COMPANION WEBSITE
CWS
www.pearson-studium.de

informatik

Harald Störrle

UML 2 für Studenten

Mit UML-Syntax-Poster

PEARSON
Studium

PEARSON Education

PEARSON Studium

Harald Störrle

UML 2 für Studenten

PEARSON Studium

ein Imprint von Pearson Education
München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

UML 2 für Studenten

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

UML 2.0 Syntax

Klassendiagramme

Analyse-Klassendiagramme stellen Geflechte von Konzepten dar, wobei Konzepte durch Klassen und ihre Beziehungen durch Assoziationen dargestellt werden. Verallgemeinerung und Spezialisierung von Konzepten werden durch Generalisierung („Vererbung") dargestellt.

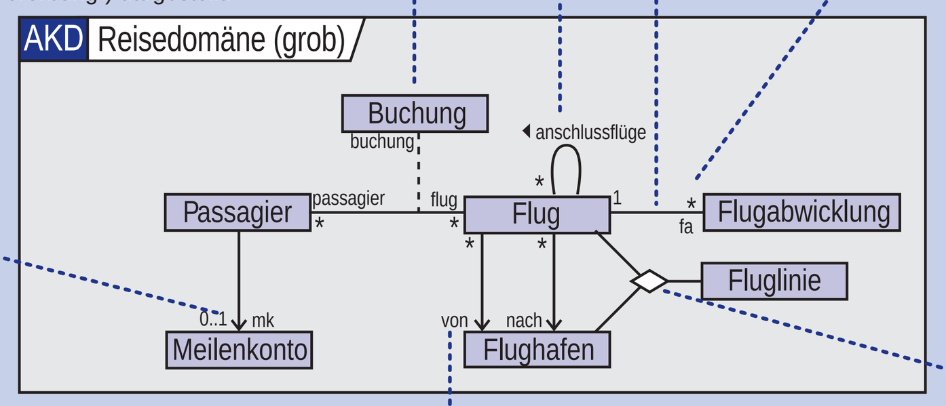
benannte ungerichtete Assoziation mit Leserichtung

Assoziationsklasse

ungerichtete Assoziation

Die Multiplizität gibt an, wie viele Objekte eines Typs an einer Assoziation teilnehmen können. Die Multiplizität wird als Intervall aus minimaler und maximaler Zahl von Teilnehmern angegeben (sind beide gleich, entfällt der obere Wert). Der Wert * gibt eine unbeschränkte Multiplizität an.

Assoziationen können durch Namen und Pfeile als **navigierbar** ausgezeichnet werden.



ternäre Assoziation

benanntes Assoziationsende (Rolle)

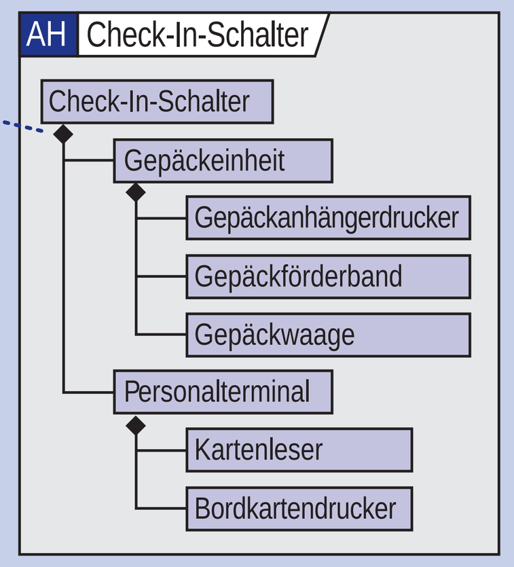
Mit einer **Aggregationshierarchie** kann der statische Aufbau verdeutlicht werden.

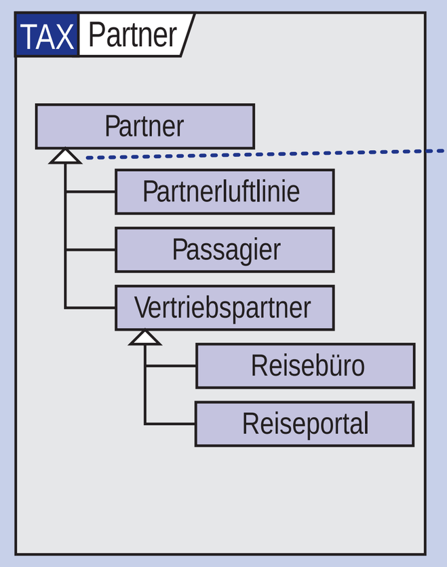
Mit einer **Taxonomie** wird eine Generalisierungs- bzw. Spezialisierungshierarchie von Begriffen aufgebaut.

Komposition ist eine Teil-Ganzes-Beziehung („by value").

Im Gegensatz dazu ist eine **Assoziation** ein reiner Verweis („by reference").

Die **Aggregation** (weiße Raute) ist undefiniert, und sollte vermieden werden.





Die **Generalisierungs-beziehung** kann in unterschiedlichen Kontexten z.B. als Generalisierung, Spezialisierung, Vererbung, oder Subtypisierung interpretiert werden.

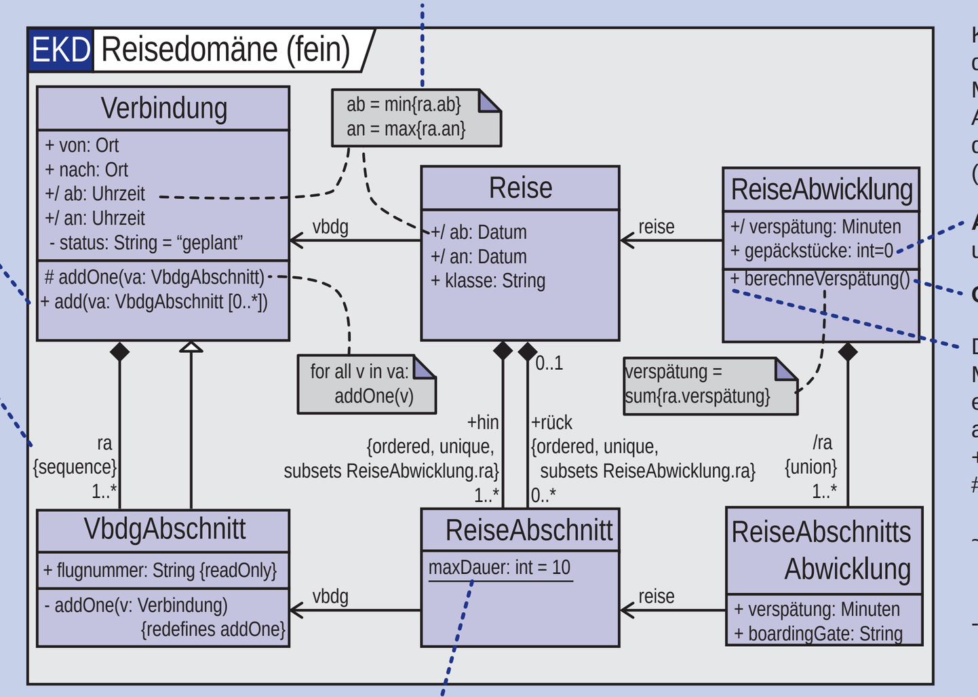
Entwurfs-Klassendiagramme stellen den Übergang von einem Analysemodell zu einem Implementationsmodell dar und dokumentieren Technologie-unabhängige Entwurfsentscheidungen.

Kommentar

Abgeleitete Merkmale werden durch einen Schrägstrich angezeigt.

Multiplizität eines Attributs

Merkmale können durch Eigenschaftswerte näher beschrieben werden.
- Die Eigenschaften **unique**/ **nonunique**, **ordered**/**un-ordered**, **seq**, **sequence**, und **bag** beziehen sich auf die Menge der Objekte in einem Merkmal,
- **readonly/unrestricted** bezieht sich auf Sichtbarkeit und Änderbarkeit,
- **composite** legt fest, dass der Teil nur in einem Ganzen existieren kann, nicht loslgelöst;
- **redefines *Attr***, **subsets *Attr***, **union** beziehen sich auf andere Merkmale der Klasse.



Klassen besitzen **Merkmale**, die sich nach statischen Merkmalen (Attribute und Assoziationen) und dynamischen Merkmalen (Operatinen) untergliedern.

Attribut mit **Typ** und **Initialwert**

Operation („Methode")

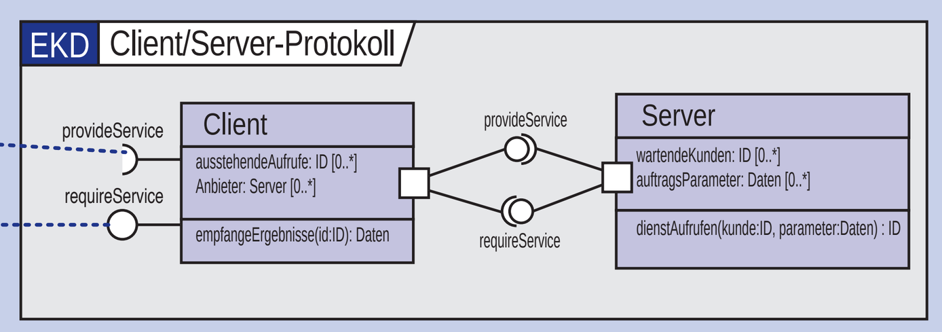
Die **Sichtbarkeit** von Merkmalen einer Klasse kann einen der folgenden Werte annehmen:
+ (public) allgemein sichtbar
# (package) innerhalb des Pakets sichtbar
~ (friend) wird durch permission-Abhängigkeit sichtbar gemacht
- (private) nur für Objekte dieser Klasse selbst sichtbar

Klassenattribute („static" in Java und C++) können für die Klasse instantiert werden.

Klassen und Anschlüsse können durch **Schnittstellen** näher beschrieben werden. Dabei kann zwischen gebotenen und genutzten Schnittstellen unterschieden werden.

genutzte **Schnittstelle** (Benutzungs-Beziehung, «uses»)

gebotene **Schnittstelle** (Realisierungs-Beziehung, «realizes»)

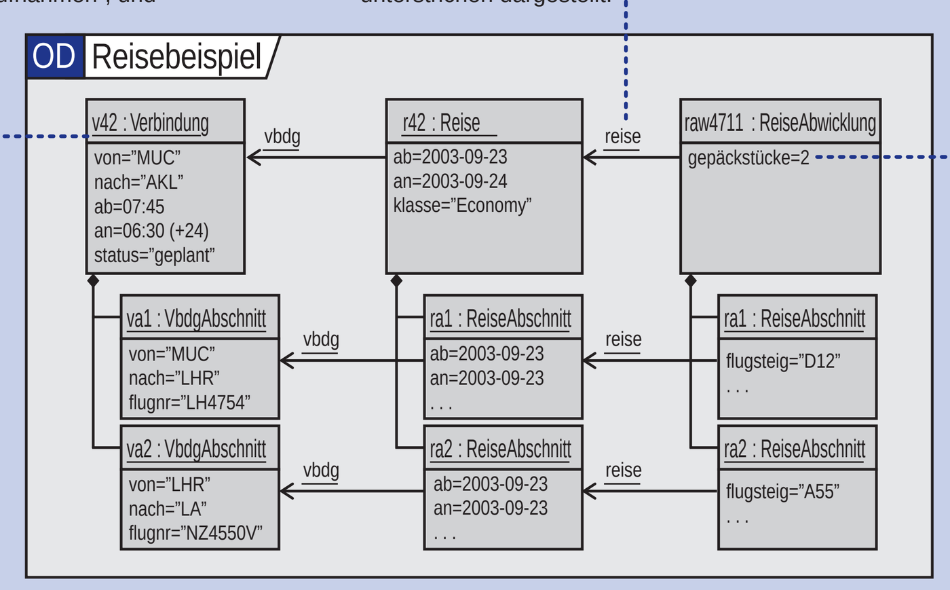


Assoziationen und Attribute sind semantisch gleichwertig, daher können beide durch die gleichen Eigenschaftswerte beschrieben werden.

Mit **Objektdiagrammen** lassen sich mögliche Instantierungen von Klassendiagrammen darstellen, also quasi „Momentaufnahmen", und damit Systemzustände.

In einem Objektdiagramm sind auch Assoziationen Instanzen, daher wird ihr Name ebenfalls unterstrichen dargestellt.

Objekte (in der UML-Terminologie „InstanceSpecification") sind Instanzen von Klassen. Sie werden durch einen (optionalen) Instanznamen vor dem Klassennamen und durch Unterstreichung angezeigt.



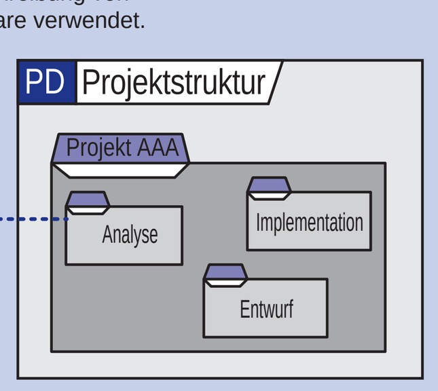
Anders als in Klassendiagrammen besitzen die Attribute in Objektdiagrammen konkrete Werte.

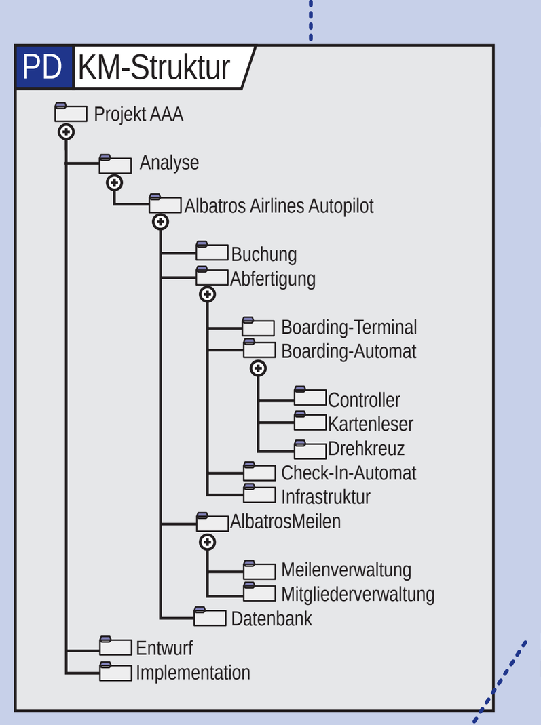
Paketdiagramme

Paketdiagramme stellen die Gruppierung von Modellelementen zu Namensräumen dar. Paketdiagramme werden gleichermaßen zur Beschreibung von Projektinfrastrukturen und Software verwendet.

Für große Aggregationshierarchien bietet sich eine baumförmige Darstellung in Anlehnung an gängige Werkzeuge an.

Pakete werden als stilisierte Hängeregistermappen mit einer Lasche links oben dargestellt. Der Name eines Pakets wird auf der Lasche oder auf der Mappe selbst dargestellt.

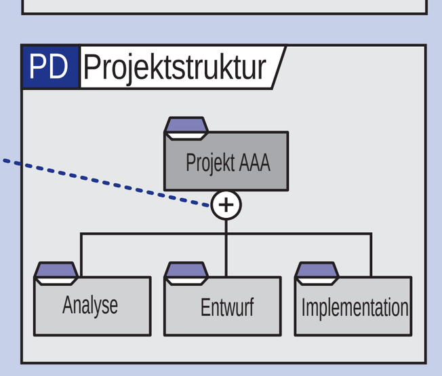




Pakete können geschachtelt sein, was entweder durch die graphische Schachtelung (links) oder die Aggregationsassoziation (rechts) angezeigt wird.

In einem Paket sind alle Elemente **sichtbar** die
- in diesem Paket **definiert** werden;
- aus einem anderen Paket **importiert** werden;
- in einem übergeordneten (!) Paket **sichtbar** sind.

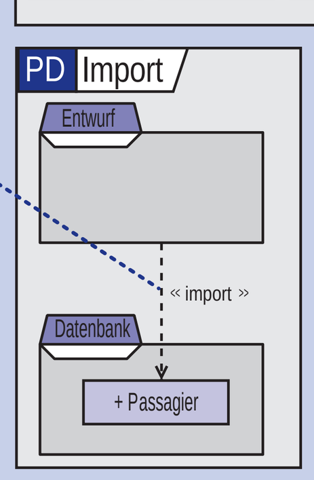
Die **Aggregationsbeziehung** für Pakete ähnelt der Komposition von Klassen

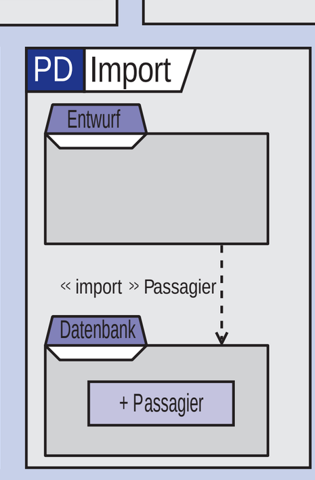


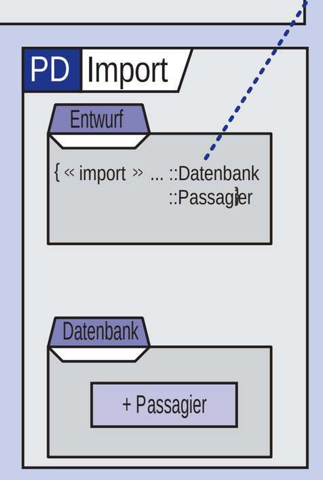
Verschiedene Pakete können durch **Importbeziehungen** miteinander in Beziehung gesetzt werden.

Die Bestandteile **qualifizierter Namen** werden durch „::" voneinander getrennt.

Um gezielt einzelne Elemente (z.B. Klassen oder Pakete) sichtbar (und damit zugreifbar) zu machen, kann der **Einzel-import** benutzt werden. Im Bild rechts sind drei notationelle Varianten von Einzelimport dargestellt: das Paket Entwurf kann jeweils auf das Element Passagier zugreifen.







Um den gesamten Inhalt eines Pakets auf einen Schlag sichtbar zu machen, kann der **Pauschalimport** benutzt werden. Er wird angezeigt durch eine Beziehung auf ein exportierendes Paket mit dem Stereotyp «**import**» aber ohne qualifizierenden Elementnamen (nicht dargestellt).

Soll ein Element nicht weiter exportiert werden dürfen, wird es mit «**access**» importiert.

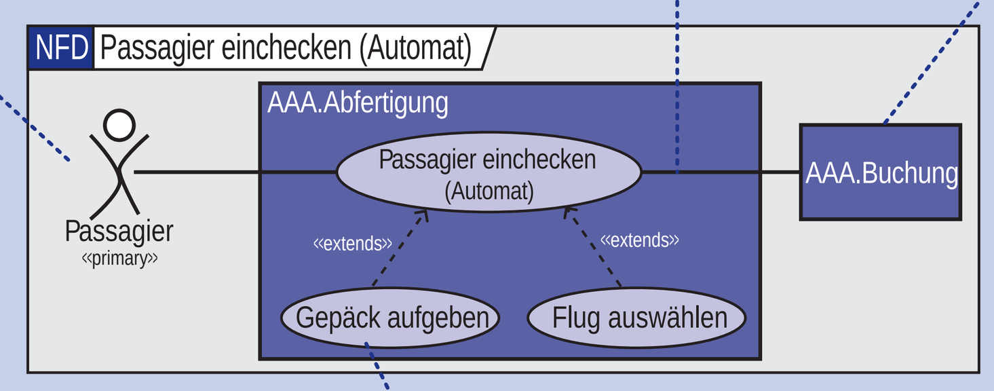
Nutzfalldiagramme

Nutzfalldiagramme sind vor allem als Übersichtsdarstellungen nützlich. Sie stellen einerseits Elemente eines Systemkontextes dar (also Akteure und Nachbarsysteme), andererseits Geschäftsprozesse bzw. Nutzfälle des Systems. Daneben können auch Abhängigkeiten zwischen Geschäftsprozessen bzw. Nutzfällen mit Inklusions- und Erweiterungsbeziehungen angezeigt werden.

Die **Assoziation** zeigt an, dass der Akteur bzw. das Nachbarsystem an diesem Geschäftsprozess teilnimmt.

Nachbarsystem

Der **Akteur** Passagier ist Teil des Systemkontextes. Das Stereotyp «primary» drückt aus, dass dieser Akteur den Geschäftsprozess auslöst.

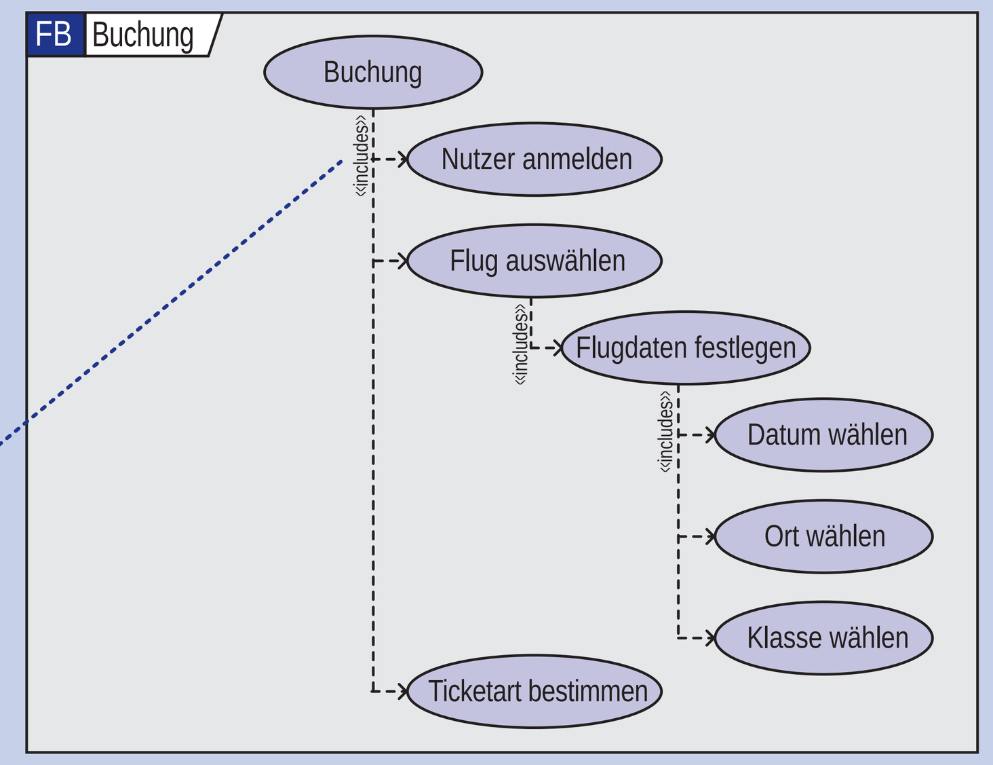


Das Layout eines Nutzfalldiagramms sollte die Leserichtung ausnutzen, und nach Möglichkeit den auslösenden Akteur links darstellen, und aufgerufene Teil- bzw. eingebundene Sonderfälle unten.

Eine optionale Variante eines Geschäftsprozesses bzw. Nutzfalles kann durch einen **Erweiterungsfall** („Sonderfall") modelliert werden. Im zu erweiternden Geschäftsprozess bzw. Nutzfall wird dazu ein **Erweiterungspunkt** definiert (hier nicht dargestellt), an dem der Sonderfall durch eine **Erweiterungs-Beziehung** angebunden wird.

Ein **Funktionsbaum** stellt die Aufrufabhängigkeiten zwischen Geschäftsprozessen bzw. Notfällen dar. Diese Darstellung ist vor allem in der Dokumentation geeignet, weniger zum Entwurf.

Analog zu Klassen können auch für Akteure und Nutzfälle Kompositionshierarchien und Taxonomien gebildet werden.



Durch das Layout des Funktionsbaums kann die **Reihenfolge** der Aufrufe (teilweise) wiedergegeben werden. Durch die Einrückung kann gleichzeitig die **Hierarchieebene** des Aufrufs angezeigt werden.

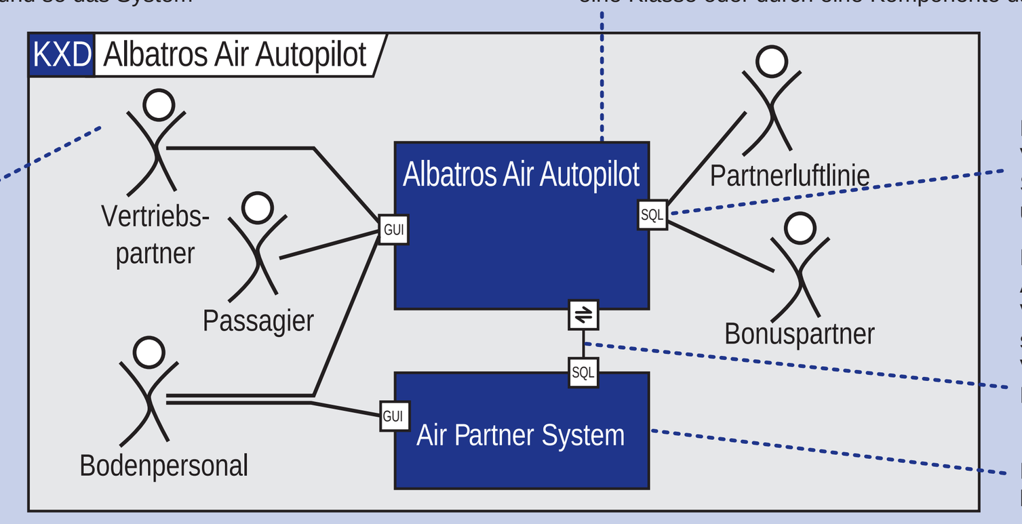
Ein Teilprozess bzw. eine Teilfunktion, die einmal oder mehrmals im Verlauf eines Geschäftsprozesses bzw. Nutzfalles aufgerufen werden kann, wird durch eine **Inklusions-Beziehung** dargestellt.

Montagediagramme

Ein **Kontextdiagramm** dient dazu, die Umgebung eines Systems zu definieren, und so das System abzugrenzen.

Ein **System** in einem Kontextdiagramm kann je nach Anwendung durch eine Klasse oder durch eine Komponente dargestellt werden.

Ein **Akteur** ist eine Rolle, die mit einem System interagiert. Diese Rolle kann von einem Menschen, einer Gruppe oder Organisation, aber auch von einer Maschine bzw. einem Nachbarsystem übernommen werden (in dem Fall empfiehlt es sich, eine stereotypisierte Notation zu verwenden).



Ein **Anschluss** stellt die Verbindung zwischen zwei Systemen bzw. einem System und einem Akteur dar.

Ein **Verbinder** verbindet Anschlüsse von Systemen. Verbinder können rein passiv sein oder mit eigenem Verhalten modelliert sein (z.B. Fehler oder Verzögerung).

Der Begriff **Nachbarsystem** bezieht sich immer auf ein „Haupt"-System, und beschreibt oft dessen Einbettung in existierende Systemlandschaften.

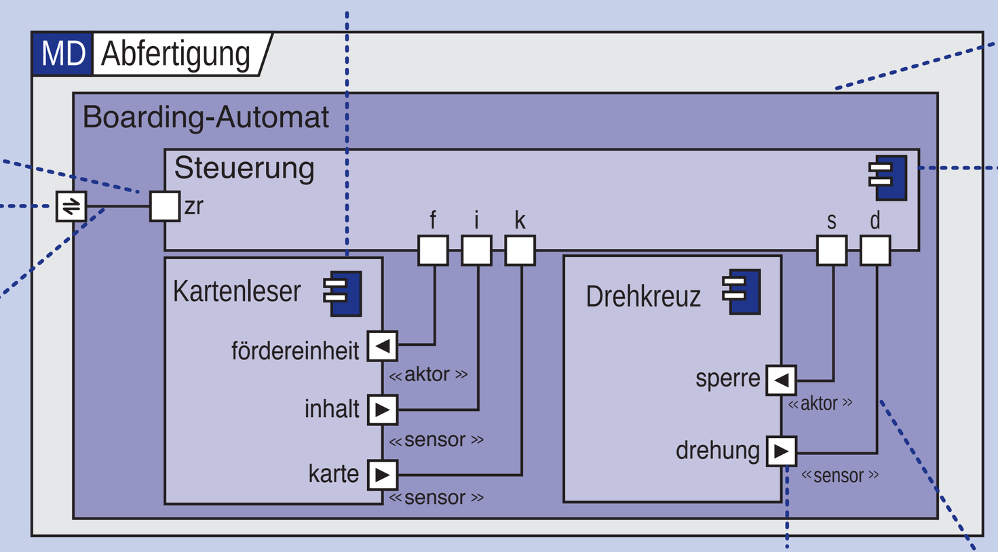
Mit einem **Montagediagramm** kann die Grobstruktur eines Systems beschrieben werden. In dieser Verwendung lassen sich auch Hardware-nahe Strukturen gut darstellen (so begründet sich z.B. auch der Name „Anschluss").

Das Verhalten der Komponente „Kartenleser" ist im gleichnamigen Zustandsautomaten rechts unten beschrieben, die Ausdrücke dort beziehen sich auf die Ports hier.

innerer Anschluss

Ein **Relaisanschluss** leitet Signale unverändert weiter (im Gegensatz zu einem **Transponderanschluss**).

Delegierungsverbinder verbinden innere und Relaisanschlüsse.



Genauso ist auch **Subsystem** ein relativer Begriff: Subsysteme können ihrerseits aus Subsystemen bestehen.

Komponenten werden durch Rechtecke mit dem Komponentensymbol in der rechten oberen Ecke dargestellt. Eine Komponente in UML 2 stellt einen „modularen Teil eines Systems dar und kapselt ihren Inhalt". Eine Komponente durchläuft den gleichen Lebenszyklus wie Software im Allgemeinen, ist also eine Einheit des Entwurfs und der Herstellung wie der Installation und Nutzung.

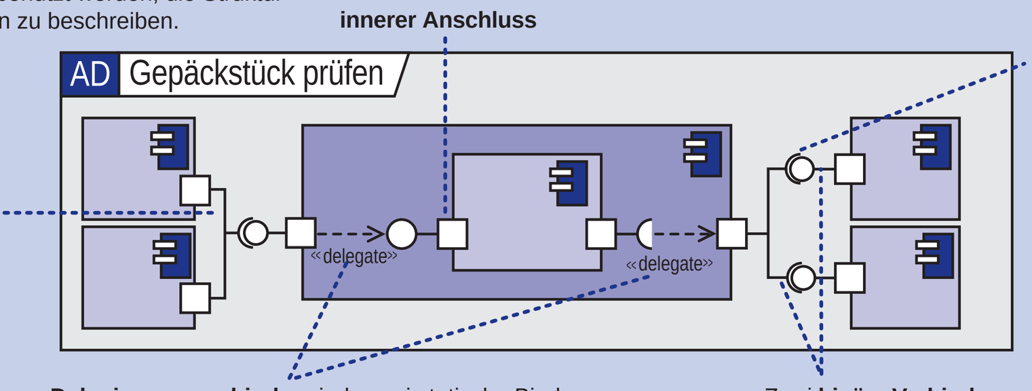
Unidirektionaler Anschluss

Verbinder

Auf einem noch feineren Detailgrad können **Montagediagramme** dazu benutzt werden, die Struktur zusammengesetzter Klassen zu beschreiben.

innerer Anschluss

Ein **ternärer Verbinder**



Anschlüsse können durch gebotene und genutzte Schnittstellen charakterisiert werden.

Delegierungsverbinder sind quasi statische Bindungen.

Zwei **binäre Verbinder**

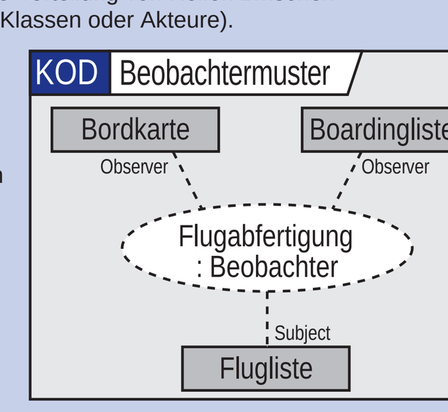
Kollaborationen

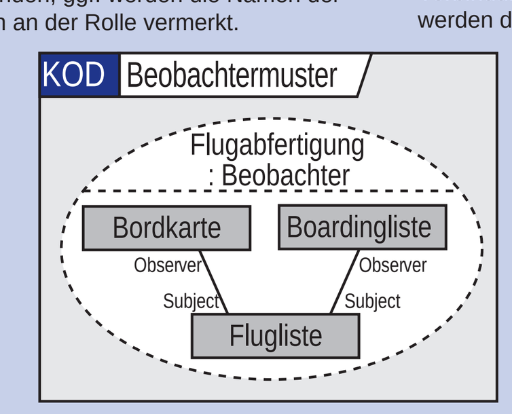
Kollaborationen definieren einen strukturellen Zusammenhang und eine Verteilung von Rollen zwischen einigen Elementen (z.B. Klassen oder Akteure).

Die **Rollen** der Kollaboration werden mit dieser durch gestrichelte Linien verbunden, ggf. werden die Namen der Rollen an der Rolle vermerkt.

Alternativ können die Rollen innerhalb der Kollaboration angezeigt werden, Verbindungen zwischen den Rollen werden durch Assoziationen dargestellt.

Insbesondere in der Dokumentation der Anwendung von Architekturstilen und Entwurfsmustern werden Kollaborationen eingesetzt.





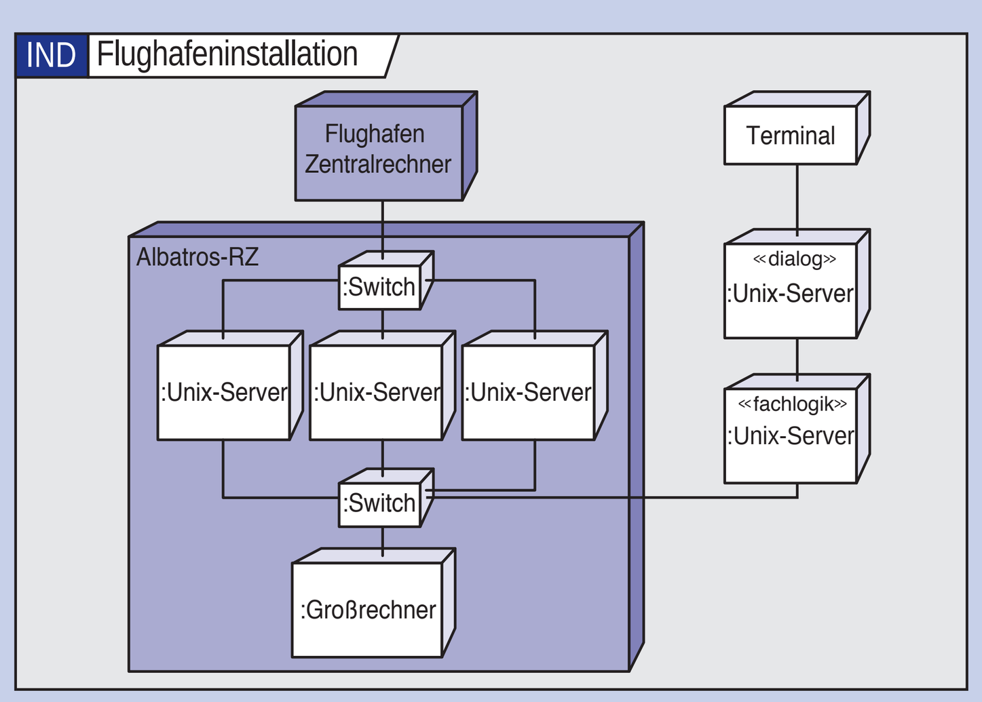
Eine Kollaboration wird als gestrichelte Ellipse dargestellt, in der die Namen der Instanz und des Typs der Kollaboration dargestellt werden.

Installationsdiagramme

Systemstrukturdiagramme stellen den physischen Aufbau eines Systems dar, also die Rechner, Netzwerke usw., die in einer Installation vorkommen.

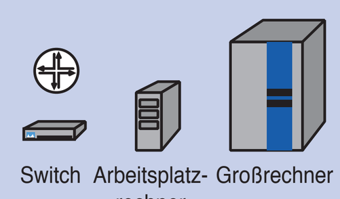
Installationsdiagramme können auf Typ- und auf Instanzebene beschrieben werden.

Ein Knoten repräsentiert einen Rechner oder eine sonstige aktive Komponente in einer Installation. Knoten können ineinander geschachtelt werden.



Eine Leitung repräsentiert eine Verbindung zwischen Knoten.

In Installationsdiagrammen werden oft sehr viele visuelle Stereotype wie die folgenden verwendet.

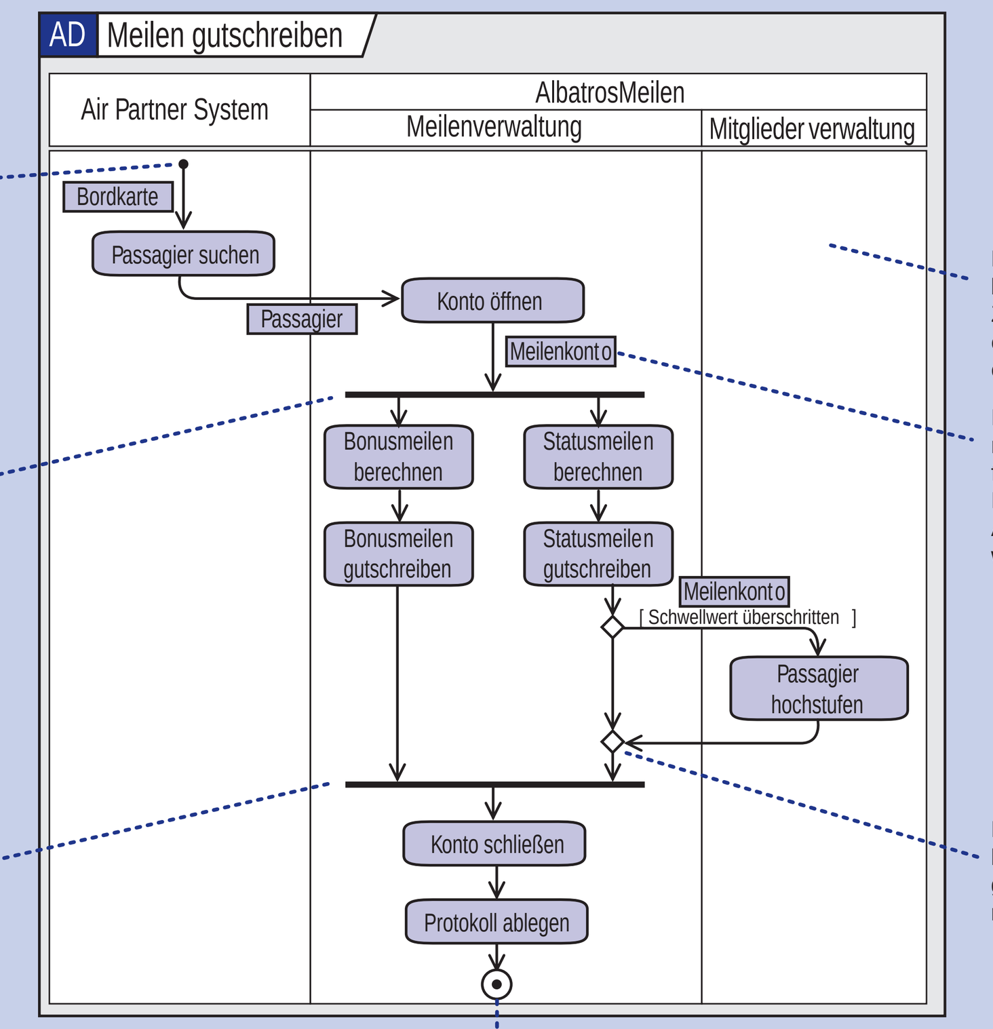


Aktivitätsdiagramme

Ein **Aktivitätsdiagramm** kann zur Modellierung von Abläufen jeder Art benutzt werden, z.B. für Geschäftsprozesse, Nutzfälle oder algorithmische Abläufe.

Ein **Anfangsknoten** markiert den Beginn eines Ablaufs.

Ein **Verzweigungsknoten** spaltet einen Kontrollfluss in mehrere auf. Die Verzweigung kann durch eine logische Bedingung gesteuert werden (z.B. or).



Eine **Partition** („Schwimmbahn" in UML 1.x) stellt einen Zuständigkeitsbereich dar, etwa ein Subsystem oder einen Akteur.

Ein **Objektflussknoten** repräsentiert Daten oder Artefakte (z.B. Dokumente, Briefe, Akten), die von einer Aktion zur nächsten geschickt werden.

Ein **Vereinigungsknoten** führt mehrere Kontrollflüsse zu einem zusammen. Die Vereinigung kann durch eine logische Bedingung gesteuert werden (z.B. or).

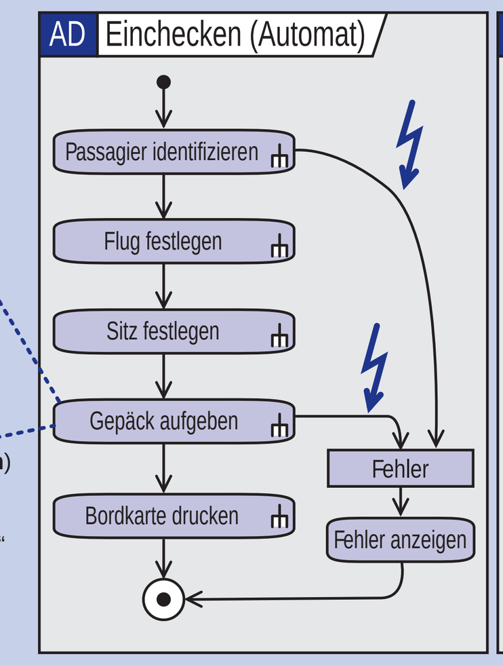
Mit **Fallunterscheidungs-knoten** werden Verzweigungen und Zusammenflüsse modelliert.

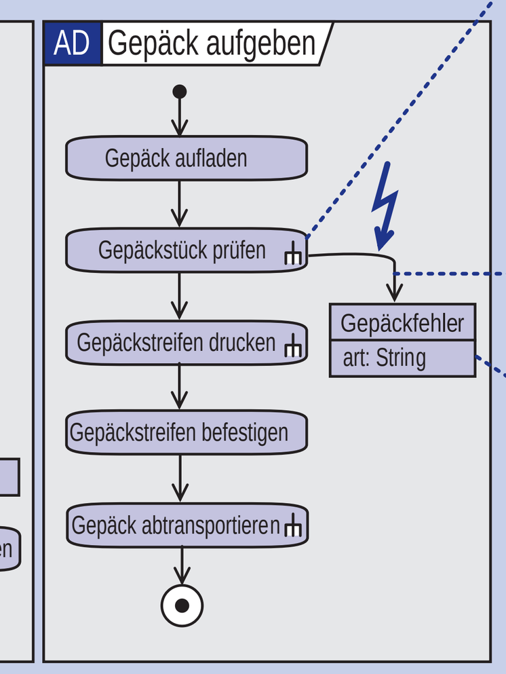
Ein **Endknoten** markiert das Ende einer Aktivität.

Aktivitätsdiagramme können auch dazu benutzt werden, um Abläufe z.B. von Geschäftsprozessen schrittweise zu verfeinern.

Eine **verfeinerte Aktion** wird durch ein Symbol in der rechten unteren Ecke angedeutet. In diesem Fall ist die Verfeinerung die Aktivität im Diagramm rechts außen.

Im Zustand „Gepäckstück aufgeben" (**geschützter Knoten**) kann eine **Ausnahme** des Typs „Fehler" ausgelöst werden, die von der Aktion „Fehler anzeigen" behandelt wird.





Im Zustand „Gepäckstück aufgeben" kann eine **Ausnahme** des Typs „Gepäckfehler" ausgelöst werden, die innerhalb der Aktivität „Gepäck aufgeben" nicht behandelt, und also zum Aufrufer der Aktivität propagiert wird (hier die gleich- namige Aktion in der Aktivität „Einchecken (Automat)").

Eine **Ausnahmekante** wird durch ein Blitzsymbol gekennzeichnet. Ein Knoten, von dem eine Ausnahmekante ausgeht, heißt „geschützter" Knoten.

Ein **Objektflussknoten** ist ein temporärer Speicher für Datenpakete, die zwischen Aktionen fließen.

Ein **Pin** ist ein notwendiger Parameter für den Aufruf einer Aktion, bzw. ein garantiertes Resultat.

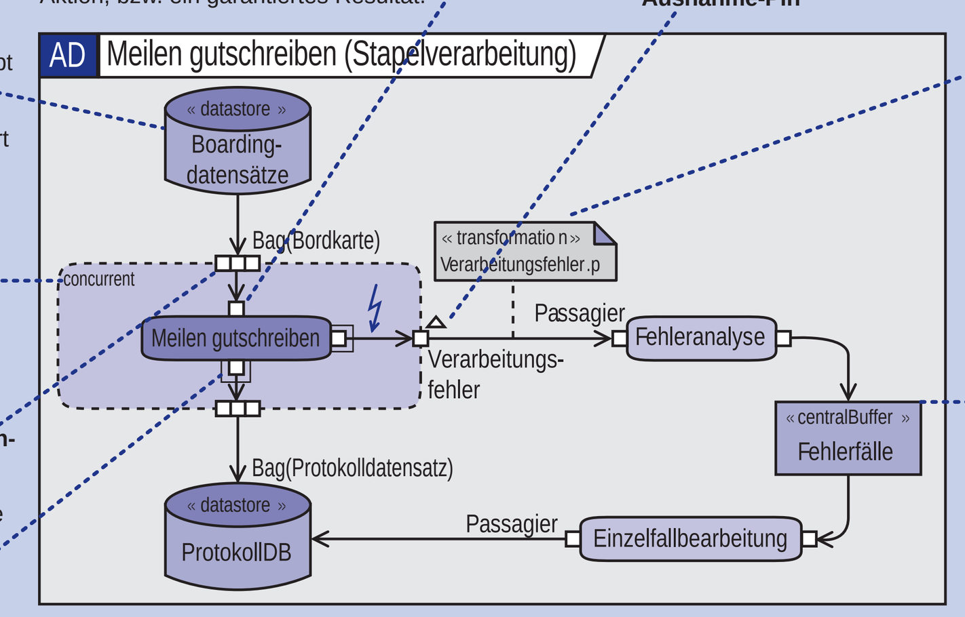
Ausnahme-Pin

Ein **Datenspeicher**(knoten) speichert alle in ihn abgelegten Datenpakete permanent, und gibt nur jeweils Kopien wieder ab.

Ein **Auffaltungsbereich** definiert einen Bereich einer Aktivität, in dem Massendaten nach verschiedenen Modi en gros verarbeitet werden:
- **concurrent**: gleichzeitig
- **iterative**: nacheinander
- **stream**: zeitlich überlappend nacheinander.

Die Parameter von Auffaltungsbereichen werden durch **Mengen-daten-Pins** markiert.

Wenn nicht alle Pins notwendige Parameter sind, kann eine Teilmenge davon als **Parameter-menge** ausgezeichnet werden.



Transitionen können durch eine **Transformationsregel** ergänzt werden, um den Dateninhalt zu manipulieren (hier: Auswahl eines Attributs). Analog dazu gibt es **Selektionsregeln** an Transitionen, die von Datenknoten ausgehen. Sie erlauben es, zwischen mehreren verfügbaren Datenpaketen zu wählen.

Ein **Datenpuffer**(knoten) speichert Datenpakete temporär, und dient vor allem als Datenweiche.

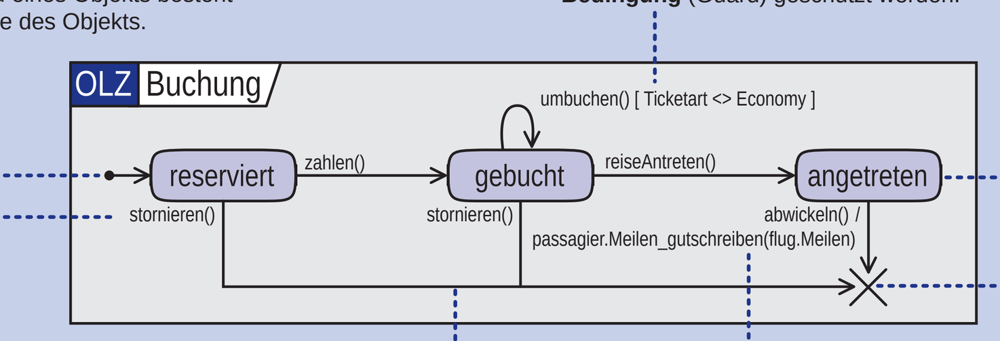
Zustandsautomaten

Ein **Objektlebenszyklus** beschreibt das Verhalten aller Objekte einer Klasse, von der Instantierung zur Termination. Der Zustand eines Objekts besteht aus den Werten aller Attribute des Objekts.

Ein **Zustandsübergang** kann durch eine **Bedingung** (Guard) geschützt werden.

Anfangszustand

Ein **Ereignis** als **Auslöser** eines Zustandsübergangs



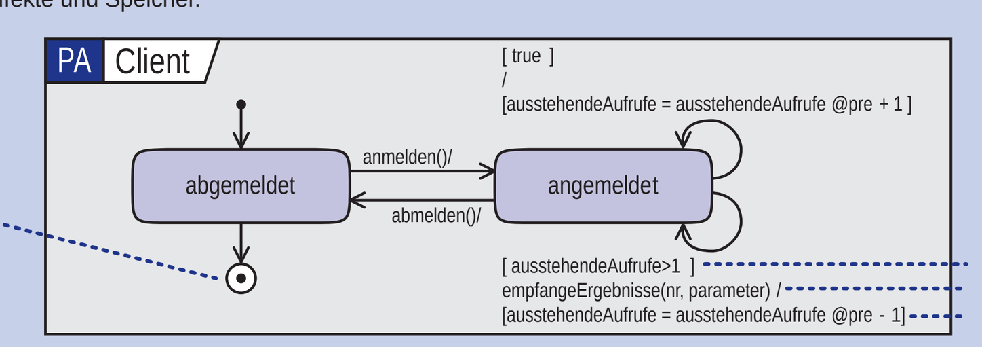
Zustand

Wenn der **Terminator-zustand** eines Zustandsautomaten erreicht wird, wird nicht nur der Zustandsautomat beendet, sondern das Objekt gelöscht, zu dem er gehört.

Ein **Zustandsübergang** kann als **Effekt** eine **Aktion** auslösen, z.B. eine Zustandsänderung des Objekts, dessen Lebenszyklus beschrieben wird.

Ein **Protokollzustandsautomat** beschreibt ein Verhalten ausschließlich über Zustände und Übergänge, aber ohne Seiteneffekte und Speicher.

Wenn der **Endzustand** eines Zustandsautomaten erreicht wird, wird die Region beendet, in der der Endzustand liegt.



Die Zustandsübergänge von Protokoll-Zustandsautomaten verfügen über eine **Vorbedingung**, einen **Auslöser** und eine **Nachbedingung** (alle optional) – jedoch nicht über einen Effekt.

Protokollzustandsautomaten beschreiben das Verhalten von Softwaresystemen, Nutzfällen oder technischen Geräten.

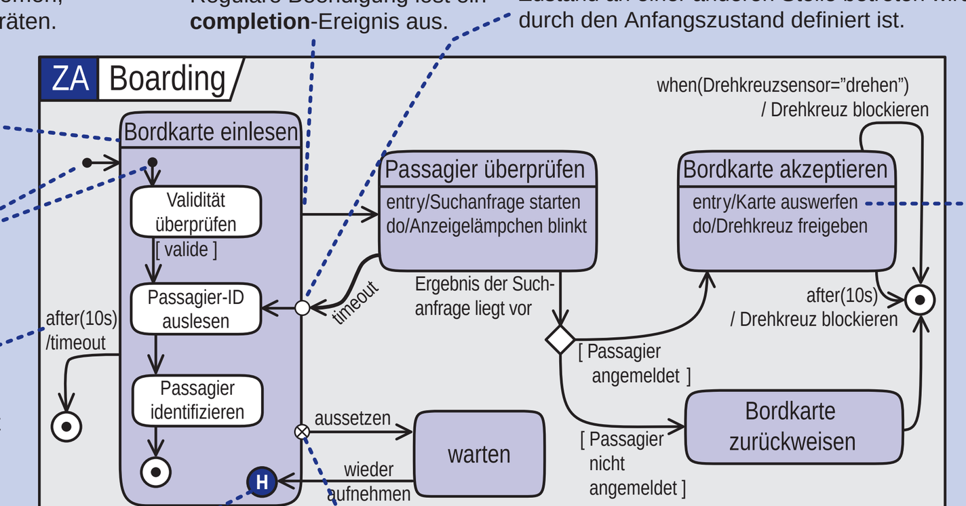
Reguläre Beendigung löst ein **completion**-Ereignis aus.

Ein **Eintrittspunkt** definiert, dass ein komplexer Zustand an einer anderen Stelle betreten wird, als durch den Anfangszustand definiert ist.

Ein **komplexer Zustand** mit einer Region.

Der **Anfangszustand** markiert den voreingestellten Startpunkt von „Boarding" bzw. „Bordkarte einlesen".

Das **Zeitereignis** after(10s) löst einen Abbruch von „Bordkarte einlesen" aus.



Ein Zustand löst von sich aus bestimmte Ereignisse aus:
- **entry** beim Betreten;
- **do** während des Aufenthaltes;
- **completion** beim Erreichen des Endzustandes einer Unter-Zustandsmaschine
- **exit** beim Verlassen.

Diese und andere Ereignisse können als Auslöser für Aktivitäten herangezogen werden.

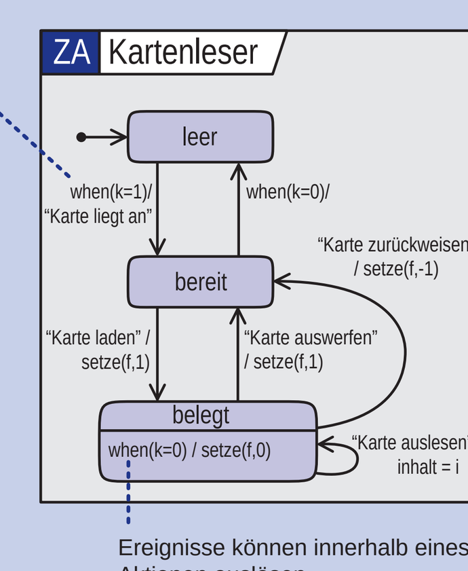
Der **Gedächtniszustand** sorgt dafür, dass nach dem Wiederaufnehmen der gleiche Zustand wie vor dem Aussetzen eingenommen wird.

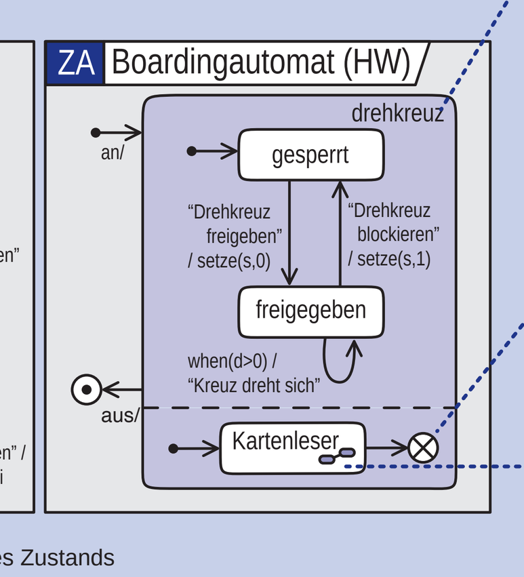
Der **Austrittspunkt** erlaubt es, von einem definierten inneren Zustand aus den Oberzustand zu verlassen.

Auch Zeit- und Änderungsereignisse können Zustandsübergänge auslösen:

- **after** definiert das Verstreichen eines Intervalls;
- **when** definiert einen Zustandswechsel.

Zustände und zeitlicher Bezugsrahmen werden über den umgebenden Classifier definiert, hier die Werte der Ports, siehe das Montagediagramm „Abfertigung" links oben.





Ereignisse können innerhalb eines Zustands Aktionen auslösen.

Ein Zustand kann eine oder mehrere **Regionen** enthalten, die wiederum Zustandsautomaten enthalten können. Wenn ein Zustand mehrere Regionen enthält, werden diese in verschiedenen Abteilen angezeigt, die durch gestrichelte Linien voneinander getrennt sind. Regionen können benannt werden. Alle Regionen werden parallel zueinander abgearbeitet.

Wenn ein **Regionsend-zustand** erreicht wird, wird der gesamte *komplexe* Zustand beendet, also auch alle parallelen Regionen.

Ein **verfeinerter Zustand** verweist auf einen Zustandsautomaten (angedeutet von dem Symbol unten links), der das Verhalten des Zustandes definiert.

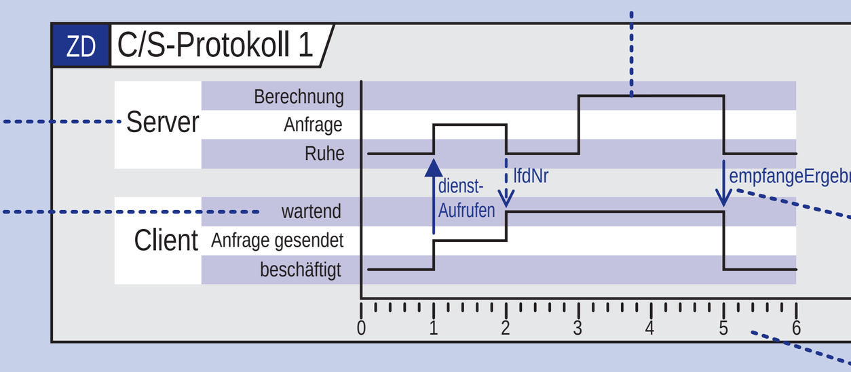
Zeitdiagramme

Zeitdiagramme sind eine Form von Interaktionsdiagrammen, die besonders zur Modellierung und Visualisierung komplexer zeitlicher Abläufe geeignet sind.

Ähnlich zu Sequenzdiagrammen besitzen auch Zeitdiagramme eine Art **Lebenslinie**, die hier aber auch den Zustand eines Interaktionspartners zu einem Zeitpunkt ausdrückt.

Interaktionspartner

Zustände eines Interaktionspartners



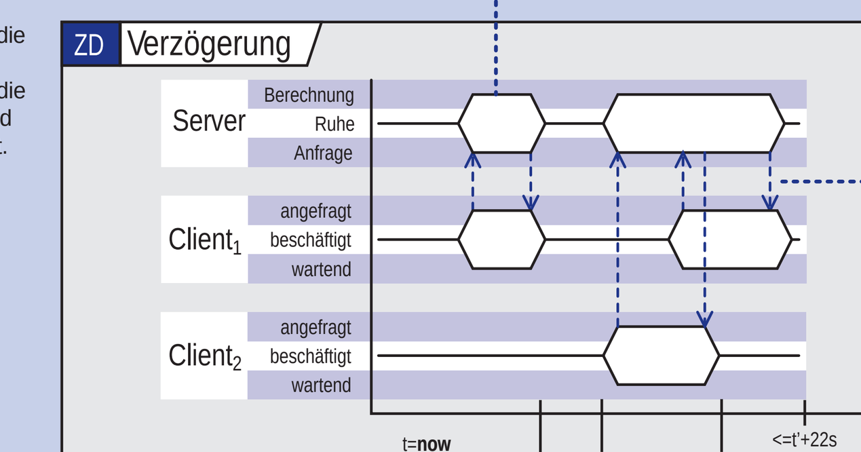
Nachrichtenaustausch zwischen Interaktionspartnern

Im Unterschied zu Sequenzdiagrammen besitzen Zeitdiagramme eine echte **Zeitskala**, hier eine absolute (metrische) Skala.

Um zeitliche Abläufe stärker abstrahiert darzustellen, können Abschnitte eines zeitlichen Ablaufes zu Phasen zusammengestellt werden.

Wenn während eines Zeitabschnitts verschiedene Zustände eingenommen werden können, ohne dass zwischen ihnen unterschieden werden soll, kann dies durch eine **Phase** ausgedrückt werden.

Mit Zeitdiagrammen lässt sich die **Koordinierung verteilter Zustände** modellieren, sofern die Zahl der Interaktionspartner und ihrer Zustände nicht zu groß ist.



Nachrichtenaustausch zwischen Interaktionspartnern

Zeitbedingungen können als Zeitpunkte oder als Intervalle, und absolut oder relativ angegeben werden. Um einen Zeitpunkt als Bezugspunkt auszuwählen, kann das Schlüsselwort **now** benutzt werden.

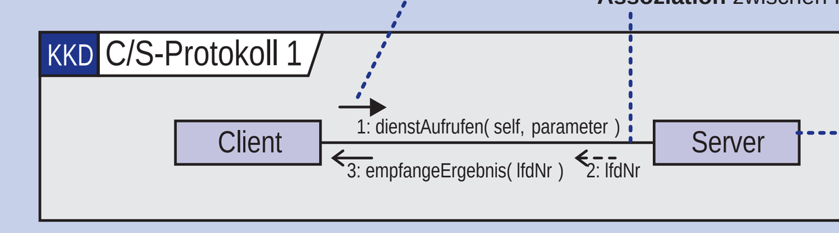
Statt einer absoluten, kann in einem Zeitdiagramm auch eine **relative Zeitskala** verwendet werden. Darauf werden nur einzelne Punkte definiert, ggf. benannt, und in einschränkenden Ausdrücken benutzt.

Kommunikationsdiagramme

Kommunikationsdiagramme („Kollaborationsdiagramme" in UML 1.x) sind eine Form von Interaktionsdiagrammen, die besonders gut dann geeignet sind, wenn wenige Nachrichten ausgetauscht werden, aber die Beziehungen zwischen den Interaktionspartnern hervorgehoben werden sollen.

Die **Reihenfolge** von Nachrichten in einem Kommunikationsdiagramm kann lediglich durch **Sequenznummern** dargestellt werden. Richtung und Typ der Nachricht werden durch einen **kleinen Pfeil** angezeigt

Assoziation zwischen Interaktionspartnern



Als **Interaktionspartner** kann jeder Classifier dienen, z.B. eine Klasse, ein Akteur oder ein Knoten.

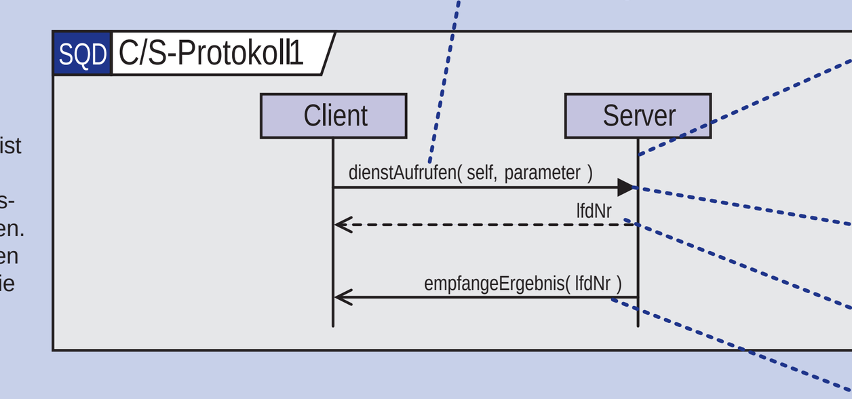
Sequenzdiagramme

Sequenzdiagramme sind eine Form von Interaktionsdiagrammen, die besonders gut dann geeignet sind, wenn wenige Interaktionspartner viele Nachrichten austauschen, und wenn komplexe Muster des Nachrichtenaustauschs zu modellieren sind.

Die Menge der **möglichen Nachrichten** an einen Interaktionspartner wird durch dessen Signatur bestimmt. In diesem Fall muss es z.B. in der Klasse „Server" eine für „Client" sichtbare Operation „dienstAufrufen" mit den entsprechenden Parametern geben.

In einem Sequenzdiagramm wird für jeden Interaktionspartner eine **Lebenslinie** angezeigt. Nachrichten zwischen den Interaktionspartnern werden durch Pfeile zwischen den Lebenslinien dargestellt.

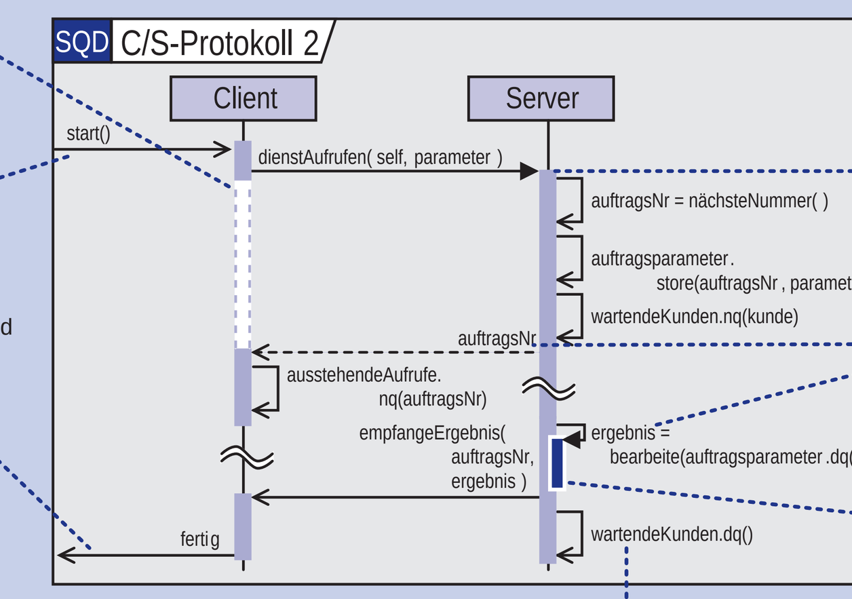
Das Sequenzdiagramm rechts ist **bedeutungsgleich** mit den gleichnamigen Kommunikations- und Zeitdiagrammen weiter oben. Die drei Diagramme beschreiben exakt die gleiche Interaktion. Die drei Diagrammtypen sind aber nicht austauschbar.



Der **Typ einer Nachricht** wird durch die Form des Pfeils angezeigt:
- **Aufrufe** sind synchron, d.h., die Aktivierung des sendenden Interaktionspartners wird ausgesetzt;
- **Antworten** reaktivieren die Aktivierung des empfangenden Interaktionspartners;
- **synchrone Nachrichten** lassen die Aktivierung unverändert.

ausgesetzte Aktivierung

Anfang und **Ende** einer Interaktion können durch Nachrichten von bzw. zum Rand des Interaktionsdiagramms dargestellt werden.



Ein **Aktivierungsbalken** zeigt an, dass die Kontrolle bei einem Interaktionspartner liegt, bzw. wie tief Aufrufe geschachtelt sind.

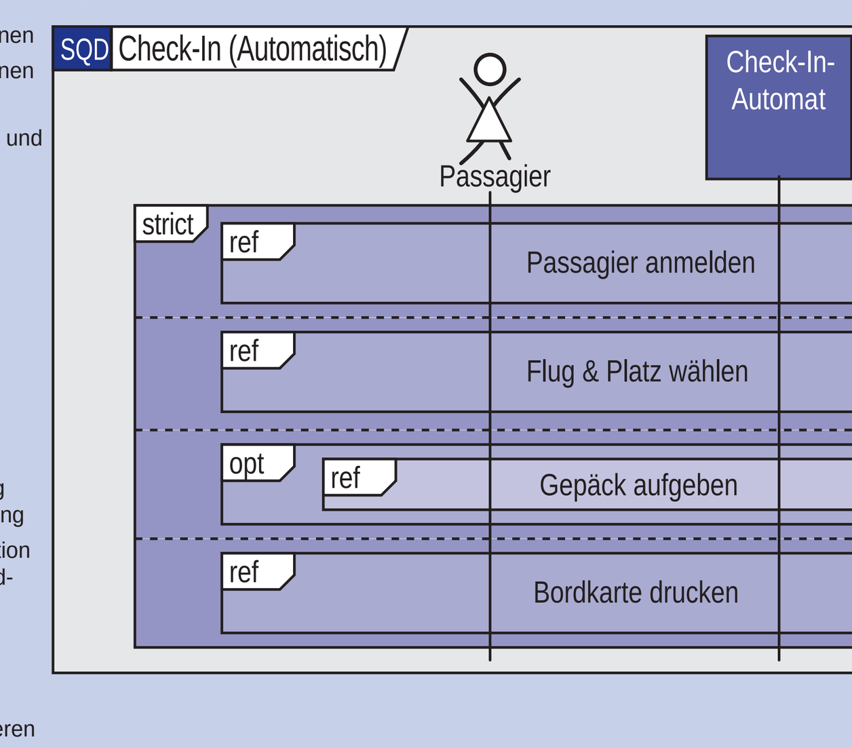
Das **Resultat** eines Aufrufs wird typischerweise in einem Attribut abgelegt. Dieses Attribut kann an der Antwort-Nachricht oder direkt im Aufruf angegeben werden.

Bei einem **Selbstaufruf** werden Aktivierungsbalken übereinander gelegt.

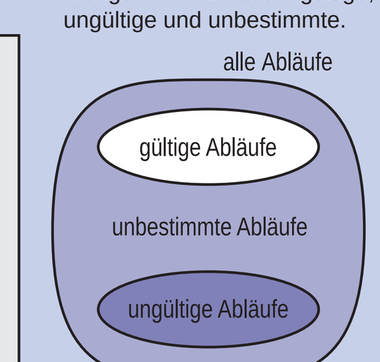
Bei einer synchronen Nachricht wird dagegen **kein Resultat** erwartet.

Komplexe Interaktionen können modular durch **Interaktionsoperatoren** aufgebaut werden. Folgende Operatoren werden von der UML bereitgestellt.

strict	Abfolge von Interaktionen
seq	Abfolge von Interaktionen je Interaktionspartner
loop	Schleife mit minimaler und maximaler Anzahl von Durchläufen
par	Parallele Ausführung (Interleaving)
critical	Ununterbrochene Ausführung („kritische Region")
ref	Referenz auf eine Interaktion
opt	optionale Ausführung
alt	alternative Ausführung nach Fallunterscheidung
brk	Abbruch einer Interaktion und Ausnahmebehandlung
ignore	Ausblenden von Nachrichten
consider	dual zu **ignore**, Ausblenden aller anderen Nachrichten



Eine Interaktion definiert drei Mengen von Abläufen: gültige, ungültige und unbestimmte.



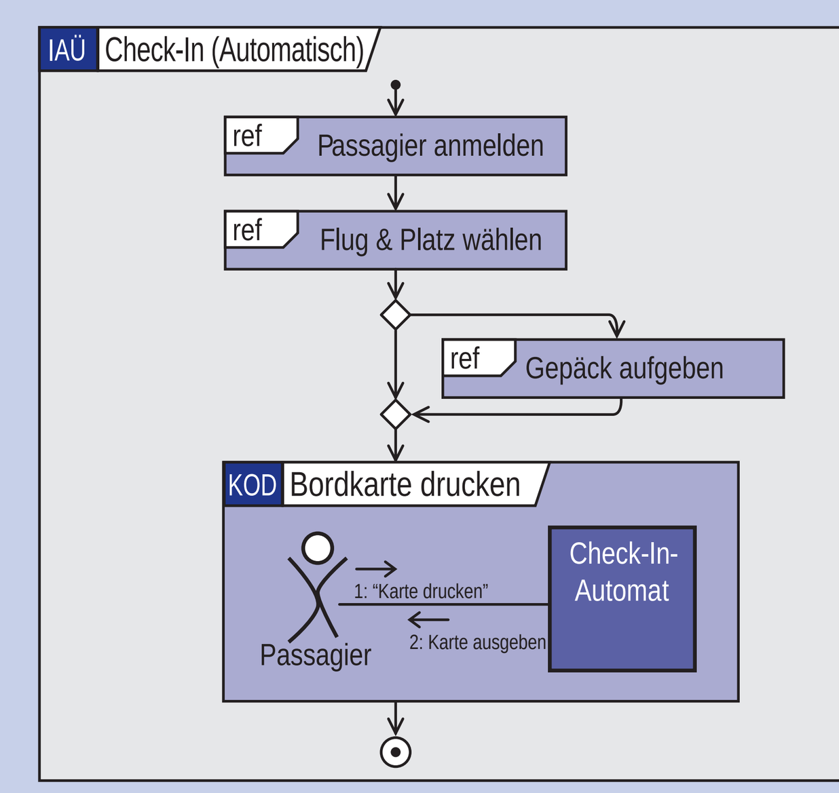
Unbestimmte Abläufe entsprechen nicht-durchgeführten Testfällen, ungültige Abläufe kann man als Gegenbeispiele verstehen.

assert	Ausschluss von Unbestimmtheit, d.h. unbestimmte Abläufe werden als ungültig definiert
neg	Definition von ungültigen Abläufen

Interaktionsübersichten

Wenn man den Zusammenhang zwischen vielen Interaktionen graphisch darstellen will, kann man **Interaktionsübersichten** nutzen. Hier können einige Ausdrucksmittel von Aktivitätsdiagrammen benutzt werden, die Semantik bleibt aber die gleiche wie bei Interaktionen.

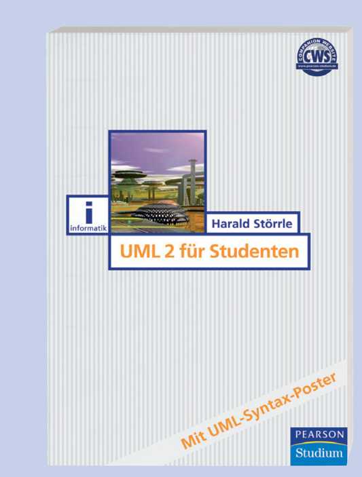
Im einfachsten Fall enthält eine Interaktionsübersicht lediglich Verweise auf Interaktionen, die anderswo definiert sind. Es ist aber auch möglich, diese Interaktionen direkt innerhalb der Interaktionsübersicht, also „in-line"zu definieren. Hierbei kann natürlich jede Art von Interaktionsdiagramm benutzt werden.



Folgende Ausdrucksmittel von Aktivitätsdiagrammen sind zulässig:
- **Aufruf** von Interaktionen
- **Einbettung** von Interaktionen
- **Fallunterscheidung** (entspricht **par**);
- **Parallelbearbeitung** (entspricht **alt**).

Andere Ausdrucksmittel sind nicht gestattet!

Die Diagramme dieses Posters stammen aus dem Buch
UML 2 für Studenten
Harald Störrle
Pearson Studium 2005
ISBN 3-8273-7143-0
€ 19,95 [D], € 20,60 [A]



www.pearson-studium.de

PEARSON Studium