cooperate

Einführung in die grafische Beschreibungssprache UML

Version 2.0

Vanessa Petrausch

Projektpartner:

Karlsruher Institut für Technologie

FZI

Gefördert durch:

Bundesministerium für Arbeit und Soziales

aus Mitteln des Ausgleichsfonds
Förderkennzeichen: 01KM141108

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Unified Modeling Language™ (UML™) ist eine durch die Object Management Group (OMG) standardisierte grafisch repräsentierte Sprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Modellen für Softwaresysteme, Geschäftsmodelle und andere Nicht-Softwaresysteme. Die UML ist damit nicht fachgebunden und kann in nahezu jedem Fachbereich eingesetzt werden. Im Folgenden wird jedoch insbesondere auf die Nutzung für Softwaresysteme eingegangen.

Bei der UML handelt es sich um eine Modellierungssprache, nicht um eine Programmiersprache. Es existieren jedoch verschiedene Werkzeuge, mit welchen automatisiert Quelltext aus UML-Diagrammen erstellt werden kann. Die grafische Darstellung besteht aus einfachen geometrischen Formen, welche, in einen logischen Zusammenhang gebracht, einen bestimmten Aspekt eines Systems darstellen, wie z.B. den Zusammenhang von verschiedenen Klassen. Es gibt verschiedene Arten von Diagrammen in der UML, welche in Struktur- und Verhaltensdiagramme aufgeteilt werden. Die Strukturdiagramme sind statische Diagramme, welche, wie der Name schon vermuten lässt, die Strukturen von Systemen darstellen. Verhaltensdiagramme dagegen beschreiben Abläufe. Sie sind dynamisch und stellen Zusammenhänge im Zeitablauf dar.

Ziel dieser Einführung in UML ist, Menschen mit Seheinschränkung den Zugang zu UML zu erleichtern und ein Verständnis für deren Aufbau zu vermitteln. Daher wird ein Schwerpunkt auf der Beschreibung der grafischen Darstellung von UML-Diagrammen gelegt, um eine gemeinsame Basis zur Diskussion bereitzustellen. Dies soll dazu beitragen, dass sich Menschen mit und ohne Sehschädigung einfacher über Softwaresysteme, die in UML modelliert sind, austauschen können.

2 Aufbau des Dokuments

Nachfolgend wird zuerst auf einige Elemente der UML eingegangen, welche in allen Diagrammtypen verfügbar sind und deshalb nicht in jedem Kapitel extra erläutert werden. Danach wird näher auf fünf Diagrammtypen eingegangen, wobei die einzelnen Elemente des entsprechenden Typs Schritt für Schritt mit Abbildungen erklärt werden. Die Auswahl der Diagrammtypen erfolgte auf Basis von häufig genutzten Diagrammen in der Praxis, welche durch mehrere Umfragen und Studien ermittelt wurden (Petre, 2014), (Dobing & Parson, 2006). Als Ergebnis der Umfragen werden das Klassendiagramm, das Anwendungsfalldiagramm, das Aktivitätsdiagramm, das Zustandsdiagramm und das Sequenzdiagramm vorgestellt. Die Abbildungen zur Erklärung stellen nicht immer vollständige Diagramme dar, sondern nur einzelne Ausschnitte. Die Beispiele sind teilweise (Kecher & Salvanos, 2015) entnommen oder daran angelehnt. Nach der Erklärung aller Elemente wird ein Beispiel beschrieben, in dem einige der vorgestellten Elemente kombiniert in einem vollständigen Diagramm verwendet werden. Als Beispielszenario dient eine Navigationsapplikation für Fußgänger, mit deren Hilfe Personen von einem Ort zum anderen navigieren können.

Anschließend gibt es in jedem Kapitel zwei Übungsaufgaben. Die erste besteht aus je fünf Multiple-Choice-Fragen, welche beantwortet werden sollen um das Verständnis des aktuellen Diagrammtyps zu überprüfen. Die Fragen wurden dabei dem UML-Quiz der TU Wien entnommen und zum Teil an die Lerninhalte dieser Schulung angepasst (Business Informatics Group, 2015). Das Quiz ist dabei eine Erweiterung des Buches (Seidl, Scholz, Huemer, & Kappel, 2015). Die zweite Übung besteht aus einer praktische Modellierungsaufgabe, in welcher textuell ein Szenario beschrieben wird, welches als Diagramm dargestellt werden soll. Diese Aufgabe kann

als umgangssprachlicher Text formuliert werden. Dabei soll textuell beschrieben werden welche Elemente verwendet werden und wie diese miteinander in Beziehung stehen. Fortgeschrittene Nutzer können zur Bearbeitung dieser Aufgabe auch PlantUML-Code verwenden. Im Anhang des Dokuments werden beispielhafte Lösungen aller Diagramme mittels einer grafischen Variante mit Alternativtext vorgestellt.

Abschließend befindet sich eine tabellarische Zusammenfassung aller Elemente mit Namen, kurzer Beschreibung und der Darstellung in textueller und grafischer Form in jedem Kapitel.

Alle Abbildungen sind zusätzlich als taktile Grafik aufbereitet. Die Grafiken werden dabei durch die Nummerierungen der Abbildungen referenziert. Alle Abbildungsnamen innerhalb dieses Dokuments sind folgendermaßen strukturiert: *Abbildung <Nummer>: Beschreibungstext*. Zur besseren Orientierung werden am Anfang jedes Unterkapitels die Seitenzahlen, auf welchen sich die enthaltenen Grafiken innerhalb der taktilen Mappe befinden, angegeben.

3 Grundelemente der UML

In diesem Abschnitt werden zuerst Elemente der UML vorgestellt, welche in allen Diagrammtypen, unabhängig von deren Typ, eingesetzt werden können. Dabei spielt es keine Rolle, ob es sich um ein Struktur- oder Verhaltensdiagramm handelt.

3.1 Darstellung

Alle Diagramme können freistehend oder innerhalb eines Rahmens dargestellt werden, welcher die Typbezeichnung des Diagramms und einen frei wählbaren Namen beinhalten kann. Der Rahmen besteht dabei laut Spezifikation aus einem Rechteck mit dem Typ und Namen am oberen linken Rand, abgegrenzt durch Linien. Diese Linien bilden mit dem Rahmen ein Pentagon, ähnlich zu einem Karteireiter in Computersystemen. Zugunsten der Übersichtlichkeit wird in diesem Dokument auf diese Notation verzichtet und es werden nur die Elemente an sich dargestellt.

3.2 Kommentare

Zu jedem Element eines Diagramms können Kommentare hinzugefügt werden. Ein Kommentar wird durch ein Rechteck mit einer umgeknickten Ecke oben rechts (Eselsohr) dargestellt. Innerhalb des Rechtecks befindet sich der eigentliche Kommentar. Dieser kann in beliebiger Form dargestellt werden, da er keine Änderung der Semantik des Diagramms bewirkt, sondern lediglich als Erläuterung dient. Der Kommentar wird mithilfe einer gestrichelten Linie an ein Element angehängt.

3.3 Verbindungen

Zwischen verschiedenen Elementen eines Diagramms können Verbindungen bestehen. Diese Verbindungen stellen den Zugriff oder Abhängigkeiten zwischen den Elementen dar. Je nach Diagrammtyp gibt es unterschiedliche Verbindungen oder Bedeutungen. Diese werden in dem jeweiligen Unterkapitel näher beschrieben. Generell besteht jede Verbindung aus einer Linie zwischen mehreren Elementen. Dabei kann diese an keinem, einem oder an beiden Enden eine Pfeilspitze erhalten. Eine Pfeilspitze deutet darauf hin, dass in Richtung dieses Elements navigiert werden kann, d.h. weitere modellierte Elemente können auf dieses zugreifen oder eine Verbindung aufbauen und z.B. Eigenschaften verwenden. Einige dieser Verbindungen, überwiegend diejenigen in Strukturdiagrammen, werden Assoziation genannt und können zusätzliche Multiplizitäten, auch Kardinalitäten genannt, besitzen. Für eine bessere Verständlichkeit dieser Notation wird diese mit konkreten Beispielen in Kapitel 4.1 erläutert.

3.4 Stereotype

Stereotype geben Auskunft über den Zweck oder die Rolle eines Elements und werden in doppelt spitzen Klammern, sogenannten Guillements, angegeben. Beispielsweise kann in einem Klassendiagramm angegeben werden, dass eine Klasse ein Interface ist, indem *<<interface>>* direkt über den Klassennamen platziert wird.

4 Strukturdiagramme

Strukturdiagramme bezeichnen die Klasse aller Diagramme, welche statische, d.h. unveränderliche, Komponenten eines Systems modellieren. Als Strukturdiagramm wird nachfolgend das Klassendiagramm vorgestellt.

4.1 Klassendiagramm

Die taktilen Grafiken des Klassendiagramms (im englischen Class Diagram) befinden sich auf den Seiten 1 – 7 der taktilen Mappe.

Das Klassendiagramm ist das am häufigsten genutzte Diagramm der UML (Petre, 2014), (Dobing & Parson, 2006). Es kann sowohl als erster Überblick über die Konzepte und das verwendete Vokabular genutzt werden, als auch als detaillierte Basis für eine automatisierte Generierung von Code in einer späteren Entwurfsphase. Das Klassendiagramm modelliert statische Strukturen, welche sich über die Zeit nicht ändern.

4.1.1 Aufbau von Klassen

Klassen beschreiben einen Art Bauplan für Objekte und definieren dabei die Struktur (Attribute einer Klasse) und deren Verhalten (Methoden der Klasse). Sie werden als Rechtecke dargestellt, welche durch horizontale Linien in drei Bereiche unterteilt werden können:

1. Im oberen Bereich steht der Name der Klasse,
2. im mittleren Bereich stehen Eigenschaften/Attribute der Klasse und
3. im unteren Bereich die Methodendeklarationen.

Der Klassenbegriff in der UML entspricht einer Klasse in einer objektorientierten Programmiersprache (OO-Sprache). So wie im Quellcode Klassen aus Eigenschaften und Methoden bestehen, gilt dies auch für die UML. Die Syntax für Attribute und Methoden ist folgendermaßen definiert:

Attribut:

Sichtbarkeit Attributname: Paket::Typ[Multiplizität] = Initialwert {Eigenschaftswerte}

Beispiel: public buchtitel: String = „Die besten Rezepte“

Methoden:

Sichtbarkeit Name (Parameterliste) [:Rückgabetyp][Multiplizität] {Eigenschaftswerte}

Beispiel: public bestellen (anzahlBücher: int): Buch {ordered}

Eigenschaftswerte:

z.B. readOnly, ordered, composite, unique

Sichtbarkeiten:

Textuelle Darstellung: public, protected, private, package

Symbolische Darstellung: + (für public), # (für protected), - (für private), ~ (für package)

Hat eine Methode keinen Rückgabetyp, wird das Schlüsselwort *„void“* genutzt. Es müssen nicht alle Elemente einer Klasse definiert werden: Ein eindeutiger Klassenname ist ausreichend. Ebenso müssen nicht alle einzelnen Elemente einer Attribut- oder Methodendefinition genutzt werden. Ein Beispiel, welches nur den Namen und Typ angibt, ist in Abbildung 1 zu sehen. Die linke Seite stellt den allgemeinen Aufbau einer Klasse dar. Die rechte Seite zeigt ein Beispiel einer Klasse *„Buch“* mit den Attributen *„titel:String“* und *„isbn:Int“*.

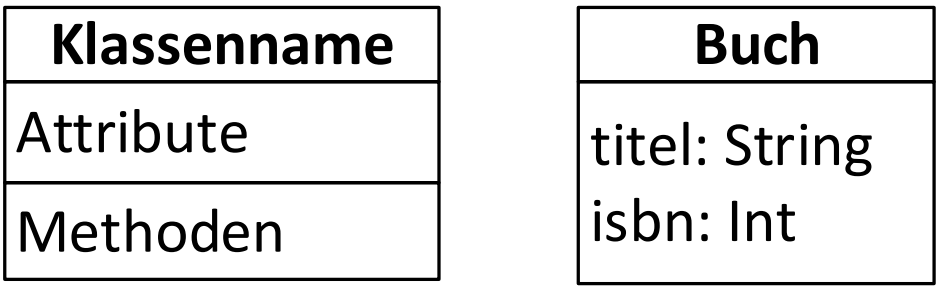


Abbildung 1: Generische Darstellung einer UML-Klasse (links) und konkrete Ausprägung einer Klasse (rechts)

4.1.2 Einfache Verbindungen zwischen Klassen

Klassen können in verschiedenen Beziehungen zueinander stehen. Verbindungen zwischen Klassen werden Assoziationen genannt. Miteinander verbundene Klassen sind potentielle Kommunikationspartnern, d.h. im Kontext von OO- Sprachen, dass diese auf Attribute und Methoden der beteiligten Klassen zugreifen können. Abbildung 2 bis Abbildung 4 zeigen drei verschiedene Assoziationen, welche sich in der Art der Navigierbarkeit unterscheiden.

- Abbildung 2 stellt die einfachste, eine ungerichtete, Assoziation dar, welche durch eine einfache durchgezogene Linie zwischen den Klassen dargestellt wird. Die Kommunikationsrichtung (Navigierbarkeit) ist hierbei nicht näher spezifiziert, d.h. es ist nicht ersichtlich, wie die Kommunikation realisiert ist. Im Beispiel sind zwei Klassen *„Buchhandlung“* und *„Kunde“* miteinander verbunden.

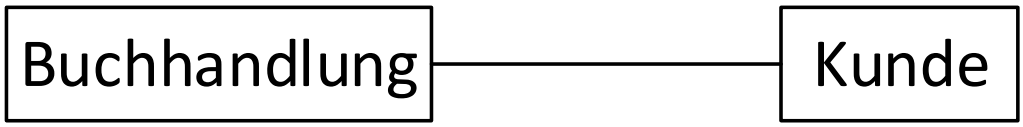


Abbildung 2: Ungerichtete Assoziation zwischen den Klassen „Buchhandlung“ und „Kunde“

- Abbildung 3 beschreibt eine bidirektionale Assoziation, welche an beiden Enden der beteiligten Klassen *„Verkäufer“* und *„Kunde“* mit einer einfachen Pfeilspitze gekennzeichnet ist. In diesem Fall liegt eine gerichtete Assoziation vor, da Pfeilrichtungen die Richtung der Kommunikation anzeigen. Im Falle einer bidirektionale Assoziation kann die Kommunikation in beide Richtungen ablaufen. Sowohl die Klasse *„Verkäufer“* als auch die Klasse *„Kunde“* können miteinander kommunizieren, d.h. potentiell auf Attribute und Methoden der jeweils anderen Klasse zugreifen.

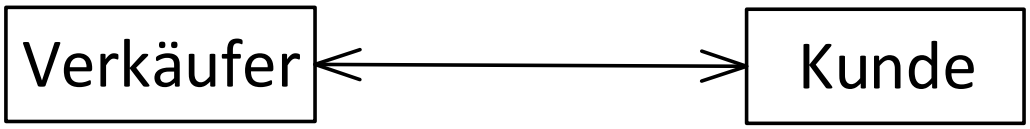


Abbildung 3: Bidirektional gerichtete Assoziation zwischen den Klassen „Verkäufer“ und „Kunde“

- Im Gegensatz dazu ist in Abbildung 4 eine gerichtete (direktionale) Assoziation gegeben. Hier kann nur die Klasse *„Bestellung“* auf Attribute und Methoden der Klasse

„Buch“ zugreifen, jedoch nicht umgekehrt, da die Pfeilrichtung nur von *„Bestellung“* nach *„Buch“* zeigt. Eine *„Bestellung“* mit mehreren *„Büchern“* kennt daher alle enthaltenen *„Bücher“*, ein *„Buch“* kennt jedoch nicht die anderen enthaltenen Bücher der *„Bestellung“*. Falls eine Kommunikationsrichtung explizit verboten werden soll, kann durch ein großes „X“ am Ende der Assoziation angezeigt werden, in welche Richtung keine Kommunikation stattfinden soll. In diesem Falle ist dies bei der Klasse *„Bestellung“*.

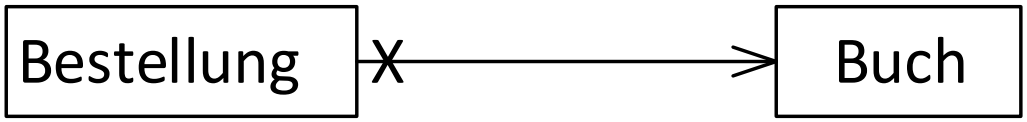


Abbildung 4: Gerichtete Assoziation von der Klasse „Bestellung“ zur Klasse „Buch“

4.1.3 Teil-Ganzes-Verbindungen zwischen Klassen

Weiterhin existieren spezielle Verbindungen zwischen Klassen, welche eine Teil-Ganzes-Beziehung darstellen. Die Begriffe können dabei wörtlich genommen werden. Bei einer Bestellung, die aus mehreren Produkten besteht, wie in Abbildung 4, wäre beispielsweise die *„Bestellung“* das „Ganze“ und die verschiedenen *„Bücher“* wären die einzelnen Teile der Gesamtbestellung. Statt „Ganzes“ kann auch der Begriff „zusammengesetztes Objekt“ verwendet werden. Bei einer solchen Beziehung unterscheidet man zwischen zwei Fällen: Komposition und Aggregation. Dargestellt werden beide durch ein zusätzliches Symbol am Ende der Assoziation, die am zusammengesetzten Objekt (dem „Ganzen“) endet. Die Komposition wird durch eine schwarz ausgefüllte Raute dargestellt, wie in Abbildung 5 zu sehen ist. Eine Aggregation dagegen, dargestellt in Abbildung 6, wird durch eine weiße Raute mit Rahmen dargestellt.

Die Komposition ist die stärkere Bindung der beiden Varianten. „Bei einer Komposition kann ein Teil immer nur in genau einem zusammengesetzten Objekt enthalten sein und die Lebensdauer der Komponenten entspricht immer der Lebensdauer des zusammengesetzten Objekts. Das zusammengesetzte Objekt wird hier als Kompositum bezeichnet“ (Lahres & Rayman, 2009). Abbildung 5 zeigt eine Komposition zwischen einer *„Buchhandlung“* als Kompositum und der Klasse *„Raum“* als Teil. Ein *„Raum“* kann außerhalb der *„Buchhandlung“* nicht existieren. Sobald die *„Buchhandlung“* nicht mehr existiert, da das Gebäude beispielsweise zerstört wird, werden auch automatisch darin enthaltene *„Räume“* mit zerstört.

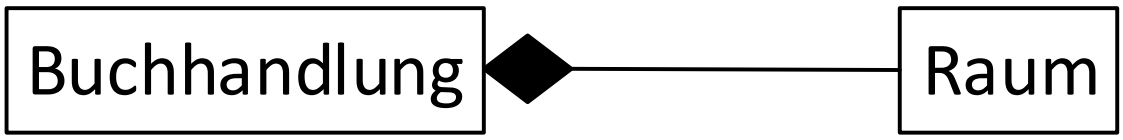


Abbildung 5: Komposition der Klassen „Buchhandlung“ und „Raum“

Bei der Aggregation ist die Bindung der Teile zu ihrem Ganzen, dem sogenannten Aggregat, nicht so streng. Sie sind nicht existenzabhängig. Wird das Aggregat gelöscht, können die Teile weiterhin existieren. Außerdem können Teile in mehreren Aggregaten involviert sein. Ein Beispiel ist in Abbildung 6 gegeben. Die *„Buchhandlung“* ist das Aggregat (das Ganze) und die Klasse *„Buch“* ist ein Teil davon. Wird die *„Buchhandlung“* geschlossen oder zerstört, können die *„Bücher“,* im Gegensatz zu den Räumen, immer noch weiter existieren und anderweitig verwendet werden.

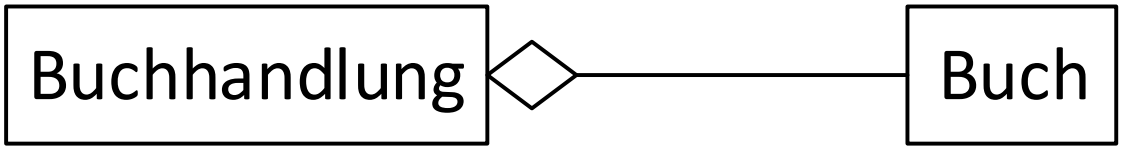


Abbildung 6: Aggregation zwischen den Klassen „Buchhandlung“ und „Buch“

4.1.4 Zusätzliche Elemente für Assoziationen: Rollen und Beschriftungen

Assoziationsenden können mit Rollen beschriftet werden, wie in Abbildung 7 zu sehen ist. Eine Rolle beschreibt dabei die Art, in der ein Objekt in der Assoziation beteiligt ist, d.h. welche Rolle das Objekt in der Verbindung einnimmt. Rollen können zusätzlich Sichtbarkeiten haben, wie bei der Beschreibung der Klasse in 4.1.1 bereits erwähnt. Das Beispiel in Abbildung 7 besteht aus den beiden Klassen „*Verkäufer*“ und „*Kunde*“. Sie sind durch eine unbestimmte Assoziation miteinander verbunden. Bei der Klasse „*Verkäufer*“ ist oberhalb der Assoziation „*+Bedienung*“, angegeben d.h. die Rolle „*Bedienung*“ hat die Sichtbarkeit „public“. Bei der Klasse „*Kunde*“ ist keine Rolle zugeordnet. Der „*Verkäufer*“ nimmt damit die Rolle der „*Bedienung*“ in einer Buchhandlung an, indem er „*Kunden*“ bei ihrem Einkauf behilflich ist.

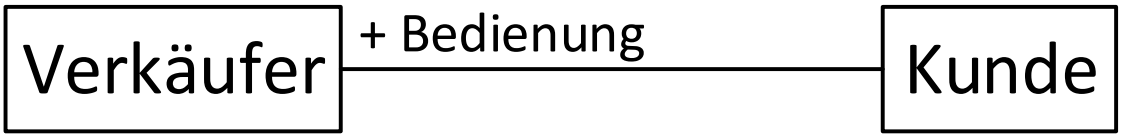


Abbildung 7: Rollenbezeichnung, die einen „Verkäufer“ in der Beziehung zu einem „Kunden“ als „Bedienung“ identifiziert

Zusätzlich können Assoziationen Beschriftungen haben, welche direkt auf oder unterhalb der Assoziation platziert werden. Dadurch kann die Bedeutung der Assoziation veranschaulicht werden. Optional kann zusätzlich die Leserichtung für eine solche Beschriftung angegeben werden. Diese wird, wie in Abbildung 8, durch ein gefülltes Dreieck neben der Beschriftung zwischen den beteiligten Klassen dargestellt. Die Leserichtung ist dabei in Richtung der Spitze des Dreiecks. Im Beispiel ist die Leserichtung der Assoziation von der Klasse „*Verkäufer*“ zur Klasse „*Kunde*“. Der Name der Assoziation lautet „*berät*“. Ein „*Verkäufer*“ „*berät*“ demnach einen „*Kunden*“ beim Einkauf.

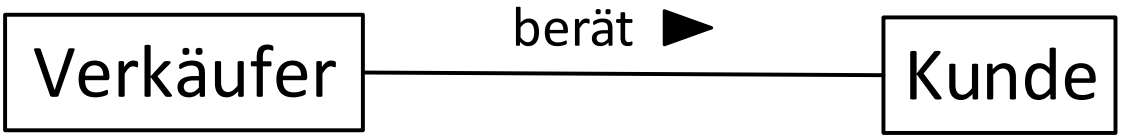


Abbildung 8: Beschriftung einer Assoziation mit „berät“ und Leserichtung von „Verkäufer“ zu „Kunde“

4.1.5 Multiplizitäten

Multiplizitäten legen die mögliche Anzahl der an einer Assoziation beteiligten Exemplare einer Klasse fest (Lahres & Rayman, 2009). Diese Angabe wird in der Nähe des Endes der Assoziation der betroffenen Klasse dargestellt. Multiplizitäten bestehen aus einer Unter- und einer Obergrenze, welche durch zwei Punkte getrennt dargestellt werden, z.B. „3..8“. Sind die Unter- und Obergrenze identisch, kann stattdessen nur eine Zahl verwendet werden, d.h. „1..1“ kann als „1“ dargestellt werden.

Verschiedene Varianten von Multiplizitäten werden anhand eines Beispiels erläutert. Das Beispiel ist in Abbildung 9 gegeben. Die beteiligten Klassen sind die Klasse *„Regal“* und *„Buch“*. Es besteht eine ungerichtete Assoziation zwischen den beiden Klassen.

- Wenn an beiden Klassen je eine *„1“* steht, darf pro *„Regal“* nur 1 *„Buch“* enthalten sein. Konkret würde das bedeuten, dass jedes *„Buch“* in einem eigenen *„Regal“* steht. Dies wird auch 1:1-Beziehung genannt. Wird keine Angabe vorgenommen, ist eine Assoziation immer eine 1:1-Beziehung.
- In Abbildung 9 steht bei der Klasse *„Regal“* eine *„1“* und bei der Klasse *„Buch“ „0..20“*. In diesem Fall sind pro *„Regal“* null bis zwanzig *„Bücher“* erlaubt. Andererseits ist ein *„Buch“* jedoch immer nur in genau einem *„Regal“* enthalten. Ein Element des *„Regals“* steht demnach mit mehreren Elementen der Klasse *„Buch“* in Verbindung. Es besteht eine 1:n-Beziehung.

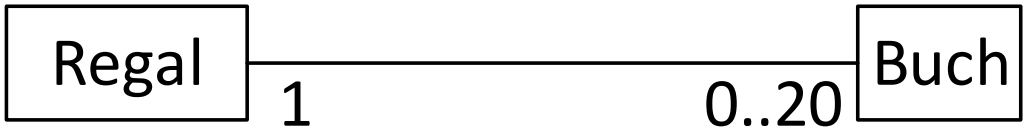


Abbildung 9: 1:n-Beziehung zwischen der Klasse „Regal“ und der Klasse „Buch“

- Würde bei der Klasse *„Regal“* auch eine Mengenangabe wie beispielsweise *„1..5“* stehen, bedeutet dies, dass ein *„Buch“* zugleich in einem bis fünf *„Regalen“* stehen könnte. Es würde eine n:m-Beziehung bestehen. Für das aktuelle Szenario gibt es jedoch keine sinnvolle Abbildung dieser Beziehung in der Realität.
- Durch einen Stern (*) kann ausgedrückt werden, dass keine Obergrenze festgelegt ist und beliebig viele Elemente beteiligt sein können. Wäre an der Klasse *„Buch“* entsprechend *„0..*“* angegeben, würden beliebig viele *„Bücher“* in ein *„Regal“* passen. Die Angabe *„0..*“* kann vereinfacht nur durch einen Stern ausgedrückt werden.
- Eine weitere, häufig vorkommende, Kombination ist *„0..1“*. Dies bedeutet, dass ein Element mit einem anderen verbunden sein kann, aber nicht muss. Wäre dies an der Klasse *„Buch“* beschrieben, würde das bedeuten, dass in einem *„Regal“* entweder kein oder genau 1 *„Buch“* stehen würde.

4.1.6 Vererbung

Bei der Vererbung oder Generalisierung erbt (übernimmt) eine Klasse Attribute und Methoden einer anderen Klasse, falls diese nicht explizit als privat deklariert und damit nicht vererbbar sind. Die Klasse, welche erbt, wird dabei als Unterklasse bezeichnet, diejenige von welcher geerbt wird als Oberklasse. Die visuelle Darstellung ist in Abbildung 10 gegeben. Eine Vererbung wird dabei als Linie zwischen den beteiligten Klassen repräsentiert mit einem nicht gefüllten Dreieck als Pfeilende am Ende der Oberklasse. Es ist sowohl möglich, dass eine Unterklasse von mehreren Oberklassen erbt, als auch, dass mehrere Unterklassen von der gleichen Oberklasse erben. Die Unterklasse kann zusätzliche Attribute oder Methoden enthalten, welche unabhängig von der Oberklasse sind. Im Beispiel ist die Oberklasse die Klasse *„Mitarbeiter“*. Von dieser Klasse erben die beiden Klassen *„Verkäufer“* und *„Abteilungsleiter“*. Offensichtlich sind diese beiden jeweils *„Mitarbeiter“* in einer Buchhandlung und haben in dieser Position ähnliche Eigenschaften. Die konkreten Ausprägungen sind jedoch verschieden, da ein *„Abteilungsleiter“* noch weitere Aufgabenbereiche hat als ein normaler *„Verkäufer“*.

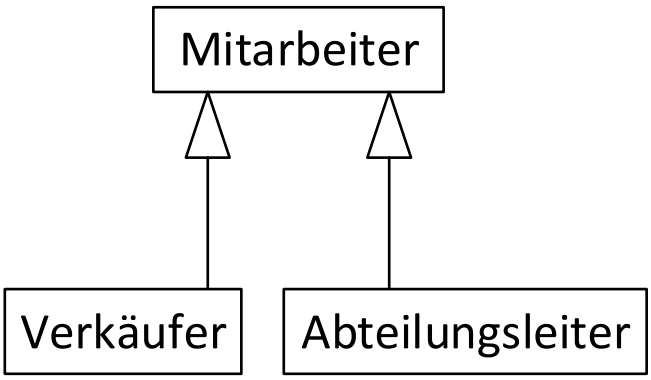


Abbildung 10: Vererbung zwischen der Klasse „Mitarbeiter“ und den Klassen „Verkäufer“ und „Abteilungsleiter“

4.1.7 Beispiel eines Klassendiagramms für eine Navigations-App

Das Beispiel der Fußgängernavigation in Abbildung 11 stellt die benötigten Komponenten zur Berechnung einer Route durch ein Navigationsgerät dar. Es enthält drei Klassen: *„NaviGerät“*, *„Route“* und *„Routenabschnitt“*. Die Klassen sind linear verbunden. Das Navigationsgerät (Klasse *„NaviGerät“*) ist mithilfe einer n:n Assoziation mit der Klasse *„Route“* verbunden, d.h. an beiden Enden ist jeweils ein Stern. Die Assoziation ist mit *„berechnet“* beschriftet. Die Klasse *„Route“* enthält zwei Attribute: Die *„Länge“* der Route und die geschätzte *„Dauer“* für diese. Beide Attribute haben den Typ *„Double“*. Die *„Route“* ist wiederum mittels einer Komposition mit der Klasse *„Routenabschnitt“* verbunden. Eine *„Route“* kann beliebig viele *„Routenabschnitte“* enthalten (Multiplizität „*“ am *„Routenabschnitt“*). Existiert die *„Route“* nicht mehr, dann hören auch *„Routenabschnitte“* auf zu existieren. Die Klasse *„Routenabschnitt“* enthält drei Attribute: *„Straßenname:String“*, *„Anfangspunkt:Position“* und *„Endpunkt:Position“*.

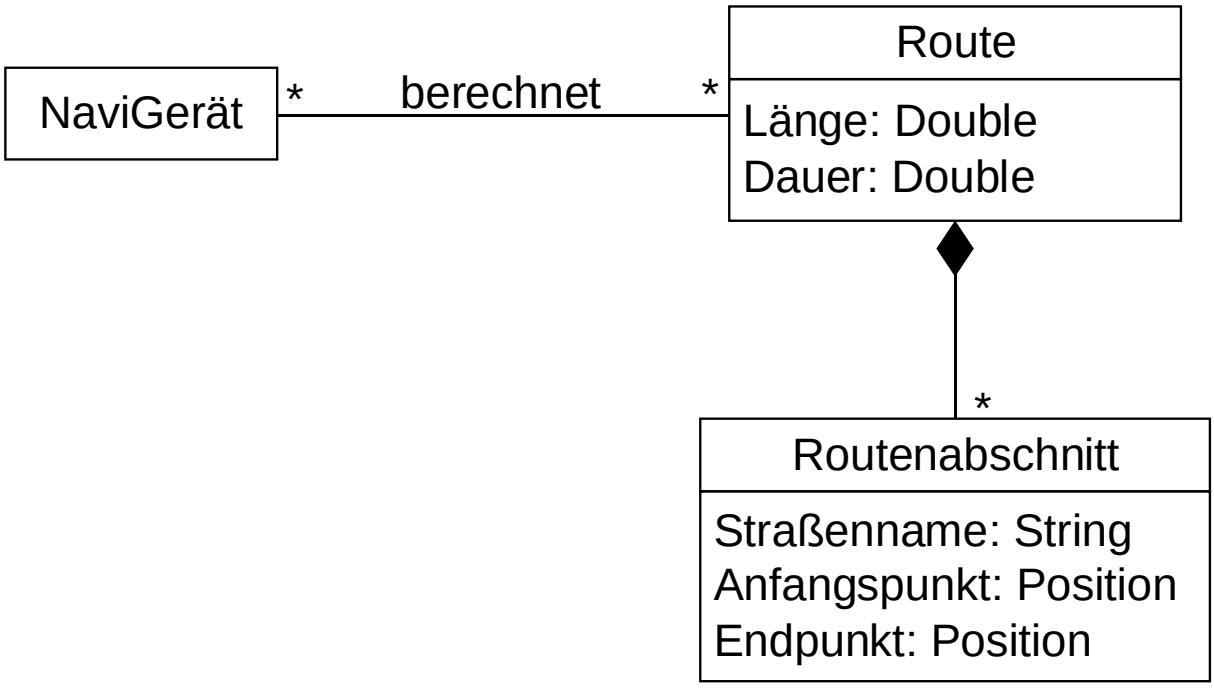


Abbildung 11: Klassendiagramm des Navigationsbeispiels

4.1.8 Verständnisfragen zum Klassendiagramm

Dieses Kapitel besteht aus fünf Multiple-Choice-Fragen (Business Informatics Group, 2015), um das Verständnis für das Klassendiagramm zu überprüfen. Jede Frage hat vier Antwortmöglichkeiten a – d. Bei jeder Frage können eine, zwei, drei oder vier Antworten richtig sein. Die Auflösung ist in Anhang A und im Detail in Anhang A.1 zu finden.

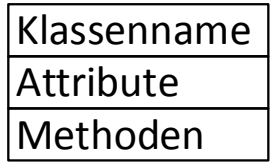
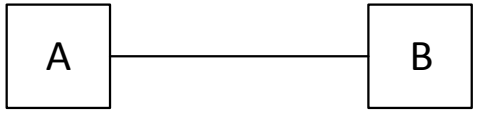
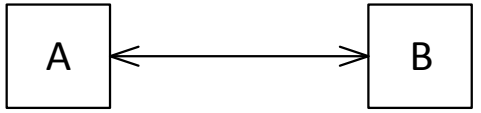
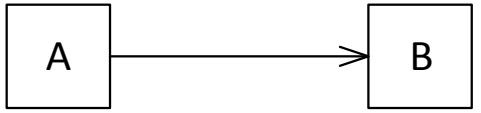
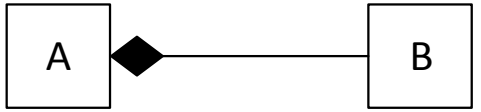
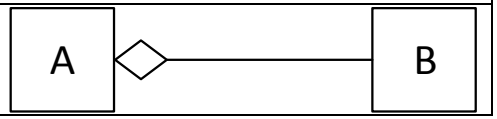
1. Ein Klassendiagramm beschreibt…
   (a) den Interaktions-Aspekt eines Systems.
   (b) den strukturellen Aspekt eines Systems.
   (c) den praktischen Aspekt eines Systems.
   (d) den dynamischen Aspekt eines Systems.
2. Assoziationen…
   (a) können Multiplizitäten an ihren Enden haben, wodurch angegeben wird, mit wie vielen Objekten auf der gegenüberliegenden Seite ein Objekt in Beziehung stehen kann.
   (b) modellieren mögliche Beziehungen zwischen Instanzen von Klassen.
   (c) müssen durch einen Assoziationsnamen identifizierbar sein.
   (d) können eine Navigationsrichtung haben, welche angibt, in welche Richtung die Navigation von einem Objekt zu seinem Partnerobjekt erfolgen kann.
3. Bei der Generalisierungsbeziehung zwischen einer spezialisierten Klasse (Unterklasse) und einer allgemeineren Klasse (Oberklasse) gelten folgende Eigenschaften:
   (a) Die spezialisierte Klasse erbt die Eigenschaften der allgemeineren.
   (b) Die spezialisierte Klasse darf ausschließlich nur von einer einzigen allgemeineren Klasse erben.
   (c) Die allgemeinere Klasse darf ausschließlich mit einer Unterklasse eine Vererbungsbeziehung eingehen.
   (d) Es dürfen keine weiteren Attribute zur spezialisierten Klasse hinzugefügt werden.
4. Eine Aggregation...
   (a) wird durch eine gefüllte Raute an einem Assoziationsende dargestellt.
   (b) drückt eine ist-ein-Beziehung aus.
   (c) drückt eine Teil-von-Beziehung aus.
   (d) wird durch eine ungefüllte Raute an einem Assoziationsende dargestellt.
5. Welche der folgenden Aussagen sind korrekt?
   (a) Bei einer Klasse im Klassendiagramm müssen immer Attribute und Methoden angegeben werden.
   (b) Klassen haben immer einen Abschnitt, der eine textuelle Beschreibung von ihnen beinhaltet.
   (c) Attribute können durch einen Typ näher beschrieben werden.
   (d) Bei Operationen können Übergabeparameter und Rückgabewert angegeben werden.

4.1.9 Modellierungsaufgabe Klassendiagramm

Ein Arzt oder eine Privatperson, welche je durch ihren Namen identifiziert werden können, möchten eine Arznei kaufen. Die Arznei hat einen Namen und soll eine Krankheit heilen. Jede Person darf maximal 50 Arzneimittel auf einmal kaufen. Jede Arznei kann mindestens eine Krankheit heilen, es gibt jedoch auch unheilbare Krankheiten. Jede Krankheit zeichnet sich durch einen Namen und eine Reihe von Symptomen aus.
Eine Beispiellösung findet sich in Anhang B und als taktile Grafik auf Seite 61 der taktilen Mappe.

4.1.10 Tabellarische Übersicht der Elemente des Klassendiagramms

Die folgende Tabelle fasst die bisherigen Informationen zusammen. Dabei werden die einzelnen Elemente abstrahiert und nicht mit konkreten Beispielen veranschaulicht. In Spalte 1 steht der Name des Elements, in Spalte 2 eine textuelle Beschreibung und in Spalte 3 die Beschreibung der grafischen Darstellung in Spalte 4. Alle Tabellen der nachfolgenden Diagrammtypen sind analog aufgebaut.

Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Klasse	Beschreibt Strukturen und Verhalten einer Menge von Objekten.	Ein Rechteck mit drei Abschnitten (2 optionalen), durch horziontale Linien getrennt. Im ersten steht der Klassenname, im zweiten die Attribute und im dritten die Methoden.	
Assoziation	Verbindungen zwischen Klassen modellieren mögliche Kommunikationspartner.	Eine einfache Linie zwischen Rechtecken, z.B. zwische Rechteck A und B.	
Bidirektionale Assoziation	Beide Klassen können aufeinander zugreifen.	Eine einfache Linie mit offener Pfeilspitze an beiden Seiten zwischen Rechtecken, z.B. zwischen Rechteck A und B.	
Gerichtete/direktionale Assoziation	Nur eine Klasse hat Zugriff auf eine andere, nicht umgekehrt.	Eine einfache Linie mit offener Pfeilspitze an einem Ende. Pfeilspitze zeigt von einem Rechteck A zum Rechteck B.	
Komposition	Teil-Ganzes-Verbindung: Eine Klasse stellt Teile einer anderen dar. Wird die „Ganze“-Klasse gelöscht, dann werden auch die verbundenen Instanzen gelöscht.	Eine einfache Linie mit schwarzer Raute am „Ganzen“-Ende. 2 Rechtecke, A und B, sind durch eine Linie verbunden. An der Seite von A ist eine schwarze Raute.	
Aggregation	Teil-Ganzes-Verbindung: Eine Klasse stellt Teile einer anderen	Eine einfache Linie mit weißer Raute am „Ganzen“- Ende.	

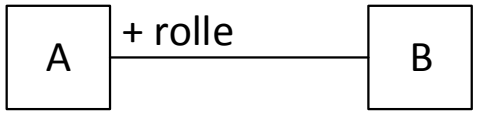
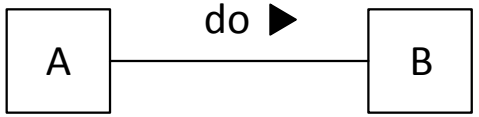
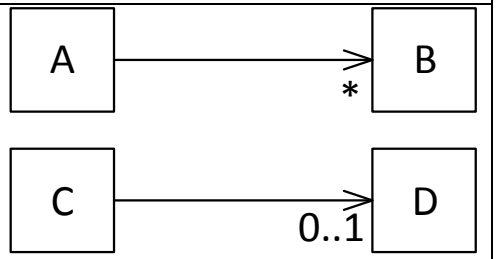
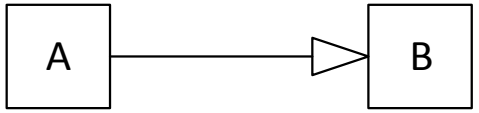
Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
	dar. Wird die „Ganze“-Klasse gelöscht, können die Teile jedoch weiterhin existieren.	2 Rechtecke, A und B, durch eine Linie verbunden. An der Seite von A ist eine ungefüllte Raute.	
Rollen	Beschreibt die Art, in der ein Objekt innerhalb einer Assoziation involviert ist. Rollen können optional Sichtbarkeiten haben.	Text, welcher ober- /unterhalb von Assoziationen, nahe am Objekt, welchem eine Rolle zugewiesen werden soll, stehen. 2 Rechtecke, A und B, durch eine einfache Linie verbunden. Oberhalb der Linie bei A steht ein *+* und das Wort *„rolle“*.	
Beschriftung von Assoziationen	Zusätzliche detailliertere Beschreibung von Assoziationen. Optional mit Angabe der Leserichtung als schwarzes Dreieck, dessen Spitze die Leserichtung angibt.	Texte ober-/unterhalb einer Assoziation, oft mittig zwischen Klassen angebracht mit optionalem schwarzen Dreieck als Leserichtung. 2 Rechtecke, A und B, sind durch eine einfache Linie verbunden. Über der Linie steht das Wort *„do“* und ein schwarzes Dreieck rechts daneben zeigt mit der Spitze zum Rechteck B.	
Multiplizität	Legt fest wie viele Elemente/Objekte in einen Ablauf/Prozess involviert sein müssen (Untergrenze) oder können (Obergrenze).	4 Rechtecke: A,B,C und D. A und B sind durch eine Linie mit Pfeil bei B verbunden. Bei B steht unterhalb der Linie ein Stern. C und D sind ebenso verbunden, es steht jedoch bei D 0..1.	
Vererbung	Vererbungsverbindung: Unterklasse erbt Attribute und Methoden der Oberklasse.	2 Rechtecke, A und B, verbunden durch eine einfache Linie. Am Ende von B ist ein ungefülltes Dreieck mit Spitze in Richtung B.	

Tabelle 1: Kurzreferenz der Elemente des Klassendiagramms

5 Verhaltensdiagramme

Verhaltensdiagramme beschreiben das dynamische Verhalten eines Systems. Im Laufe des Kapitels werden vier Diagrammtypen vorgestellt: das Anwendungsfalldiagramm, das Aktivitätsdiagramm, das Zustandsdiagram und zuletzt das Sequenzdiagramm.

5.1 Anwendungsfalldiagramm

Die taktilen Grafiken des Anwendungsfalldiagramms (im englischen Use Case Diagram) befinden sich auf den Seiten 8 – 16 der taktilen Mappe.

Das Anwendungsfalldiagramm stellt die Anforderungen an ein System dar. Es werden Kernfunktionen der Software aus Nutzersicht herausgearbeitet und zueinander in Beziehung gesetzt. Das Anwendungsfalldiagramm beantwortet damit die folgenden Fragen:

- Was wird beschrieben? (Das System)
- Wer interagiert mit dem System? (Die Akteure)
- Was kann der Akteur machen? (Die Anwendungsfälle)
  (Seidl, Scholz, Huemer, & Kappel, 2015)

5.1.1 Die Hauptkomponenten: Akteur, Anwendungsfall und das System

Hauptkomponenten des Anwendungsfalldiagramms sind der Akteur, das System und die Anforderungen, welche durch sogenannten Anwendungsfälle repräsentiert werden. Akteure sind Personen oder Maschinensysteme, welche Rollen definieren wie ein Benutzer mit einem System interagiert. Das System ist die zu entwickelnde Software mit den enthaltenen Anwendungsfällen. Die Anwendungsfälle werden häufig als Ellipsen mit dem Funktionsnamen darin dargestellt, wie in Abbildung 12 zu sehen ist. Alternativ können Anwendungsfälle auch als Rechteck mit einer kleinen Ellipse in einer oberen Ecke und dem Namen dargestellt werden. Diese Notation wird jedoch kaum verwendet, weshalb hier die Ellipsenform verwendet wird.

Abbildung 12: Ein Anwendungsfall „Buch kaufen“

Akteure können sowohl menschliche Personen darstellen als auch Maschinen, wie z.B. ein Webserver. Menschliche Akteure werden als Strichmännchen dargestellt. Die Bezeichnung steht unmittelbar darunter. Maschinelle Akteure könnten ebenfalls als Strichmännchen symbolisiert werden, werden jedoch häufiger als Rechteck mit dem Stereotyp <<Akteur>> und dessen Namen dargestellt. Stereotype können unabhängig vom Typ für jeden Akteur angegeben werden, d.h. auch bei einem Strichmännchen kann als Stereotyp <<Akteur>> oder <<Human>> verwendet werden, auch wenn die grafische Darstellung dies bereits eindeutig zeigt. Durch diese Angabe kann jedoch die Unterscheidung verschiedener Typen von Akteuren erleichtert werden, insbesondere falls gleiche Symbole verwendet werden. Eine detaillierte Darstellung beider Varianten von Akteuren ist in Abbildung 13 gegeben. Links ist ein menschlicher Akteur als Strichmännchen mit der Bezeichnung „*Kunde*“ gegeben und rechts ein maschineller Akteur als Rechteck mit der Stereotypangabe *<<Akteur>>* und dem Namen „*Kasse*“.

Abbildung 13: Strichmännchen- (links) und Rechtecknotation (rechts) zur Darstellung von Akteuren

5.1.2 Einfache Assoziationen

Eine Darstellung mit allen drei Hauptelementen ist in Abbildung 14 gegeben. Das System, als Rechteck mit dem Namen *„Buchhandlung“* oben links dargestellt, enthält den Anwendungsfall *„Buch kaufen“*. Dieser Anwendungsfall ist mit einem Akteur *„Kunde“* verbunden. Hierbei ist es wichtig zu beachten, dass Akteure immer außerhalb des Systems sind und daher daneben dargestellt werden. Die durchgezogene Verbindungslinie zwischen dem Akteur und dem Anwendungsfall stellt eine Assoziation dar. Sie bedeutet, dass der Akteur den Anwendungsfall in einer beliebigen Form verwendet. Der Akteur tauscht demnach Informationen mit der durch den Anwendungsfall realisierten Systemfunktionalität aus. Beispielsweise kann er eine Funktion des Systems starten oder ihm von einer Funktion des Systems Daten übergeben werden.

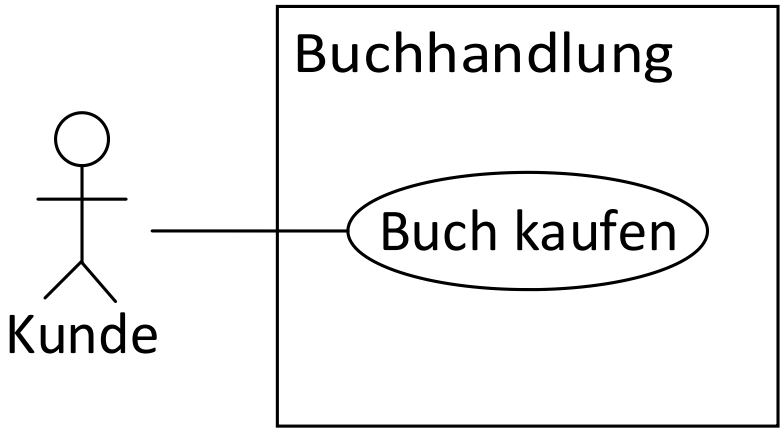


Abbildung 14: Darstellung eines Gesamtsystems mit einem Akteur, einem System und einem Anwendungsfall

Ist ein Anwendungsfall mit zwei Akteuren verbunden, kann dieser nur ausgeführt werden, wenn beide Akteure involviert sind. Ebenso kann am Akteurende der Assoziation eine Multiplizität stehen, sodass mehrere gleiche Akteure bei der Ausführung des Anwendungsfalls involviert sein müssen. Das Beispiel in Abbildung 15 zeigt das System *„Buchhandlung“* mit den Anwendungsfall *„Buch kaufen“* und den beiden Akteuren *„Verkäufer“* und *„Kunde“*. Beide Akteure sind jeweils mit dem Anwendungsfall verbunden. Die Multiplizität „2“ ist bei der Assoziation mit dem Akteure *„Verkäufer“* am Assoziationsende des *„Verkäufers“*. In diesem Fall müssen immer zwei *„Verkäufer“* an dem Anwendungsfall *„Buch kaufen“* beteiligt sein. Ebenfalls muss zusätzlich ein *„Kunde“* beteiligt sein, da dieser ebenfalls mit dem Anwendungsfall verbunden ist. Insgesamt müssen demnach drei Akteure bei der Ausführung der Funktion *„Buch kaufen“* involviert sein.

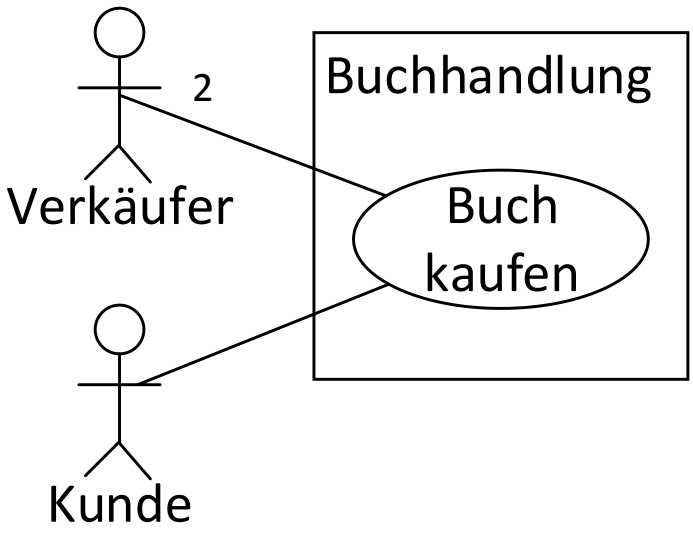


Abbildung 15: Verbindung eines Anwendungsfalls mit zwei verschiedenen Akteuren

5.1.3 Die speziellen Assoziationen include und extend

Weitere Assoziationen können zwischen verschiedenen Anwendungsfällen gegeben sein. Diese Assoziationen werden als gestrichelte Linie mit einer offenen Pfeilspitze dargestellt. Es existieren zwei verschiedene Arten von Assoziationen zwischen Anwendungsfällen, weshalb neben oder auf der Linie ein Schlüsselwort zur Unterscheidung angegeben wird. Schlüsselwörter in der UML, die zur Spezifizierung von Verbindungslinien oder anderen geometrischen Formen verwendet werden, werden immer zwischen doppelte spitze Klammern (Guillements) gestellt.

Eine include-Assoziation bedeutet, dass ein Anwendungsfall das Verhalten eines anderen Anwendungsfalls integriert. In Abbildung 16 wird der Anwendungsfall *„Bezahlen“* von dem Anwendungsfall *„Buch kaufen“* eingebunden. *„Buch kaufen“* wird dabei als Basisanwendungsfall bezeichnet und *„Bezahlen“* als eingebundener Anwendungsfall. Die Basis benötigt immer den eingebundenen Anwendungsfall und dessen Funktionalität, um vollständig ausgeführt werden zu können. Die Pfeilrichtung der Assoziation zeigt daher in Richtung des eingebundenen Anwendungsfalles, in diesem Beispiel also von *„Buch kaufen“* zu *„Bezahlen“* mit dem Schlüsselwort <<include>> oberhalb der Assoziation. Möchte jemand ein *„Buch kaufen“*, so muss er in diesem Zuge immer auch den Anwendungsfall *„Bezahlen“* ausführen. Der eingebundene Anwendungsfall kann dagegen auch eigenständig ausgeführt werden (Seidl, Scholz, Huemer, & Kappel, 2015).

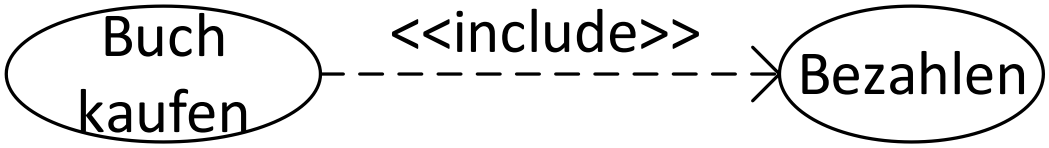


Abbildung 16: Include-Assoziation vom Anwendungsfall „Buch kaufen“ zu „Bezahlen“

Eine extend-Assoziation bedeutet, dass ein Anwendungsfall einen anderen Anwendungsfall erweitern kann, wie in Abbildung 17 dargestellt. Der Anwendungsfall *„Buch kaufen“* wird als Basisanwendungsfall bezeichnet und der Anwendungsfall *„Als Geschenk verpacken“* als der erweiternde Anwendungsfall. *„Buch kaufen“* kann das Verhalten von *„Als Geschenk verpacken“* durch Aktivierung bei Bedarf nutzen. Beide Anwendungsfälle können jedoch auch unabhängig voneinander ausgeführt werden. Ob ein Anwendungsfall einen anderen erweitert wird durch einen Erweiterungspunkt (englisch: *extension point)* bestimmt. Falls spezifizierte

Bedingungen an extend-Assoziationen als wahr ausgewertet werden, wird der Anwendungsfall erweitert. Ein Erweiterungspunkt wird durch einen ungefüllten Kreis auf der Assoziation dargestellt. Bedingungen und Namen des Erweiterungspunktes werden innerhalb einer Notiz (Rechteck mit einem „Eselsohr“) dargestellt und mittels einer gestrichelten Linie an den Erweiterungspunkt (Kreis) angehängt. Der Erweiterungspunkt wird ebenfalls innerhalb des Basisanwendungsfalls notiert, indem der Name des Anwendungsfalls durch eine horizontale Linie vom Schlüsselwort *„extension point:“* gefolgt vom Namen des Erweiterungspunkts getrennt wird. Während eines *„Buchkaufes“* kann dieses demnach optional *„als Geschenk verpackt“* werden, falls die Bedingung *„condition:{Adressat! = Käufer} extension point: Verpackung“* erfüllt ist. Beide Anwendungsfälle können auch unabhängig voneinander ausgeführt werden. In diesem Fall zeigt die Assoziationsspitze vom erweiternden Anwendungsfall *„Als Geschenk verpacken“* zum Basisanwendungsfall *„Buch kaufen|extension point: Verpackung“* mit dem Schlüsselwort *<<extend>>* oberhalb der Assoziation.

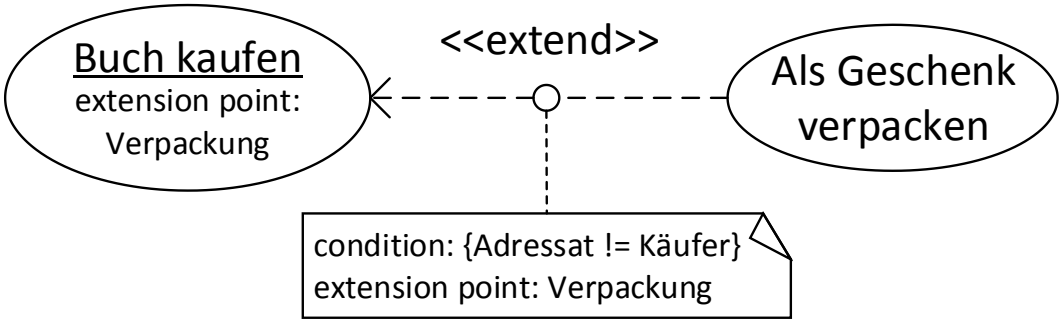


Abbildung 17: Extend-Assoziation vom Anwendungsfall „Als Geschenk verpacken“ zu „Buch kaufen“

5.1.4 Vererbung zwischen Anwendungsfällen und Akteuren

Wie bereits im Klassendiagramm vorgestellt, gibt es auch im Anwendungsfalldiagramm das Konzept der Vererbung oder Generalisierung. Hierbei werden jedoch zwei Fälle unterschieden: Die Vererbung zwischen Anwendungsfällen und die Vererbung bei Akteuren. Das Symbol für die Vererbung ist wie im Klassendiagramm: ein nicht gefülltes Dreieck. Erbt ein Anwendungsfall von einem anderen, übernimmt er dessen Verhalten und Assoziationen. Ein Beispiel ist in Abbildung 18 gegeben, in welchem der Anwendungsfall *„Mit Kreditkarte zahlen“* vom Anwendungsfall *„Bezahlen“* erbt. Die *„Bezahlung mit Kreditkarte“* ist eine Art des *„Bezahlens“,* erfordert jedoch zusätzliche Schritte wie z.B. einen Pin eingeben, welcher beim normalen Bezahlen mit Bargeld entfällt.

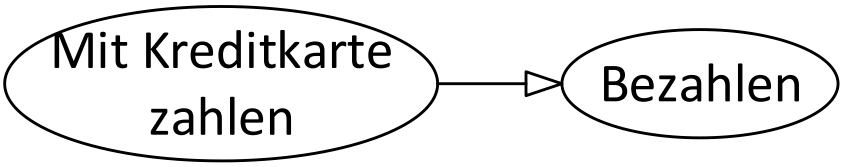


Abbildung 18: Der Anwendungsfall „Mit Kreditkarte zahlen“ erbt vom Anwendungsfall „Bezahlen“

Abbildung 19 zeigt dagegen die Vererbung bei Akteuren. Der Akteur *„Abteilungsleiter“* erbt dabei vom Akteur *„Mitarbeiter“.* Damit übernimmt der *„Abteilungsleiter“* die Verbindungen zu Anwendungsfällen vom *„Mitarbeiter“*, d.h. er ist in allen Anwendungsfällen involviert, in denen auch der *„Mitarbeiter“* involviert ist, kann darüber hinaus jedoch noch mit anderen Anwendungsfällen in Verbindung stehen.

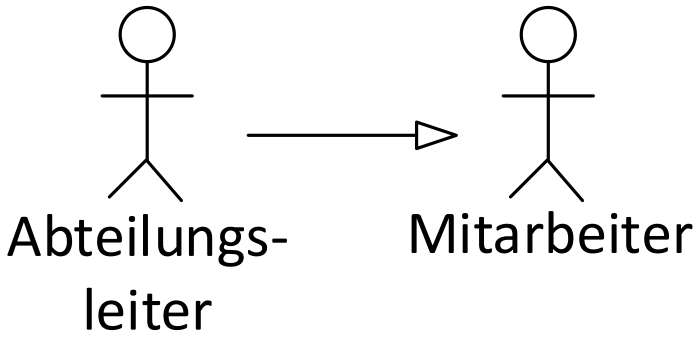


Abbildung 19: Der Akteur „Abteilungsleiter“ erbt vom Akteur „Mitarbeiter“

Erben zwei Akteure von einem Akteur, welcher mit einem Anwendungsfall verbunden ist, müssen nicht beide Akteure bei der Ausführung des Anwendungsfalls beteiligt sein, sondern es genügt ein Akteur. In Abbildung 20 ist ein System *„Buchhandlung“* mit dem Anwendungsfall *„Buch kaufen“* dargestellt. Die Akteure *„Abteilungsleiter“* und *„Verkäufer“* erben von einem dritten Akteur *„Mitarbeiter“*, welcher mittels einer Assoziation mit dem Anwendungsfall *„Buch kaufen“* verbunden ist. Im Gegensatz zu dem Szenario aus Abbildung 15 muss hier entweder ein *„Abteilungsleiter“* oder ein *„Verkäufer“* am Anwendungsfall *„Buch kaufen“* beteiligt sein, nicht beide, da der Anwendungsfall hier nur *eine* Assoziation besitzt.

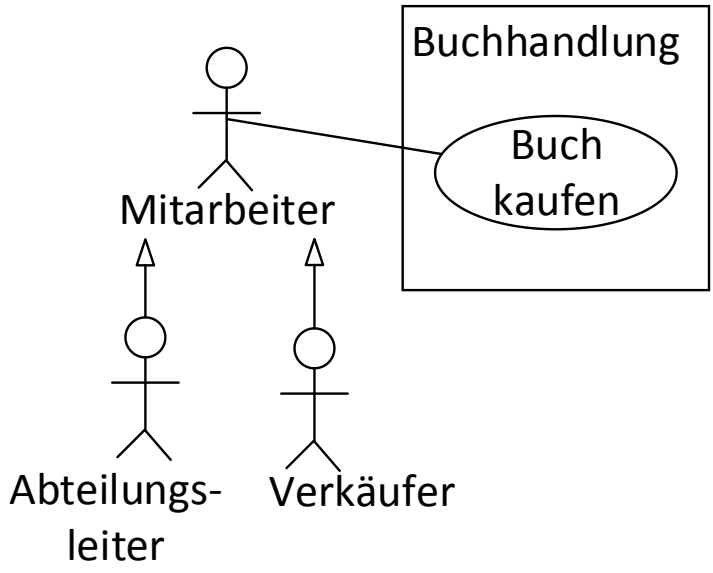


Abbildung 20: Verhalten bei der Ausführung eines Anwendungsfalls bei Vererbung von Akteuren

5.1.5 Beispiel eines Anwendungsfalldiagramms für eine Navigations-App

Das Beispiel in Abbildung 21 zeigt die Benutzung des Navigationsgeräts für das Navigieren zu einem Zielort. Es besteht insgesamt aus zwei Akteuren und drei Anwendungsfällen. Der erste Akteur ist ein menschlicher Akteur (Strichmännchen) mit dem Bezeichner *„Person“*. Der zweite Akteur ist ein maschineller Akteur mit der Bezeichnung *„GPS Satellitensystem“*. Dieser Akteur wird als Rechteck mit dem Stereotyp *<<Akteur>>* dargestellt. Das System *„Navigationsapp“* beinhaltet drei Anwendungsfälle. Der Anwendungsfall *„Zum Ziel gehen“* bindet die Anwendungsfälle *„Zielort eingeben“* und *„Aktuelle Position ermitteln“* ein, d.h. sie sind über eine include-Assoziation verbunden. Der Akteur *„Person“* kann mit den Anwendungsfällen *„Zielort eingeben“* und *„Zum Ziel gehen“* interagieren, d.h. es existiert eine Assoziation. Das

„GPS Satellitensystem“ ist mit dem Anwendungsfall *„Aktuelle Position ermitteln“* verbunden, da diese über Satelliten ermittelt werden muss.

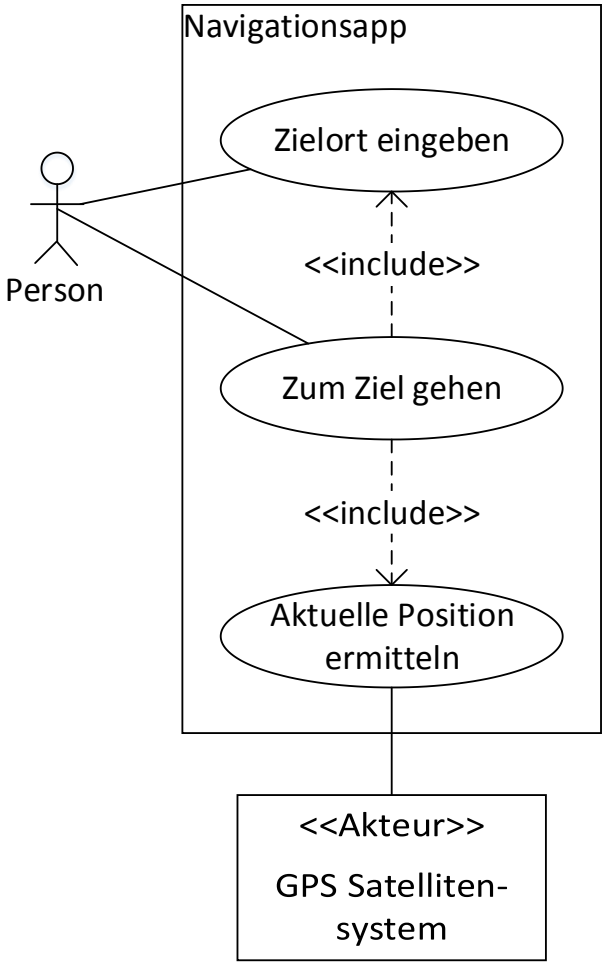


Abbildung 21: Anwendungsfalldiagramm des Navigationsbeispiels

5.1.6 Verständnisfragen zum Anwendungsfalldiagramm

Dieses Kapitel besteht aus fünf Multiple-Choice-Fragen (Business Informatics Group, 2015), um das Verständnis für das Anwendungsfalldiagramm zu überprüfen. Jede Frage hat vier Antwortmöglichkeiten a–d. Bei jeder Frage können eine, zwei, drei oder vier Antworten richtig sein. Die Auflösung ist in Anhang A und im Detail in Anhang A.2 zu finden.

1. Welche Aussagen treffen zu, wenn ein Akteur B von einem Akteur A erbt?
   (a) A kann mit denselben Anwendungsfällen wie B kommunizieren.
   (b) B kann mit denselben Anwendungsfällen wie A kommunizieren.
   (c) B erbt alle Assoziationen von A.
   (d) A erbt alle Assoziationen von B.
2. Akteure in einem Anwendungsfalldiagramm...
   (a) können das beschriebene System benutzen.
   (b) befinden sich immer innerhalb des beschriebenen Systems.
   (c) interagieren mit Hilfe von <<include>>-Beziehungen mit dem System.
   (d) stellen Rollen der Benutzer des beschriebenen Systems dar.
3. Die Assoziation zwischen einem Akteur und einem Anwendungsfall...
   (a) wird durch eine gestrichelte Linie modelliert.

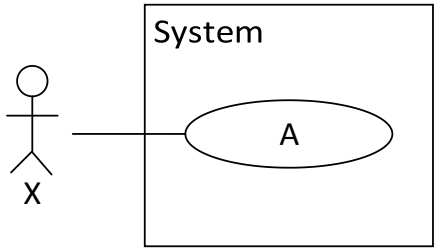
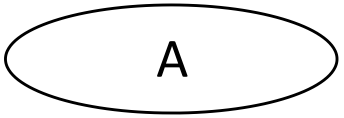
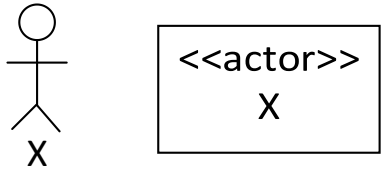
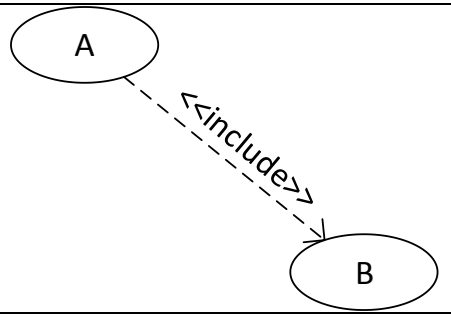
   (b) wird durch eine Kante mit Pfeilspitze auf der Seite des Anwendungsfalls modelliert.
   (c) kann Multiplizitäten aufweisen.
   (d) ist binär.

4. Welche der folgenden Aussagen über Anwendungsfälle sind korrekt?
   (a) Anwendungsfälle dürfen untereinander in keiner Generalisierungsbeziehung stehen
   (b) Anwendungsfälle müssen nicht zwingend benannt werden
   (c) Anwendungsfälle dürfen binäre, aber keine n-ären, Assoziationen besitzen
   (d) Ein Anwendungsfall wird als Rechteck mit abgerundeten Ecken dargestellt.
5. Das Anwendungsfalldiagramm...
   (a) beinhaltet Akteure und Anwendungsfälle.
   (b) beschreibt zeitliche Abläufe innerhalb eines Systems.
   (c) beschreibt, wer was mit dem zu entwickelnden System macht.
   (d) beschreibt, wie Funktionen im System angeboten werden.

5.1.7 Modellierungsaufgabe Anwendungsfalldiagramm

Ein Kunde möchte in einer Apotheke etwas kaufen. Dabei kann es sich um einen Arzt oder eine Privatperson handeln. Jeder Verkauf an einen Kunden wird durch einen Apotheker durchgeführt. Es versteht sich von selbst, dass bei jedem Verkaufsgespräch eine Beratung stattfindet. Falls keine Kunden in der Apotheke sind, widmen sich die Apotheker der Bestellung oder Herstellung von Arzneien, wobei bei der Herstellung mindestens zwei Apotheker zusammen arbeiten müssen.
Eine Beispiellösung findet sich in Anhang B und als taktile Grafik auf Seite 62 der taktilen Mappe.

5.1.8 Tabellarische Übersicht der Elemente des Anwendungsfalldiagramms

Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
System	Das System beinhaltet die Anwendungsfälle und stellt durch ein Rechteck die Grenze zwischen Akteuren und dem technischen System dar.	Ein Rechteck mit dem Namen „*System*“ oben links in der Ecke. Darin enthalten ist eine Ellipse mit dem Namen „A“. Außerhalb des Rechtecks ist ein Strichmännchen mit Namen „*X*“ darunter. Von diesem gibt es eine Linie zur Ellipse.	
Anwendungsfall	Funktionalität, welche vom System erbracht werden soll.	Eine Ellipse mit einem „*A*“ darin.	
Akteur	Rolle, welche der Anwender des Systems während der Interaktion mit diesem einnimmt.	Links ist ein Strichmännchen mit einem „*X*“ darunter. Rechts ist ein Rechteck mit „*<<actor>>*“ oben mittig und dem Namen „*X*“ in der nächsten Zeile.	
Include-Assoziation	Einbindung von Anwendungsfällen während der Ausführung. Ein Basisanwendungsfall bindet einen anderen ein. Dieser muss zwingend während der Ausführung des Basisanwendungsfalls eingebunden werden.	Zwei Ellipsen sind mit „*A*“ und „*B*“ beschriftet. Dazwischen ist eine gestrichelte Linie mit dem Wort „*<<include>>*“ darauf. Am Linienende von „*B*“ ist eine offene Pfeilspitze.	

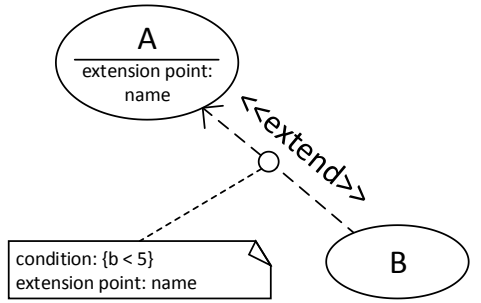
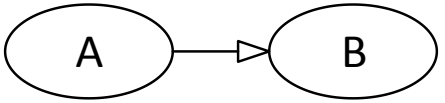
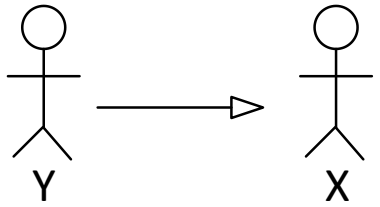
Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Extend-Assoziation	Erweiterung von Anwendungsfällen während der Ausführung. Optionale Einbindung eines erweiternden Anwendungsfalls innerhalb eines Basisanwendungsfalls.	Zwei Ellipsen sind mit „*A*“ und „*B*“ beschriftet. Dazwischen ist eine gestrichelte Linie mit dem Wort „<<*extend*>>“ darauf. Am Linienende von A ist eine offene Pfeilspitze. Mittig auf der Linie ist ein Kreis, welcher mit einer fein gestrichelten Linie mit einem Rechteck mit umgeklappter Ecke (Eselsohr) verbunden ist. Darin enthalten ist der Text: „*condition: {b<5} extension point: name*“	
Vererbung bei Anwendungsfällen	Der erbende Anwendungsfall übernimmt alle Eigenschaften und das gesamte Verhalten von dem Anwendungsfall, von dem er erbt.	Zwei Ellipsen, „*A*“ und „*B*“ sind mit einer Verbindungslinie mit einem nicht gefülltem Dreieck an der Ellipse von „*B*“ verbunden.	
Vererbung bei Akteuren	Erbt ein Akteur von einem anderen, ist dieser in allen Anwendungsfällen involviert, in denen der andere Akteur involviert ist.	Zwei Strichmännchen sind mit „*Y*“ und „*X*“ beschriftet. Dazwischen ist eine Linie mit einem nicht gefülltem Dreieck am Ende mit Spitze in Richtung „*X*“ .	

Tabelle 2: Kurzreferenz der Elemente des Anwendungsfalldiagramms

5.2 Aktivitätsdiagramm

Die taktilen Grafiken des Aktivitätsdiagramms (im englischen Activity Diagram) befinden sich auf den Seiten 17–27 der taktilen Mappe.

Aktivitätsdiagramme sind vielseitig einsetzbar. Sie werden häufig zur Modellierung des Verhaltens mittels Kontroll-und Datenflüssen verwendet. Sie können ebenfalls eingesetzt werden, um die Abläufe innerhalb eines Anwendungsfalls zu spezifizieren. Im Wesentlichen besteht ein Aktivitätsdiagramm aus Knoten und Kanten, wie in einem Graphen. Die Knoten repräsentieren Aktionen, welche elementare Vorgänge im System darstellen. Diese werden durch Kanten verbunden. Der Ablauffluss von Aktivitätsdiagrammen ist ähnlich zu der Arbeitsweise von Petri Netzen. Pro Startknoten wird im Aktivitätsdiagramm ein sogenanntes Token initiiert und beginnt durch das Diagramm zu wandern, bis es am Ende zerstört wird.

5.2.1 Aktionen und Kanten

Aktionen sind atomar, d.h. sie können nicht weiter zerlegt werden und stellen eine Einheit ausführbarer Funktionalität dar (Kecher & Salvanos, 2015). Aktionen sind das am häufigsten verwendete Element. Alle anderen Elemente werden fast ausschließlich verwendet, um Aktionen in eine Reihenfolge zu bringen. Aktionen werden als Rechteck mit abgerundeten Ecken und dem Namen mittigen dargestellt. Ein Beispiel stellt in Abbildung 22 die Aktion *„Buch auswählen“* dar.

Abbildung 22: Einzelne Aktion „Buch auswählen“

Aktionen werden mithilfe von Kanten, sogenannten Kontrollflüssen, miteinander verknüpft. Kontrollflüsse werden als Linien mit einer offenen Pfeilspitze an einem Ende dargestellt. Alle Kontrollflüsse weisen eine definierte Richtung auf, welche die Reihenfolge der Aktionen festlegt. Diese haben keine Beschriftung, da sie keine Ereignisse, sondern nur die Reihenfolge und Abläufe der Aktionen darstellen, woher auch die Bezeichnung Kontrollfluss stammt. Das Beispiel in Abbildung 23 zeigt einen Kontrollfluss von der Aktion *„Buchhandlung betreten“* zur Aktion *„Buch auswählen“*. Dadurch ist leicht zu sehen, dass die Buchhandlung erst betreten werden muss, bevor die Aktion *„Buch auswählen“* ausgeführt werden kann.

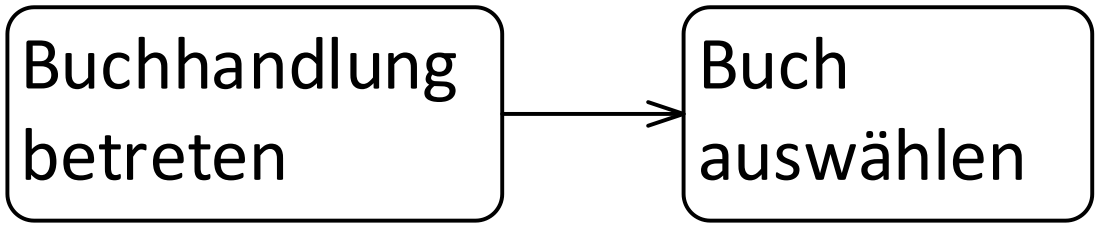


Abbildung 23: Beispiel einer Transition zwischen zwei Aktionen „Buchhandlung betreten“ und „Buch auswählen“

5.2.2 Start-, Endknoten und Flussende

Den Startpunkt eines Aktivitätsdiagramms bildet der Startknoten, dargestellt durch einen schwarzen Kreis in Abbildung 24. In ihm beginnt der Kontrollfluss des Diagramms. Sollen mehrere Aktionen gleichzeitig durchgeführt werden, können mehrere Startknoten modelliert werden. Pro Startknoten wird dabei ein Token initiiert, welches seinen Weg durch das Aktivitätsdiagramm startet.

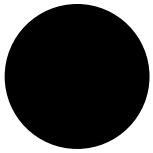


Abbildung 24: Startknoten

Es existieren zwei verschiedene Endknoten in Aktivitätsdiagrammen. Das Flussende, dargestellt in Abbildung 25 als Kreis mit einem X darin, beendet nur den in ihn führenden Fluss und zerstört das Token. Sollten parallele Flüsse gestartet sein, werden diese dadurch nicht beeinflusst. Parallele Flüsse werden in Kapitel 5.2.5 erklärt.

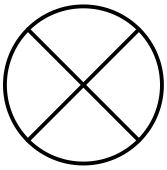


Abbildung 25: Flussende

Ein Endknoten, dargestellt durch einen weißen Kreis mit einem kleineren schwarzen Kreis darin, wie in Abbildung 26, beendet dagegen alle Kontrollflüsse des Aktivitätsdiagramms und damit auch die Aktivität selbst. Sollten noch Tokens in anderen, parallelen Kontrollflüssen existieren, werden diese ebenfalls beendet und zerstört. Es können mehrere Endknoten innerhalb eines Diagramms existieren.

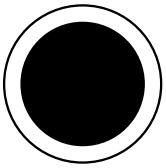


Abbildung 26: Endknoten

5.2.3 Aktivitäten

Mehrere Aktionen zusammen können in einer Aktivität gekapselt werden, welche beispielsweise die Durchführung eines Anwendungsfalls in einzelnen Schritten beschreibt. Aktivitäten sind nicht atomar, sondern repräsentieren Aktionsfolgen. Aktivitäten werden ebenfalls durch abgerundete Rechtecke repräsentiert. Innerhalb der Aktivitäten wird die Aktionsfolge dargestellt. Der Name der Aktivität ist oben links innerhalb des Rechtecks notiert. Ein Beispiel der Aktivität *„Buch kaufen“* ist in Abbildung 27 dargestellt. Die Aktivität besteht aus einem *„Start-* und *Endknoten“* und insgesamt vier Aktionen, welche linear durchgeführt werden. Vom *„Startknoten“* aus beginnt der Fluss und führt in die erste Aktion *„Buchhandlung betreten“*. Danach werden die Aktionen *„Buch auswählen“* und *„Buch bezahlen“* ausgeführt. Abschließend wird die Aktion *„Buchhandlung verlassen“* durchgeführt und mittels des *„Endknotens“* die Aktivität beendet.

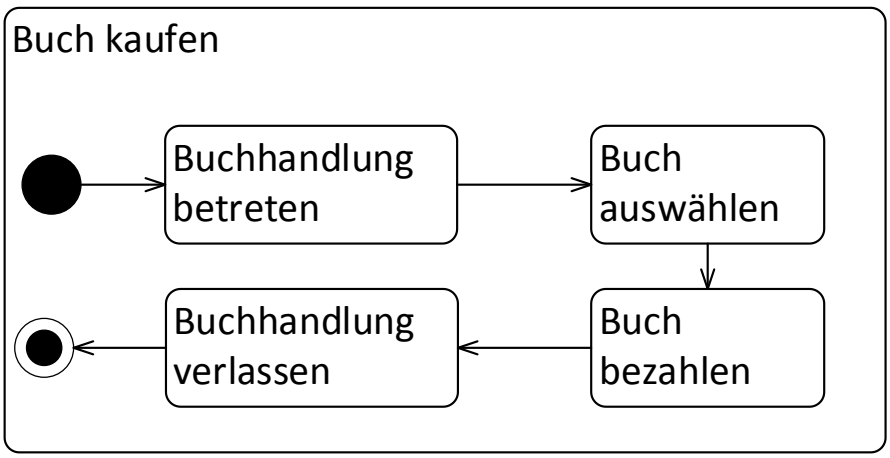


Abbildung 27: Beispiel der Aktivität „Buch kaufen“ bestehend aus mehreren Aktionen

Aktivitäten können zusätzlich Vor- und Nachbedingungen haben, welche vor, bzw. nach der Ausführung einer Aktivität erfüllt sein müssen. Um diese zu spezifizieren, werden sie in das Rechteck der Aktivität geschrieben, indem zuerst der Typ (Vor- oder Nachbedingung) in Guillements und dann die Bedingung angegeben wird. Mit dem Schlüsselwort *„precondition“* wird eine Vorbedingung und mit *„postcondition“* eine Nachbedingung angegeben. Beide werden oben rechts innerhalb der Aktivität platziert. Dargestellt ist diese Syntax in Abbildung 28, welche die Aktivität *„Wissen erweitern“* darstellt. Als Vorbedingung gilt, dass eine *„Wissenslücke“* vorhanden ist, als Nachbedingung der Aktivität, dass man *„neues Wissen“* erworben hat. Entsprechend steht am oberen rechten Rand der Aktivität *„<<precondition>> Wissenslücke“* und *„<<postcondition>> neues Wissen“*. Die Aktivität beginnt mit einem *„Startknoten“* von welchem der Kontrollfluss über die Aktionen *„Sachbuch kaufen“* und *„Sachbuch lesen“* zum *„Endknoten“* fließt.

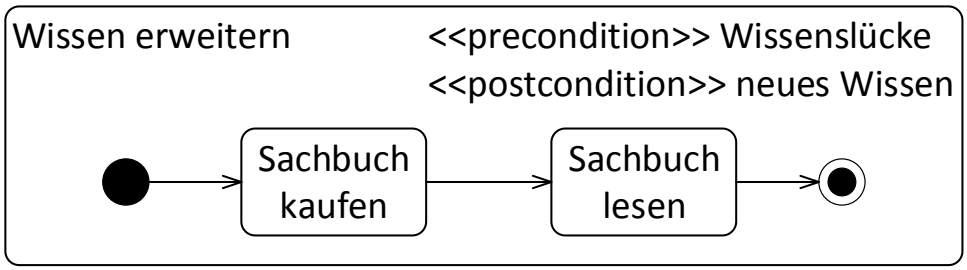


Abbildung 28: Beispiel einer Aktivität mit Vor- und Nachbedingung

5.2.4 Bedingte Verzweigung von Flüssen

Mithilfe bedingter Verzweigungen können alternative Flüsse modelliert werden. Eine bedingte Verzweigung wird mithilfe eines Entscheidungsknotens, welcher wie in Abbildung 29 als weißer Diamant oder Raute dargestellt wird, herbeigeführt. Der Entscheidungsknoten hat dabei immer genau einen eingehenden Kontrollfluss und mindestens zwei ausgehende Kontrollflüsse. Die ausgehenden Kontrollflüsse beinhalten Guards, d.h. Bedingungen, welche den jeweiligen Kontrollfluss schalten. Diese Guards werden in eckigen Klammern an den jeweiligen Kontrollfluss notiert. Die ausgehenden Kontrollflüsse sollten dabei immer alle möglichen Entscheidungen abdecken. Dies kann einfach durch einen „else“ oder „sonst“ Kontrollfluss gelöst werden, welcher immer wahr wird, falls alle anderen Bedingungen nicht zu wahr ausgewertet werden können. Im Idealfall sollten alle Bedingungen ausgehender Kontrollflüsse disjunkt sein. Falls die Bedingungen an mehreren Kontrollflüssen als wahr ausgewertet werden, wird zufällig (nichtdeterministisch) ein Kontrollfluss ausgewählt. Im Beispiel sind alle Flüsse disjunkt und man kann zwischen *„Fantasy“*, *„Krimi“* oder *„sonst“* wählen. Das ankommende Token wandert hierbei nur den gewählten Fluss entlang.

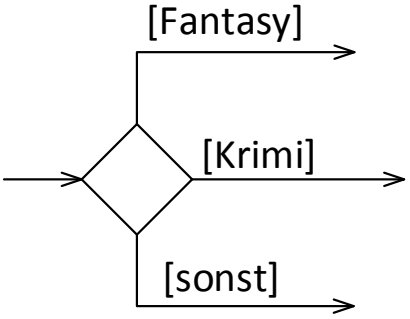


Abbildung 29: Darstellung einer bedingten Verzweigung mit drei Alternativen

Um die verschiedenen Kontrollflüsse wieder zusammenzuführen, wird ein Verbindungsknoten benötigt. Dieser ist visuell identisch zum Entscheidungsknoten mit dem Unterschied, dass mehrere Kontrollflüsse eingehen und nur ein Kontrollfluss davon ausgeht. Weiterhin stehen keine Bedingungen an den Kontrollflüssen.

Abbildung 30 stellt ein einfaches Beispiel einer bedingten Verzweigung dar. Vom *„Startknoten“* aus findet sofort ein Übergang zur Aktion *„Buch suchen“* statt. Von diesem gelangt der Kontrollfluss in einen Entscheidungsknoten, von welchem zwei Alternativen gewählt werden können. Eine Bedingung lautet *„[Fantasy]“,* die andere *„[sonst]“.* Wird das Genre *„Fantasy“* gewählt, wird die Aktion *„Tolkien wählen“* ausgeführt. Danach wird der Fluss sofort wieder durch einen Verbindungsknoten zusammengeführt. Im „[*sonst*]“ Fall wird die Aktion *„Bestseller wählen“* ausgeführt und danach der Kontrollfluss in den Verbindungsknoten geführt. Nach der Zusammenführung wird in die Aktion *„Buch kaufen“* und von dieser zum *„Endknoten“* gewechselt, welcher die Aktionsfolge beendet.

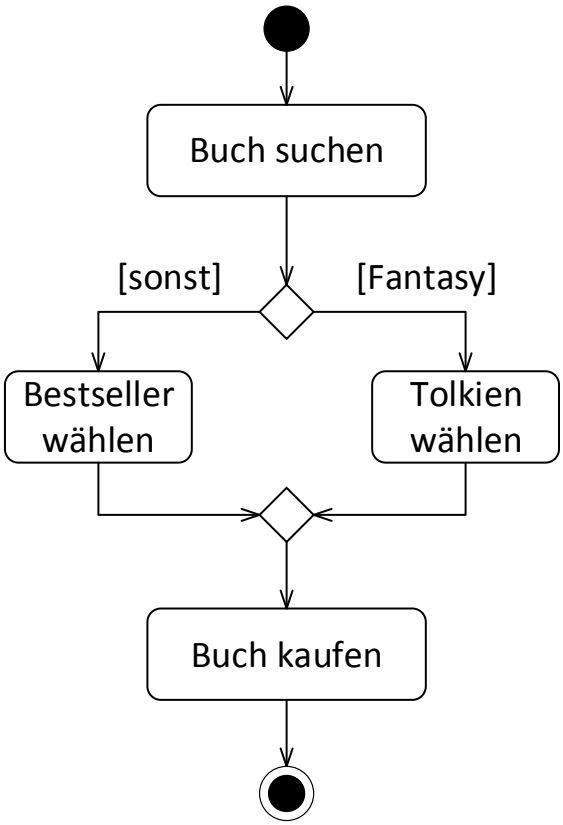


Abbildung 30: Beispiel einer bedingten Verzweigung

Mithilfe bedingter Verzweigungen können ebenfalls Schleifen konstruiert werden, sodass bei bestimmten Bedingungen einige Aktionen mehrmals wiederholt werden. Ein oder mehrere Kontrollflüsse zeigen dabei von der bedingten Verzweigung auf bereits besuchte Aktionen oder andere Knoten wie Verbindungsknoten, sodass diese dadurch mehrfach ausgeführt werden können. Abbildung 31 zeigt ein Beispiel einer Schleife. Vom *Startknoten* aus wird die Aktion „*Buch wählen*“ ausgeführt. Danach wird die Bedingung *„[Summe < Gutscheinwert]“* ausgewertet. Wird diese Bedingung zu wahr ausgewertet, wird wiederrum die Aktion „*Buch wählen*“ ausgeführt und danach wieder die Bedingung überprüft. Solange die Summe der Preise aller Bücher demnach kleiner dem Wert des Gutscheines entspricht, können weiterhin neue Bücher ausgewählt werden. Wird diese Bedingung nicht mehr zu wahr ausgewertet, wird die Aktion „*Bezahlen*“ ausgeführt und danach in den *Endknoten* gewechselt.

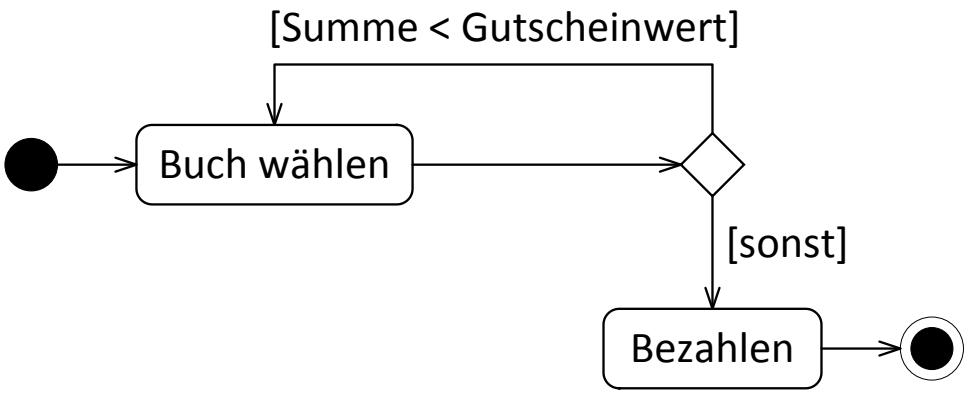


Abbildung 31: Darstellung einer Schleife durch bedingte Verzweigungen

5.2.5 Parallelisierung und Synchronisierung von Flüssen

Abläufe können auch nebenläufig verzweigt werden. Im Gegensatz zur bedingten Verzweigung wird dabei nicht nur ein Fluss ausgeführt, sondern alle ausgehenden Flüsse parallel. Dargestellt wird dies durch einen Balken mit einem eingehenden Kontrollfluss und mindestens zwei ausgehenden Kontrollflüssen wie auf der linken Seite in Abbildung 32. Dabei wird für jeden ausgehenden Kontrollfluss ein Token erzeugt, welches ab sofort diesem Kontrollfluss folgt. Die rechte Seite zeigt die Zusammenführung aller parallelen Flüsse zu einem einzigen. Die Tokens sammeln sich an diesem Symbol, und es wird gewartet, bis alle Flüsse synchronisiert werden können, bevor der ausgehende Kontrollfluss aktiv wird und ein Token weiterleitet. Symbolisch wird dies erneut durch einen Balken mit mindestens zwei eingehenden Kontrollflüssen und genau einem ausgehenden Kontrollfluss dargestellt. Eine Parallelisierung muss nicht zwingend wieder zusammengeführt werden. Alle parallelen Flüsse könnten ebenso in einem Flussende oder Endknoten führen. Ebenfalls kann ein Synchronisierungsknoten zugleich ein Parallelisierungsknoten sein, indem mehrere Kanten ein- und ausgehen. Dabei wird demzufolge gewartet, bis alle Flüsse die Synchronisierung erreicht haben um danach wieder in verschiedene Flüsse aufgespalten zu werden.

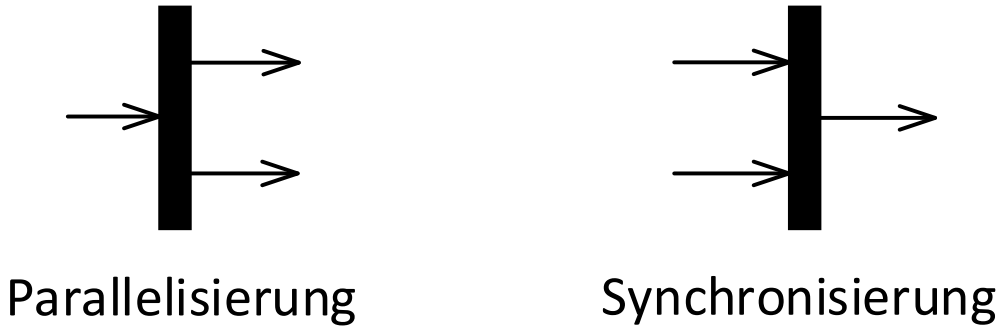


Abbildung 32: Darstellung der Parallelisierung und Synchronisierung von Kontrollflüssen

Ein Beispiel für die Parallelisierung und Synchronisierung von Kontrollflüssen ist in Abbildung 33 gegeben. Vom *„Startknoten“* wird die Aktion *„Buchhandlung betreten“* erreicht. Danach wird der Kontrollfluss in zwei parallele Kontrollflüsse aufgeteilt, d.h. von der Aktion *„Buchhandlung betreten“* geht ein Kontrollfluss in einen Balken und von diesem geht je ein Kontrollfluss zu den Aktionen *„Umsehen“* und *„Kaffee trinken“*. Von diesen beiden geht dann je ein Kontrollfluss wieder in einen Balken zur Synchronisierung der Flüsse. Von diesem erreicht man die Aktion *„Buchhandlung verlassen“* und danach direkt den *„Endknoten“*.

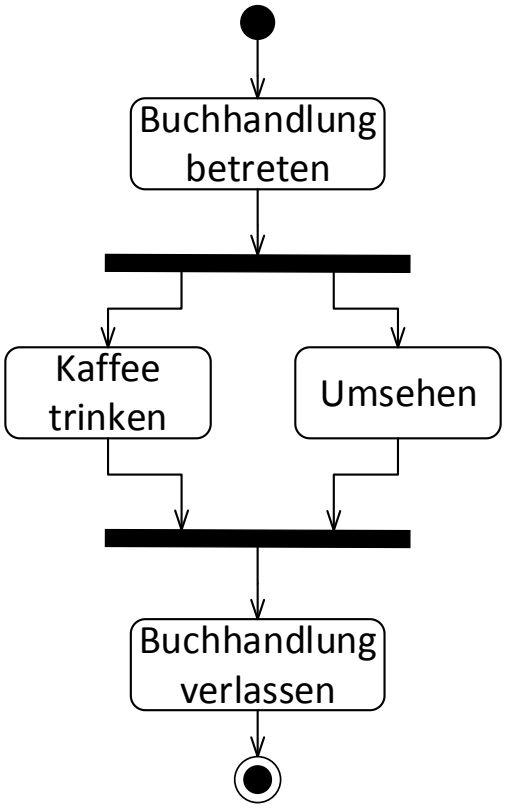


Abbildung 33: Beispiel paralleler Kontrollflüsse

5.2.6 Aktivitätsbereiche: Swimlanes

Mithilfe von Aktivitätsbereichen, oder dem mehr verbreiteten englischen Begriff „Swimlanes“, können Aktionen zu Organisationseinheiten gruppiert werden. Es können beliebig viele Bereiche erzeugt werden. Ebenfalls ist es möglich die Bereiche nochmals zu verfeinern, um hierarchische Bereiche zu bilden. Die Aktivitätsbereiche werden mithilfe von Rechtecken dargestellt. Der Name des Bereichs steht dabei oberhalb des Bereiches in einem eigenen Rechteck. Aktivitätsbereiche können sowohl vertikal als auch horizontal angeordnet werden. Auch beide Varianten sind in einem Diagramm möglich, um eine sehr detaillierte Unterteilung zu erreichen. Die einzelnen Aktionen werden in dem Bereich platziert, in dessen Zuständigkeit die Aktion gehört, z.B. wer die Aktion ausführt. Ein Beispiel ist in Abbildung 34 dargestellt. Die Aktivitätsbereiche sind mit den Namen „*Kunde*“ und „*Mitarbeiter*“ beschriftet und vertikal angeordnet. Die Aktionsfolge fängt im Bereich des „*Kunden*“ mit einem *Startknoten* an. Danach wird die Aktion „*Buch suchen*“ vom Kunden ausgeführt. Der „*Mitarbeiter*“ zeigt dem Kunden daraufhin das Regal, in welchem das Buch zu finden ist. Die Aktion ist mit „*Regal zeigen*“ beschriftet. Zuletzt wird die Aktion „*kaufen*“ vom Kunden ausgeführt.

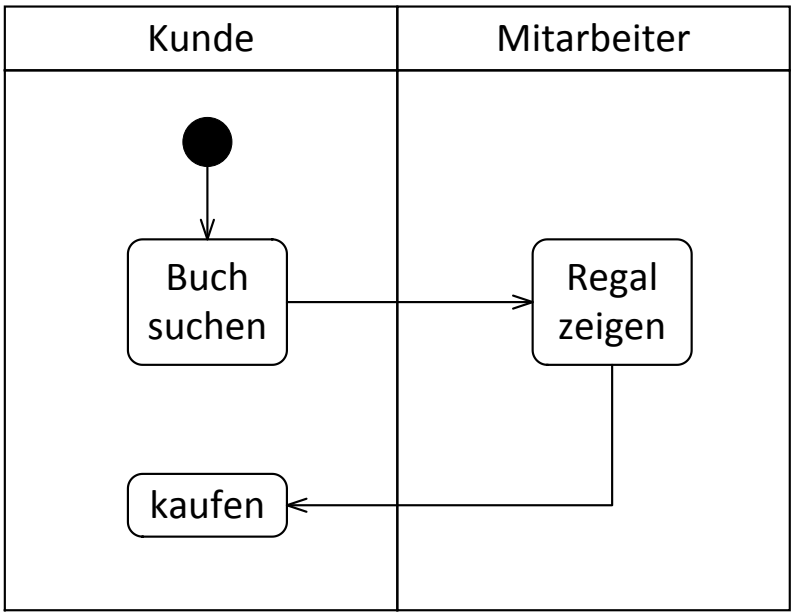


Abbildung 34: Beispiel von zwei Aktivitätsbereichen "Kunde" und "Mitarbeiter"

5.2.7 Beispiel eines Aktivitätsdiagramms für eine Navigations-App

Das Aktivitätsdiagramm in Abbildung 35 zeigt die für die Navigation benötigte Aktionsreihenfolge von der Eingabe eines Zieles, bis zur Ausgabe der endgültigen Route. Das Diagramm beginnt mit einem *„Startknoten“*, von welchem sofort eine Parallelisierung der weiteren Aktionen erfolgt. Eine Seite der parallelen Verarbeitung enthält nur die Aktion *„Aktuelle Position ermitteln“*. Die andere Seite startet mit der Aktion *„Zielort eingeben“* nach der Parallelisierung. Danach wird eine bedingte Verzweigung ausgeführt. Wird die Bedingung *„[Abkürzungen einbeziehen]“* als wahr ausgewertet, wird die Aktion *„Wege einbeziehen“* ausgeführt und danach die Verzweigung zusammengeführt. Im *„[sonst]“* Fall wird der Fluss sofort wieder zusammengeführt. Nachdem diese Aktionen durchgeführt wurden, endet die parallele Ausführung, und die beiden Flüsse werden wieder synchronisiert. Nach der Synchronisierung werden die Aktionen *„Route berechnen“* und *„Route ausgeben“* nacheinander ausgeführt, bevor der *„Endknoten“* erreicht wird.

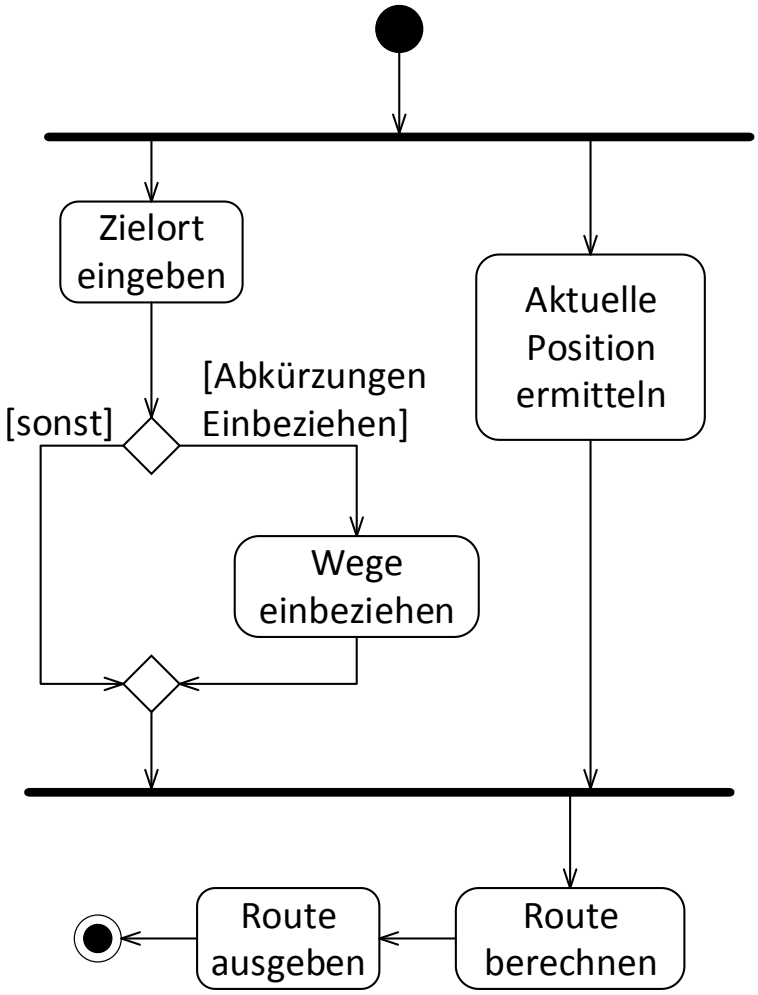


Abbildung 35: Aktivitätsdiagramm des Navigationsbeispiels

5.2.8 Verständnisfragen zum Aktivitätsdiagramm

Dieses Kapitel besteht aus fünf Multiple-Choice-Fragen (Business Informatics Group, 2015), um das Verständnis für das Aktivitätsdiagramm zu überprüfen. Jede Frage hat vier Antwortmöglichkeiten a–d. Bei jeder Frage können eine, zwei, drei oder vier Antworten richtig sein. Die Auflösung ist in Anhang A und im Detail in Anhang A.3 zu finden.

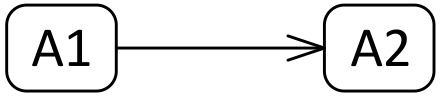
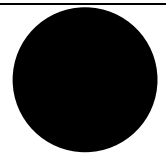
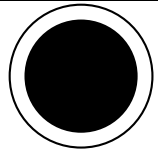
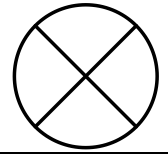
1. Ein Startknoten in einem Aktivitätsdiagramm…
   (a) versorgt alle ausgehenden Kanten mit Token.
   (b) darf pro Aktivität genau einmal vorkommen.
   (c) wird als weißer Kreis dargestellt.
   (d) stellt den Beginn eines Aktivitätsablaufes dar.
2. Ein Synchronisierungsknoten in einem Aktivitätsdiagramm...
   (a) führt nebenläufige Abläufe wieder zusammen.
   (b) wird als ungefüllte Raute dargestellt.
   (c) vereinigt Token, sobald diese an allen eingehenden Kanten vorhanden sind.
   (d) führt alternative Abläufe wieder zusammen.
3. Ein Endknoten in einem Aktivitätsdiagramm...
   (a) beendet alle Abläufe einer Aktivität.
   (b) verbietet die Ausführung weiterer Aktionen innerhalb der Aktivität.
   (c) kommt pro Aktivität genau einmal vor.
   (d) wird als schwarzer Kreis dargestellt.

4. Ein Parallelisierungsknoten in einem Aktivitätsdiagramm...
   (a) ist nur dann formal gültig, wenn es auch einen zugehörigen Synchronisierungsknoten gibt.
   (b) dient zur Modellierung der Aufspaltung in nebenläufige Abläufe.
   (c) dupliziert eingehende Token für alle ausgehenden Kanten.
   (d) stellt eine Notationsvariante des Entscheidungsknotens dar.
5. Welche der folgenden Aussagen treffen auf Aktionen des Aktivitätsdiagramms zu?
   (a) Aktionen können in Aktivitäten zu größeren Einheiten zusammengefasst werden.
   (b) Aktionen sind atomar.
   (c) Aktionen werden mithilfe von Assoziationen miteinander verbunden.
   (d) Aktionen beinhalten mehrere Aktivitäten.

5.2.9 Modellierungsaufgabe Aktivitätsdiagramm

Ein Kunde möchte eine Arznei auf Basis von Naturprodukten in einer Apotheke erstehen. Dazu sucht er eine Apotheke auf und beschreibt seine Krankheit. Falls ein Naturprodukt vorhanden ist, kauft er dieses und geht wieder. Falls jedoch nicht das gewünschte Produkt vorhanden ist, muss er sich mit einem Standardprodukt begnügen. Er regt jedoch an, dass das Naturprodukt beim nächsten Mal vorhanden sein soll, welches der Apotheker verspricht. Der Kunde verlässt daraufhin die Apotheke.
Eine Beispiellösung findet sich in Anhang B und als taktile Grafik auf Seite 63 der taktilen Mappe.

5.2.10 Tabellarische Übersicht der Elemente des Aktivitätsdiagramms

Name	Beschreibung	Beschreibung der grafische Darstellung	Grafische Darstellung
Aktivität	Beschreibt nutzerdefiniertes Verhalten. Kann Vor- und Nachbedingungen, Aktionen, Kontrollelemente etc. enthalten.	Ein Rechteck mit abgerundeten Ecken und dem Namen „*Aktivität*“ mittig platziert.	
Aktion	Basiselement einer Aktivität. Kann nicht weiter zerlegt werden.	Ein Rechteck mit abgerundeten Ecken und dem Namen „*Aktion*“ mittig.	
Kante/Kontrollfluss	Verbindung zwischen Aktionen	Eine Linie mit einer offenen Pfeilspitze. Verbindet hier die Aktion „*A1*“ mit Aktion „*A2*“, beide durch Rechtecke mit abgerundeten Ecken dargestellt.	
Startknoten	Start der Ausführung einer Aktivität/Aktionsfolge	Schwarzer Kreis	
Endknoten	Ende der Ausführung aller Kontrollflüsse	Weißer Kreis mit kleinerem schwarzen Kreis enthalten	
Flussende	Beendet den aktuellen Fluss des Diagramms, andere Flüsse werden dadurch nicht beeinflusst.	Weißer Kreis mit einem Kreuz oder X darin.	

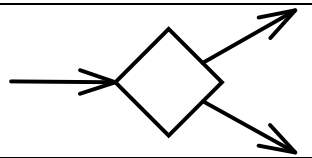
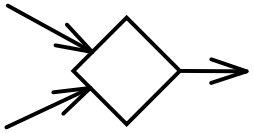
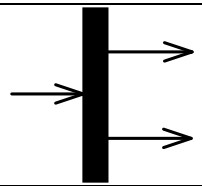
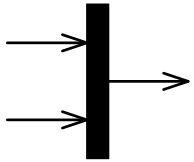
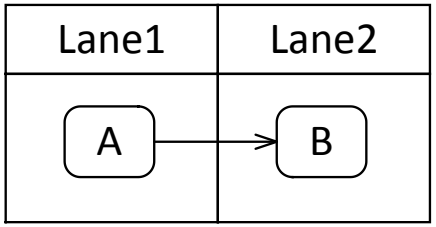
Name	Beschreibung	Beschreibung der grafische Darstellung	Grafische Darstellung
Verzweigung	Entscheidungsknoten, von welchem mehrere alternative Kontrollflüsse möglich sind	Ein weißer Diamant mit einer eingehenden und mindestens zwei ausgehenden Kontrollflüssen.	
Zusammenführung	Verbindungsknoten: Zusammenführung von einem oder mehreren Kontrollflüssen in einen Kontrollfluss	Ein weißer Diamant mit mindestens zwei eingehenden und genau einem ausgehenden Kontrollfluss.	
Parallelisierung	Aufteilung eines Kontrollflusses in mehrere parallele Kontrollflüsse.	Ein schwarzer Balken mit einem eingehenden und mindestens zwei ausgehenden Kontrollflüssen.	
Synchronisierung	Zusammenführung mehrerer paralleler Kontrollflüsse zu einem Kontrollfluss.	Ein schwarzer Balken mit mindestens zwei eingehenden Kontrollflüssen und einem ausgehenden Kontrollfluss.	
Aktivitätsbereich / Swimlane	Gruppierung von Aktionsknoten in Organisationseinheiten	Vertikale oder horizontale Rechtecke mit dem Namen abgegrenzt durch ein eigenes Rechteck. *„Lane1“* und *„Lane2“* sind vertikal angeordnet. Von Aktion *„A“* in *„Lane1“* gibt es ein Kontrollfluss zu Aktion *„B“* in *„Lane2“*.	

Tabelle 3: Kurzreferenz der Elemente des Aktivitätsdiagramms

5.3 Zustandsdiagramm

Die taktilen Grafiken des Zustandsdiagramms (im englischen State Chart) befinden sich auf den Seiten 28–42 der taktilen Mappe.

Ein Zustandsdiagramm beschreibt die erreichbaren Zustände eines Systems und deren Übergänge ineinander, d.h. es wird das dynamische Verhalten eines Systems modelliert. Eine wichtige Eigenschaft, welche damit dargestellt werden kann, ist die Lebensdauer eines Objektes von der Erstellung bis zur Zerstörung (Terminierung). Ein häufiges Anwendungsfeld ist die Darstellung der Reaktionen eines Systems, z.B. bei der Modellierung des Verhaltens von Benutzeroberflächen, die häufig auf Befehle von Benutzern reagieren. Das Zustandsdiagramm basiert auf deterministischen endlichen Automaten und wird daher auch Zustandsautomat genannt.

5.3.1 Zustände

Hauptbestandteile von Zustandsdiagrammen sind die Zustände, welche als Rechtecke mit abgerundeten Ecken dargestellt werden. Zwischen diesen Zuständen bestehen gerichtete Verbindungen, genannt Transitionen. Ein Zustand beschreibt eine Situation, in der bestimmte Bedingungen gelten. Während eines Zustands können Events (im deutschen: Ereignisse) und Aktionen stattfinden, sogenannte interne Aktionen. Events und daraus resultierende Aktionen werden durch die Kombination *„event/aktion“* dargestellt. Sobald ein Event auftritt, wird die dazugehörige Aktion ausgeführt. Ein Event kann eine beliebige Bezeichnung haben, es existieren jedoch drei vorgegebene Events: *„entry“, „exit“* und *„do“*. Die Events treten jeweils beim Betreten, Verlassen und während der Ausführung des Zustands auf. Das Beispiel in Abbildung 36 stellt den Zustand einer *„Buchhandlung“* dar. Der Zustand enthält die internen Aktionen *„entry/türÖffnen()“, „do/umschauen()“* und *„exit/türSchließen()“.* Beim Betreten der Buchhandlung, d.h. beim Betreten des Zustandes, wird die Funktion *„türÖffnen()“* aufgerufen. Dies wird durch das Schlüsselwort *„entry“* dargestellt. Diese Aktion muss beendet werden, bevor eine andere Aktion stattfinden kann. Die Aktion *„umschauen()“* mit dem Schlüsselwort *„do“* wird nach Abschluss der *„entry“*-Aktion beim Betreten des Zustandes gestartet, und wird so lange ausgeführt, bis sie endet oder der Zustand verlassen wird (Kecher & Salvanos, 2015). Die *„exit“*-Aktion wird vor dem Verlassen des Zustandes, jedoch nach *„entry“*- und *„do“*-Aktionen, ausgeführt. Erst nach der vollständigen Ausführung der *„exit“*-Aktion kann der Zustand verlassen werden, d.h. die Funktion *„türSchließen()“* muss beendet sein, bevor ein Übergang in einen anderen Zustand möglich ist.

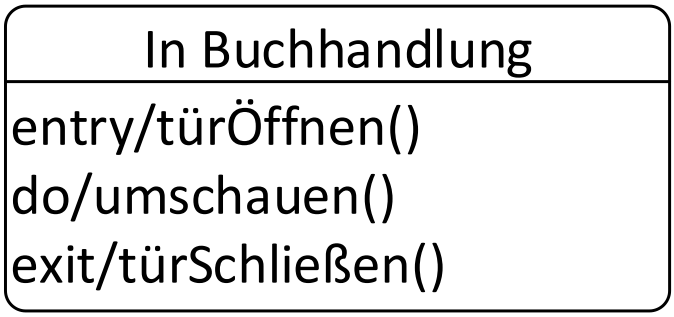


Abbildung 36: Zustand „In Buchhandlung“ mit drei internen Aktionen

5.3.2 Spezielle Zustände: Start, Ende und Terminator

Der Startzustand zeigt den Startpunkt des Zustandsdiagramms an und wird durch einen schwarzen Kreis, wie in Abbildung 37 zu sehen, dargestellt. In einem einfachen Zustandsdiagramm darf es nur einen Startzustand geben, welcher nach Starten des Zustandsdiagramms sofort verlassen werden muss. Details, wann es mehrere Startzustände geben kann werden in Kapitel 5.3.5 erklärt. Die ausgehende Transition darf nicht durch Bedingungen, welche deren Ausführung verzögern würden, eingeschränkt sein. Der Startzustand darf keine eingehende Transition und nur eine ausgehende Transition haben.

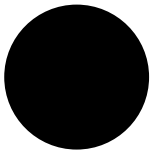


Abbildung 37: Startzustand

Im Gegensatz dazu können Zustandsdiagramme beliebig viele Endzustände und Terminatoren haben. Ein Endzustand wird als weißer Kreis mit einem kleineren schwarzen Kreis darin, wie in Abbildung 38 zu sehen, dargestellt. Er beendet die Ausführung einer Region oder Ebene von Zuständen. Das Zustandsdiagramm kann demnach weiterhin aktiv sein, falls mehrere Ablaufpfade existieren. Details wie diese entstehen, werden in den Kapiteln 5.3.4 und 5.3.5 gegeben.

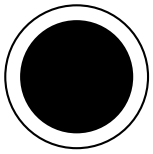


Abbildung 38: Endzustand

Im Gegensatz dazu beendet der in Abbildung 39 durch ein „X“ dargestellte Terminator die Ausführung des gesamten Zustandsdiagramms. Terminatoren werden häufig verwendet, um abrupte Abbrüche zu erzwingen, z.B. im Falle eines Fehlers.

Abbildung 39: Terminator

5.3.3 Zustandsübergänge: Transitionen

Zustände sind über Zustandsübergänge, sogenannte Transitionen, miteinander verbunden. Mit deren Hilfe kann von einem Zustand in einen anderen gewechselt werden. Eine Transition wird durch eine Linie mit offener Pfeilspitze dargestellt. Die einfachste Transition ist die ε-Transition (in Worten: epsilon-Transition). Diese kann immer ausgeführt werden, sofern interne Aktionen abgeschlossen sind. Eine Transition kann ebenfalls durch Events ausgelöst werden. Es existieren verschiedene Events, wie z.B. die Ankunft eines Signals, ein abgelaufener Zeitgeber oder die Erfüllung einer Bedingung. Diese Events werden an den Transition notiert, wie in Abbildung 40 für das Event *„bestellen“* gezeigt. Mithilfe dieses Events kann vom Zustand *„Buch nicht vorhanden“* in den Zustand *„Buch vorhanden“* gewechselt werden.

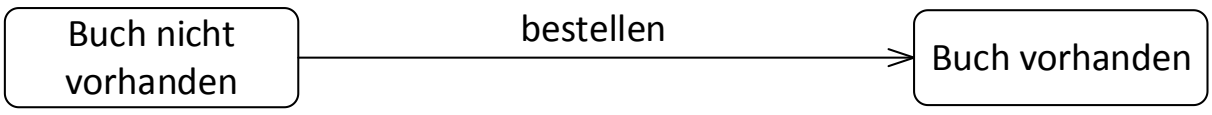


Abbildung 40: Transition zwischen zwei Zuständen mit einem Event

Zusätzlich zu Events können noch sogenannte Guards (dt. Bedingung) oder Effekte für Transitionen gelten. Die nächsten beiden Beispiele sind wie das Beispiel in Abbildung 40 aufgebaut mit einer Transition *„bestellen“* zwischen den Zuständen *„Buch nicht vorhanden“* und *„Buch vorhanden“*. Es kommen jedoch zusätzliche Transitionen und Beschriftungen dazu.

Im darauf aufbauenden Beispiel aus Abbildung 41 wird eine zusätzliche Transition hinzugefügt. Beide Transitionen werden mit Guards annotiert. Eine Transition wird nur ausgelöst,

wenn modellierte Bedingungen wahr sind. Im Beispiel wird überprüft, ob das Buch, welches bestellt werden soll, verfügbar ist. Der Guard wird in eckigen Klammern hinter dem Event notiert. In diesem Beispiel gibt es zwei Transitionen. Eine Transition ist mit *„bestellen[verfügbar]“* beschriftet. Diese wird ausgelöst, falls bei der Überprüfung der Bedingung *„[verfügbar]“* diese zu *wahr* ausgewertet wird. In diesem Fall wird vom Zustand *„Buch nicht vorhanden“* in den Zustand *„Buch vorhanden“* gewechselt. Falls diese Überprüfung nicht wahr ist, wird eine reflexive Transition (Ausgangs- und Zielzustand sind identisch) vom Zustand *„Buch nicht vorhanden“* zum Zustand *„Buch nicht vorhanden“* ausgelöst. In diesem Fall wird die Bedingung „[*nicht verfügbar*]“ zu *wahr* ausgewertet. Bei einer reflexiven Transition muss immer ein Event oder Guard angegeben werden, da der Zustand sonst endlos betreten und verlassen wird.

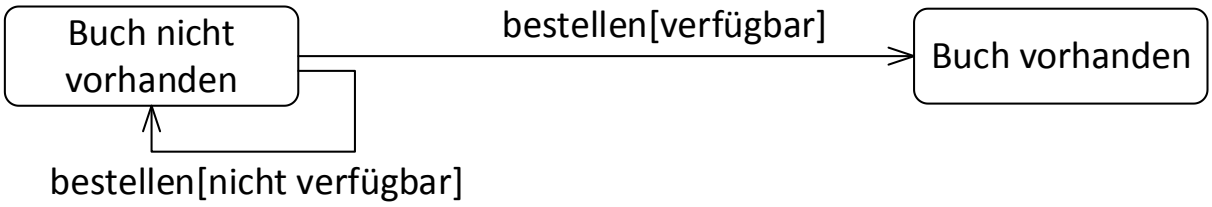


Abbildung 41: Transition zwischen zwei Zuständen mit Events und Guards

Das Beispiel in Abbildung 42 fügt zusätzlich noch eine Aktion, oder auch Effekt genannt, hinzu. Ein Effekt wird während der Ausführung einer Transition durchgeführt. Dieser wird mit einem Schrägstrich (Slash) hinter dem Event oder den Guard notiert. Im Beispiel ist ein Effekt nur an der reflexiven Transition angegeben: *„bestellen[nicht verfügbar]/drucken()“*. Ist demnach ein Buch nicht verfügbar, wird der erneute Druck des Buches ausgelöst. Die andere Transition ist identisch mit dem vorigen Beispiel.

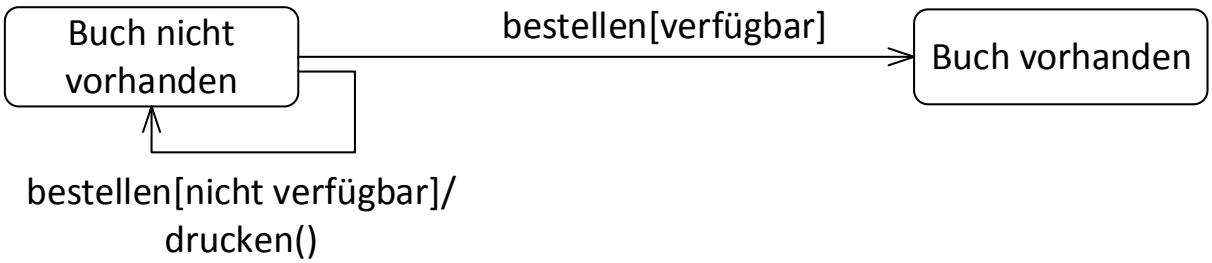


Abbildung 42: Transition zwischen zwei Zuständen mit Event, Guard und Aktion

5.3.4 Bedingte Verzweigungen und Parallelisierung

Ebenso wie im Aktivitätsdiagramm, können bedingte Verzweigungen oder parallele Ablaufpfade modelliert werden. Die Notationen sind dabei nahezu identisch zu denjenigen im Aktivitätsdiagramm. Anstelle der Aktionen werden jedoch entsprechend Zustände eingesetzt. Im Gegensatz zum Aktivitätsdiagramm muss eine bedingte Verzweigung nicht mithilfe eines Verbindungsknotens zusammengeführt werden, die Verwendung eines solchen Verbindungsknotens erhöht jedoch die Übersichtlichkeit. Alle anderen Elemente sind identisch und können in den Kapiteln 5.2.4 und 5.2.5 nachgelesen werden. Obwohl es die Notation paralleler Abschnitte gibt, werden diese in Zustandsdiagrammen sehr selten verwendet. Parallele Zustände werde mit orthogonalen Zuständen modelliert, welche teilweise davor eine Parallelisierung zur Verdeutlichung modelliert haben, dies ist jedoch ebenfalls unüblich. Orthogonale Zustände werden in Kapitel 5.3.5.3 behandelt.

5.3.5 Hierarchische/Komplexe Zustände

Wie bereits am Anfang angedeutet, können Zustände hierarchisch angeordnet sein. Bei hierarchischen Zuständen können Zustände wiederum vollständige Zustandsdiagramme enthalten. Diese Zustände nennt man zusammengesetzte Zustände (Composite States) oder auch komplexe Zustände. Mithilfe solcher Zustände können z.B. verschiedene Abstraktionsebenen des gleichen Zustandsdiagramms modelliert werden. In zusammengesetzten Zustandsdiagrammen können mehrere Zustände gleichzeitig aktiv sein. Anschaulich ist dies in Abbildung 43 gezeigt. Diese Abbildung ist kein UML-Diagramm, vereinfacht jedoch die Erklärung von komplexen Zuständen. Der abgebildete Baum zeigt die Hierarchie von sieben Zustände. Die verschiedenen Ebenen des Baumes werden als unterschiedliche geometrische Formen dargestellt um die taktile Unterscheidung zu vereinfachen. Eine Legende der Formen ist unterhalb des Baumes angegeben. Die Wurzel des Baumes ist dabei die oberste Ebene des Zustandsdiagramms, als Kreis dargestellt, und enthält den Zustand *„Buchhandlung“*. Innerhalb von diesem Zustand sind entsprechend weiter verschachtelte Zustände enthalten. Die Wurzel hat zwei direkte Kinder: *„Offen“* und *„Geschlossen“*, welche als Rechtecke dargestellt werden. Beide Zustände haben jeweils Unterzustände, welche als Sechsecke dargestellt sind. *„Offen“* hat *„Arbeit“* und *„Pause“* als Kinder und symbolisiert damit die Zustände, in denen ein Mitarbeiter während der Öffnungszeiten der Buchhandlung sein kann. Der Zustand *„Geschlossen“* hat die Kinder *„Verlassen“* und *„Inventur“* und stellt damit die Zustände dar, wenn die Buchhandlung geschlossen ist: entweder ist die Buchhandlung verlassen, oder es wird Inventur gemacht. Die Zustände *„Offen“* und *„Geschlossen“* sind damit selbst wieder zusammengesetzte Zustände. Die jeweiligen beiden Unterzustände der zusammengesetzten Zustände sind einfache Zustände und verzweigen sich nicht weiter. In jeder Hierarchieebene muss immer ein Zustand aktiv sein. Beispielsweise könnten die Zustände *„Offen“* und *„Arbeit“* aktiv sein. Den Zustand *„Buchhandlung“* als Wurzel könnte man ebenfalls als aktiv bezeichnen, dies ist jedoch nicht innerhalb der UML Spezifikation definiert.

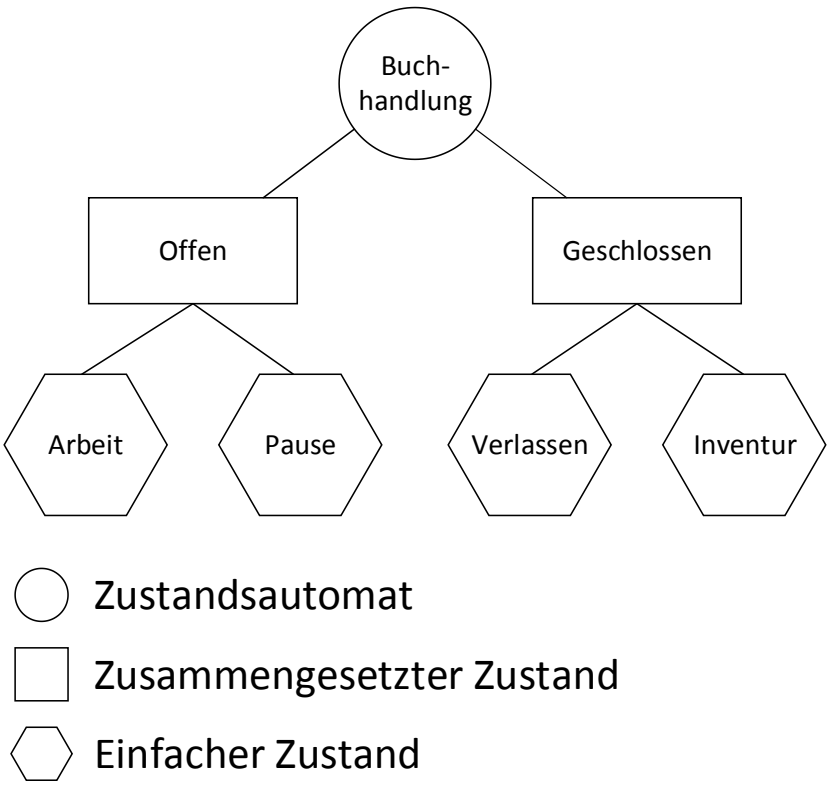


Abbildung 43: Darstellung sieben hierarchischer Zustände als Baum

Abbildung 44 zeigt, wie ein zugehöriges Zustandsdiagramm zum Baum aus Abbildung 43 ohne Transitionen aussehen könnte. Das gesamte Zustandsdiagramm ist in diesem Fall in ein Rechteck eingebettet mit der Bezeichnung des Zustandsdiagramms *„Buchhandlung“* oben links innerhalb eines Pentagons, d.h. er wird durch eine waagerechte Linie unter dem Namen, einer schrägen nach oben laufenden Linie und dann einer senkrechten Linie abgegrenzt. Innerhalb dieses Rechteckes sind die restlichen Zustände angeordnet. Die hierarchischen Zustände *„Offen“* und *„Geschlossen“* beinhalten die zugehörigen Unterzustände. Alle Zustände, zusammengesetzte Zustände und Unterzustände werden dabei genau wie einfache Zustände als Rechteck mit abgerundeten Ecken dargestellt. Unterzustände befinden sich jedoch innerhalb des zugehörigen Oberzustands. Der Name des Oberzustandes ist dabei ähnlich wie bei dem Karteireiter am oberen Rand des Rechteckes dargestellt. Die Unterzustände enthalten dann wieder die Namen direkt mittig. In diesem Beispiel enthält der Zustand *„Offen“* seine Unterzustände *„Arbeit“* und *„Pause“*. *„Geschlossen“* enthält die Zustände *„Verlassen“* und *„Inventur“*.

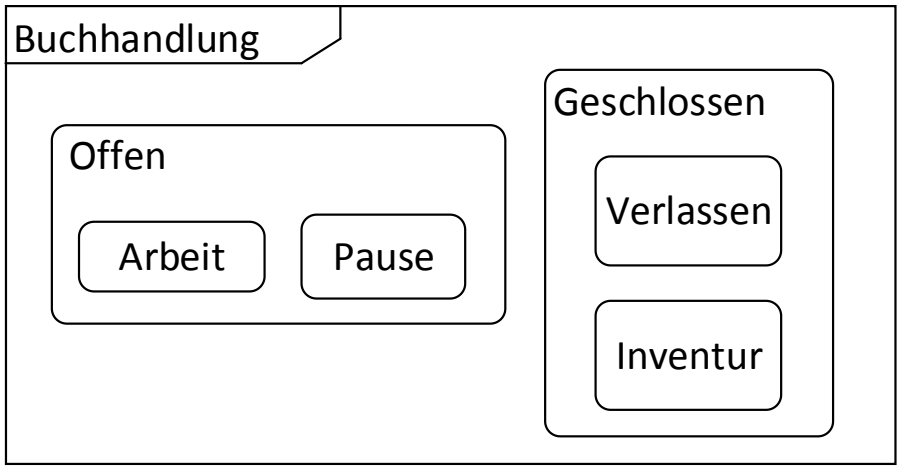


Abbildung 44: Beispielhafte Darstellung der hierarchischen Zustände des Baumes als Zustandsdiagramm

5.3.5.1 Betreten von komplexen Zuständen

Komplexe Zustände können durch eine Transition zum komplexen Zustand oder direkt zu einem eingebetteten Unterzustand betreten werden. Die nachfolgenden Beispiele zeigen diese unterschiedlichen Vorgehensweisen.

Beim Betreten spielt es unter anderem eine Rolle, ob innerhalb des Unterzustandsdiagramms ein Startzustand existiert oder nicht. Start- und Endzustände können innerhalb komplexer Zustände modelliert werden, müssen es aber nicht. Als Beispiel wird das Diagramm aus

Abbildung 44 herangezogen. Der zusammengesetzte Zustand *„Offen“* besitzt folgende Unterzustände: vom *„Startzustand“* gelangt man zum Zustand *„Arbeit“* und von diesem zum Zustand *„Pause“*. Der Zustand außerhalb des zusammengesetzten Zustandes ist der Zustand *„Geschlossen“*. Alle weiteren Transitionen werden bei den jeweiligen Varianten erklärt.

Abbildung 45 stellt den Standard-Eintritt in einen zusammengesetzten Zustand dar. Vom Zustand *„Geschlossen“* zeigt eine Transition auf den Rand des zusammengesetzten Zustands *„Offen“*. Wird diese ausgeführt, wird automatisch der *„Startzustand“* des Zustands *„Offen“* erreicht. Sollte kein Startzustand innerhalb des zusammengesetzten Zustands existieren, bedeutet dies, dass der Zustandsautomat sich in dem zusammengesetzten Zustand befindet, ohne die Unterzustände zu betreten. Dieses Verhalten ist zwar syntaktisch korrekt, sollte jedoch vermieden werden.

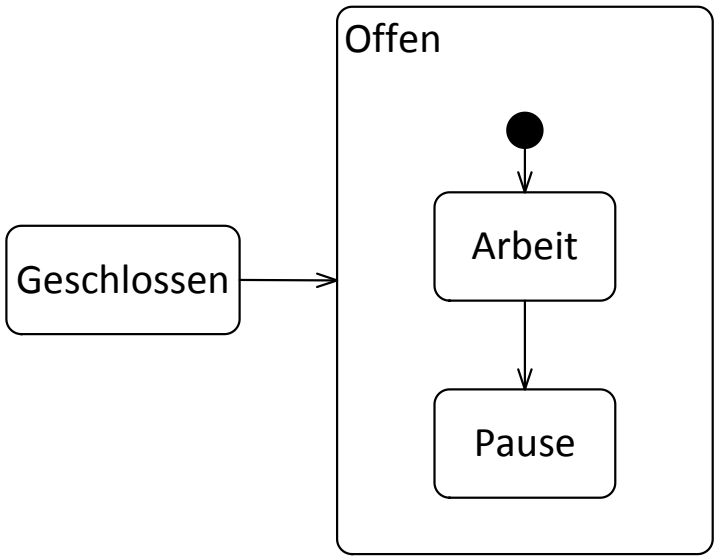


Abbildung 45: Standardeintritt in einen zusammengesetzten Zustand

Abbildung 46 zeigt eine weitere Variante des Standard-Eintritts. Der außerhalb gelegene Zustand *„Geschlossen“* ist in diesem Fall nicht eingezeichnet, da er keine Transition besitzt. Der Zustand *„Pause“* hat jedoch eine zusätzliche Transition auf den Rand des zusammengesetzten Zustands. Diese geht im Vergleich zum vorigen Beispiel vom Inneren des zusammengesetzten Zustands auf den Rand und nicht von einem Zustand außerhalb. Der Effekt dieser Transition ist derselbe wie im Standardfall: Es wird eine Transition auf den *„Startzustand“* des zusammengesetzten Zustands durchgeführt. Dabei wird der zusammengesetzte Zustand nicht verlassen, sodass kein Eintritts- oder Austrittsverhalten (entry- und exit-Aktionen) ausgeführt werden.

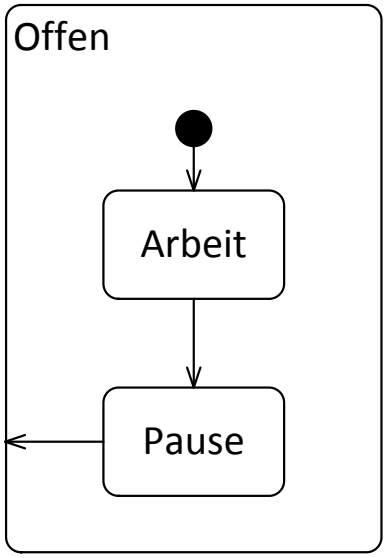


Abbildung 46: Zweite Variante des Standard-Eintritts

Abbildung 47 zeigt das explizite Betreten eines bestimmten Zustands innerhalb eines zusammengesetzten Zustands. In diesem Fall gibt es eine Transition vom außerhalb gelegenen Zustand *„Geschlossen“* direkt zum Zustand *„Pause“*, welcher ein Unterzustand des zusammengesetzten Zustands *„Offen“* ist. Wird diese Transition ausgeführt, wird somit direkt der Zustand *„Pause“* betreten, ohne dass vorher der *„Startzustand“* aktiv wird. Kommen nur explizite Eintrittspunkte in einen zusammengesetzten Zustand vor, kann man auf die Modellierung eines Startzustands verzichten, da dieser niemals betreten werden würde.

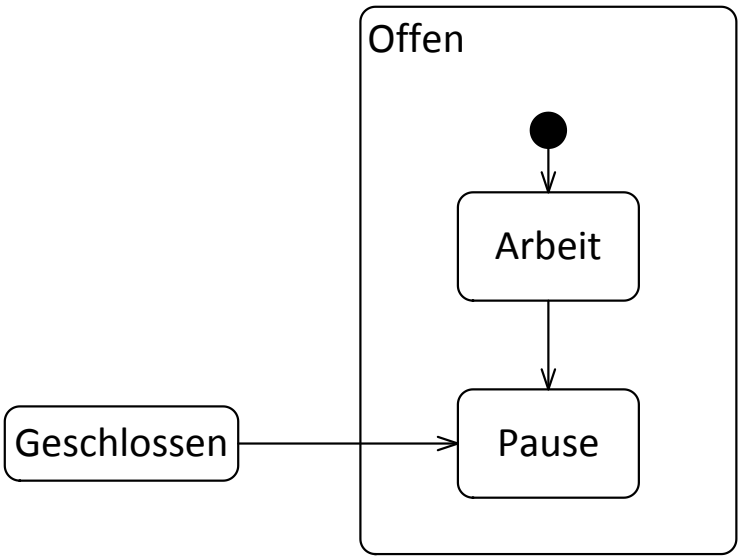


Abbildung 47: Explizites Betreten eines Unterzustands in einem zusammengesetzten Zustand

5.3.5.2 Verlassen eines komplexen Zustands

Genau wie es verschiedene Varianten gibt, um einen zusammengesetzten Zustand zu betreten, gibt es auch mehrere Varianten, diesen zu verlassen. Die drei verschiedenen Varianten sind in den Abbildungen 46 - 48 dargestellt. Alle Diagramme enthalten den identischen zusammengesetzten Zustand *„Offen“*, welcher einen *„Startzustand“* enthält, von welchem man zum Zustand *„Arbeit“* gelangt. Von diesem Zustand kann man einen *„Endzustand“* oder den Zustand *„Pause“* betreten.

Enthält ein zusammengesetzter Zustand einen Endzustand, wird bei dessen Erreichen eine ε-Transition vom Rand des zusammengesetzten Zustands ausgeführt, welche modelliert werden muss. In Abbildung 48 ist dies durch die Transition vom Rand des Zustands *„Offen“* zum außerhalb liegenden Zustand *„Geschlossen“* gegeben. Wird der Endzustand erreicht, ist daher als nächster Zustand *„Geschlossen“* aktiv.

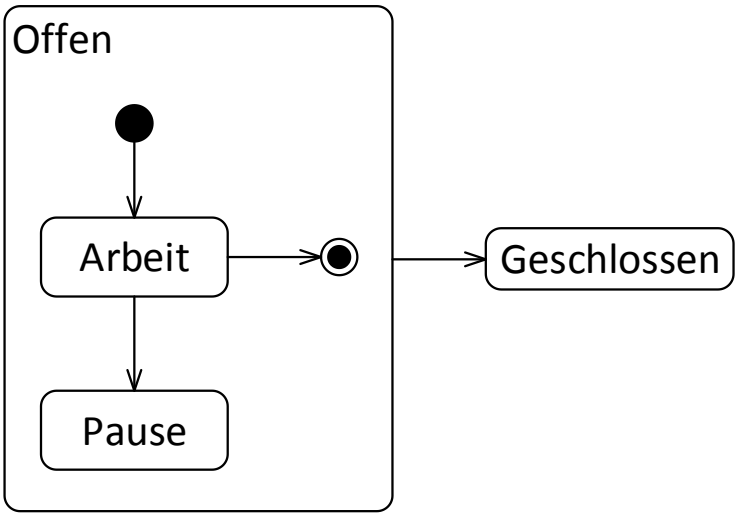


Abbildung 48: Verlassen eines zusammengesetzten Zustands mithilfe eines modellierten Endzustand

Existiert eine Transition vom Rand des zusammengesetzten Zustands mit Events und Guards, kann diese jederzeit von allen Unterzuständen des zusammengesetzten Zustands ausgeführt werden, sobald die entsprechenden Events aufgetreten und Guards erfüllt sind. In Abbildung 49 wird dies durch die Transition mit dem Event und der Aktion *„Feuerausbruch/rausgehen“* zum Zustand *„Abgebrannt“* modelliert. In diesem Fall wird wiederum der zusammengesetzte Zustand *„Offen“* inaktiv und der Zustand *„Abgebrannt“* wird aktiv.

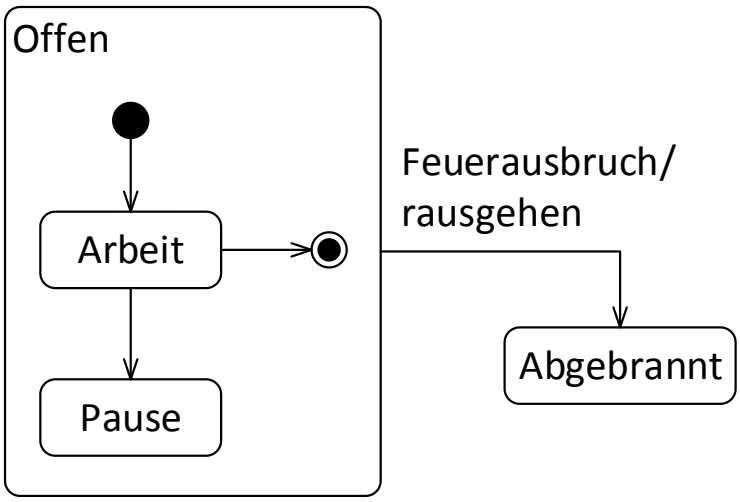


Abbildung 49: Verlassen eines zusammengesetzten Zustands mithilfe eines Events

Die letzte Möglichkeit besteht aus einer Transition von einem der Unterzustände zu einem Zustand außerhalb des zusammengesetzten Zustands. Dieser Fall, dargestellt in Abbildung 50, ist vom Unterzustand *„Pause“* zum außerhalb liegenden Zustand *„Feierabend“* gegeben. Auch diese Transition kann Events, Guards und Aktionen modellieren. Sobald die Bedingungen der Transition erfüllt wären, könnte diese daher vom aktiven Zustand *„Pause“* ausgeführt werden. Im Beispiel sind jedoch keine Events oder Guards modelliert, sodass diese Transition immer vom Zustand *„Pause“* durchgeführt werden kann. Wie in den anderen Fällen wird der zusammengesetzte Zustand dadurch inaktiv und der Zustand *„Feierabend“* wird aktiv.

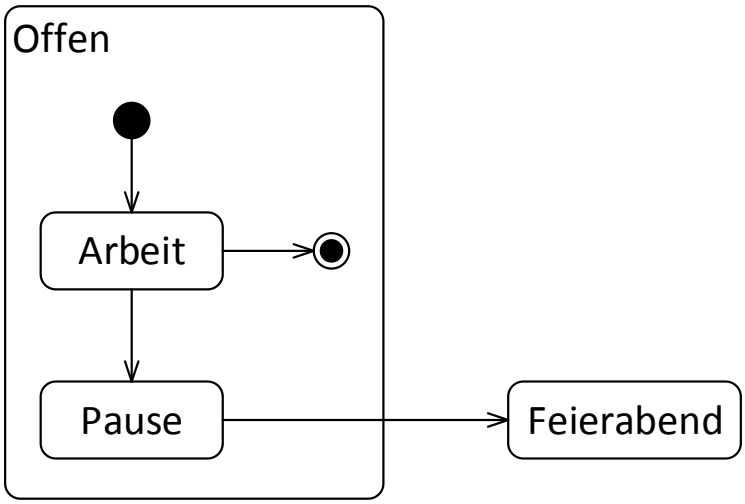


Abbildung 50: Direktes Verlassen eines zusammengesetzten Zustands

5.3.5.3 Orthogonale Zustände

Zustände können nicht nur weitere Zustände enthalten, sondern können ebenso in mehrere Regionen aufgeteilt werden, welche parallel ausgeführt werden. Solche Zustände nennt man

orthogonale Zustände. Diese gehören ebenfalls zu den zusammengesetzten Zuständen. Die Regionen werden dabei mithilfe von waagrecht gestrichelten Linien voneinander getrennt. Jede Region kann dabei ein vollständiges Zustandsdiagramm enthalten. Transitionen zwischen den einzelnen Regionen sind nicht erlaubt. Wird ein Zustand, welcher mehreren Regionen enthält, betreten, werden die Startzustände aller Regionen aktiv und führen nebenläufig die spezifizierten Abläufe durch. Ein orthogonaler Zustand wird erst verlassen, wenn alle Regionen ihren Endzustand erreicht haben. Falls ein modellierter Terminator erreicht wird, wird jedoch der gesamte orthogonale Zustand sofort verlassen. Ein Beispiel ist in Abbildung 51 gegeben. Der zusammengesetzte Zustand heißt *„In Buchhandlung“* und besteht aus den beiden Regionen *„Geist“* und *„Körper“*. Die Namen der Regionen werden oben links in den Bereich geschrieben. Die Regionen im Beispiel unterscheiden zwischen den unterschiedlichen Zuständen, in denen sich der Besucher einer Buchhandlung geistig und körperlich befindet. Die Region *„Geist“* enthält einen *„Startzustand“,* von dem man in den Zustand *„Suchend“* gelangt. Von diesem gelangt man sofort in den *„Endzustand“*. Die Region *„Körper“* besteht ebenfalls aus einem *„Startzustand“*. Von diesem gelangt man jedoch in den Zustand *„Gehend“* und danach in den *„Endzustand“*.

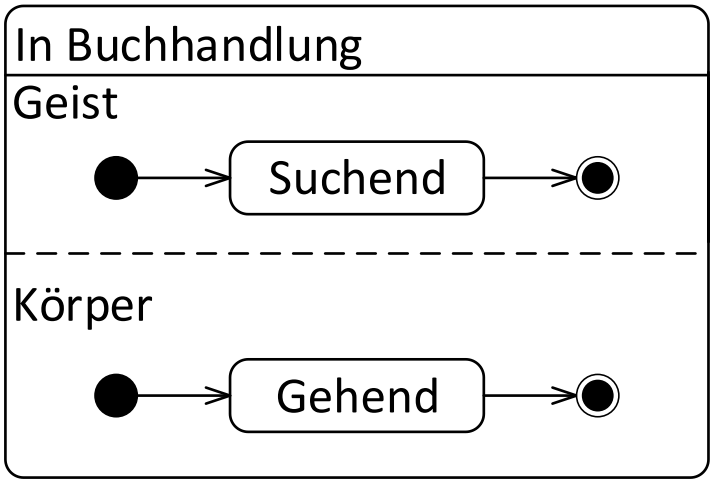


Abbildung 51: Orthogonaler Zustand „In Buchhandlung“ mit zwei Regionen „Geist“ und „Körper“

5.3.6 Beispiel eines Zustandsdiagramms für eine Navigations-App

Das Zustandsdiagramm für das Navigationsbeispiel in Abbildung 52 zeigt verschiedene Zustände, welche das Navigationsgerät erreichen kann. In diesem Fall wird zwischen den beiden Zuständen unterschieden, ob das Navigationsgerät ein- oder ausgeschaltet ist. Entsprechend besteht das Diagramm aus dem einfachen Zustand *„Navi Aus“* und dem komplexen Zustand *„Navi Ein“*. Vom *„Startzustand“* gelangt man zu einem einfachen Zustand *„Navi Aus“*. Vom Zustand *„Navi Aus“* gelangt man mittels der Transition *„ein“* zum Zustand *„Navi Ein“*. Mithilfe der Transition *„aus“* kann man umgekehrt den Zustand wechseln. Durch die Transition *„ein“* wird der Startzustand des komplexen Zustands betreten. Von diesem gelangt man automatisch in den Zustand *„Kein GPS Signal“*. Steht ein GPS Signal zur Verfügung, wird die Transition *„GPS empfangen“* ausgeführt und der Zustand *„GPS Signal“* wird betreten. Bricht in diesem Zustand der Empfang des Signals ab oder wird ausgeschaltet, wird die Transition und die Aktion *„Kein GPS Signal erreichbar/Hinweis ausgeben“* ausgeführt, und man gelangt wieder in den Zustand *„Kein GPS Signal“*.

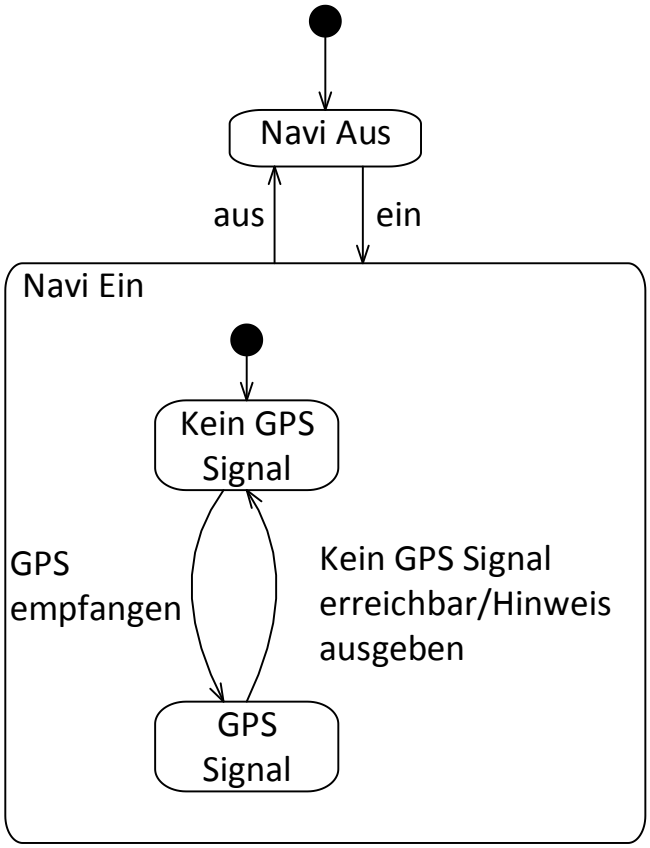


Abbildung 52: Zustandsdiagramm des Navigationsbeispiels

5.3.7 Verständnisfragen zum Zustandsdiagramm

Dieses Kapitel besteht aus fünf Multiple-Choice-Fragen (Business Informatics Group, 2015), um das Verständnis für das Zustandsdiagramm zu überprüfen. Jede Frage hat vier Antwortmöglichkeiten a–d. Bei jeder Frage können eine, zwei, drei oder vier Antworten richtig sein. Die Auflösung ist in Anhang A und im Detail in Anhang A.4 zu finden.

1. Bei einem aktiven orthogonalen Zustand…
   (a) ist immer genau ein Subzustand aktiv.
   (b) ist mindestens ein Subzustand in jeder Region aktiv.
   (c) kann auch kein Subzustand aktiv sein.
   (d) sind Transitionen zwischen den Regionen verboten.
2. In einem Zustandsdiagramm können folgende Elemente modelliert werden:
   (a) In den Startzustand eingehende Transitionen.
   (b) Events, die Zustandsübergänge auslösen.
   (c) Mögliche Zustandsübergänge von einem Zustand zum anderen.
   (d) Vom Endzustand ausgehende Transitionen.
3. Mit welchen Angaben kann eine Transition beschriftet sein?
   (a) Mit einer Aktivität, die während der Transition auszuführen ist.
   (b) Mit dem Event, das nach der Transition ausgeführt werden soll.
   (c) Mit dem Event, das während der Transition ausgeführt wird.
   (d) Mit dem Event, das die Transition auslöst.
4. Welche der folgenden Aussagen über Zustandsdiagramme bzw. deren Inhalt sind korrekt?
   (a) Pro Zustandsdiagramm kann es mehrere Endzustände geben.

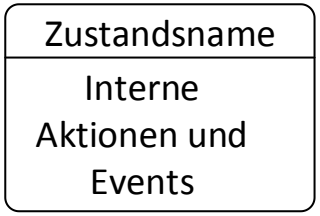
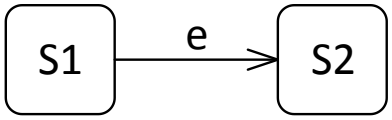
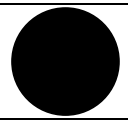
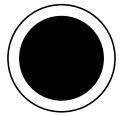
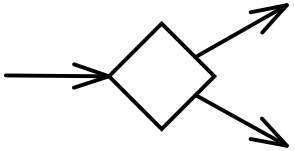
(b) An einer Transition können Bedingungen, Events und Zustände angegeben werden.
(c) Der Startzustand besitzt genau eine ausgehende und beliebig viele eingehende Transitionen.
(d) Events lösen Transitionen aus.

5. In welcher Syntax wird die Beschriftung einer Transition angeschrieben?
(a) [Aktivität] Event /Bedingung
(b) [Bedingung]Aktivität/ Event
(c) Aktivität[Bedingung]/ Event
(d) Event [Bedingung]/Aktivität

5.3.8 Modellierungsaufgabe Zustandsdiagramm

Zu Beginn des Lebens ist der Mensch gesund. Falls er jedoch zu oft im Regen spaziert, kann er sich erkälten. Ruhe sollte dabei helfen eine Erkältung loszuwerden. Manche Personen gehen jedoch weiterhin arbeiten, sodass die Erkältung eine ernsthafte Krankheit nach sich zieht. Da sollte man schnell eine Arznei zu sich nehmen. Schlägt diese gut an, wird man wieder gesund. Ansonsten kann die Krankheit fortdauern. Hoffentlich führt diese nicht zu einem bösen Ende. Eine Beispiellösung findet sich in Anhang B und als taktile Grafik auf Seite 64 der taktilen Mappe.

5.3.9 Tabellarische Übersicht der Elemente des Zustandsdiagramms

Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Zustand	Beschreibt den aktuellen Zustand des Systems. Kann vordefinierte Aktivitäten beinhalten, die z.B. beim Betreten, Verlassen oder während des Zustands ausgeführt werden.	Rechteck mit abgerundeten Ecken. Zwei Teile bei internen Aktivitäten: oben steht der Name *„Zustandsname“*, unten durch eine waagrechte Linie getrennt die Aktionen oder Events. Hier als Platzhalter nur *„Interne Aktionen und Events“*.	
Transition	Zustandsübergang zwischen verschiedenen Zuständen	Eine Transition ist eine Linie mit offener Pfeilspitze an einem Ende und optionalen Beschriftungen ober-/unterhalb der Linie. Hier verbindet sie zwei Rechtecke *„S1“* und *„S2“*. Event *„e“* steht oberhalb der Linie.	
Startzustand	Startpunkt eines Zustandsdiagramms	Schwarzer Kreis	
Endzustand	Endpunkt eines Zustandsdiagramms (Objekt kann ewig im Endzustand existieren)	Weißer Kreis mit kleinerem schwarzen Kreis darin enthalten.	
Terminator	Terminierung des Zustandsdiagramms und Löschung des modellierten Objektes	Ein großes X	
Verzweigung	Entscheidungsknoten von dem mehrere alternative Transitionen ablaufen können	Ein weißer Diamant mit einer eingehenden und mindestens zwei ausgehenden Transitionen.	

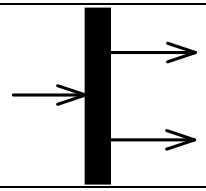
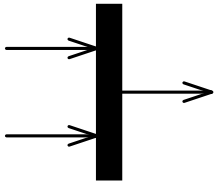
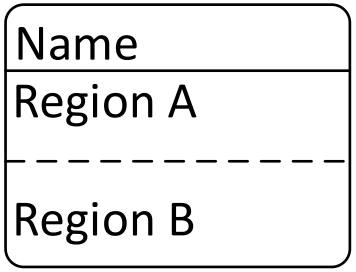
Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Parallelisierung	Aufteilung einer Transition in mehrere parallele Transitionen	Ein schwarzer Balken mit einer eingehenden und mindestens zwei ausgehenden Transitionen.	
Synchronisierung	Zusammenführung mehrerer paralleler Transitionen in eine Transition	Ein schwarzer Balken mit mindestens zwei eingehenden und einer ausgehenden Transitionen.	
Orthogonaler Zustand	Aufteilung eines Zustandes in mehrere parallel ausgeführte Regionen	Abgerundetes Rechteck als Zustand. Waagrechte Linie trennt Name von Inhalt. Regionen werden durch gestrichelte waagrechte Linien abgegrenzt. Die Namen der Regionen stehen oben links innerhalb der Region. Hier „*Region A*“ und „*Region B*“.	

Tabelle 4: Kurzreferenz der Elemente des Zustandsdiagramms

5.4 Sequenzdiagramm

Die taktilen Grafiken des Sequenzdiagramms befinden sich auf den Seiten 43–60 der taktilen Mappe

Sequenzdiagramme gehören zur Kategorie der Interaktionsdiagramme innerhalb der Verhaltensdiagramme. Wie das Aktivitäts- und Zustandsdiagramm beschreibt es Abläufe und keine Strukturen, es konzentriert sich jedoch auf den Nachrichtenfluss zwischen verschieden Objekten, d.h. wie diese miteinander interagieren. Sequenzdiagramme zeigen den zeitlichen Ablauf von Nachrichten zwischen Objekten, geben jedoch keine Informationen über die Beziehungen zwischen diesen. Es werden demnach nicht alle möglichen Ablaufpfade von Nachrichten modelliert, sondern ein konkretes, mögliches Szenario von Nachrichtenverläufen. Ein Sequenzdiagramm kann beispielsweise ein Klassendiagramm näher beschreiben, indem jede Klasse als Teilnehmerobjekt des Sequenzdiagramms modelliert wird. Zugehörige Methoden der Klassen sind Nachrichten zwischen den Teilnehmern. Es kann ebenfalls zur Präzisierung der Vorgänge des Aktivitätsdiagramms oder zur Darstellung der Interaktion zwischen Akteuren und Anwendungsfällen verwendet werden.

5.4.1 Lebenslinie und Objekte

Die Hauptkomponenten des Sequenzdiagramms sind die Teilnehmer und der zeitliche Ablauf der Interaktion zwischen Teilnehmern. Die Teilnehmer werden horizontal nebeneinander angeordnet, die Zeit verläuft vertikal. Ein menschlicher Teilnehmer wird dabei als Strichmännchen dargestellt, ein Systemteilnehmer oder Objekt als Rechteck, wie bereits bei den Anwendungsfällen eingeführt. Die Symbole der Teilnehmer werden als Kopf der Lebenslinie bezeichnet. Alle Teilnehmer können mit einem Namen und Typ identifiziert werden. Der Name steht dabei getrennt durch einen Doppelpunkte vor dem Typ. In Abbildung 53 sind beide Varianten dargestellt. Der menschliche Teilnehmer trägt den Namen „*Alice*“ und ist vom Typ „*Kunde*“. Der Systemteilnehmer hat keinen Namen spezifiziert, sondern nur den Typ „*:System*“. Der Name von Teilnehmern ist optional, jedoch muss der Typ zwingend angegeben werden. Wird kein Name vergeben, wir dennoch ein Doppelpunkt vor den Typ des Teilnehmers gestellt. Um den zeitlichen Verlauf darzustellen werden Lebenslinien von jedem Teilnehmer ausgehend modelliert. Eine Lebenslinie verläuft dabei vertikal und wird als gestrichelte Linie dargestellt, wie in Abbildung 53 ebenfalls zu sehen ist.

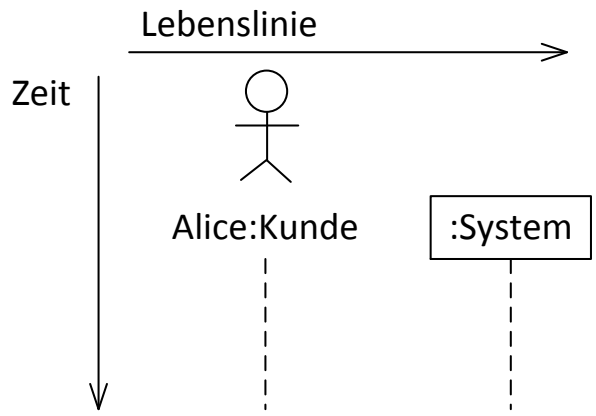


Abbildung 53: Die Zeit- und Interaktionsachse mit zwei Teilnehmern

Die Lebenslinien stellen die passive Lebenszeit eines Teilnehmers dar. Die aktive Teilnahme (Aktivitätsbereich) wird durch Ausführungsbalken dargestellt. Ein Ausführungsbalken wird dabei als schmales Rechteck über die Lebenslinie gelegt, wie in Abbildung 54 dargestellt. Eine einfache Darstellung eines Aktivitätsbereichs stellt der Teilnehmer „*Alice:Kunde*“ dar. Alice hat

einen einfachen Ausführungsbalken auf der Lebenslinie angelegt. Sie ist zuerst passiv, dann aktiv und danach wieder passiv, da der Ausführungsbalken endet. „*Bob:Kunde*“ dagegen besitzt übereinanderliegende Ausführungsbalken. Dabei wird der zweite Balken leicht versetzt über dem ersten dargestellt. Diese Darstellung indiziert eine parallele Tätigkeit des Teilnehmers. „*Bob:Kunde“* ist ebenfalls zuerst passiv, dann aktiv, dann führt er kurze Zeit eine Tätigkeit parallel aus bevor beide Ausführungsbalken beendet werden und er wieder passiv agiert. Die Beendigung einer Lebenslinie während der Interaktion kann durch ein Stopp-Symbol dargestellt werden, welches durch ein „X“ auf der Lebenslinie dargestellt wird. In Abbildung 54 ist dies bei dem Teilnehmer „*Carol:Kunde*“ dargestellt. Die Beendigung einer Lebenslinie bedeutet die Zerstörung eines Teilnehmers. Der Teilnehmer kann sich danach nicht mehr an der Interaktion beteiligen.

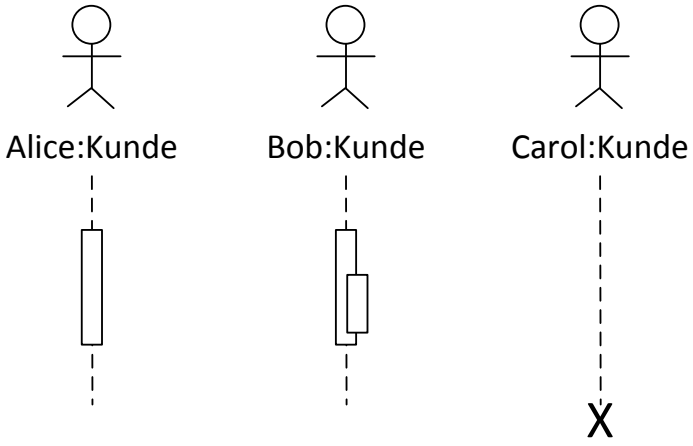


Abbildung 54: Drei Varianten der Lebenslinien mit Ausführungsbalken und Beendigung

5.4.2 Zustandsinvariante

Mithilfe der Zustandsinvariante kann ein Zustand modelliert werden, welcher ein Teilnehmer erfüllen muss, um an der Interaktion teilnehmen zu können. Die Zustandsinvariante wird immer vor den folgenden Interaktionen geprüft. Es gibt drei Varianten zur Darstellung der Invariante, welche alle in Abbildung 55 dargestellt sind. Alle drei Beispiele spezifizieren die gleiche Invariante: „*befüllt == true*“. Die erste Variante besteht aus einem Rechteck mit abgerundeten Ecken in welchem die Invariante spezifiziert wird. Das Rechteck wird mittig auf die Lebenslinie positioniert. Die zweite Variante wird ebenfalls mittig platziert besteht jedoch nur aus einem Text ohne Umrahmungen. Die Invariante wird hierbei in geschweifte Klammern gesetzt. Die dritte Möglichkeit besteht aus einer Notiz an der Lebenslinie. Die Invariante wird dabei innerhalb eines Notizzettels (Rechteck mit Eselsohr) dargestellt und mithilfe einer gestrichelten Linie mit der Lebenslinie verbunden für welche diese gelten soll. Die Verbindung mit der Lebenslinie definiert dabei den Zeitpunkt für die Prüfung der Invariante.

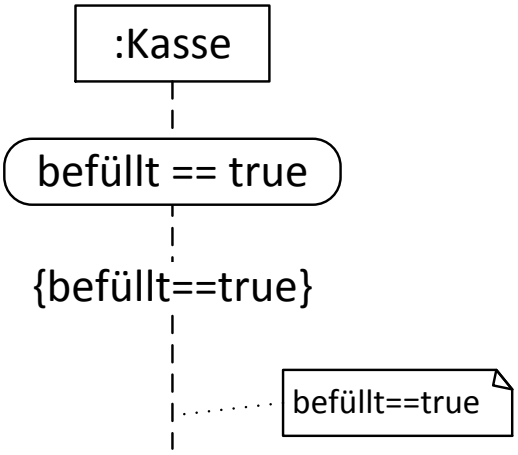


Abbildung 55: Die Zustandsinvariante von Lebenslinien

5.4.3 Nachrichten

Nachrichten stellen die Hauptkomponente eines Sequenzdiagramms dar. Mit ihnen werden die Kommunikation und deren Richtung zwischen Teilnehmern definiert. Nachrichten werden als waagerechte Pfeile zwischen den Teilnehmern dargestellt. Dabei können sowohl die Linienart (einfach, gestrichelt), als auch die Pfeilspitze (gefülltes Dreieck, offen) je nach Kontext variieren. Die folgenden Unterkapitel erklären einzeln die verschiedenen Bedeutungen. Nachrichten sind oberhalb der Linie mit ihrem Namen beschriftet und können optional nummeriert werden.

5.4.3.1 Synchrone und Asynchrone Nachrichten

Nachrichten können sowohl synchron als auch asynchron sein. Der Unterschied zwischen einer synchronen und asynchronen Nachricht besteht in der Verarbeitung der Nachrichten. Bei einer synchronen Nachricht blockiert die Lebenslinie des Senders bis er eine Antwort des Empfängers erhalten hat. Erst danach kann er weitere Nachrichten senden. Bei einer asynchronen Nachricht ist diese nicht der Fall und der Sender kann ohne Unterbrechung mit seinen Tätigkeiten fortfahren. Eine synchrone und eine asynchrone Nachricht zwischen einem Teilnehmer „*:Kunde*“ und „*:Mitarbeiter*“ sind in Abbildung 56 dargestellt. Eine synchrone Nachricht wird durch eine ausgefüllte Pfeilspitze dargestellt. Der Pfeil zeigt dabei von „*:Kunde*“ zu „*:Mitarbeiter*“ und ist mit „*synchron rufen*“ beschriftet. Die asynchrone Nachricht ist mit einer offenen Pfeilspitze dargestellt und mit „*asynchron rufen*“ beschriftet. In beiden Fällen handelt es sich um eine einfache Linie.

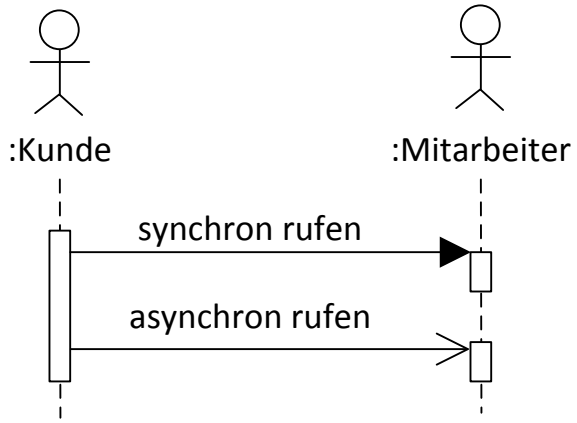


Abbildung 56: Darstellung von synchroner und asynchroner Kommunikation zwischen einem Kunden und einem Mitarbeiter

Nachrichten müssen nicht zwingend an andere Teilnehmer gerichtet sein. Es ist ebenfalls möglich Nachrichten an sich selbst zu senden. Abbildung 57 zeigt ein solches Szenario, in welchem *„Alice:Kunde“* die Nachricht *„Buch wählen“* an sich selbst verschickt. Die eingehende Nachricht zeigt dabei auf einen parallelen Ausführungsbalken des Teilnehmers.

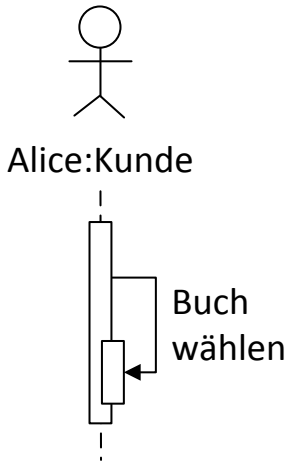


Abbildung 57: Reflexive Nachricht "Buch wählen"

Bei synchronen Nachrichten wartet der Sender auf eine Antwortnachricht des Empfängers, bevor er mit anderen Tätigkeiten fortfährt. Nach UML-Spezifikation muss diese Antwortnachricht jedoch nicht immer explizit modelliert werden, sondern ist optional. Eine Antwortnachricht zeigt vom Empfänger der synchronen Nachricht zum Sender und wird als gestrichelte Linie mit einer offenen oder gefüllten Pfeilspitze dargestellt, wie in Abbildung 58 dargestellt. Die Bezeichnung der Antwortnachricht ist identisch mit dem Namen der initialen Nachricht beschriftet, um die Zugehörigkeit darzustellen. In dem abgebildeten Beispiel sendet ein „*:Kunde*“ die synchrone Nachricht „*rufen*“ an einen „*:Mitarbeiter*“. Der Mitarbeiter antwortet ebenfalls mit der Nachricht „*rufen*“.

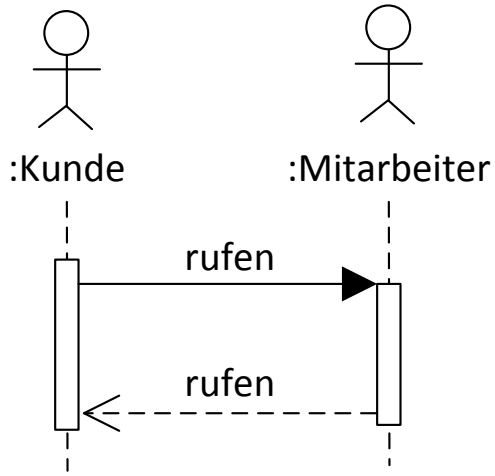


Abbildung 58: Die Antwortnachricht

5.4.3.2 Verlorene und gefunden Nachrichten

Nachrichten können ebenfalls ohne Empfänger oder Sender modelliert werden. Eine sogenannte verlorene Nachricht wird zwar gesendet, ihr Empfänger wird jedoch nicht modelliert. Anstelle des Empfängers wir ein gefüllter Kreis an das Pfeilende gesetzt, um anzudeuten, dass der Empfänger nicht spezifiziert ist. Dies kann der Fall sein, wenn der Empfänger zum Modellierungszeitpunkt unbekannt ist, außerhalb des Diagrammkontextes liegt oder dass diese

Nachricht tatsächlich nie ankommt. Eine gefundene Nachricht dagegen wird empfangen, der Absender ist jedoch unbekannt. Dies wird modelliert, indem die Nachricht von einem gefüllten Kreis ausgeht. Gefundenen Nachrichten können verwendet werden, falls der Sender außerhalb des Diagrammkontextes liegt, irrelevant ist oder einfach nicht modelliert werden soll (Kecher & Salvanos, 2015). Ein Beispiel ist in Abbildung 59 gegeben. Der Teilnehmer „:*Kunde*“ erhält eine gefundene Nachricht „*Geschenk kaufen*“. Da er sich nicht sicher ist, was er kaufen soll fordert er eine Beratung von einem „:*Mitarbeiter*“ an mittels der asynchronen Nachricht „*Beratung anfordern*“. Der „:*Mitarbeiter*“ möchte diesem Wunsch jedoch nicht nachkommen und ignoriert den Kunden. Dies wird mit der verlorenen Nachricht „*ignorieren*“ modelliert.

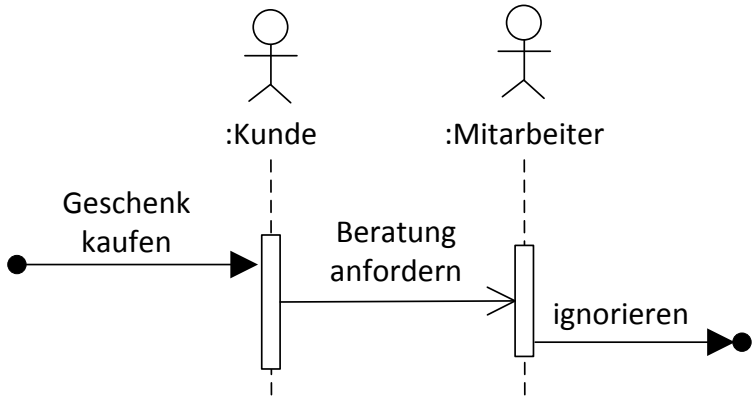


Abbildung 59: Gefundene und verlorene Nachrichten

5.4.3.3 Zeitbedingungen von Nachrichten

Bisher wurde angenommen, dass das Versenden einer Nachricht keine Zeit verbraucht, d.h. die Dauer beträgt Null. Es ist jedoch möglich die Zeitdauer einer Nachricht oder Zeitbedingungen zwischen Nachrichten zu modellieren. Vier Möglichkeiten sind in der nachfolgenden Abbildung 60 dargestellt (Kecher & Salvanos, 2015). Alle Nachrichten sind von einem Teilnehmer „:*Kunde*“ an einen Teilnehmer „:*Mitarbeiter*“ gerichtet.

rufen d=dauer: Bei der ersten Nachricht wir mittels der Angabe „*d=dauer*“ die Dauer der Übermittlung gemessen und dem Attribut „*d*“ zugewiesen.

rufen{0..1 sec.}: Diese Zeitdarstellung indiziert, dass das Übermitteln der Nachricht höchstens 1 Sekunde dauern darf.

rufen: Bei dieser Nachricht wird der Zeitpunkt der Sendung der Nachricht und des Eintreffens beim Empfänger explizit angegeben. Mithilfe der Annotation „*{t=now}*“ an der Lebenslinie des Empfängers wird der Startpunkt angegeben. Der Zeitpunkt des Eintreffens liegt 1 Sekunde später. Dies wird ersichtlich aus der Annotation „*{t+1 sec.}*“ an der Lebenslinie des Empfängers. Zusätzlich wird hier der Nachrichtenpfeil nicht waagerecht dargestellt, sondern schräg nach unten, um den Zeitverlauf zusätzlich zu betonen. Der Ausführungsbalken des Empfängers bleibt jedoch an der Stelle an der die Nachricht ohne Zeitverzögerung ankommen würde. In diesem Fall liegt dieser demnach oberhalb der Berührung der Pfeilspitze mit der Lebenslinie des Empfängers.

rufen/anschreien: Die letzten beiden Nachrichten stellen eine Zeitbedingung zwischen den Nachrichten dar. Die Nachrichten selbst werden waagerecht modelliert. An der Lebenslinie des Senders werden jedoch zwei Markierungen angebracht. Ein waagerechter Strich beim Senden der ersten Nachricht „*rufen*“ und einer beim Senden der zweiten Nachricht „*an-*

schreien“. Zwischen diesen beiden Markierungen wird die Zeitbedingung notiert, welche maximal zwischen dem Senden der beiden Nachrichten vergehen darf. In diesem Fall ist dies maximal 5 Minuten. Die Zeitbedingung wird als Intervall in geschweifte Klammern gesetzt: *„{0..5 min.}“*. Zwei Pfeile zeigen von der Zeitbedingung jeweils zu den waagrechten Linien auf der Lebenslinie, d.h. einer zeigt nach oben, der andere nach unten.

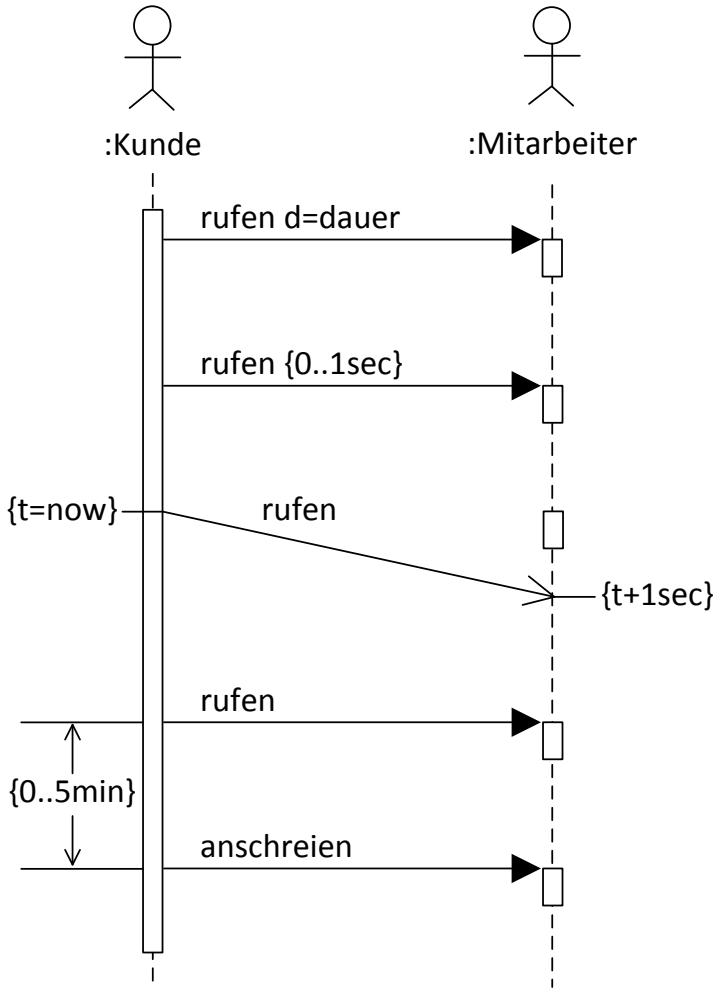


Abbildung 60: Modellierung von Zeitbedingungen von Nachrichten

5.4.3.4 Argumente von Nachrichten

Sendenachrichten können nicht nur mit einfachen Namen beschriftet werden, sondern können zusätzliche Argumente übergeben bekommen. Diese Angaben sind jedoch optional. Argumente werden in runden Klammern hinter den Namen gestellt und durch Kommas getrennt. Jedes Argument kann einen „Parameter-Namen“ haben, welchem einen Wert mittels einem „=“ zugewiesen wird. Dieser Wert kann ein spezifizierter „Argument-Wert“ sein, ein Bindestrich, um einen unspezifizierten Wert darzustellen oder aus einem Stern bestehen, welcher eine beliebige, möglicherweise noch nicht spezifizierte Nachricht modelliert. Beispiele für verschiedene Varianten sind in Abbildung 61 (Kecher & Salvanos, 2015) dargestellt. Die Nachrichten gehen dabei wieder von einem „:*Kunden*“ zu einem „:*Mitarbeiter*“.

- **rufen:** Eine normale Nachricht mit einem Namen und keinen Argumenten.
- **Preise addieren(10,8):** Eine Nachricht mit einem Namen und zwei Zahlen als Argumenten, in runden Klammern angegeben.

- **Addieren(sum1=10, sum2=8):** Bei dieser Nachricht werden die Argument-Werte, „10“ und „8“, explizit Parameter-Namen, „sum1“ und „sum2“, zugewiesen. Die Zuweisung erfolgt durch ein Gleichheitszeichen.
- **Buch bestellen (titel=-):** Eine Nachricht mit einem Namen und einem Parameter-Namen. Der Wert des Arguments „*titel*“ ist jedoch nicht angegeben, sondern wird durch einen Bindestrich als nicht spezifiziert modelliert.
- **Buch bestellen(-):** Die Nachricht besitzt einen Namen, Argumente der Nachricht sind jedoch nicht spezifiziert, dargestellt durch einen Bindestrich.
- ***:** Nachricht, welche vollständig unspezifiziert ist. Weder ein Name, noch Argumente sind angegeben.

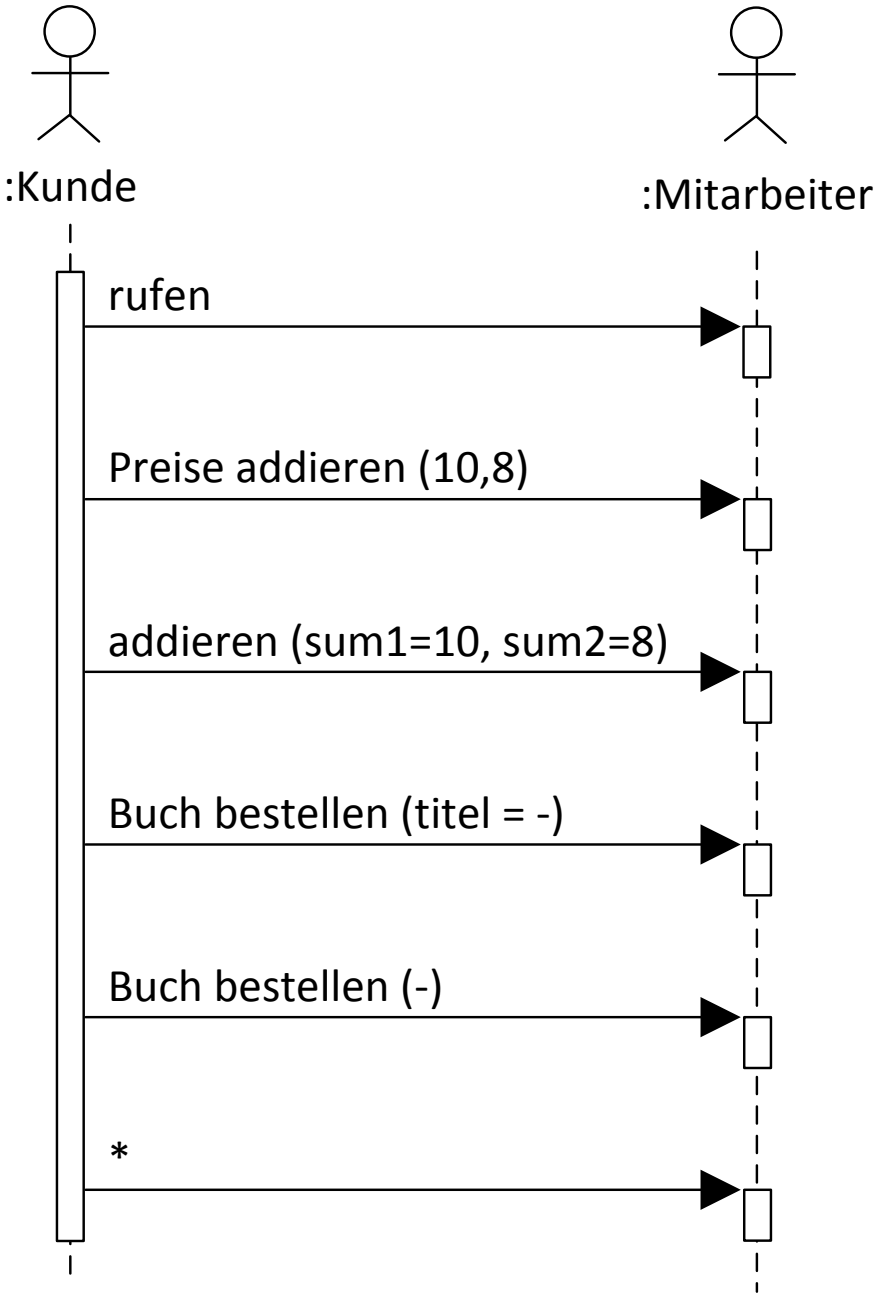


Abbildung 61: Argumentübertragung bei Nachrichten

5.4.3.5 Rückgabewerte bei Nachrichten

Ebenso wie die Sendenachrichten, können auch Antwortnachrichten mit optionalen Angaben versehen werden. Antwortnachrichten wiederholen die Bezeichnung der Sendenachricht entweder identisch oder mit zusätzlichen Rückgabewerten. Ein Rückgabewert wird durch einen Doppelpunkt getrennt hinter die Bezeichnung der Argumente notiert. Wie bei den Sendenachrichten gibt es auch hier mehrere Optionen, welche in Abbildung 62 dargestellt werden. Insgesamt werden 5 Nachrichten mit Antworten zwischen den Teilnehmern „:*Kunde*“ und „:*Mitarbeiter*“ ausgetauscht (Kecher & Salvanos, 2015). Die Nachrichten von Sender und Empfänger werden nachfolgend aufgelistet und in den Zeilen danach erklärt.

- **Sender: Preise addieren (10,8), Antwort: Preise addieren(10,8):18**
  In diesem Fall enthält die Antwortnachricht einen einfachen Rückgabewert, welcher mithilfe eines Doppelpunkts hinter die ursprüngliche Nachricht gesetzt wird. Der Rückgabewert gibt hierbei den Wert der Funktion „Preise addieren(10,8)“ an.
- **Sender: Preise addieren (10,8), Antwort: sum=Preise addieren(10,8)**
  Die Antwortnachricht wird einem Attribut zugewiesen. Das Attribut wird dabei der ursprünglichen Nachricht vorangestellt und mittels einem Gleichheitszeichen zugewiesen.
- **Sender: Preise addieren (10,8, out sum), Antwort: Preise addieren(10,8, out sum:18)**
  Es kann ein expliziter Ausgabeparameter spezifiziert werden, welchem bei der Antwortnachricht mittels eines Doppelpunktes ein Wert zugewiesen wird. Der Ausgabeparameter wird mittels des Schlüsselworts „*out*“ und einem Namen erstellt und ist in beiden Nachrichten enthalten. Der Sender enthält nur die Definition, die Antwort dagegen weist dem Parameter mithilfe eines Doppelpunktes einen Wert zu.
- **Sender: Preise addieren (10,8), Antwort: Preise addieren(-,-):18**
  Falls nur der Rückgabewert interessant ist, können bei einer Antwortnachricht die Argumente, welche mit übergeben wurden, durch einen Bindestrich ersetzt werden. Damit sind diese Argumente nicht länger spezifiziert.
- **Sender: *, Antwort: ***
  Die Antwortnachricht ist, ebenso wie die Sendenachricht, nicht spezifiziert.

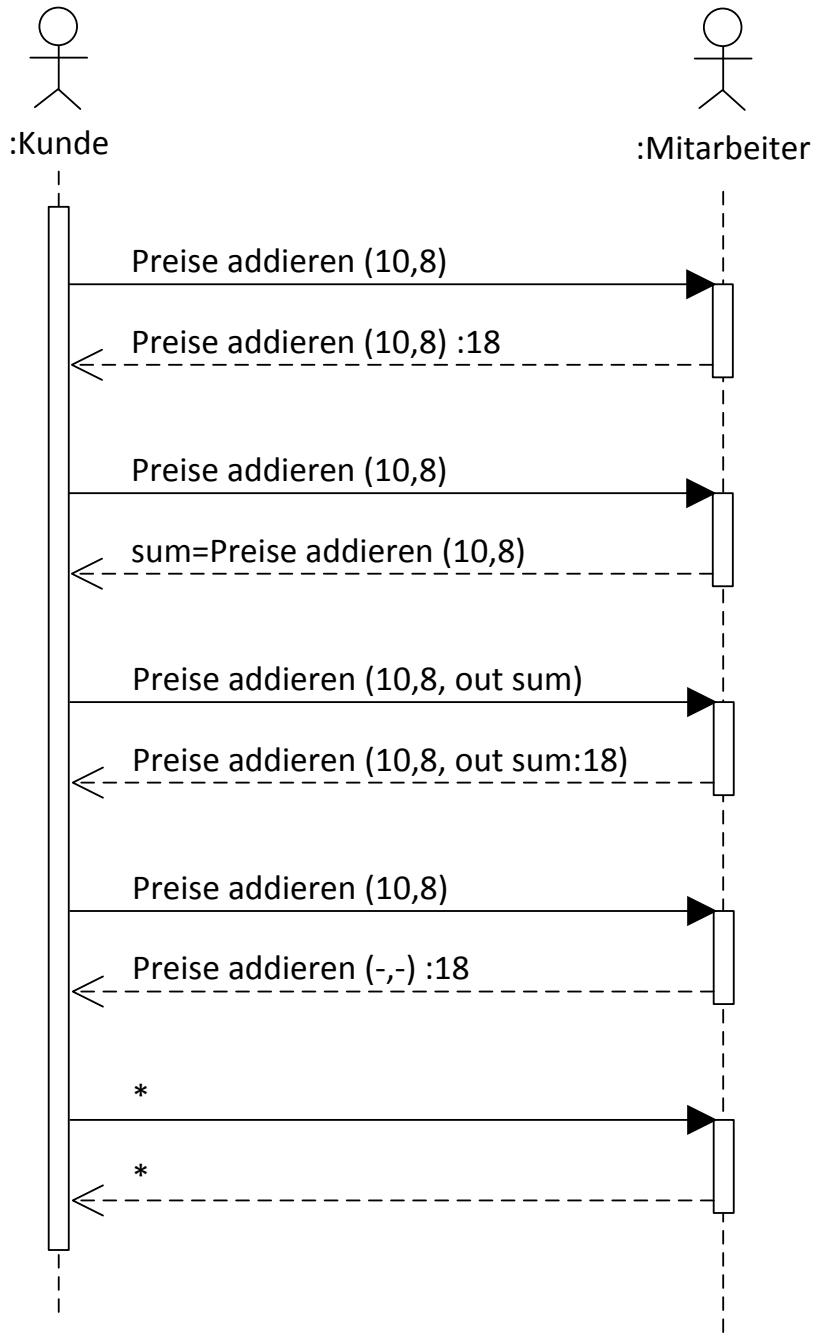


Abbildung 62: Rückgabewerte von Nachrichten mit Argumenten

5.4.4 Erzeugen und Zerstören von Teilnehmern

Teilnehmer können während der Interaktion aktiv von anderen Teilnehmern erzeugt oder zerstört werden. Das Beispiel der Erstellung und Zerstörung eines Buches durch seinen Autor ist in Abbildung 63 gegeben. Eine Erzeugungsnachricht wird als gestrichelte Linie mit einer offenen Pfeilspitze dargestellt. Der Kopf der neu erzeugten Lebenslinie wird dabei auf der Höhe der Erzeugungsnachricht dargestellt, sodass der Pfeil mittig auf den Kopf zeigt. Wird eine Lebenslinie mittels einer Nachricht zerstört, wird eine Destruktionsnachricht gesendet. Diese besteht aus einer Linie mit einer gefüllten Pfeilspitze und zeigt auf einen Aktivitätsbereich des Teilnehmers. Der Aktivitätsbereich endet nach dieser Nachricht und die Lebenslinie wird mithilfe eines „X“ als zerstört gekennzeichnet. In Abbildung 63 erzeugt der Teilnehmer „:*Autor*“ mittels der Nachricht „*schreiben*“ ein „*Buch*“-Teilnehmer. Direkt danach wird die Destruktionsnachricht „*Zerstören*“ an das Buchobjekt geschickt und zerstört dieses wieder.

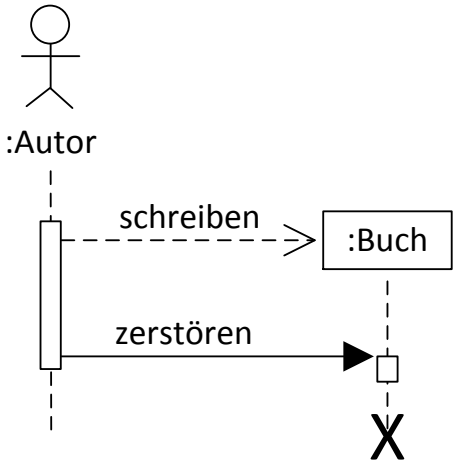


Abbildung 63: Erstellung und Zerstörung eines Buch-Objekts

5.4.5 Kombinierte Fragmente

Mit kombinierten Fragmenten können Kontrollstrukturen in einem Sequenzdiagramm modelliert werden. Bisher werden alle Nachrichten nacheinander abgearbeitet. Mithilfe von kombinierten Fragmenten können beispielsweise Bedingungen der Ausführung gesetzt werden oder parallele Nachrichtenflüsse modelliert werden. Insgesamt definiert die UML 12 verschiedene kombinierte Fragmente. Einige davon behandeln jedoch sehr spezielle Fälle, sodass in dieser Schulung nur fünf vorgestellt werden.

Alle kombinierten Fragmente bestehen aus einem Interaktions-Operator, welcher die Art des Fragments angibt, aus einem optionalen Namen und einem oder mehreren Interaktions-Operanden. Interaktions-Operanden stellen horizontale Bereiche des kombinierten Fragments dar und enthalten den eigentlichen Nachrichtenaustausch. Mehrere Operanden werden durch waagerecht gestrichelte Linien voneinander abgegrenzt. Kombinierte Fragmente dürfen wiederum weitere kombinierte Fragmente enthalten. Dargestellt werden kombinierte Fragmente mit einem Rechteck, welcher den Gültigkeitsbereich abgrenzt. Dabei werden alle involvierte Lebenslinien der Teilnehmer von dem Rechteck eingefasst. Interaktions-Operatoren und Namen werden oben links in einem Pentagon notiert.

5.4.5.1 Alternative Ausführung

Eine alternative Ausführung entspricht einem „if-then-else“-Konstrukt und wird mit dem Interaktions-Operator „alt“ gekennzeichnet. Ein „alt“-Fragment besteht immer aus mindestens zwei Interaktions-Operanden. Es darf immer nur genau ein Interaktions-Operand ausgeführt werden. Die Bedingungen werden in eckigen Klammern in die jeweiligen Operanden oben links notiert. Die Bedingungen müssen disjunkt sein, d.h. sich gegenseitig ausschließen. Es sollte außerdem alle alternativen Ausführungsalternativen abgedeckt werden. Ein Beispiel ist in Abbildung 64 dargestellt. Als Teilnehmer sind ein „:*Kunde*“ und eine „:*Buchhandlung*“ involviert. Zuerst schickt der Kunde eine Nachricht „*Buchserie suchen*“ an die Buchhandlung, von welcher er eine Antwortnachricht mit der gleichen Beschriftung erhält. Danach beginnt das kombinierte Fragment, welches über beiden Lebenslinien liegt. Es werden die beiden Fälle „*Buch vorhanden*“ und „*sonst*“ unterschieden. Ist ein Buch der Buchserie vorhanden wird die Nachricht „*kaufen*“ vom Kunden an die Buchhandlung gesendet, sonst wird die Nachricht „*bestellen*“ gesendet. Danach endet sowohl das kombinierte Fragment, als auch das Sequenzdiagramm.

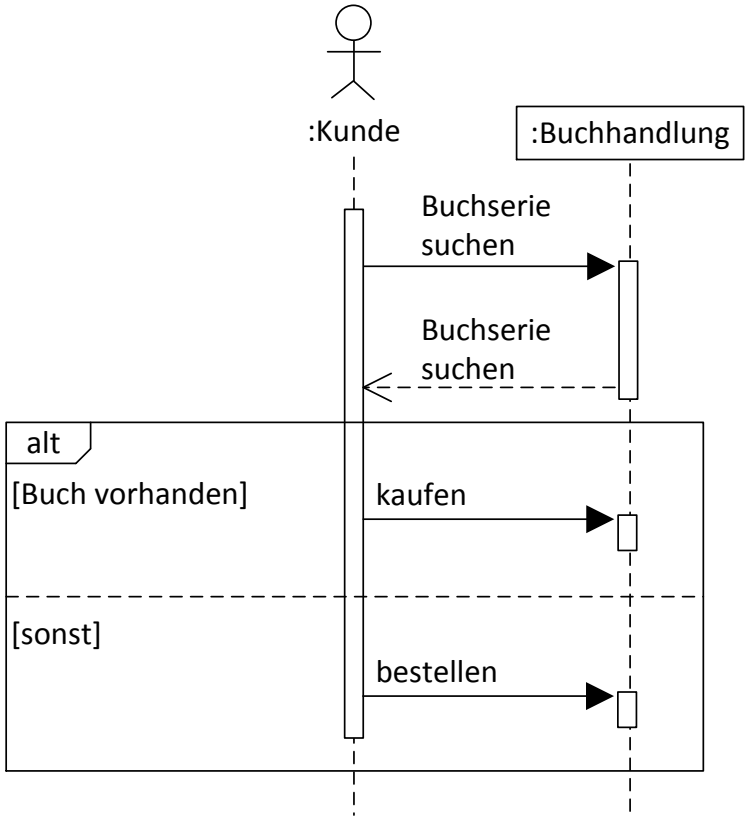


Abbildung 64: Darstellung von alternativen Nachrichtenflüssen

5.4.5.2 Optionale Ausführung

Sollen Nachrichten nur unter bestimmten Bedingungen ausgeführt werden, kann der Interaktions-Operator *„opt“* verwendet werden. Wie im alternativen Fall wird die Bedingung in eckigen Klammern oben links notiert. Im Gegensatz zur Alterative gibt es jedoch nur einen Interaktions-Operand und nicht mehrere. Im Beispiel in Abbildung 65 ist wiederrum ein *„:Kunde“* und eine *„:Buchhandlung“* beteiligt. Der *„:Kunde“* schickt jedoch nur die Nachricht *„kaufen“* zur *„:Buchhandlung“*, falls die Bedingung *„[Empfehlung == gut]“* gilt.

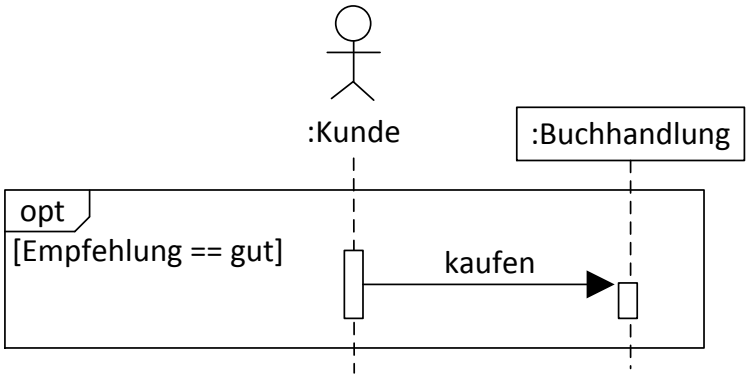


Abbildung 65: Optionale Nachrichtenflüsse anhand von Bedingungen

5.4.5.3 Parallele Ausführung

Operanden können nicht nur bedingt ausgeführt werden, sondern auch parallel. Parallele Operanden werden wie Alternativen ebenfalls durch waagerechte gestrichelte Linien voneinander getrennt, es werden jedoch keine Bedingungen angegeben. Als Interaktions-Operator wird *„par“* verwendet. Der Austausch der Nachrichten aus den verschiedenen Operanden kann in beliebiger Reihenfolge ausgeführt werden. Die Reihenfolge innerhalb der einzelnen

Operanden muss jedoch eingehalten werden. Das Beispiel in Abbildung 66 enthält zwei Operanden und drei Teilnehmer: „:*Abteilungsleiter*“, „*MA1:Mitarbeiter*“ und „*MA2:Mitarbeiter*“. Zur Vereinfachung der Erklärung wurden die Nachrichten nummeriert. Im ersten parallelen Abschnitt schickt der „:*Abteilungsleiter*“ zuerst „*MA1:Mitarbeiter*“ die Nachricht „*1.1 Bücher einräumen*“, bevor er „*MA2:Mitarbeiter*“ die Nachricht „*1.2 Bücher einräumen*“ sendet. Im zweiten Abschnitt schickt „*MA1:Mitarbeiter“* die Nachricht „2.1. *Bücher diskutieren*“ an „*MA2:Mitarbeiter*“, worauf dieser mit der gleichen Nachricht „*2.2 Bücher diskutieren*“ antwortet. Da die Reihenfolge innerhalb der Operanden eingehalten werden muss, d.h. 1.1 vor 1.2 und 2.1 vor 2.2, sind folgende Ausführreihenfolgen möglich:

- 1.1 → 2.1 → 1.2 → 2.2
- 1.1 → 2.1 → 2.2 → 1.2
- 1.1 → 1.2 → 2.1 → 2.2
- 2.1 → 2.2 → 1.1 → 1.2
- 2.1 → 1.1 → 2.2 → 1.2
- 2.1 → 1.1 → 1.2 → 2.2

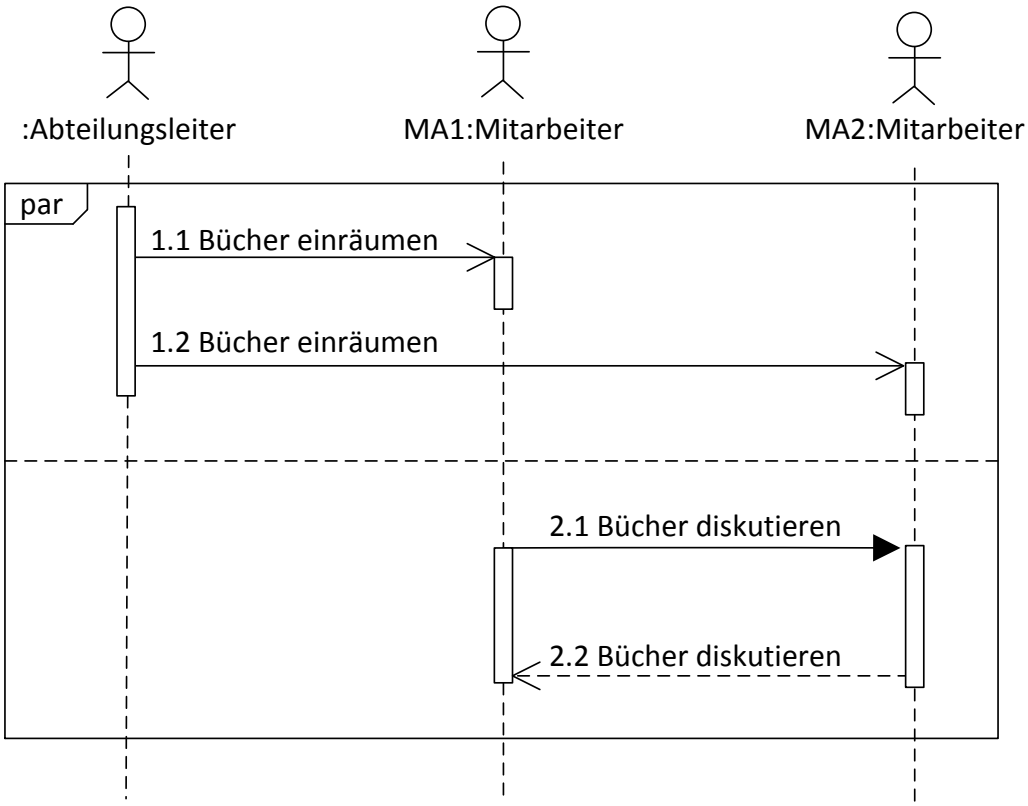


Abbildung 66: Darstellung von parallelen Nachrichtenflüssen

5.4.5.4 Kritischer Bereich

Innerhalb eines kritischen Bereiches, welcher mit „*critical*“ eingeleitet wird, darf die Ausführung der enthaltenen Nachrichten nicht unterbrochen werden. Alle Interaktionen innerhalb des definierten Bereichs werden als atomar angesehen und sind damit unteilbar. Die strikte Reihenfolge kann nicht durch andere Fragmente aufgehoben werden. In Abbildung 67 ist ein Beispiel mit zwei menschlichen Teilnehmern („*MA1:Mitarbeiter*“ und „*MA2:Mitarbeiter*“) und einem Systemteilnehmer „:*Ambulanz*“ gegeben. Zuerst sendet „*MA2:Mitarbeiter“* die asynchrone Nachricht „*Unfall melden*“ an „*MA1:Mitarbeiter*“. Daraufhin beginnt der kritische Be-

reich, durch das Schlüsselwort „*critical*“ definiert: „*MA1:Mitarbeiter*“ sendet zuerst die reflexive Nachricht „*Erste Hilfe leisten(MA2)*“ an sich selbst und danach die synchrone Nachricht „*anfordern*“ zur „*:Ambulanz*“. Die Ambulanz sendet eine nicht spezifizierte Antwortnachricht zurück an „*MA1:Mitarbeiter*“. Während „*MA1:Mitarbeiter*“ diese beiden Nachrichten schickt, darf keine andere Nachricht geschickt werden bis diese beendet sind.

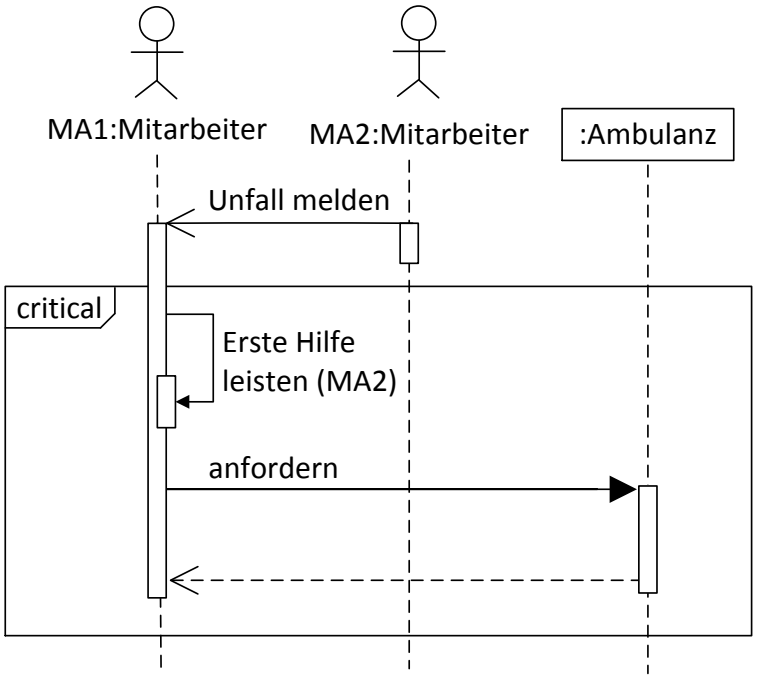


Abbildung 67: Darstellung eines nicht unterbrechbaren Bereichs "critical"

5.4.5.5 Wiederholte Ausführung: Schleifen

Der Schleifen-Operator, gekennzeichnet mit „*loop*“ spezifiziert die wiederholte Ausführung von Interaktions-Operanden. Eine Bedingung definiert dabei die Anzahl an Iterationen und wird in runden Klammern hinter dem Schlüsselwort „*loop*“ festgelegt. Es ist sowohl möglich eine feste Anzahl anzugeben, ein Intervall oder ein Ausdruck, welcher zu „wahr“ und „falsch“ ausgewertet werden kann (Kecher & Salvanos, 2015). Möglich sind demnach folgende Ausdrücke:

- Loop(8): feste Anzahl von genau 8 Wiederholungen
- Loop(2,5): Minimal zwei und maximal 5 Wiederholungen
- Loop(0,4): Maximal 4 Wiederholungen
- Loop(*) oder loop: Beliebige Anzahl an Wiederholungen
- Loop(x < 3): Wiederholung bis der Ausdruck nicht mehr zu wahr ausgewertet wird

Im Beispiel in Abbildung 68 ist folgende Schleife definiert:“ *loop(MA.einräumen!=true)“.* Solange diese Bedingung wahr ist schickt ein Teilnehmer „*:Abteilungsleiter*“ die asynchrone Nachricht mit einem Parameter „*einräumen(Buch)*“ an den Teilnehmer „*:MA:Mitarbeiter*“. Die Nachricht wird demnach so lange wiederholt bis der Mitarbeiter mit dem einräumen der übergebenen Bücher beginnt.

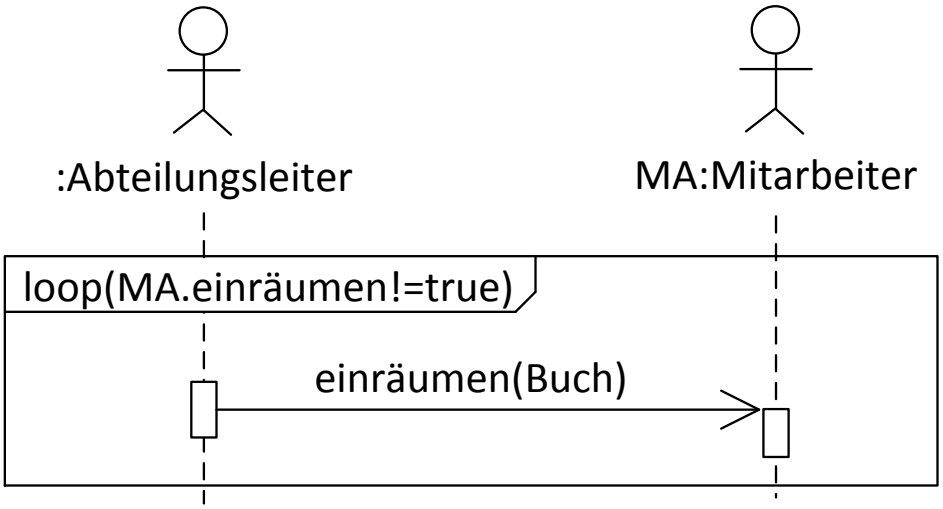


Abbildung 68: Mehrfache Durchführung von Interaktionen: die Schleife

5.4.6 Beispiel eines Sequenzdiagramms für eine Navigations-App

Das Navigationsbeispiel für das Sequenzdiagramm besteht aus insgesamt drei Teilnehmern: ein menschlicher Teilnehmer „:Person“ und zwei Systemteilnehmern *„:Navigerät“* und „:*GPS-System*“. Die Interaktion fängt mit dem Senden der synchronen Nachricht *„Ziel bestimmen“* von der „:Person“ zum „:*Navigerät*“ an. Daraufhin sendet das „:*Navigerät*“ eine synchrone Nachricht *„Aktuelle Position ermitteln“* an das „:*GPS-System*“, damit es die Route vom aktuellen Standort zum Zielort ermitteln kann. Das „:*GPS-System*“ antwortet daraufhin mit der Nachricht *„Aktuelle Position ermitteln:Pos(X,Y)“*. Es sendet somit die X und Y Koordinaten der aktuellen Position der *„:Person“*. Danach endet die Lebenslinie. Das „:*Navigerät*“ sendet daraufhin die Antwort *„Ziel bestimmen“* an die „:*Person*“. Die Lebenslinie des „:*Navigeräts*“ endet nach dieser Nachricht. Die „:*Person*“ sendet daraufhin innerhalb einer Schleife, welche nicht näher spezifiziert ist und daher nur aus dem Schlüsselwort *„loop“* besteht, eine reflexive Nachricht *„Zum Ziel navigieren“* an sich selbst. Danach endet das gesamte Sequenzdiagramm.

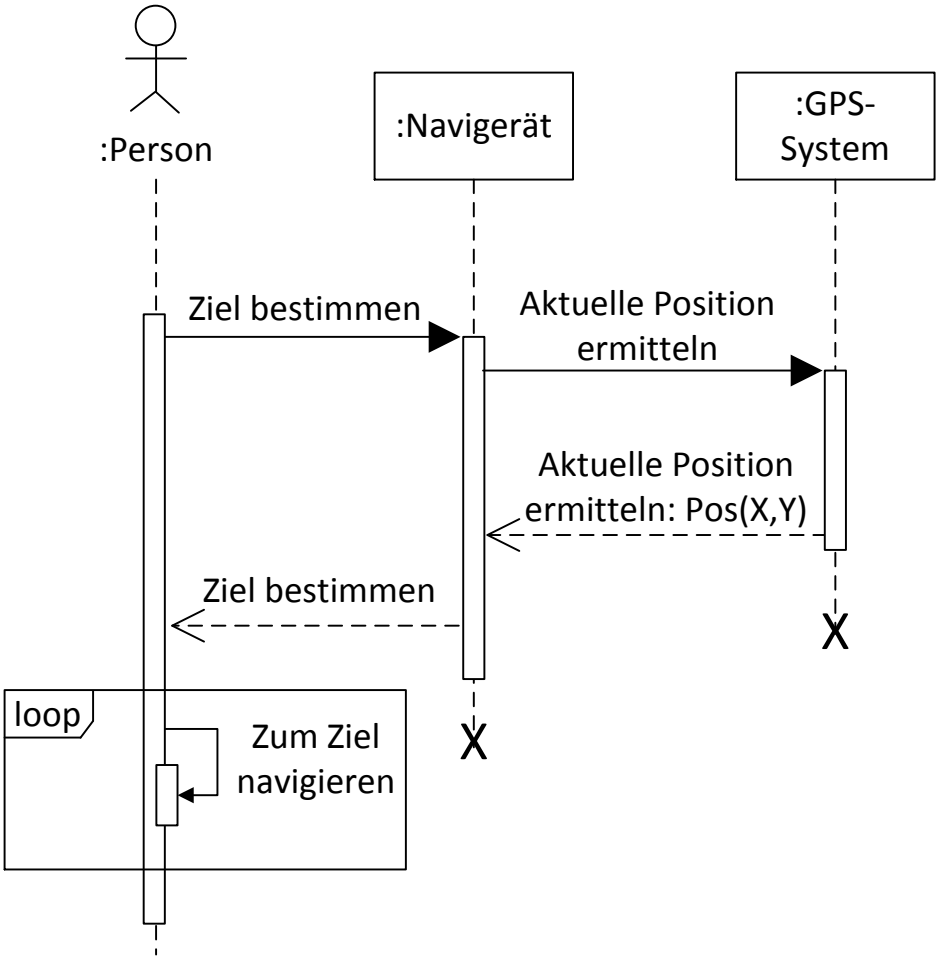


Abbildung 69: Sequenzdiagramm des Navigationsbeispiels

5.4.7 Verständnisfragen zum Sequenzdiagramm

Dieses Kapitel besteht aus fünf Multiple-Choice-Fragen (Business Informatics Group, 2015), um das Verständnis für das Zustandsdiagramm zu überprüfen. Jede Frage hat vier Antwortmöglichkeiten a–d. Bei jeder Frage können eine, zwei, drei oder vier Antworten richtig sein. Die Auflösung ist in Anhang A und im Detail in Anhang A.5 zu finden.

1. Welche der folgenden Eigenschaften weist der alt-Operator in einem Sequenzdiagramm auf?

- (a) Alle Operanden in einem alt-Operator, für die keine explizite Überwachungsbedingung angegeben wurde, werden in jedem Fall ausgeführt.
- (b) Die Entscheidung, welcher Operand ausgeführt werden soll, geschieht mit Hilfe von Überwachungsbedingungen.
- (c) Zur Laufzeit können optional auch mehrere Operanden ausgeführt werden.
- (d) Mit dem alt-Operator können alternative Interaktionsabläufe dargestellt werden.

2. Welche Aussagen über synchrone und asynchrone Kommunikation von Objekten in einem Sequenzdiagramm treffen zu??

- (a) Die Modellierung einer Antwortnachricht bei synchroner Kommunikation ist optional.
- (b) Synchrone Kommunikation wird mit einer geschlossenen Pfeilspitze modelliert, asynchrone Kommunikation mit einer offenen.

(c) Bei synchroner Kommunikation wartet der Sender, bis die Interaktion beendet wurde, bevor er fortfährt.
(d) Bei asynchroner Kommunikation wartet der Sender, bis die Interaktion beendet wurde, bevor er fortfährt.

3. Welche der folgenden Aussagen bzgl. Zustandsinvarianten in einem Sequenzdiagramm sind korrekt?
(a) Eine Zustandsinvariante wird vor dem Eintritt des darauffolgenden Ereignisses ausgewertet.
(b) Eine Zustandsinvariante bezieht sich immer auf eine bestimmte Lebenslinie.
(c) Eine Auswertung der Zustandsinvariante kann jederzeit während der Nachrichtenabfolge erfolgen.
(d) Bei einer Zustandsinvariante handelt es sich um die Annahme, dass eine bestimmte Bedingung im Lauf des Lebenszyklus erfüllt wird.

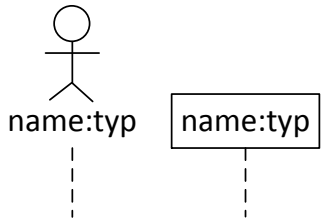
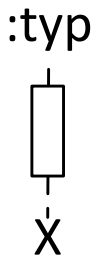
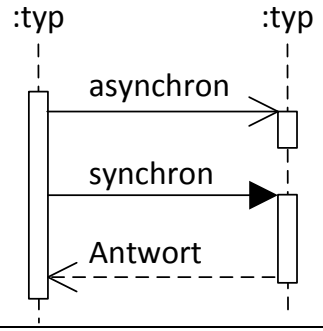
4. In Sequenzdiagrammen ...
(a) ...stellt die Zeit keine eigene Dimension dar.
(b) ...wird das Intra-Objektverhalten beschrieben.
(c) ...wird der zeitliche Ablauf von nur einem Objekt verfolgt.
(d) ...werden mögliche Nachrichtenabläufe zwischen Teilnehmern modelliert.

5. Welche allgemeinen Aussagen über Sequenzdiagramme sind korrekt?
(a) alt, par, loop und if sind kombinierte Fragmente.
(b) Bei asynchroner Kommunikation wartet der Sender auf eine Antwort vom Empfänger.
(c) Mit einem Sequenzdiagramm kann die Interaktion eines Systems mit seiner Umwelt modelliert werden.
(d) Die vertikale Achse ist die Interaktionsachse.

5.4.8 Modellierungsaufgabe Sequenzdiagramm

Ein Kunde möchte sich Medikamente von seinem Rezept in einer Apotheke abholen. Dafür sucht er eine Apotheke auf und überreicht dem Angestellten das Rezept. Dieser sucht daraufhin alle Medikamente zusammen und bestellt bei Bedarf fehlende Medikamente. Nachdem er alle Medikamente gefunden hat überreicht er dem Kunden diese.
Eine Beispiellösung findet sich in Anhang B und als taktile Grafik auf Seite 65 der taktilen Mappe.

5.4.9 Tabellarische Übersicht der Elemente des Sequenzdiagramms

Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Teilnehmer	Ein Teilnehmer kann ein menschlicher Teilnehmer sein oder ein Objekt, welcher mit anderen Teilnehmern kommuniziert. Der Name des Teilnehmers besteht aus: „name:typ“. Der „name“ ist dabei optional, „:typ“ muss immer angegeben sein.	Ein menschlicher Teilnehmer wird als Strichmännchen dargestellt. Der Name und Typ stehen direkt unter dem Strichmännchen und über seiner Lebenslinie. Ein Objekt oder Systemteilnehmer wird als Rechteck mit dem Namen und Typ darin dargestellt.	
Lebenslinie	Lebenslinien verlaufen vertikal und zeigen die Existenzlaufzeit von Teilnehmern an. Pro Teilnehmer gibt es nur eine Lebenslinie. Es gibt passive und aktive Zeit der Lebenslinie. Nur während der aktiven Phase kann ein Teilnehmer an der Interaktion teilnehmen.	Eine gestrichelte Linie, welche vertikal vom Kopf des Teilnehmers an verläuft stellt die passive Zeit der Lebenslinie dar. Ein schmaler Balken auf der Lebenslinie stellt die aktive Zeit dar. Ein „X“ am Ende markiert die Beendigung einer Lebenslinie.	
Nachrichten	Eine Nachricht stellt die Kommunikation, z.B. Methodenaufrufe zwischen Teilnehmer dar. Nachrichten können nur zwischen Aktivitätsbereichen verlaufen, werden als Pfeile dargestellt und besitzen einen Namen oberhalb des Pfeils.	Asynchrone Nachrichten werden als Pfeil mit offener Spitze modelliert. Synchrone Nachrichten werden als Pfeil mit gefüllter Spitze modelliert. Eine Antwortnachricht ist ein gestrichelter Pfeil mit offener oder gefüllter Pfeilspitze. Alle Nachrichten haben den Namen oberhalb der Pfeile notiert.	

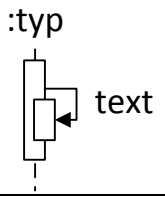
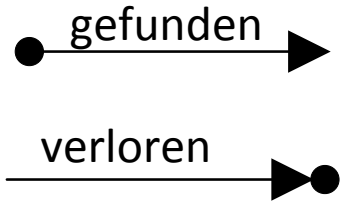
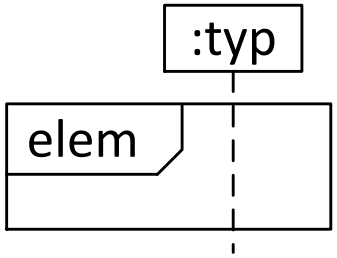
Name	Beschreibung	Beschreibung der grafischen Darstellung	Grafische Darstellung
Reflexive Nachricht	Sendet ein Teilnehmer eine Nachricht an sich selbst, so ruft er eine Methode auf sich selbst auf.	Ein Pfeil mit gefüllter Spitze, welcher von der Lebenslinie auf eine parallel modellierte Lebenslinie zeigt und damit ein „U auf der Seite“ formt. Der Name steht neben dem Pfeil.	
Gefundene und verlorene Nachricht	Nachrichten, welche keinen spezifizierten Sender oder Empfänger besitzen, da diese nicht modelliert werden sollen oder nicht relevant sind.	Eine gefundenen Nachricht ist ein Pfeil, welcher als Anfang einen schwarzen Kreis enthält. Eine verlorene Nachricht mündet in einen schwarzen Kreis und nicht in einer Lebenslinie.	
Kombinierte Fragmente	Mit kombinierten Fragmenten können Kontrollstrukturen (Interaktionsoperator) höherer Programmiersprachen ausgedrückt werden, wie z.B. parallele Ausführung, „If-Then-Else“-Konstrukte, etc. Ein Fragment kann in mehrere voneinander abgetrennte Bereiche (Operanden) aufgeteilt werden und ineinander geschachtelt werden.	Es wird als rechteckiger Block dargestellt. In der linken, oberen Ecke in einem Pentagon ist der Typ der Kontrollstruktur angegeben. Operanden werden durch eine waagrecht gestrichelte Linie im Rechteck abgegrenzt.	

Tabelle 5: Kurzreferenz der Elemente des Sequenzdiagramms

Literaturverzeichnis

Business Informatics Group, T. U. (2015). *UML Quiz*. Von http://elearning.uml.ac.at/ abgerufen

Dobing, B., & Parson, J. (2006). How UML is used. *Communications of the ACM*, S. 109-113.

Kecher, C., & Salvanos, A. (2015). *UML 2.5: Das umfassende Handbuch.* Galileo Computing. Rheinwerk Verlag.

Lahres, B., & Rayman, G. (2009). *Praxisbuch Objektorientierung: Das umfassende Handbuch.* Galileo Press.

Langer, P., Mayerhofer, T., Wimmer, M., & Kappel, G. (2015). *On the Usage of UML: Initial Results of Analyzing Open UML Models.*

Petre, M. (2014). “No shit” or “Oh, shit!”: responses to observations on the use of UML in professional practice. *Software & Systems Modeling*, S. 1225-1235.

Rupp, C., Queins, S., & Zengler, B. (2007). *UML 2 glasklar - Praxiswissen für die UML-Modellierung.* München: Hanser Verlag München Wien.

Schäling, B. (2. 6 2015). *Der moderne Softwareentwicklungsprozess mit UML*. Von http://www.highscore.de/uml/ abgerufen

Seidl, M., Scholz, M., Huemer, C., & Kappel, G. (2015). *UML @Classroom - An Introduction to Object-Oriented Modeling.* Springer.

Anhang

In diesem Abschnitt sind Lösungen für die Übungsaufgaben zu finden. Zuerst werden die Antworten zu den Multiple-Choice-Fragen gegeben und anschließend beispielhafte grafische Diagramme zu der praktischen Modellierungsübung gezeigt. Die Grafiken zeigen Lösungsvorschläge, d.h. die Benennung und Anordnung der Elemente muss nicht identisch zu der von Ihnen erstellten Lösung sein.

Anhang A Lösungen der Multiple-Choice-Fragen

Die Lösungen der Multiple-Choice-Fragen der Kapitel 4.1.8, 5.1.6, 5.2.8, 5.3.7 und 5.4.7 sind tabellarisch angegeben. In der ersten Spalte steht die Nummerierung der Fragen von 1 bis 5. Die zweite Spalte listet die Antworten des Klassendiagramms, die dritte die des Anwendungsfalldiagramms, die vierte Spalte enthält die Lösungen des Aktivitätsdiagramms, die fünfte die des Zustandsdiagramms und die sechste Spalte die des Sequenzdiagramms.

Fragen	**Klassendiagramm**	**Anwendungsfalldiagramm**	**Aktivitätsdiagramm**	**Zustandsdiagramm**	**Sequenzdiagramm**
1	b	b,c	a,d	b,d	b,d
2	a,b,d	a,d	a,c	a,b,c	b,c
3	a	c,d	a,b	a,d	a,b
4	c,d	c	b,c	a,d	d
5	c,d	a,c	a,b	d	c

Tabelle 6: Lösungen der Multiple-Choice-Fragen

Nachfolgend werden für jeden Diagrammtyp Begründungen für die jeweils falschen Antworten gegeben. Die Begründungen fangen in einer neuen Zeile nach der Antwort an und sind hauptsächlich dem Quiz (Business Informatics Group, 2015) entnommen. Visuell sind die Begründungen durch Kursivdruck und Einfärbung gekennzeichnet. Die Diagrammtypen werden dabei in der Reihenfolge ihres Auftretens innerhalb der Schulung gelistet: Klassendiagramm, Anwendungsfalldiagramm, Aktivitätsdiagramm, Zustandsdiagramm, Sequenzdiagramm.

Anhang A.1 Antworten des Klassendiagramms

1. Ein Klassendiagramm beschreibt…
   (a) den Interaktions-Aspekt eines Systems.
   Es beschreibt den strukturellen Aspekt eines Systems.
   (b) den strukturellen Aspekt eines Systems.
   Richtig
   (c) den praktischen Aspekt eines Systems.
   Es beschreibt den strukturellen Aspekt eines Systems.
   (d) den dynamischen Aspekt eines Systems.
   Es beschreibt den strukturellen Aspekt eines Systems.
2. Assoziationen…
   (a) können Multiplizitäten an ihren Enden haben, wodurch angegeben wird, mit wie vielen Objekten auf der gegenüberliegenden Seite ein Objekt in Beziehung stehen kann.
   Richtig
   (b) modellieren mögliche Beziehungen zwischen Instanzen von Klassen.
   Richtig
   (c) müssen durch einen Assoziationsnamen identifizierbar sein.
   Die Angabe eines Namens ist optional.
   (d) können eine Navigationsrichtung haben, welche angibt, in welche Richtung die Navigation von einem Objekt zu seinem Partnerobjekt erfolgen kann.
   Richtig
3. Bei der Generalisierungsbeziehung zwischen einer spezialisierten Klasse (Unterklasse) und einer allgemeineren Klasse (Oberklasse) gelten folgende Eigenschaften:
   (a) Die spezialisierte Klasse erbt die Eigenschaften der allgemeineren.
   Richtig
   (b) Die spezialisierte Klasse darf ausschließlich nur von einer einzigen allgemeineren Klasse erben.
   Gilt nicht, da in UML Mehrfachvererbung zulässig ist.
   (c) Die allgemeinere Klasse darf ausschließlich mit einer Unterklasse eine Vererbungsbeziehung eingehen.
   Gilt nicht, da in UML Mehrfachvererbung zulässig ist.
   (d) Es dürfen keine weiteren Attribute zur spezialisierten Klasse hinzugefügt werden.
   Falsch, denn dann würde Vererbung keinen Sinn machen.
4. Eine Aggregation...
   (a) wird durch eine gefüllte Raute an einem Assoziationsende dargestellt.
   Die Aggregation wird durch eine nicht gefüllte Raute dargestellt. Eine gefüllte Raute stellt die Komposition dar.
   (b) drückt eine ist-ein-Beziehung aus.
   Die Aggregation drückt eine Teil-von-Beziehung aus.
   (c) drückt eine Teil-von-Beziehung aus.
   Richtig
   (d) wird durch eine ungefüllte Raute an einem Assoziationsende dargestellt.
   Richtig

5. Welche der folgenden Aussagen sind korrekt?
   (a) Bei einer Klasse im Klassendiagramm müssen immer Attribute und Methoden angegeben werden.
   Diese Angaben sind optional, nur der Klassenname ist verpflichtend anzugeben.
   (b) Klassen haben immer einen Abschnitt, der eine textuelle Beschreibung von ihnen beinhaltet.
   Nein, ein derartiger Abschnitt existiert nicht, es sind Abschnitte für Name, Attribute und Methoden vorgesehen.
   (c) Attribute können durch einen Typ näher beschrieben werden.
   Richtig
   (d) Bei Operationen können Übergabeparameter und Rückgabewert angegeben werden
   Richtig

Anhang A.2 Antworten des Anwendungsfalldiagramms

1. Welche Aussagen treffen zu, wenn ein Akteur B von einem Akteur A erbt?
   (a) A kann mit denselben Anwendungsfällen wie B kommunizieren.
   Nein, es ist genau umgekehrt. B kann nun mit den Anwendungsfällen von A kommunizieren.
   (b) B kann mit denselben Anwendungsfällen wie A kommunizieren.
   Richtig
   (c) B erbt alle Assoziationen von A.
   Richtig
   (d) A erbt alle Assoziationen von B.
   Es ist genau umgekehrt. B erbt die Assoziationen von A.
2. Akteure in einem Anwendungsfalldiagramm...
   (a) können das beschriebene System benutzen.
   Richtig
   (b) befinden sich immer innerhalb des beschriebenen Systems.
   Sie befinden sich immer außerhalb und interagieren mit dem System.
   (c) interagieren mit Hilfe von<<include>>-Beziehungen mit dem System.
   Sie interagieren mit Hilfe von Assoziationen mit dem System.
   (d) stellen Rollen der Benutzer des beschriebenen Systems dar.
   Richtig
3. Die Assoziation zwischen einem Akteur und einem Anwendungsfall...
   (a) wird durch eine gestrichelte Linie modelliert.
   Sie wird durch eine durchgezogene Linie modelliert.
   (b) wird durch eine Kante mit Pfeilspitze auf der Seite des Anwendungsfalls modelliert.
   Die Kante ist ungerichtet.
   (c) kann Multiplizitäten aufweisen.
   Richtig
   (d) ist binär.
   Richtig

4. Welche der folgenden Aussagen über Anwendungsfälle sind korrekt?
   (a) Anwendungsfälle dürfen untereinander in keiner Generalisierungsbeziehung stehen
   Anwendungsfälle können in einer Vererbungsbeziehung, d.h. Generalisierungsbeziehung stehen.
   (b) Anwendungsfälle müssen nicht zwingend benannt werden
   Ein Name muss immer gegeben sein.
   (c) Anwendungsfälle dürfen binäre, aber keine n-ären, Assoziationen besitzen
   Richtig
   (d) Ein Anwendungsfall wird als Rechteck mit abgerundeten Ecken dargestellt.
   Die Darstellung erfolgt als Ellipse oder als Rechteck mit einer Ellipse enthalten.
5. Das Anwendungsfalldiagramm…
   (a) beinhaltet Akteure und Anwendungsfälle.
   Richtig
   (b) beschreibt zeitliche Abläufe innerhalb eines Systems.
   Zeitliche Abhängigkeiten können nicht modelliert werden.
   (c) beschreibt, wer was mit dem zu entwickelnden System macht.
   Richtig
   (d) beschreibt, wie Funktionen im System angeboten werden.
   Ein Anwendungsfalldiagramm beschreibt nur welche Funktionen existieren, nicht wie diese ausgeführt werden.

Anhang A.3 Antworten des Aktivitätsdiagramms

1. Ein Startknoten in einem Aktivitätsdiagramm…
   (a) versorgt alle ausgehenden Kanten mit Token.
   Richtig
   (b) darf pro Aktivität genau einmal vorkommen.
   Es kann auch mehrere Initialknoten geben. Mehrere Initialknoten ermöglichen das Starten von parallelen Abläufen.
   (c) wird als weißer Kreis dargestellt.
   Ein Startknoten ist ein schwarzer Kreis.
   (d) stellt den Beginn eines Aktivitätsablaufes dar.
   Richtig
2. Ein Synchronisierungsknoten in einem Aktivitätsdiagramm...
   (a) führt nebenläufige Abläufe wieder zusammen.
   Richtig
   (b) wird als ungefüllte Raute dargestellt.
   Eine ungefüllte Raute ist ein Entscheidungs-/Vereinigungsknoten. Ein Synchronisierungsknoten ist ein schwarzer Balken.
   (c) vereinigt Token, sobald diese an allen eingehenden Kanten vorhanden sind.
   Richtig
   (d) führt alternative Abläufe wieder zusammen.
   Ein Vereinigungsknoten führt alternative Abläufe wieder zusammen. Ein Synchronisierungsknoten führt nebenläufige Abläufe wieder zusammen.

3. Ein Endknoten in einem Aktivitätsdiagramm...

(a) beendet alle Abläufe einer Aktivität.
Richtig
(b) verbietet die Ausführung weiterer Aktionen innerhalb der Aktivität.
Richtig
(c) kommt pro Aktivität genau einmal vor.
Es dürfen mehrere Endknoten modelliert werden. Der erste, der erreicht wird beendet das Aktivitätsdiagramm.
(d) wird als schwarzer Kreis dargestellt.
Ein schwarzer Kreis ist ein Startknoten. Ein Endknoten ist ein weißer Kreis mit einem kleineren schwarzen Kreis darin.

4. Ein Parallelisierungsknoten in einem Aktivitätsdiagramm...
(a) ist nur dann formal gültig, wenn es auch einen zugehörigen Synchronisierungsknoten gibt.
Dies ist nicht notwendig. Es könnten alle parallel Flüsse in Endknoten führen, sodass eine Synchronisierung nicht mehr notwendig ist.
(b) dient zur Modellierung der Aufspaltung in nebenläufige Abläufe.
Richtig
(c) dupliziert eingehende Token für alle ausgehenden Kanten.
Richtig
(d) stellt eine Notationsvariante des Entscheidungsknotens dar.
Bei einem Entscheidungsknoten wird nur ein Fluss danach ausgeführt, bei einem Parallelisierungsknoten, werden mehrere Flüsse nebenläufig (parallel) ausgeführt.

5. Welche der folgenden Aussagen treffen auf Aktionen des Aktivitätsdiagramms zu?
(a) Aktionen können in Aktivitäten zu größeren Einheiten zusammengefasst werden.
Richtig
(b) Aktionen sind atomar.
Richtig
(c) Aktionen werden mithilfe von Assoziationen miteinander verbunden.
Aktionen werden mithilfe von Kontrollflüssen miteinander verbunden.
(d) Aktionen beinhalten mehrere Aktivitäten.
Es ist umgekehrt: Eine Aktivität beinhaltet mehrere Aktionen.

Anhang A.4 Antworten des Zustandsdiagramms

1. Bei einem aktiven orthogonalen Zustand…
(a) ist immer genau ein Subzustand aktiv.
Es ist je ein Subzustand in allen Regionen aktiv
(b) ist mindestens ein Subzustand in jeder Region aktiv.
Richtig
(c) kann auch kein Subzustand aktiv sein.
Nein, beim Betreten eines orthogonalen Zustands wird immer ein Subzustand in jeder Region betreten. Dies kann ein modellierter Startzustand oder ein direkt Betretener Subzustand sein.
(d) sind Transitionen zwischen den Regionen verboten.
Richtig

2. In einem Zustandsdiagramm können folgende Elemente modelliert werden:

(a) In den Startzustand eingehende Transitionen.
Der Startzustand darf keine eingehende Transition besitzen und genau eine ausgehende, welche keine Bedingungen enthält.
(b) Events, die Zustandsübergänge auslösen.
Richtig
(c) Mögliche Zustandsübergänge von einem Zustand zum anderen.
Richtig
(d) Vom Endzustand ausgehende Transitionen.
In den Endzustand dürfen nur Transitionen eingehen, keine mehr ausgehen.

3. Mit welchen Angaben kann eine Transition beschriftet sein?
(a) Mit einer Aktivität, die während der Transition auszuführen ist.
Richtig
(b) Mit dem Event, das nach der Transition ausgeführt werden soll.
Ein Event kann eine Transition auslösen. Ein Event tritt ein und kann nicht ausgeführt werden.
(c) Mit dem Event, das während der Transition ausgeführt wird.
Ein Event kann eine Transition auslösen. Während der Transition kann eine Aktivität ausgeführt werden.
(d) Mit dem Event, das die Transition auslöst.
Richtig

4. Welche der folgenden Aussagen über Zustandsdiagramme bzw. deren Inhalt sind korrekt?
(a) Pro Zustandsdiagramm kann es mehrere Endzustände geben.
Richtig
(b) An einer Transition können Bedingungen, Events und Zustände angegeben werden.
An einer Transition können Bedingungen, Events und Aktivitäten angegeben werden.
(c) Der Startzustand besitzt genau eine ausgehende und beliebig viele eingehende Transitionen.
Der Startzustand darf keine eingehenden Transitionen haben.
(d) Events lösen Transitionen aus.
Richtig

5. In welcher Syntax wird die Beschriftung einer Transition geschrieben?
(a) [Aktivität] Event /Bedingung
Richtig ist (d)
(b) [Bedingung]Aktivität/ Event
Richtig ist (d)
(c) Aktivität[Bedingung]/ Event
Richtig ist (d)
(d) Event [Bedingung]/Aktivität
Richtig

Anhang A.5 Antworten des Sequenzdiagramms

1. Welche der folgenden Eigenschaften weist der alt-Operator in einem Sequenzdiagramm auf?
   (a) Alle Operanden in einem alt-Operator, für die keine explizite Überwachungsbedingung angegeben wurde, werden in jedem Fall ausgeführt.
   Es wird immer nur ein Operand eines alt-Operators ausgeführt (und zwar der, dessen Überwachungsbedingung zutrifft -- treffen mehrere Überwachungsbedingungen zu, so wird EINER der Operanden ausgeführt)
   (b) Die Entscheidung, welcher Operand ausgeführt werden soll, geschieht mit Hilfe von Überwachungsbedingungen.
   Richtig
   (c) Zur Laufzeit können optional auch mehrere Operanden ausgeführt werden.
   Es wird immer nur ein Operand eines alt-Operators ausgeführt (und zwar der, dessen Überwachungsbedingung zutrifft - treffen mehrere Überwachungsbedingungen zu, so wird EINER der Operanden ausgeführt (welcher ist nicht definiert)).
   (d) Mit dem alt-Operator können alternative Interaktionsabläufe dargestellt werden.
   Richtig

2. Welche Aussagen über synchrone und asynchrone Kommunikation von Objekten in einem Sequenzdiagramm treffen zu??
   (a) Die Modellierung einer Antwortnachricht bei synchroner Kommunikation ist optional.
   Richtig
   (b) Synchrone Kommunikation wird mit einer geschlossenen Pfeilspitze modelliert, asynchrone Kommunikation mit einer offenen.
   Richtig
   (c) Bei synchroner Kommunikation wartet der Sender, bis die Interaktion beendet wurde, bevor er fortfährt.
   Richtig
   (d) Bei asynchroner Kommunikation wartet der Sender, bis die Interaktion beendet wurde, bevor er fortfährt.
   Es ist genau umgekehrt, wie in Antwort c.

3. Welche der folgenden Aussagen bzgl. Zustandsinvarianten in einem Sequenzdiagramm sind korrekt?
   (a) Eine Zustandsinvariante wird vor dem Eintritt des darauffolgenden Ereignisses ausgewertet.
   Richtig
   (b) Eine Zustandsinvariante bezieht sich immer auf eine bestimmte Lebenslinie.
   Richtig
   (c) Eine Auswertung der Zustandsinvariante kann jederzeit während der Nachrichtenabfolge erfolgen.
   Die Zustandsinvariante muss vor Eintritt des darauffolgenden Ereignisses ausgewertet werden.
   (d) Bei einer Zustandsinvariante handelt es sich um die Annahme, dass eine bestimmte Bedingung im Lauf des Lebenszyklus erfüllt wird.

Eine Zustandsinvariante stellt eine Zusicherung dar, dass eine bestimmte Bedingung zu einem bestimmten Zeitpunkt des Interaktionsablaufs erfüllt sein muss.

4. In Sequenzdiagrammen ...
   (a) ...stellt die Zeit keine eigene Dimension dar.
   Die zwei Dimensionen in einem Sequenzdiagramm sind die Zeitachse (vertikal) und die Interaktionspartner (horizontal).
   (b) ...wird das Intra-Objektverhalten beschrieben.
   Es wird das Inter-Objektverhalten - also das Verhalten zwischen mehreren Objekten beschrieben.
   (c) ...wird der zeitliche Ablauf von nur einem Objekt verfolgt.
   Es wird der zeitliche Ablauf mehrerer Objekte mit dem Fokus auf die Interaktion der Objekte untereinander beschrieben, wobei der zeitliche Ablauf jedes Objekts auf einer separaten Lebenslinie modelliert wird.
   (d) ...können weitere Sequenzdiagramme referenziert werden.
   Richtig
5. Welche allgemeinen Aussagen über Sequenzdiagramme sind korrekt?
   (a) alt, par, loop und if sind kombinierte Fragmente.
   if existiert nicht.
   (b) Bei asynchroner Kommunikation wartet der Sender auf eine Antwort vom Empfänger.
   Bei synchroner Kommunikation wartet der Sender auf eine Antwort vom Empfänger.
   (c) Mit einem Sequenzdiagramm kann die Interaktion eines Systems mit seiner Umwelt modelliert werden.
   Richtig
   (d) Die vertikale Achse ist die Interaktionsachse.
   Vertikale Achse: Zeitachse; Horizontale Achse: Interaktionspartner

Anhang B Lösungen der praktischen Aufgaben

Nachfolgend sind Lösungsvorschläge der Modellierungsaufgaben der Kapitel 4.1.9, 5.1.7, 5.2.9 5.3.8 und 5.4.8 in der Reihenfolge der Aufgabenstellung angegeben. Diese Lösungen dienen als Beispiel. Bezeichnungen der Elemente oder Anordnung können von Ihrer Lösung abweichen, der Grundaufbau sollte jedoch ähnlich sein. Taktile Versionen der Lösungsdiagramme sind in der taktilen Mappe im Anhang auf den Seiten 61 – 65 dargestellt.

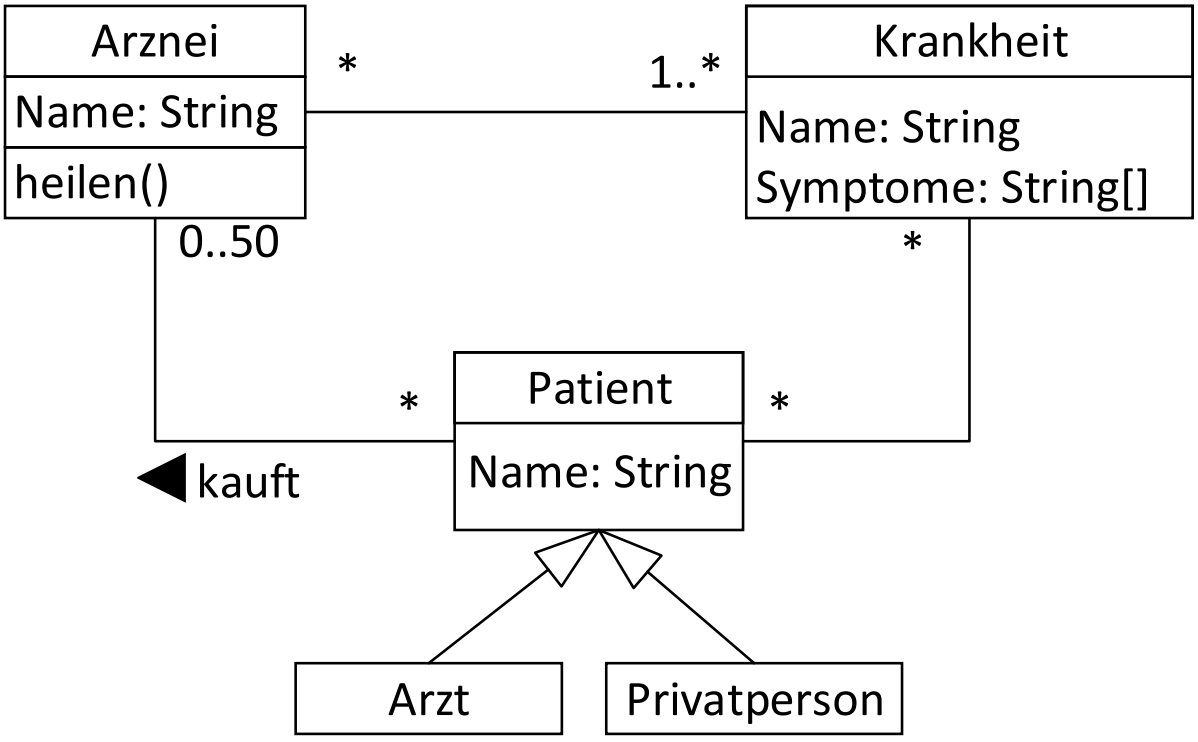


Abbildung A 1: Lösungsvorschlag für das Klassendiagramm

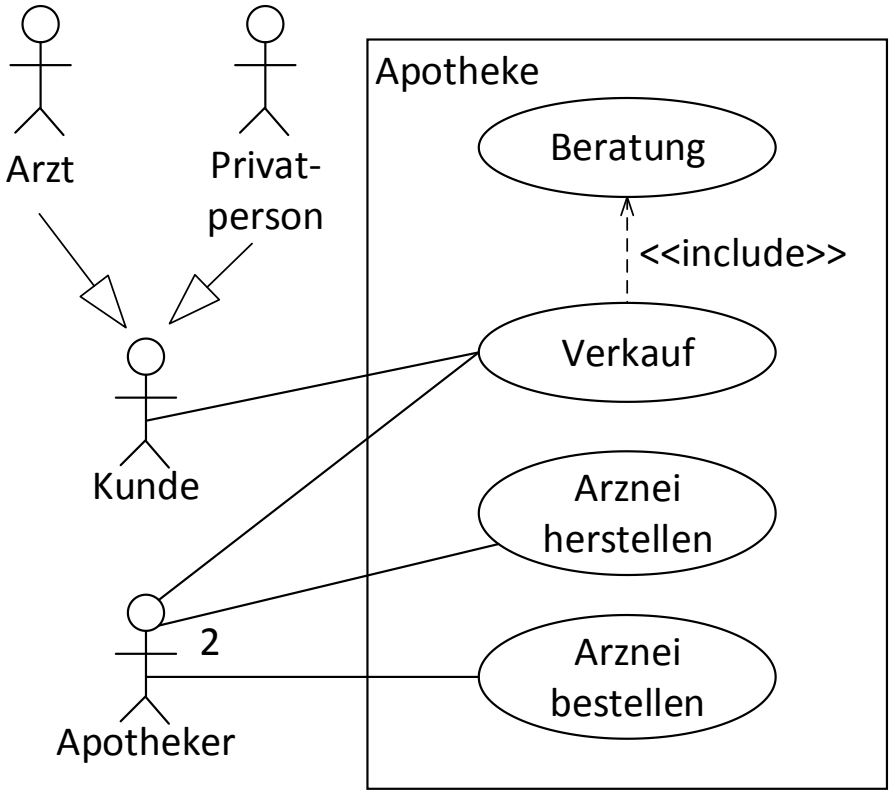


Abbildung A 2: Lösungsvorschlag für das Anwendungsfalldiagramm

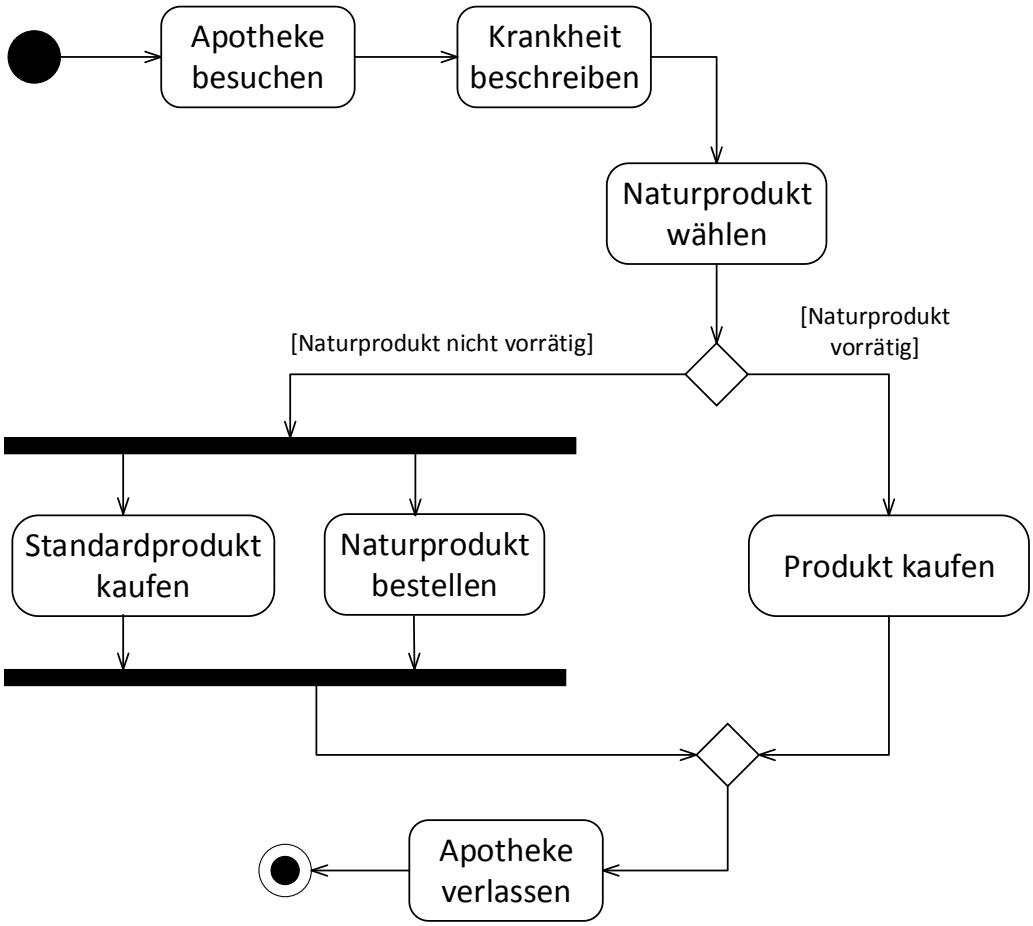


Abbildung A 3: Lösungsvorschlag für das Aktivitätsdiagramm

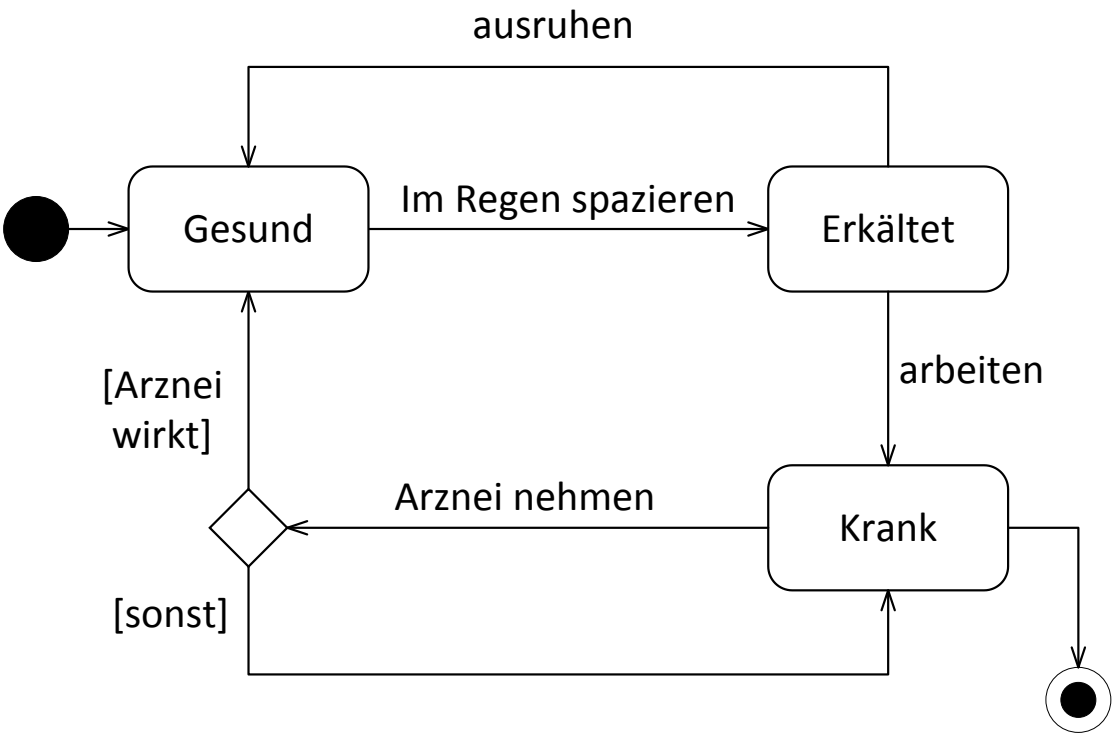


Abbildung A 4: Lösungsvorschlag für das Zustandsdiagramm

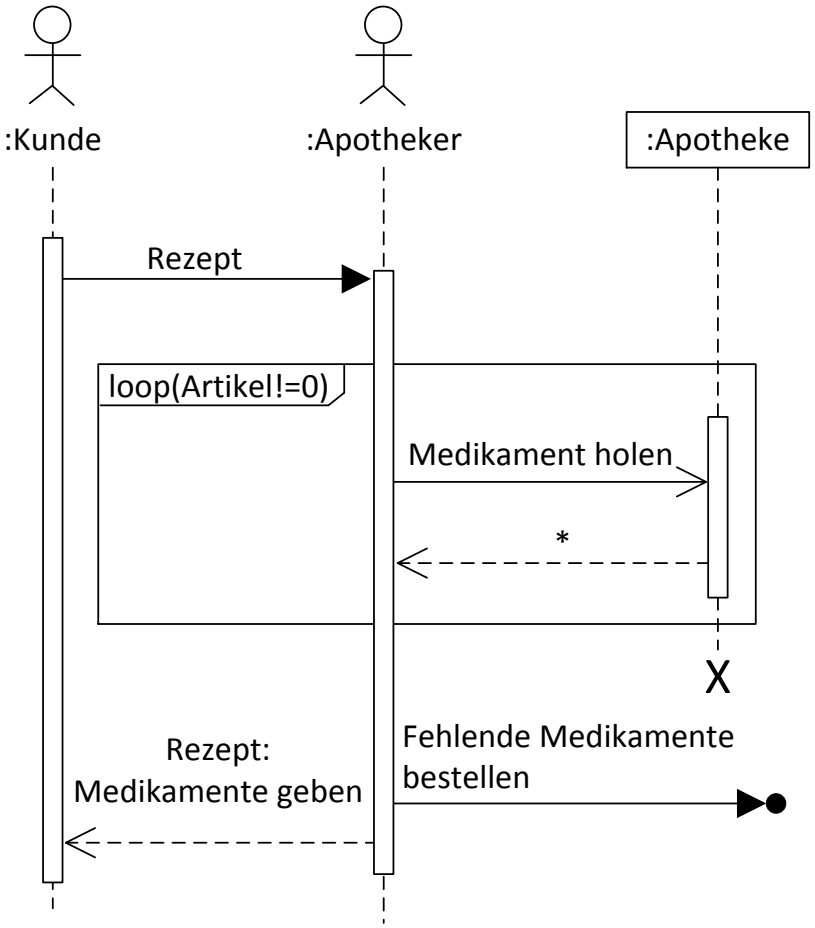


Abbildung A 5: Lösungsvorschlag für das Sequenzdiagramm