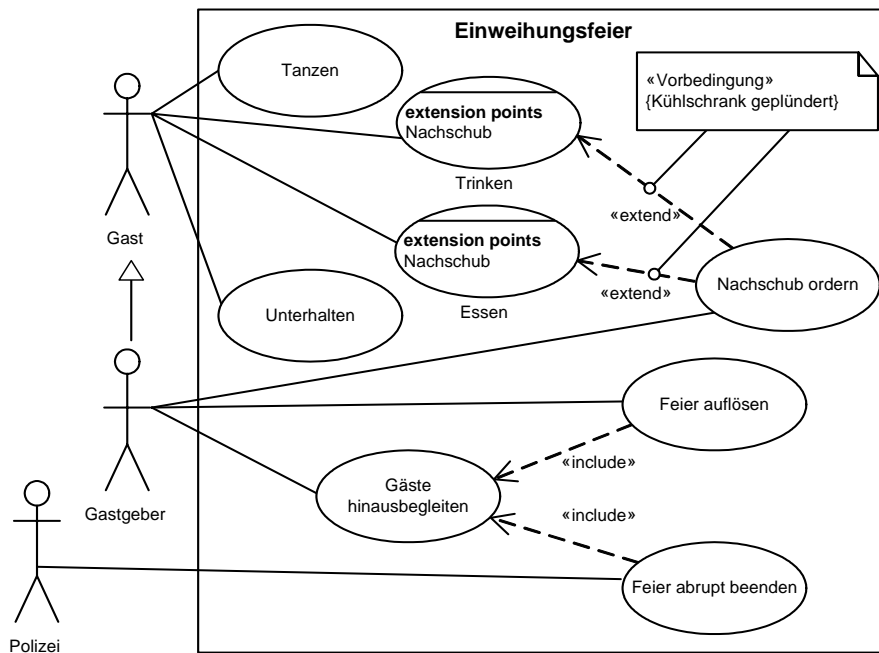
Use-Case-Väter

Seit Mitte der 90er Jahre etablierte sich diese Art der Analyse in abgewandelter Form als fester Bestandteil in zahlreichen Vorgehensmodellen (siehe zum Beispiel [Kru00]).

Für den tieferen Einstieg in die allgemeine Methodik der Use-Case-Analyse empfehlen sich [Coc98] oder [Arm00], für die Anwendung bei der Entwicklung technischer Systeme [HRu02].

Die UML ermöglicht die Use-Case-Modellierung seit der ersten Version. Außer einigen kleinen Schönheitskorrekturen hat sich auch in der UML 2 nichts an den Use-Case-Diagrammen geändert. Explizit herausgearbeitet wird im neuen Standard jedoch die Möglichkeit, dass beliebige Classifier (also auch explizit Klassen, Schnittstellen, Komponenten, ...) Use-Cases realisieren können. Obwohl dies auch in älteren Versionen nicht verboten war, wurde in der Praxis kaum Gebrauch davon gemacht. Meist wird die Use-Case-Analyse für die Beschreibung eines kompletten Systems verwendet. Abstrakt betrachtet ist ein Use-Case jedoch nur die Repräsentation eines Verhaltens, das einem Nutzer angeboten wird. Wer den Use-Case realisiert bzw. dieses Verhalten anbietet, ist in der Use-Case-Analyse nur zweitrangig. Seit dem Update auf UML 2 gehören Use-Case-Diagramme daher auch nicht mehr zu den (statischen) Strukturdiagrammen, sondern zu den (dynamischen) Verhaltensdiagrammen.

## 9.2 Anwendungsbeispiel



**Abbildung 9.2:** Die wichtigsten Use-Cases auf einer Einweihungsfeier

Das Anwendungsbeispiel zeigt, dass das System, das die Use-Cases realisiert, nicht zwangsläufig aus Hard- oder Software besteht. Die Modellierung der „realen Welt“ und von Geschäftsprozessen mittels Use-Cases ist ebenso möglich wie die Darstellung von technischen Systemprozessen in Echtzeitsystemen.

## 9.3 Anwendung im Projekt

### 9.3.1 Typische Anwendungsbereiche

Use-Case-Diagramme ermöglichen eine „Black Box“-Sicht auf das betrachtete System. Damit können Sie anwendernah und unabhängig von internen technischen Abläufen das System von seiner Umwelt abgrenzen und die elementaren Systemanforderungen finden. Modellieren Sie Use-Cases, wenn Sie

Fokus:  
„Black Box“-Sicht

- die funktionalen Dienstleistungen eines Systems oder einer Komponente „auf einen Blick“ zeigen wollen;
- Ihr System aus der Nutzersicht in handhabbare, logische Teile zerlegen wollen;
- die Außenschnittstellen und Kommunikationspartner des Systems modellieren möchten;
- komplexe Systeme leicht verständlich und auf hohem Abstraktionsniveau darstellen möchten oder
- planbare Einheiten, das heißt Inkremente, für Ihre Entwicklung benötigen.

Vor allem in der Anforderungsanalyse sollten Sie Use-Case-Diagramme einsetzen. Weil sie leicht verständlich sind, bieten sie eine gute Grundlage für die Kommunikation zwischen Anwendern, Entwicklern und Analytikern. Sie verschaffen Ihnen einen Überblick über das System und seine Einbettung in einen größeren Kontext. Gerade die Darstellung der beteiligten Akteure und der Systemgrenzen liefert essenzielle Informationen für die weitere Systementwicklung. Sie bewahrt Sie vor bösen Überraschungen und legt von Anfang an fest, was zu Ihrem System gehört und was nicht, was Sie entwickeln (Kosten!) und welchen Schnittstellen Sie gerecht werden müssen (Anforderungen!).

Multitalent in der  
Anforderungs-  
analyse

Use-Case-Diagramme bieten sich vor allem in frühen Phasen eines Projektes oder bei der Entwicklung neuer oder erweiterter Komponenten an. Mit Hilfe von Use-Cases können Sie die Entwicklung eines Systems oder einer Komponente planen. Dieses Use-Case-getriebene Vorgehen ist eine geeignete Basis für eine inkrementelle Systementwicklung. Ein Use-Case kann dabei einem Inkrement entsprechen, das der Reihe nach priorisiert, analysiert, entworfen, implementiert und getestet wird.

Planung und  
Inkrementbildung

### 9.3.2 Use-Cases und danach?

Natürlich ist es nicht damit getan, ein Use-Case-Diagramm zu zeichnen (wobei der Aufwand für das Finden aller Akteure, das richtige Schneiden der Use-Cases und die Festlegung der Systemgrenzen nicht zu unterschätzen ist!). Use-Cases sind erst dann „vollständig“, wenn die dahinter stehenden Abläufe beschrieben sind. Je nach deren Natur, dem Zweck der Dokumentation und dem Zielpublikum sollten Sie unterschiedliche Mittel zur Beschreibung der Use-Cases, genauer der Use-Case-Abläufe, einsetzen. Tabelle 9.1 gibt Ihnen hierfür eine Entscheidungsgrundlage.



10-13, 15

**Tabelle 9.1:** Beschreibungsmöglichkeiten für Use-Cases

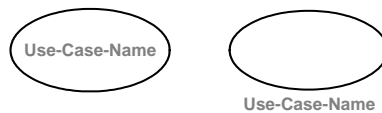
Merkmale des Use-Case	Empfohlene Notation zur Use-Case-Beschreibung	Referenz
kurze klare Abläufe, wenige Sonderfälle	(strukturierter) Text	
ablauf- oder schrittorientiert (1., 2., ...)	Aktivitätsdiagramm	Kapitel 10
einfache daten- oder entitätsorientierte Abläufe (viele Entitäten)	Kommunikationsdiagramm	Kapitel 13
komplexe daten- oder entitätsorientierte Abläufe (viele Entitäten)	Sequenzdiagramm	Kapitel 12
kein „typischer“ Ablauf, gleichwahrscheinliches Auftreten von Abfolgen und Ereignissen	Zustandsautomat	Kapitel 11
Use-Case bündelt viele Szenarien	Interaktionsübersichtsdiagramm	Kapitel 15

Unabhängig von ihrer Art sollte eine Use-Case-Beschreibung den Namen des Use-Cases, die Ablaufbeschreibungen, zugehörige Akteure, Vorbedingungen, Nachbedingungen und Ausnahmen enthalten. Ein Beispiel einer derartigen Use-Case-Beschreibung finden Sie auf unserer Homepage [www.uml-glasklar.de](http://www.uml-glasklar.de). [Coc98] diskutiert zudem unterschiedliche Möglichkeiten und Schablonen, natürlich-sprachlich beschriebene Use-Case-Abläufe zu strukturieren. Erweiterte Beschreibungsschablonen, die auch die nichtfunktionalen Aspekte des Systems berücksichtigen und daher auch für eher technisch orientierte Systeme geeignet sind, bieten [HRu02] und [www.sophist.de](http://www.sophist.de) an.

## 9.4 Notationselemente

### 9.4.1 Use-Case

#### Notation



**Abbildung 9.3:** Die Standardnotationen für einen Use-Case

Notation

In aller Regel wird ein Use-Case durch eine Ellipse dargestellt. Der Name des Use-Cases wird dabei inner- oder unterhalb der Ellipse notiert.

#### Beschreibung

Ein Use-Case, im Deutschen Anwendungsfall genannt, beschreibt eine Menge von Aktionen, die, schrittweise ausgeführt, ein spezielles Verhalten formen.

So umfasst z.B. der Use-Case „Datei speichern“ alle Aktionen, die nötig sind, um eine Datei auf einem Medium abzulegen. Also etwa die Aktionen Menüpunkt anklicken, Verzeichnis auswählen, Dateiname vergeben und mit OK bestätigen. Ein Use-Case bildet eine Art Hülle, die auch Sonder- und Fehlerfallaktionen einschließen kann (denken Sie daran, dass der Speicherplatz erschöpft sein bzw. der vergebene Dateiname unzulässige Zeichen enthalten kann).

Use-Case = Hülle für Standard-, Sonder-, Fehlerfall

Ein Use-Case wird stets von einem Akteur ausgelöst bzw. instanziiert (Trigger; Menüpunkt anklicken) und führt zu einem fachlichen Ergebnis (Datei auf dem Medium abgelegt). Die Bezeichnung des Use-Case spiegelt die Sicht des Akteurs wieder (z.B. „Film ausleihen“) und nicht die des Systems (müsste dann ja „Film verleihen“ heißen).

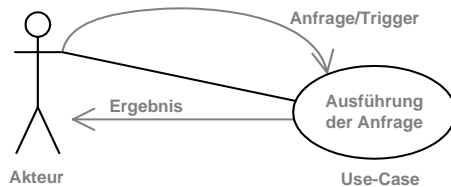


Abbildung 9.4: Ausführung eines Use-Cases

Ein Use-Case darf auch gleichzeitig mehrfach instanziiert werden (gleichzeitiges Abspeichern von mehreren Dateien). Unterschiedliche Use-Cases sind parallel instanzierbar (Datei speichern, Datei drucken, Datei kopieren, ...). D.h., auch wenn Sie in einem Diagramm nur fünf Use-Cases sehen, können Hunderte von realen Abläufen gleichzeitig durch das System abgewickelt werden.

Mehrfache Instanziierung

Betrachten Sie Use-Cases immer als abgeschlossene Einheiten, die ein funktionales Verhalten widerspiegeln und bei denen interne Strukturen irrelevant sind. Der Fokus liegt auf der an der Schnittstelle angebotenen Dienstleistung – welche Daten oder Zustände manipuliert werden, sieht der Nutzer eines Use-Cases nicht. Ihn interessiert nur der Auslöser eines Use-Case-Ablaufs, die Kommunikation (Interaktion oder Kollaboration, „Wer muss wann was liefern?“) zwischen Akteuren und Use-Cases und das am Ende stehende Ergebnis. Ein Use-Case-Ablauf ist dann zu Ende, wenn keine Kommunikation zwischen Akteur und Use-Case mehr erfolgt – mit anderen Worten: der Ablauf „zur Ruhe gekommen“ ist.

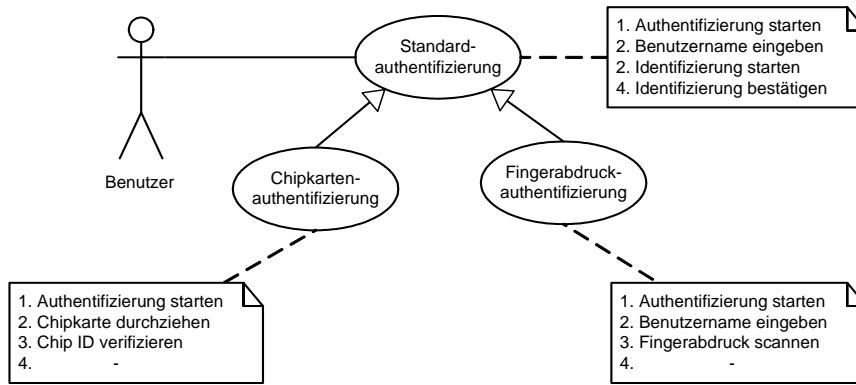
Neben der Modellierung von einzelnen, autarken Use-Cases dürfen Sie Use-Cases auch in Beziehung setzen. Dadurch verknüpfen Sie die Abläufe der einzelnen Use-Cases zu *einem* Ablauf.

Beziehungsgeflechte

So erlaubt die von den Klassen bekannte *Generalisierungs- und Spezialisierungsbeziehung* allgemeingültiges Verhalten in einem Basis-Use-Case zu definieren und in spezialisierten Use-Cases zu überschreiben oder zu vervollständigen.

Abbildung 9.5 zeigt den Basis-Use-Case Standardauthentifizierung, der eine Aktionsfolge mit der Texteingabe von Benutzernamen und Passwort vorsieht. Moderne Authentifizierungsverfahren ermöglichen aber auch technische Varianten dieses Ablaufs: Authentifizierung mittels Fingerabdruck oder per Chipkarte. Die zugehörigen Abläufe sind entsprechend in den spezialisierten Use-Cases überschrieben. Beachten Sie, dass sich zusätzlich zum Verhalten auch die Beziehungen zu Akteuren vererben.

Generalisierung und Spezialisierung



**Abbildung 9.5:** Eine Generalisierungsbeziehung zwischen Use-Cases. Die Kommentare verdeutlichen die Einzelschritte.



9.4, 9.5

Weitere Beziehungen zwischen Use-Cases beschreiben wir aufgrund ihres Umfangs in eigenen Unterkapiteln (Kapitel 9.4, 9.5).

Die UML gewährt Ihnen bei der Notation von Use-Cases viel Freiraum. Die verbindlichste Vorschrift besteht darin, dass Sie einen Use-Case bezeichnen müssen. Für die Namensgebung bietet sich zur besseren Verständlichkeit die Form „Substantiv Verb“ oder „substantiviertes Verb“ an (zum Beispiel: „Raum buchen“, „Raumbuchung“).

Use-Case =  
Classifier

Klassennotation



3.4.1

Da ein Use-Case im Metamodellsinne einen Classifier darstellt, dürfen Sie statt der üblichen Ellipsennotation auch die von Klassen bekannte Rechtecknotation verwenden (Kapitel 3.4.1). Die Ellipse wird dann als kleines Icon in die rechte, obere Ecke des Use-Cases gesetzt.



**Abbildung 9.6:** Use-Case in Ellipsen- und Standard-Classifier-Notation

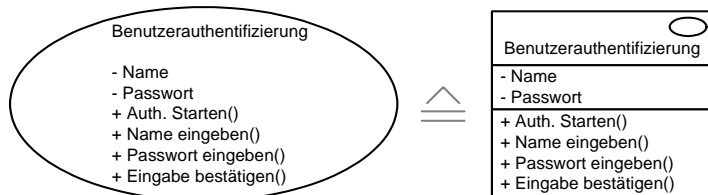
Attribute und  
Operationen von  
Use-Cases

Klassen



3.4.1

Die Rechtecknotation ist praktischer, wenn Sie einem Use-Case Attribute oder Operationen zuordnen wollen. Dieses Vorgehen ist optional möglich, aber in der Praxis eher unüblich. Namensgebung, Syntax und Aufteilung in Compartments erfolgen vollkommen analog zu Klassen (Kapitel 3.4.1):



**Abbildung 9.7:** Use-Case mit Attributen und Operationen

Stereotype



3.4.11

Die Vergabe eines Stereotyps ist ebenfalls optional möglich (Abbildung 9.8). Kapitel 3.4.11 geht detaillierter auf die Stereotypisierung ein.





Abbildung 9.8: Stereotypisierter Use-Case

Um die Verknüpfung (engl. traceability) zwischen einem Use-Case und seiner detaillierten Verhaltensbeschreibung herzustellen, können Sie einen einzelnen Use-Case als Diagramm darstellen. Das zugeordnete Verhaltensdiagramm (Teil III, Verhaltensmodellierung) wird, wie Abbildung 9.9 zeigt, in das Use-Case-Diagramm geschachtelt eingezeichnet.

Verhaltensmodellierung



Teil III

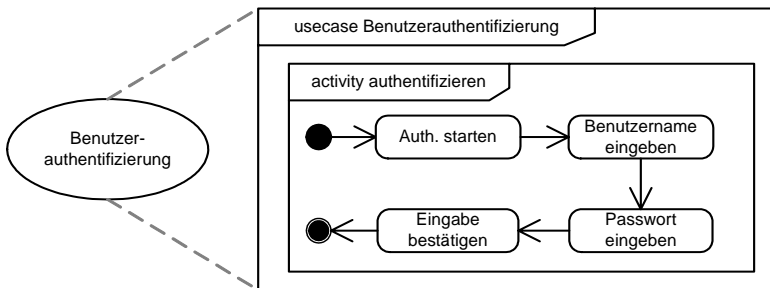


Abbildung 9.9: Detaillierte Verhaltensbeschreibung eines Use-Case

## Anwendung

Abbildung 9.10 zeigt die verschiedenen Notationsmöglichkeiten von Use-Cases.

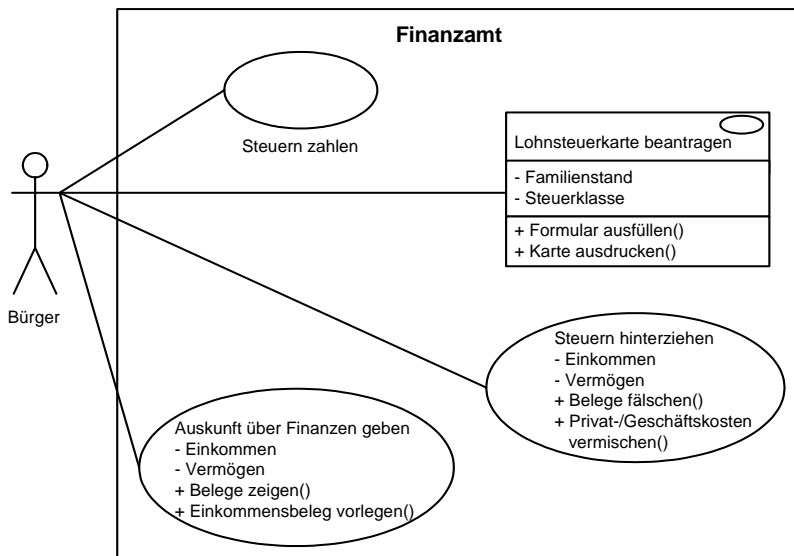


Abbildung 9.10: Use-Cases im Rahmen eines Anwendungsbeispiels

## 9.4.2 System (Betrachtungsgegenstand)

### Notation

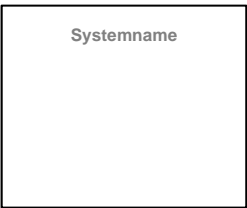


Abbildung 9.11: Notation für das System


Notation

Das System wird als rechteckiger Kasten abgebildet, wobei die Kanten des Systems die Systemgrenzen darstellen. Der Name des Systems wird innerhalb des Kastens angegeben.

### Beschreibung

Das System ist diejenige Einheit, die das Verhalten, welches durch die Use-Cases beschrieben wird, realisiert und anbietet. Unter Umständen zergliedert sich das System, und einzelne Bestandteile realisieren Teilaufgaben; insgesamt jedoch muss das Verhalten „nach außen“ ganzheitlich angeboten werden.

Wie bereits erwähnt, ist *ein System* nicht die einzige Einheit, die einen Use-Case realisieren kann. Gemäß der UML-Spezifikation kann jeder Classifier (Kapitel 3.1.5) einen Use-Case realisieren. Das bedeutet konkret, dass Sie in Ihren Modellen Verhalten in Form von Use-Cases insbesondere auch Klassen (Kapitel 3), Schnittstellen (Kapitel 3.4.4), Komponenten oder Subsystemen (Kapitel 7) zuordnen können.

System =  
Classifier  
 3, 3.1.5,  
3.4.4, 7

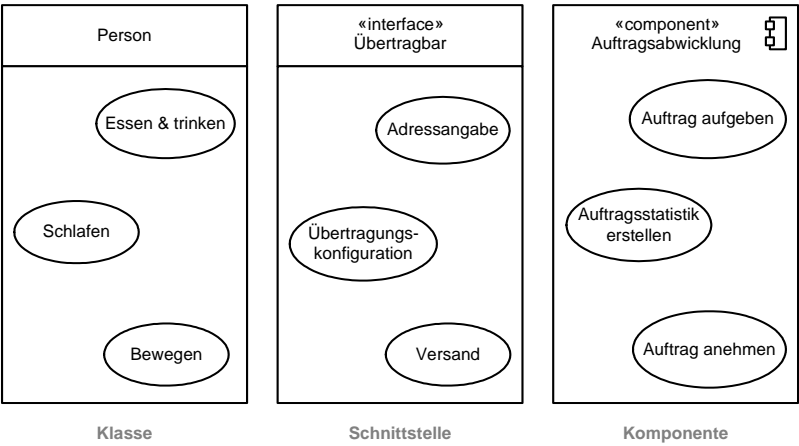


Abbildung 9.12: Eine Auswahl von Classifiern, denen Sie Use-Cases zuordnen dürfen und die diese dann realisieren

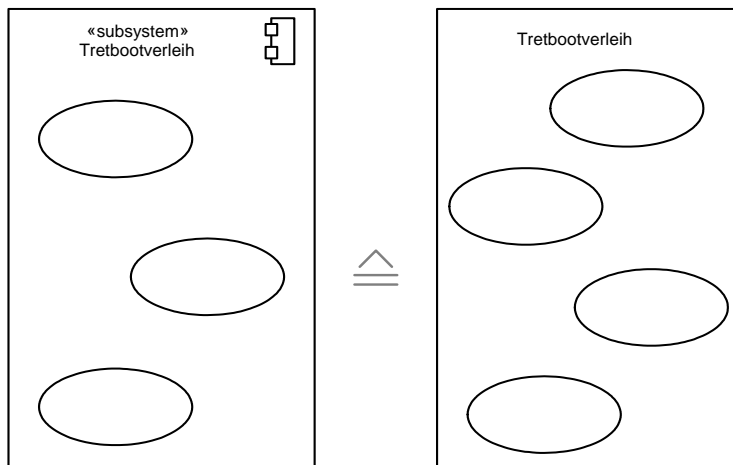
Der in der Praxis am häufigsten verwendete Classifier „System“ lässt sich als stereotypisierte Komponente mit dem Standardstereotyp «subsystem»<sup>2</sup> (Kapitel 7) auffassen.



7

Abbildung 9.13 verdeutlicht dies. Sie sehen im linken Diagramm, die vollständig, notierte Version eines Subsystems Tretbootverleih, während im rechten Diagramm die in der Praxis gängige Kurzform dargestellt wird.

Kurzform  
der Darstellung



**Abbildung 9.13:** Ein System in UML ist eigentlich ein «subsystem»

Dieses Konzept ermöglicht die Zuordnung von Use-Cases zu beliebigen Classifiern als Einheiten, die Verhalten realisieren und anbieten können. In der UML-Spezifikation sind diese Einheiten mit dem englischen Wort *subject* (im Deutschen etwa Betrachtungsgegenstand oder Fachgebiet) bezeichnet, um zu unterstreichen, dass ein System fast alles sein kann: von einem Teil der Umwelt, in der wir leben (zum Beispiel in der Geschäftsprozessanalyse: Kfz-Vermittlung, Supermarkt, ...), über technische Geräte und Systeme (Waschmaschine, Autopilot, Fahrzeug, ...) bis hin zu Teilsystemen oder Komponenten (Browsersoftware, Datenbank, ...).

Einsatzvarianten

Wir möchten Sie an dieser Stelle vor allzu extensiver Nutzung aller zulässigen Notationsmittel eines Use-Case-Diagramms warnen – auch wenn die Spezifikation dies formal zulässt. Ein Use-Case-Diagramm ist ein klassisches Analysediagramm, das den sanften Einstieg in die Systementwicklung ermöglicht, von vielen ohne tiefgreifendes UML-Wissen verstanden wird und daher vom Grundsatz her einfach gehalten werden soll.

Heimat  
Systemanalyse

Zur Beschreibung von Verhalten und Schnittstellen bieten sich je nach Interessenschwerpunkt bessere Notationsmittel an (Kapitel 2.2).



2.2

Es ist im Übrigen nicht zwingend notwendig, das System zu modellieren. Ein Use-Case-Diagramm ist auch ohne Angabe der Einheit, die den Use-Case realisiert, vollständig. So haben Sie die Möglichkeit, sich zunächst auf die Verhaltensdefinition zu beschränken und erst in einem späteren Schritt dieses Verhalten entsprechenden Einheiten zuzuweisen und damit Verantwortlichkeiten festzulegen.

<sup>2</sup> In der UML wird das Gesamtsystem als Top-Level-Subsystem betrachtet.

## Anwendung

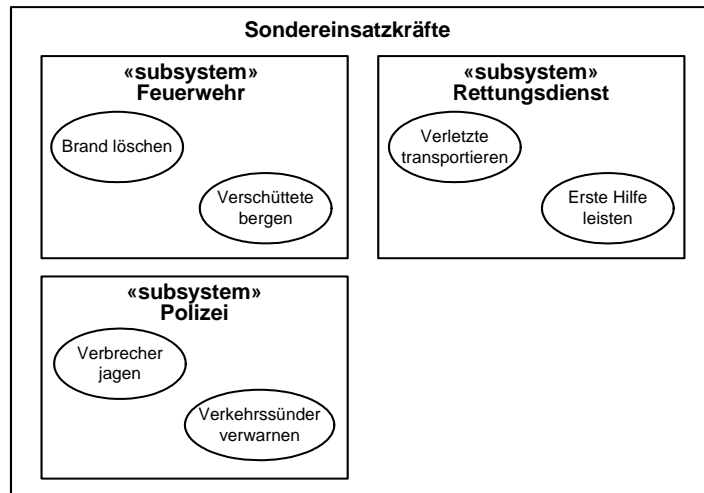


Abbildung 9.14: System und Subsysteme, die die Use-Cases realisieren

Abbildung 9.14 zeigt exemplarisch eine Abgrenzung verschiedener Teilbereiche und ordnet beispielsweise Use-Cases wie Brand löschen und Verschüttete bergen dem Subsystem Feuerwehr zu.

### 9.4.3 Akteur

#### Notation



Abbildung 9.15: Die gebräuchlichste Notation für einen Akteur – das Strichmännchen.

#### Beschreibung

Externe  
Interaktionspartner

Ein Akteur interagiert mit dem System, steht aber immer außerhalb davon. Beachten Sie zudem, dass ein Akteur lediglich eine Rolle darstellt, die mit dem System interagiert. Ein Akteur muss nicht zwangsläufig eine natürliche Person sein, sondern kann auch ein Sensor, ein Zeitereignis oder ein anderes Gerät sein. Weil der Akteur für eine Rolle steht, ist es zwingend erforderlich, dass er mit einem Namen versehen wird.

Die UML fasst den Begriff des Akteurs sehr weit und losgelöst von Use-Case-Diagrammen, wenngleich Akteure zumeist in diesen Diagrammen verwendet werden.

## Akteure in Use-Case-Diagrammen

In einem Use-Case-Diagramm interagiert ein Akteur mit einem Use-Case, indem er dessen Ausführung anstößt oder an der Ausführung *aktiv oder passiv* beteiligt ist. Zwischen dem Use-Case und dem Akteur findet ein wechselseitiger Austausch von Signalen und Daten statt. Das „Drehbuch“ dafür liefert die Verhaltensdefinition des internen Use-Case-Ablaufs.

Die Beteiligung eines Akteurs an einem Use-Case-Ablauf wird durch eine Assoziation (Kapitel 3.4.7) zwischen dem Akteur und dem Use-Case dargestellt. Die Assoziationen müssen binär sein, das bedeutet, dass an einer Assoziation genau zwei Partner beteiligt sind. Um den Ein- und Ausgabefluss darzustellen, dürfen die Assoziationen zudem gerichtet sein.

Assoziation



3.4.7

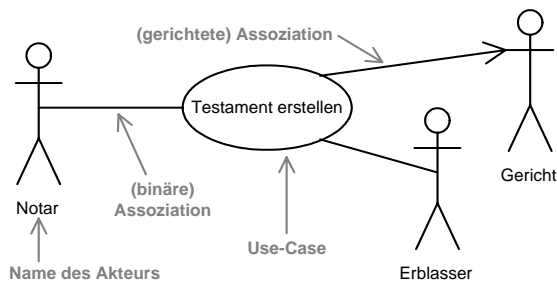


Abbildung 9.16: Assoziationen zwischen Use-Case und Akteuren

Die Beziehung lässt sich weiter mit den aus Klassendiagrammen bekannten Mitteln (Kapitel 3.4.7) verfeinern. Sie können insbesondere Rollen, Assoziationsnamen und Multiplizitäten antragen.

Beschriftung der  
Assoziation

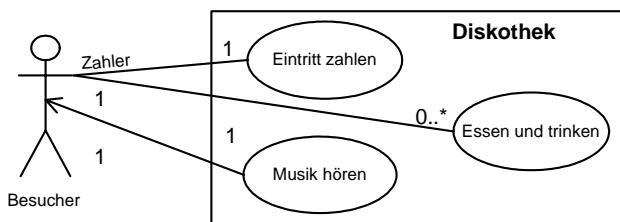


Abbildung 9.17: Ausführliche Assoziationen zwischen Akteur und Use-Cases

Das Beispiel *Diskothek* zeigt verschiedene Notationen von Assoziationen zwischen Akteur und Use-Case. Die mittlere Assoziation zeigt, dass ein Akteur mehrmals den Anwendungsfall *Essen und Trinken* anstoßen kann, aber nicht muss. Dies hängt davon ab, wie groß der Hunger und der Durst und das Stehvermögen des Besuchers sind. Die Assoziation zwischen dem Besucher und dem Use-Case *Musik hören* ist eine gerichtete Assoziation, die zeigt, dass der Informationsfluss nur einseitig in Richtung Akteur verläuft.

Bei dem Use-Case *Eintritt zahlen* nimmt der Akteur die Rolle des *Zahlers* ein. Außerdem drückt die Assoziation zwischen dem Akteur und dem Use-Case aus, dass genau ein *Zahler* genau einmal den Use-Case *Eintritt zahlen* anstößt bzw. daran beteiligt ist.

Wenn Sie das System modellieren, müssen Sie die Akteure immer außerhalb der Systemgrenzen (Rahmen) anbringen.

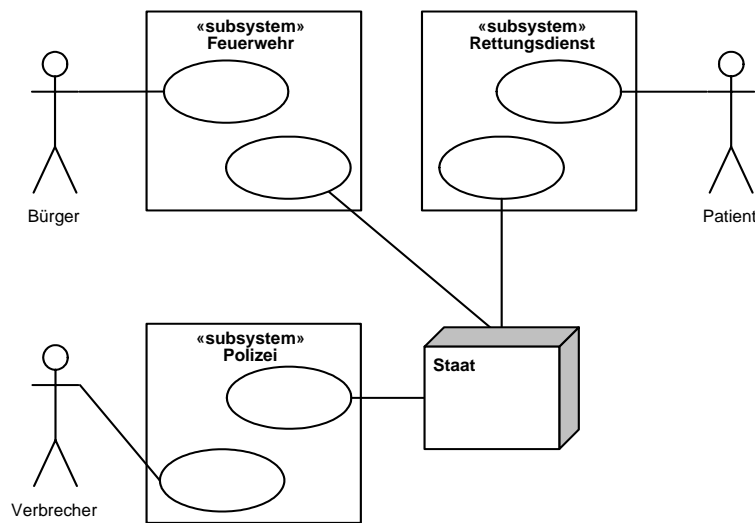


Abbildung 9.18: Akteure stehen außerhalb des Systems

Alternative Darstellungsformen und Zusatzinformationen

Akteur = spezialisierte Klasse

Ein Akteur im Sinne des UML-Metamodells ist eine spezialisierte Klasse, die einigen Beschränkungen unterliegt. So dürfen Sie zum Beispiel, wie bereits erwähnt, nur binäre Assoziationen an einen Akteur antragen. Zudem muss er sich außerhalb des Systems befinden.



Dafür steht Ihnen aber das umfangreiche Repertoire einer Klasse (Kapitel 3.4.1) zur Verfügung. Sie können dem Akteur Attribute und Operationen zuordnen oder ihn mit einem Stereotyp (Kapitel 3.4.11) versehen.

Notationsverhalten

Aus der Spezialisierungsbeziehung ergibt sich auch die Rechtecksnotation, in der der Stereotyp «actor» anzeigt, dass es sich nicht um eine „normale“ Klasse, sondern um eine „spezialisierte“ Klasse handelt.

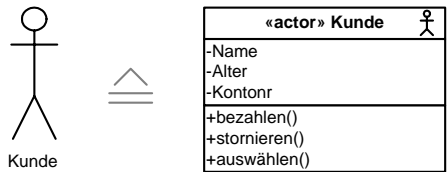


Abbildung 9.20: Die Klasse „Kunde“ – ein Akteur

Grafische Eigenkreationen

Zudem haben Sie die Möglichkeit, statt des Strichmännchens oder des Rechtecks eigene, definierte Symbole zu verwenden. Diese sind häufig eingängiger als ein Strichmännchen. Abbildung 9.21 zeigt einige typische Vertreter. Den Stereotyp «actor» können Sie optional angeben.

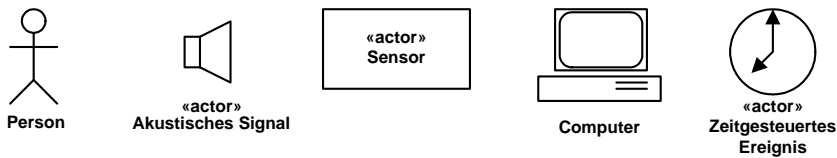


Abbildung 9.21: Alle meine Akteure ...

### Beziehungen zwischen Akteuren

Akteure dürfen „untereinander“ mit einer binären Assoziation verbunden werden.<sup>3</sup> Somit können Sie das Beziehungsgeflecht der Akteure außerhalb Ihres Systems darstellen. Tun Sie das nur in Fällen, in denen es für das Systemverständnis unabdingbar ist, denn eigentlich sollten Sie sich um Ihr System und nicht um dessen Umwelt kümmern.

Beziehungs-  
geflechte außer-  
halb Ihres Systems



Abbildung 9.22: Eine binäre Assoziation zwischen zwei Akteuren

Ein weitere Klasseneigenschaft, die Akteure besitzen, ist die Generalisierungs- und Spezialisierungsmöglichkeit (Kapitel 3.4.6). Hierbei wird die Kommunikationsbeziehung zwischen Akteur und Use-Case weitervererbt: Der spezialisierte Akteur ist an den gleichen Use-Case-Abläufen beteiligt wie der vererbende Akteur.

Generalisierung



3.4.6

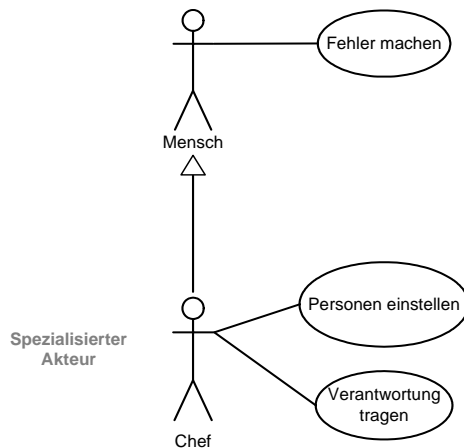


Abbildung 9.23: Auch ein Chef ist ein Mensch und macht Fehler

<sup>3</sup> Dies folgt aus der Tatsache, dass Akteure selbst spezialisierte Klassen sind und Akteure mit Klassen mittels Assoziationen in Beziehung stehen dürfen (siehe Abschnitt Akteure in weiteren UML-Diagrammen).

Erlaubte  
Assoziationen  
eines Akteurs

Akteure in weiteren UML-Diagrammen

Die UML erlaubt Akteuren nicht nur Beziehungen zu Use-Cases, sondern auch zu Komponenten (darunter auch Subsystemen), Knoten und Klassen (siehe Abbildung 9.25).

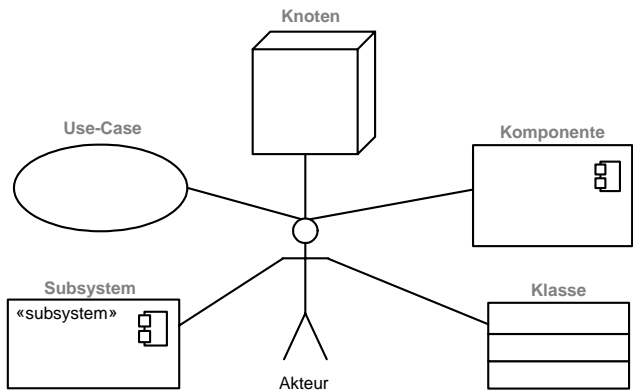


Abbildung 9.25: Erlaubte Assoziationen eines Akteurs

Ein Akteur darf dementsprechend auch in den Diagrammen modelliert werden, in denen diese Elemente abgebildet werden.

Kontextdiagramme

Nebenbei bemerkt erlaubt dieser Freiheitsgrad, das aus der Strukturierten Analyse [DPI79] bekannte Kontextdiagramm mit UML-Notationsmitteln nachzubilden. Dieses Diagramm dient vorwiegend der Abgrenzung von Systemen zu Nachbarsystemen und zeigt den Ein- und Ausgangsfluss.

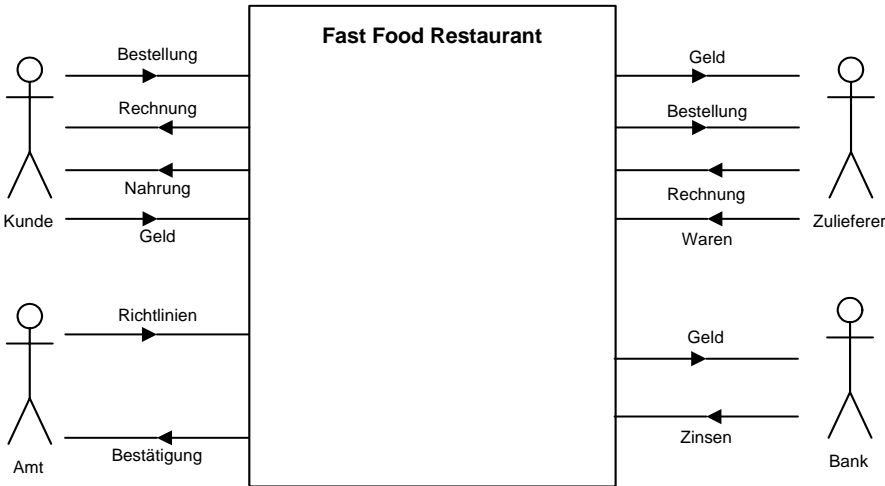


Abbildung 9.26: Auch mit der UML kann man Kontextdiagramme modellieren

Die Assoziationen zwischen Akteur und System in Abbildung 9.26 wurden zusätzlich mit Informationsflüssen versehen.



## Anwendung

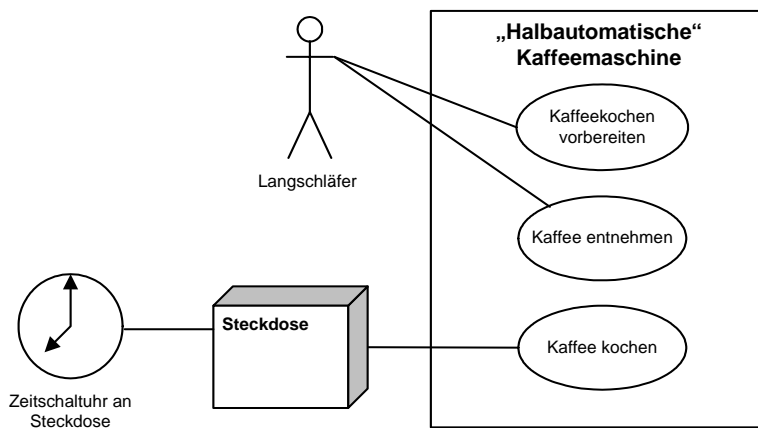


Abbildung 9.27: Morgens endlich länger schlafen

Abbildung 9.27 zeigt die Anwendung verschiedener Darstellungen für Akteure. Einerseits wird hier ein Strichmännchen in der Rolle des Langschläfers als Akteur verwendet. Andererseits ist die Steckdose ein Akteur, der durch eine Zeitschaltuhr geschaltet wird.

### 9.4.4 «include»-Beziehung

#### Notation

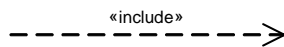


Abbildung 9.28: Die «include»-Beziehung

Die «include»-Beziehung wird durch eine unterbrochene gerichtete Kante mit dem Schlüsselwort «include» dargestellt. Nähere Informationen zu dieser Abhängigkeitsbeziehung finden Sie in Kapitel 3.4.9 „Abhängigkeitsbeziehung“ und in Kapitel 5.4.4 „Abhängigkeiten“.

Abhängigkeitsbeziehung



3.4.9, 5.4.4

#### Beschreibung

Die «include»-Beziehung visualisiert, dass ein Use-Case (A) das Verhalten eines anderen Use-Case (B) importiert.

Verhaltensimport

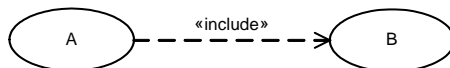
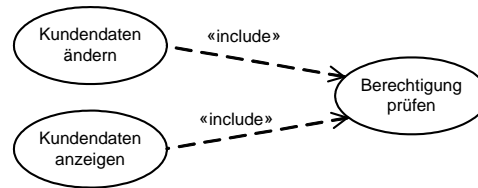


Abbildung 9.29: Use-Case A inkludiert Use-Case B

So importiert der Use-Case Kundendaten ändern in Abbildung 9.30 den Use-Case Berechtigung prüfen. Das bedeutet, dass während der Use-Case Kundendaten ändern instanziiert ist (abläuft), in einem Ablaufschritt der Use-Case Berechtigung prüfen gerufen wird, dann abläuft und sein Ergebnis in Kundendaten ändern genutzt wird.

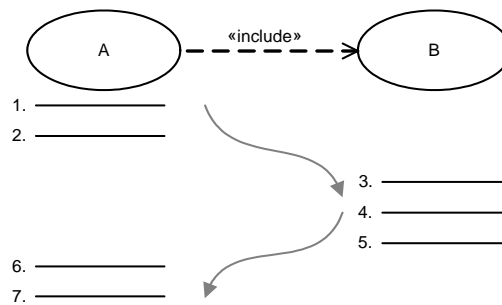


**Abbildung 9.30:** Zentral beschriebenes Verhalten inkludieren

Mehrfache  
Inklusion

Beispiel 9.30 zeigt, dass ein Use-Case durch unterschiedliche Use-Cases mehrfach inkludiert werden darf. Dadurch können Sie mehrfach benötigtes Verhalten einmalig an zentraler Stelle beschreiben und beliebig oft nutzen. Formulieren Sie deshalb das ausgelagerte Verhalten so, dass es in beliebigen Use-Case-Abläufen nutzbar ist.

Eine «include»-Beziehung ist *nicht optional*; das Verhalten wird *immer* inkludiert. Erst durch die Inklusion ergibt sich ein (sinnvolles) Gesamtverhalten. Der Use-Case, der das Verhalten einbindet, ist somit abhängig vom Use-Case, den er inkludiert. Deswegen wird die Notation vom abhängigen, importierenden zum unabhängigen inkludierten Use-Case gerichtet.



**Abbildung 9.31:** Der Ablauf einer «include»-Beziehung

Gemeinsames  
Gesamtverhalten

Abbildung 9.31 zeigt das Zusammenspiel der Use-Case-Abläufe. Nachdem zum Beispiel die ersten beiden Schritte von Use-Case A abgearbeitet sind, werden die inkludierten Schritte 3 bis 5 von Use-Case B durchlaufen, bevor 6 und 7 den Gesamtablauf komplettieren. Das Verhalten von Use-Case A setzt sich somit aus den Schritten 1-7 zusammen. Das Ergebnis von Use-Case B (Schritt 5) wird in den Schritten 6 und 7 von Use-Case A genutzt. Beachten Sie, dass hier keine Parallelität vorliegt, „nach außen“ tritt *ein gemeinsames Gesamtverhalten* auf.

Verbot von Zyklen

Ein Use-Case darf sich weder selber inkludieren noch einen anderen Use-Case, der wiederum ihn selber inkludiert (Verbot zyklischer Abhängigkeiten). Dies bedeutet: Wenn Use-Case A den Use-Case B inkludiert, darf Use-Case B nicht Use-Case A inkludieren.

Entscheidend ist, dass durch «include»-Beziehungen bzw. die Anordnung der Use-Cases im Diagramm keine zeitlichen Reihenfolgen vorgegeben sind. Nur die Verhaltensbeschreibung des Use-Case legt fest, in welcher Reihenfolge inkludiert wird.

Keine zeitlichen  
Reihenfolgen

## Anwendung

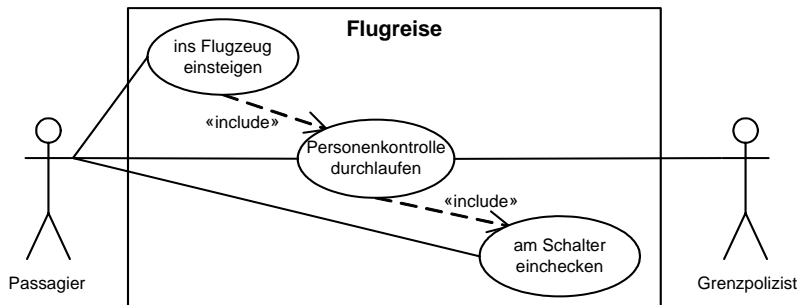


Abbildung 9.33: «include»-Beziehungen in der Anwendung

In dem Anwendungsbeispiel einer Flugreise wird der Use-Case ins Flugzeug einsteigen hierarchisch zerlegt. Dadurch wird ausgedrückt, dass der Use-Case am Schalter einchecken komplett im Use-Case Personenkontrolle durchlaufen enthalten ist. Dieser wiederum wird vollständig im Use-Case ins Flugzeug einsteigen durchlaufen.

### 9.4.5 «extend»-Beziehung

#### Notation

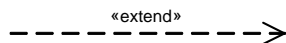


Abbildung 9.34: Die «extend»-Beziehung

Eine «extend»-Beziehung ist eine unterbrochene gerichtete Kante mit der Bezeichnung «extend» vom erweiternden Use-Case zum erweiterten Use-Case.

Notation

#### Beschreibung

Die «extend»-Beziehung zeigt an, dass das Verhalten eines Use-Case (A) durch einen anderen Use-Case (B) erweitert werden kann, aber nicht muss.

Verhaltens-  
erweiterung

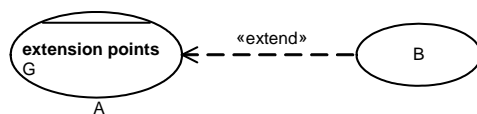
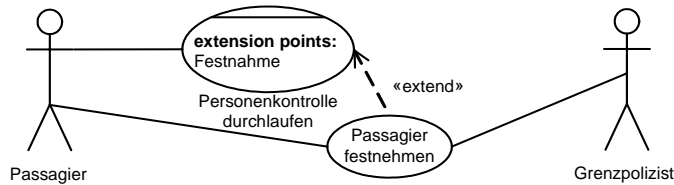


Abbildung 9.35: Use-Case B erweitert Use-Case A

Abbildung 9.36 zeigt den Use-Case Personenkontrolle durchführen, der in bestimmten Fällen durch Passagier festnehmen erweitert wird.



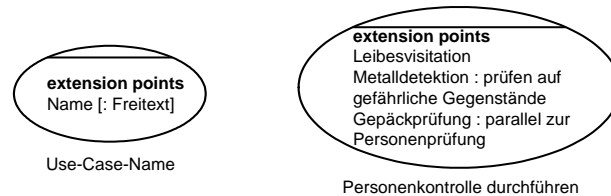
**Abbildung 9.36:** «extend»-Beziehungen erweitern Use-Cases

Extension point

Den Zeitpunkt, an dem ein Verhalten eines Use-Case erweitert werden kann, bezeichnet man als *Erweiterungspunkt* (engl. extension point). Ein Use-Case darf mehrere Erweiterungspunkte besitzen.

Die Erweiterungspunkte werden in einem mit *extension points* bezeichneten Abschnitt innerhalb der Use-Case-Ellipse dargestellt und müssen benannt sein. Die Bezeichnung des Use-Cases verschiebt sich unter die Ellipse.

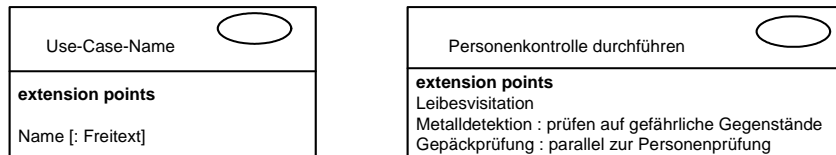
Sie dürfen optional an den Namen des Erweiterungspunktes einen beliebigen Text anhängen. Dieser Text ist durch einen Doppelpunkt vom Namen des Erweiterungspunktes abzutrennen. Sie haben dadurch die Möglichkeit, den Erweiterungspunkt präziser zu beschreiben oder die Auftretsstelle im Use-Case-Ablauf anzugeben.



**Abbildung 9.37:** Ein Use-Case mit Erweiterungspunkten

Alternative Darstellung

Bei einer großen Anzahl an Erweiterungspunkten empfiehlt sich, die Use-Cases in der Rechtecksnotation darzustellen (Abbildung 9.38).



**Abbildung 9.38:** Erweiterungspunkte eines Use-Cases in der Rechtecksnotation

Neben dem Erweiterungspunkt können Sie zudem eine Bedingung für die Erweiterung angeben. Die Bedingung wird bei Erreichen des Erweiterungspunktes geprüft. Ist die Bedingung wahr, wird der Ablauf erweitert, das heißt der entsprechende referenzierte Use-Case durchlaufen. Ist die Bedingung nicht erfüllt, läuft der Use-Case-Ablauf „normal“ weiter.

Die entsprechenden Bedingungen und der zugehörige Erweiterungspunkt werden als Notizzettel (condition) an die «extend»-Beziehung notiert.

Bedingungen  
der Erweiterung  
notieren

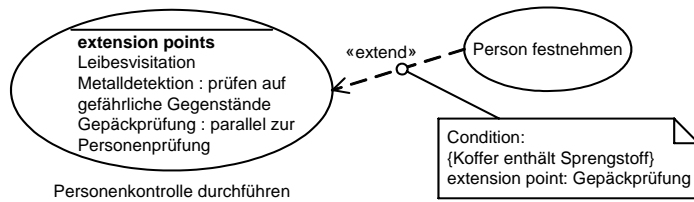


Abbildung 9.39: Die Bedingungen eines extension points als Notiz

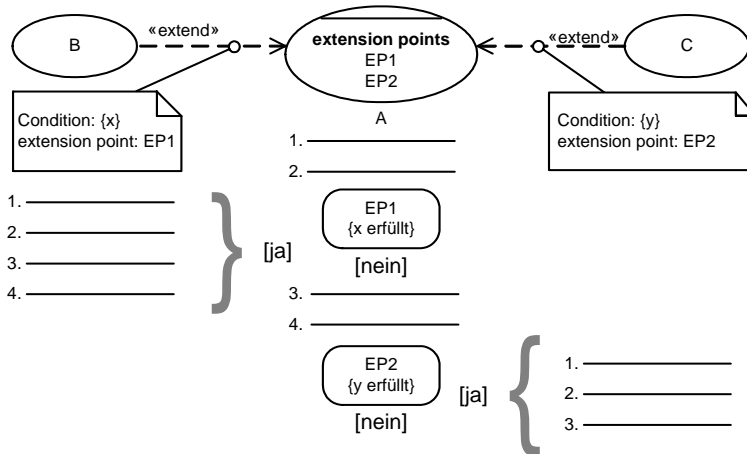


Abbildung 9.40: Der Ablauf einer «extend»-Beziehung

Die Angabe der Bedingung ist optional. Fehlt die Bedingung, wird der Use-Case immer erweitert. Ein erweiternder Use-Case kann mehrere Use-Cases erweitern und auch selbst erweitert sein. In der Praxis werden Generalisierungs- und «extend»-Beziehungen leider häufig falsch eingesetzt, deshalb hier eine kurze Übersicht:

Die «extend»-Beziehung dient nur der Verhaltenserweiterung, die Bedingungen unterliegt (das heißt, das Verhalten eines Use-Cases kann, muss aber nicht erweitert werden). Die Generalisierungsbeziehung hingegen kopiert und überschreibt das Verhalten des Basis-Use-Case.

Die Generalisierung ist nicht die Obermenge der «extend»-Beziehung, obwohl beide das Verhalten des Basis-Use-Case durch Spezialisierung erweitern. Die Beziehungstypen unterscheiden sich darin, dass nur die Generalisierung zusätzlich das Überschreiben des Verhaltens ermöglicht und dass nur bei der «extend»-Beziehung die Erweiterung durch Bedingungen steuerbar ist. Bei der Generalisierung dagegen wird immer erweitert; die Erweiterung erfolgt zur Spezifikationszeit und nicht zur Ablaufzeit.

Die Generalisierung ist nur in wenigen Fällen geeignet, da sie häufig zu semantischen Zweideutigkeiten führt. Denkbar wäre eine Generalisierung, wenn mehrere alternative Handlungen in verschiedenen Schritten gleich sind, also mehrere «extend»-Beziehungen vonnöten wären.

Anwendung

Im folgenden Beispiel wird die Verwendung der «extend»-Beziehung nochmals verdeutlicht. Dargestellt ist die mehrfache Erweiterung des Use-Cases Nahrung verspeisen und die Erweiterung des erweiternden Use-Cases Mängel reklamieren durch den Use-Case Zahlung verweigern.

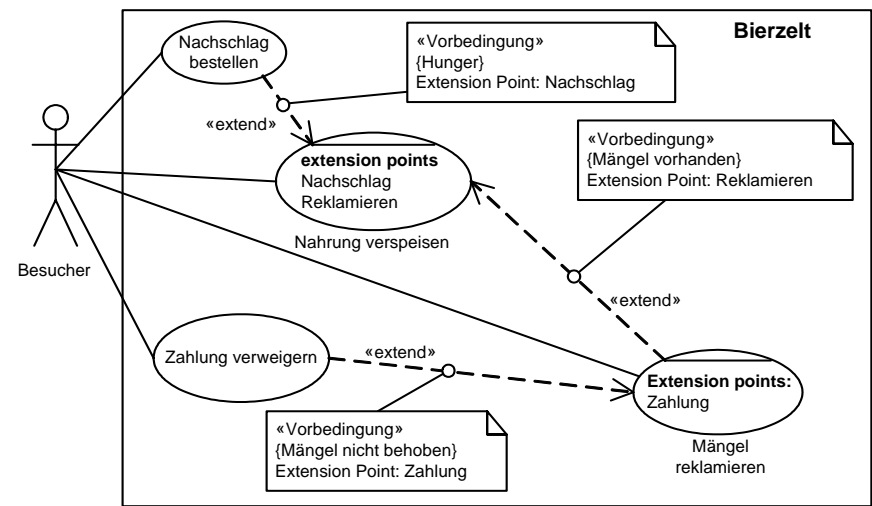


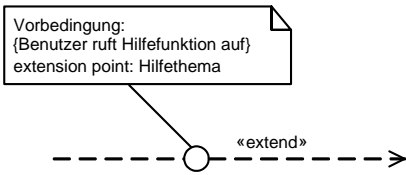
Abbildung 9.41: Die «extend»-Beziehung im Einsatz

Tabelle 9.2 stellt nochmals die «include»- und «extend»-Beziehung gegenüber.

Tabelle 9.2: Vergleich «include» - und «extend»-Beziehung

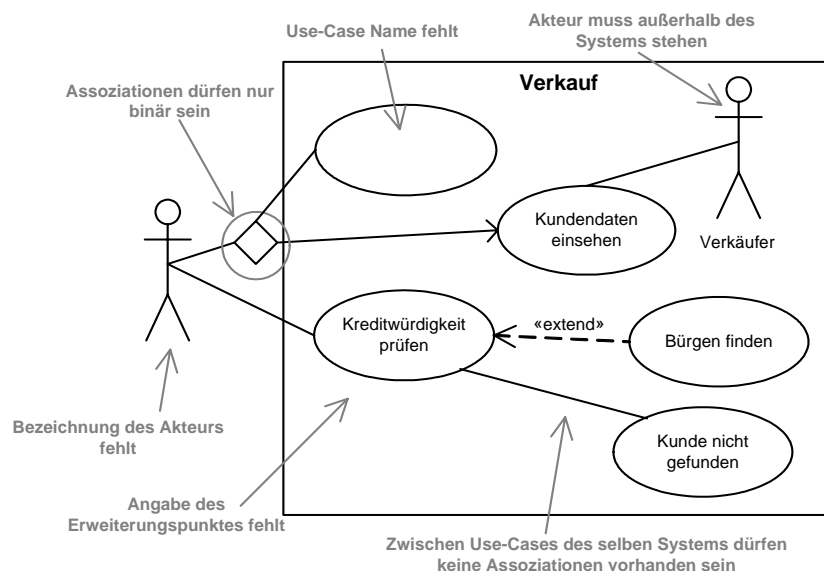
	«include»-Beziehung	«extend»-Beziehung
Notation		
Bedeutung	Ablauf von A schließt immer Ablauf von B ein.	Ablauf von A kann, muss aber nicht durch Ablauf von B erweitert werden.
Wann wird die Beziehung genutzt?	Ablauf von B kann in verschiedenen Use-Cases genutzt werden. Hierarchische, funktionale Zerlegung.	A besitzt neben Normalverhalten auslagerbare Sonderfälle.
Bedeutung für die Modellierung	A ist meist unvollständig und wird erst durch Inklusion B zu einem vollständigen Ablauf. B ist häufig künstlich zur Redundanzvermeidung gebildet.	A ist meist vollständig und kann durch B optional erweitert werden. B ist meist in sich vollständig.
Abhängigkeiten	A muss B bei der Modellierung berücksichtigen. B wird <b>unabhängig von A</b> modelliert, um die Nutzung durch weitere Use-Cases sicherzustellen (Wiederverwendbarkeit), B muss in sich <b>nicht vollständig</b> sein („B weiß nicht, durch wen er inkludiert wird“).	A muss durch Angabe von Erweiterungspunkten auf die Erweiterung durch B vorbereitet werden. B wird <b>in sich vollständig</b> und <b>unabhängig</b> von A modelliert („B weiß nicht, wen er erweitert“).

## 9.5 UML 2 Update

UML 1.x	UML 2
Ein Akteur darf unbenannt sein.	Ein Akteur muss einen Namen haben.
Bei der «extend»-Beziehung werden die Vorbedingungen nur in der Nähe der entsprechenden Relation angetragen:	Die Vorbedingung und die entsprechenden <i>extension points</i> werden als Notiz an die Erweiterungsbeziehung angehängt: 
Nur Pakete können Use-Cases besitzen.	Classifier im Allgemeinen können Use-Cases besitzen, siehe Kapitel 3.
Als Realisierer von Use-Cases werden (Sub-) Systeme impliziert, obwohl jeder Classifier einen Use-Case realisieren darf.	Es wird deutlich herausgestellt, dass alle Classifier Use-Cases realisieren können, siehe Kapitel 7.

## 9.6 Do & Don't

- Verwenden Sie Use-Cases nicht zur detaillierten Beschreibung von Operationen oder Funktionen.
- Spezifizieren Sie nicht-funktionale Anforderungen nicht mittels Use-Cases.
- Vermeiden Sie eine zu starke funktionale Zerlegung mittels Use-Cases, wenn Sie nicht strukturiert, sondern objektorientiert vorgehen.
- Greifen Sie zur Modellierung von internen Strukturen, deren Abläufen oder von Reihenfolgen auf andere Verhaltensdiagramme zurück – auch wenn die Use-Cases in einem Diagramm oft eine zeitliche Reihenfolge vorzugeben scheinen.
- Erstellen Sie Use-Cases aus der Sicht des Anwenders und nicht aus der Sicht des Entwicklers.
- Modellieren Sie nicht zu viele Use-Cases (Empfehlung: maximal zehn Use-Cases pro Diagramm).
- Gehen Sie mit den erlaubten Beziehungen (include und extend) zwischen den Use-Cases sparsam um, denn sie reduzieren die intuitive Verständlichkeit für den Leser.
- Verfeinern Sie die Use-Cases durch eine geeignete Technik.
- Benennen Sie den Akteur immer in der Einzahl, vergeben Sie nie konkrete Namen wie Frau Meier (der Akteur ist eine Rolle).
- Schreiben Sie, sobald Sie eine extend Beziehung einsetzen, alle UC Bezeichnungen einheitlich unten hin.



**Abbildung 9.42:** Mögliche Notationsfehler in einem Use-Case-Diagramm

## 9.7 Literatur

UML Superstructure. OMG Adopted Specification, ptc/03-08-02. Chapter 16

- [Arm00] **Armour, F.:** Advanced Use Case Modelling. Addison-Wesley GmbH. 2000.
- [Coc98] **Cockburn, A.:** Writing Effective Use Cases. Addison-Wesley GmbH, 1998.
- [DP179] **DeMarco T., Plauger P.J.:** Structured Analysis and System Specification. Prentice Hall, 1979.
- [HRu02] **Hruschka, P., Rupp, C.:** Agile Softwareentwicklung für Embedded Real-Time Systems mit der UML. Hanser, 2002.
- [Jac92] **Jacobson, I.:** Object-Oriented Software Engineering – A Use Case Driven Approach. Addison Wesley Longman, 1992.
- [Kru00] **Kruchten, P.:** The Rational Unified Process – An Introduction. 2<sup>nd</sup> ed. Addison Wesley Longman, 2000.
- [MPa84] **McMenamin S. M., Palmer J. F.:** Essential System Analysis. Prentice Hall, Eaglewood Cliffs 1984.