

**Andreas de Vries**



Version: 4.10.2001

Dieses Skript ist geistiges Eigentum von Andreas de Vries. Es unterliegt der GNU *General Public License*.

Es ist daher frei zur nichtkommerziellen Nutzung und darf zur nichtkommerziellen Nutzung als Ganzes oder in Auszügen kopiert werden, vorausgesetzt, dass sich dieser Copyright-Vermerk auf jeder Kopie befindet.

**Danksagung.** Für wertvolle Hinweise möchte ich bei meinen Kollegen Bernd Blümel und Ralf Berning (beide FH Bochum) bedanken.

# Inhalt

- 1 Einführung 3**
- 2 Die UML-Notation mit Hilfe eines Fallbeispiels 5**
  - 2.1 Die Wissensträger 5
  - 2.2 Das Fallbeispiel 6
- 3 Modellierung von Geschäftsprozessen 7**
  - 3.1 Was ist ein Geschäftssystem? 7
  - 3.2 Szenarios 8
  - 3.3 Die zwei Sichten auf das Geschäftssystem 8
  - 3.4 Die externe Sicht 9
    - 3.4.1 Anwendungsfalldiagramme (use case diagrams) 10
    - 3.4.2 Aktivitätsdiagramme (activity diagrams) 15
    - 3.4.3 Sequenzdiagramme (sequence diagrams) 20
  - 3.5 Die interne Sicht 24
    - 3.5.1 Organisationseinheiten als Pakete (packages) 25
    - 3.5.2 Klassendiagramme (class diagrams) 26
    - 3.5.3 Aktivitätsdiagramme 31
  - 3.6 Zusammenfassung: Modellierung von Geschäftsprozessen mit UML 31
- 4 Das Modell des IT-Systems 34**
  - 4.1 Die Sicht von außen 34
    - 4.1.1 Die Benutzeroberfläche und die Gebrauchsanweisung 34
    - 4.1.2 Das Anwendungsfalldiagramm des IT-Systems 36
    - 4.1.3 Das Sequenzdiagramm des IT-Systems 37
  - 4.2 Die strukturelle Sicht 38
    - 4.2.1 Objekte und Klassen 38
    - 4.2.2 Generalisierung, Spezialisierung und Vererbung 40
    - 4.2.3 Zusammenfassung Klassen und Objekte 41
    - 4.2.4 Klassendiagramme und statische Geschäftsregeln 42
    - 4.2.5 Zustandsdiagramme und die Verhaltenssicht 48
  - 4.3 Die Ablaufsicht - Kollaborations- und Sequenzdiagramme 53
    - 4.3.1 Kollaborationsdiagramm (collaboration diagram) 54
    - 4.3.2 Erstellen eines Kollaborationsdiagramms 55
  - 4.4 Zusammenfassung der Modellierung eines IT-Systems 55



# 1 Einführung

Wir haben bisher die sogenannten klassischen Analysemethoden des Software-Engineerings kennengelernt, die Strukturierte Analyse und die Moderne Strukturierte Analyse. Die „Software-Krise“ der 60er Jahre konnte damit zwar nicht komplett gelöst werden, jedoch wurde erst mit diesen Methoden die Entwicklung komplexer Systeme im großen Maßstab ermöglicht.

Das große technisch-historische Verdienst der klassischen Methoden ist es, dass Software-Entwicklung nicht mehr als künstlerische Aktivität genialer Programmierer durchgeführt wurde, sondern als ingenieurmäßige Produktion angesehen wurde.

Seit den 70er Jahren wurde immer mehr Software in den Unternehmen (und in den öffentlichen Verwaltungen) eingesetzt. Dabei rückte sie zunehmend in geschäfts-tragende Rollen: Mittlerweile führt jedes größere Unternehmen seine Finanzbuchhaltung mit Computersystemen durch, und selbst Produktionsprozesse und Kundenbeziehungen werden wesentlich von Software unterstützt. Das belegt nicht zuletzt der fortschreitende Einsatz von Softwarepaketen wie SAP, Baan oder J.D. Edwards genauso wie die aktuelle Diskussion um Knowledge Warehouses, Data Mining oder das World Wide Web.

Kurz gesagt: Der Ausfall der IT würde mittlerweile jedes größere Unternehmen innerhalb weniger Tage in den Ruin treiben. Keine Bank der Welt würde ohne ihre die mel-derechtlichen Erfordernisse erfüllende Software ihre Lizenz behalten können - und selbst wenn: Ohne die dv-technische Abwicklung des Zahlungsverkehrs - einem Kerngeschäft - könnte sie ihre existenziellen Geschäftsprozesse gar nicht ausführen.

Die Systeme sind immer komplexer geworden. Auch kann längst nicht jedes System auf der „grünen Wiese“ entstehen. Das klappt höchstens bei ganz neuen Geschäftsaktivitäten wie in den letzten Jahren bei der Entwicklung des e-Commerce und der zunehmenden Nutzung des Internets. Eigenschaften wie **Wartbarkeit** und **Wiederverwendbarkeit** rücken in den Vordergrund.

Dieser Hintergrund verhalf dem Anfang der 80er Jahre entwickelten *Objektorientierten Ansatz* zum Durchbruch. Denn eines der systematischen Hauptanliegen der Objektorientierung ist genau Wartbarkeit und Wiederverwendbarkeit der Objekte. Während der 90er Jahre entstanden Dutzende von Methoden zur objektorientierten Analyse und objektorientiertem Design - und ebenso viele Darstellungsformen. Eine *einheitliche Modellierungssprache* schien unerlässlich.

Zu Beginn der 90er Jahre waren objektorientierte Methoden von Grady Booch und James Rumbaugh weitverbreitet. Im Oktober 1994 begannen bei der Rational Software Corporation erste offizielle Aktivitäten zur Schaffung einer einheitlichen Modellierungssprache. Man einigte sich zunächst auf die Standardisierung einer *Notation* - noch nicht einer *Methode*.

Es wurde die *Booch-Methode* von Grady Booch, die *Object-Modelling Technique (OMT)* von James Rumbaugh und das *Object-Oriented Software Engineering (OOSE)* oder *Objectory* von Ivar Jacobson zusammengeführt, mit Elementen weiterer Methoden angereichert und im **Juni 1996** unter dem Namen **UML (Unified Modeling Language)** in der Version 0.9 veröffentlicht.

Dabei wurde keine vollständig neue Notation entworfen, sondern bestehende und akzeptierte Diagrammtypen der verschiedenen objektorientierten Methoden ergänzt oder vereinfacht. Auch Darstellungsmittel, die bei strukturierten Methoden verwendet werden, findet man in UML wieder: Die „Aktivitätsdiagramme“ der UML enthalten beispielsweise Ansätze der Datenflussdiagramme und der Petri-Netze.

Bei der Weiterentwicklung der UML wurden namhafte Firmen wie IBM, Oracle, Microsoft, Digital, Hewlett-Packard und Unisys einbezogen. 1997 wurde die UML in der Version 1.1 bei der Object Management Group (OMG) zur Standardisierung eingereicht und im September 1997 akzeptiert. Die bisher letzte Version 1.3 wurde im Juni 1999 veröffentlicht. Siehe <http://www.omg.org>.

## 2 Die UML-Notation mit Hilfe eines Fallbeispiels

Notation ist ein Schlüsselwerkzeug für Software-Entwickler. Mit Diagrammen und Notationen werden abstrakte Konzepte *visualisiert*, d.h. sichtbar gemacht. Software-Entwickler, genau wie Musiker, Schriftsteller oder Ingenieure, müssen die jeweils benutzte Notation beherrschen - hier die UML, dort Noten, Schriftzeichen oder Konstruktionspläne.

Die Unified Modeling Language UML ermöglicht es, Systeme in Worten und Bildern zu beschreiben. Es können ganz verschiedene Dinge modelliert werden: Software-Systeme, Geschäftssysteme, Kundengruppen, usw.

Ähnlich wie die Erstellung von Bauplänen durch einen Architekten oder von Notenblättern durch einen Komponisten hat die Modellierung von Software zwei verschiedenartige schwierige Aspekte: Einerseits geht es darum, die Notation zu beherrschen und korrekt anzuwenden. Andererseits aber, und das ist vielleicht noch viel schwieriger, soll *modelliert* werden, d.h. man muss kreativ und innovativ sein, das betrachtete System muss in einen Kontext eingebettet werden und effizient Lösungen für Probleme der realen Welt liefern. Es geht um das Entwerfen und das Design von Software. Wir werden in der vorliegenden Arbeit nur bedingt diesen zweiten Aspekt vermitteln können. Schwerpunkt wird es sein, die Notation der UML darzustellen.

Die Diagramme der UML - die verbreitetsten sind sicherlich die Anwendungsfalldiagramme, die Klassendiagramme und die Sequenzdiagramme - sind nicht wirklich neu. Neu an der UML ist die *weltweite Einigung* auf eine Modellierungssprache, die von der Object Management Group (OMG) standardisiert wurde.

Wir werden in drei Schritten vorgehen: Zunächst werden wir unser Fallbeispiel, das Geschäftssystem des UML Airports, mit UML modellieren, dann das IT-System, und schließlich die Systemintegration. Aber halt: Der aufmerksame Leser wird erkannt haben, dass die Modellierung des Geschäftssystems in die Phase der **Analyse** gehört, während die beiden anderen Tätigkeiten bereits zur Phase des **Entwurfs** gehören. Man erahnt schon, welchen enormen Vorteil UML verspricht: Ein in der Analyse erstelltes Diagramm kann von den Software-Designern gelesen werden, und ein UML-kundiger Auftraggeber wird sogar die Entwurfsdiagramme deuten können. Die Erfüllung des alten Traums der Objektorientierung, die Kommunikation zwischen Auftraggebern und Software-Entwicklern zu verbessern, rückt dank UML in greifbare Nähe...

### 2.1 Die Wissensträger

Ohne die Beherrschung der jeweiligen fachlichen Grundlagen ist es nicht möglich, ein brauchbares Modell zu erstellen und zu verifizieren. Wesentlich ist, dass in den Analyse-Prozess **Wissensträger** mit einbezogen werden, die mit den Systemen und den Abläufen vertraut sind, die modelliert werden sollen. Wie findet man solche Wissensträger? Folgende Personengruppen kommen besonders in Frage:

- an Geschäftsabläufen beteiligte Personen, die ausführen, steuern und überwachen;
- Anwender von gleichen oder ähnlichen IT-Systemen;

- Kunden (sie sind oft kritische und ideenreiche Wissensträger!);
- Geschäftspartner;
- Fachexperten;
- das Management.

## 2.2 Das Fallbeispiel

Wir werden im Folgenden die UML-Notation anhand des Fallbeispiels ***Einführung eines IT-Systems in die Passagierabfertigung am UML Airport*** kennen lernen. Es ist eng angelehnt an das Fallbeispiel aus Grässle *et al.* (2000). Die Passagierabfertigung soll aus dem ***Check-In*** und dem ***Boarding*** bestehen.

## 3 Modellierung von Geschäftsprozessen

Um ein Geschäftssystem zu modellieren, müssen wir zunächst klären, was ein Geschäftssystem ist.

### 3.1 Was ist ein Geschäftssystem?

Unter einem **Geschäftsprozess (business process, workflow)** versteht man einen Vorgang oder Ablauf innerhalb einer betrieblichen Organisationsstruktur, um ein geschäftlich relevantes Ziel zu erreichen.

Betrachtet man den UML Airport, findet man eine ganze Reihe von Geschäftsprozessen:

- Ziel unseres Passagiers ist es, in den Urlaub zu fliegen. Dazu muss er eine Reise buchen, zum UML Airport fahren, einchecken, ins Flugzeug steigen, am Zielort wieder aussteigen und zum Hotel fahren.
- Ziel der Kioskbesitzerin am UML Airport ist es, ihre angebotenen Waren zu verkaufen. Dafür kauft sie günstig Waren ein und bietet sie zu einem höheren Preis an.
- Die Passagierabfertigung möchte Reisende einchecken. Dazu nimmt eine Mitarbeiterin das Ticket und das Gepäck entgegen, erfragt die Sitzplatzwünsche und bedient ein IT-System. Am Ende des Vorgangs erhält der Passagier seine Boardingkarte, auf der sein reservierter Sitzplatz im Flugzug und sein Gate vermerkt sind.

Ein Geschäftsprozess wird also meist in mehreren Arbeitsschritten erledigt. Diese Arbeitsschritte nennt man **Aktivitäten**. Sie werden in einer vorgegebenen Ablauffolge durchgeführt und können nacheinander oder parallel ablaufen. So kann der Passagier sich am Duty-Free Shop ein Parfüm kaufen, während sein Gepäck gerade in das Flugzeug nach London verladen wird.

Aktivitäten sind dynamischer Natur. Daneben gibt es bei Geschäftsprozessen noch statische Aspekte zu berücksichtigen. Dazu gehören die **Organisationsstrukturen** des Unternehmens, die sich wiederum aus den **Mitarbeitern** sind, sowie die **Geschäftsobjekte** wie Tickets oder Aufträge.

Aktivitäten, Organisationsstrukturen und Geschäftsobjekte zusammen bilden ein **Geschäftssystem (business system)**, also gewissermaßen

Geschäftssystem = Aktivitäten + Orgastruktur + Geschäftsobjekte.

Aus betriebswirtschaftlicher Sicht ist ein Geschäftssystem die Wertschöpfungskette, die die Leistungserstellung beschreibt. Ein **Unternehmen** umfasst ein oder mehrere Geschäftssysteme, jedes Geschäftssystem für sich erbringt eine wirtschaftliche Leistung. Zusammengefasst ergibt sich also die Gliederung eines Unternehmens gemäß *Abbildung 1*.

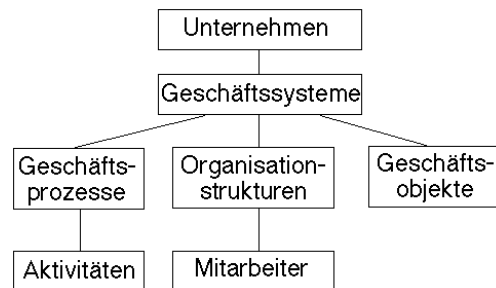


Abbildung 1 Gliederung eines Unternehmens

### 3.2 Szenarios

Ein geeigneter Ansatz, die Analyse zu beginnen, ist es Szenarien durchzuspielen: **Szenarien** sind Abfolgen von Schritten und Abläufen, die mit den Geschäftsaktivitäten zu tun haben. Ein solches *typisches* Szenario ist in *Abbildung 2* dargestellt. Es ist gewissermaßen eines der „Normalfälle“. Allerdings ist es nur eines von vielen möglichen Szenarien. Für die Passagierabfertigung sind beispielsweise folgende Abweichungen denkbar:

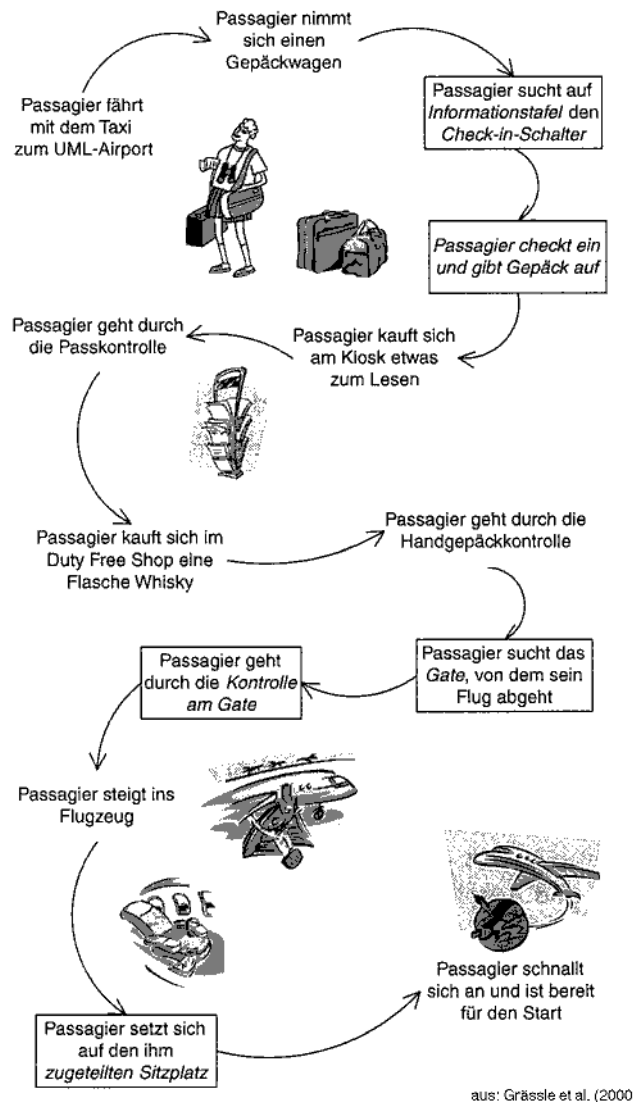
- Der Passagier hat nur Handgepäck (Geschäftsreisender).
- Der Passagier kauft nichts am Kiosk
- Der Passagier kommt sehr spät und muss so schnell wie möglich einchecken
- Der Passagier ist mit dem Flugzeug angekommen und muss lediglich umsteigen, d.h. er wird den Transitbereich nicht verlassen.
- Der Passagier checkt ein, verschläft aber in einem der nicht wirklich bequemen Sessel im Wartebereich seinen Flug trotz mehrfacher Aufrufe.
- Der Passagier verliert seine Boardingkarte
- Der Passagier kommt nicht durch die Passkontrolle, weil sein Pass abgelaufen ist.

### 3.3 Die zwei Sichten auf das Geschäftssystem

In unserem Fallbeispiel soll ein IT-System in den Unternehmensbereich Passagierabfertigung integriert werden. Daher genügt es, diesen Unternehmensbereich zu betrachten und nur das Geschäftssystem Passagierabfertigung zu modellieren.

Ein Geschäftssystem sollte aus zwei Sichten betrachtet werden: Der internen Sicht und der externen Sicht, s. *Abbildung 3*. Betrachtet man das Geschäftssystem von außen, nimmt man die Rolle eines Kunden, Geschäftspartners oder Lieferanten ein.

Innerhalb des Geschäftssystems befinden sich die Mitarbeiter, die die Anforderung der Umgebung erfüllen und die dafür erforderlichen Geschäftsprozesse abwickeln. Hinter diesen Geschäftsprozessen stehen *Arbeitsabläufe* und *IT-Systeme*. Diese interne Sicht bleibt Außenstehenden im Normalfall verborgen, das Geschäftssystem selbst bleibt nach außen eine *Black-box*.



aus: Grässle et al. (2000)

Abbildung 2 Typisches Szenario zum Fallbeispiel **Passagierabfertigung**. Diejenigen Stationen des Szenarios, die zur Passagierabfertigung gehören, sind eingerahmt und kursiv gedruckt.

Für die Modellierung der Geschäftsprozesse ist es sehr hilfreich, für das betrachtete Geschäftssystem stets eine schematische Darstellung wie in *Abbildung 3* zu erstellen. Sie enthält bereits wichtige Elemente, die später für die UML-Diagramme benötigt werden. Davon abgesehen erhält man damit bereits im Vorfeld der Analyse eine erste grobe Strukturierung des in der Regel komplexen Systems und eine Abgrenzung zu anderen Geschäftssystemen. Denn es klärt direkt die ersten wichtigen Fragen: Was ist eigentlich das betrachtete Geschäftssystem? Gehört die Abteilung Gepäckabfertigung nicht auch dazu? Wer benötigt Dienstleistungen von ihm und wer liefert welche?

### 3.4 Die externe Sicht

Relevant für die externe Sicht ist die Erstellung von Sachgütern und das Angebot von Dienstleistungen. In der externen Sicht sind nur diejenigen Aktivitäten von Interesse,

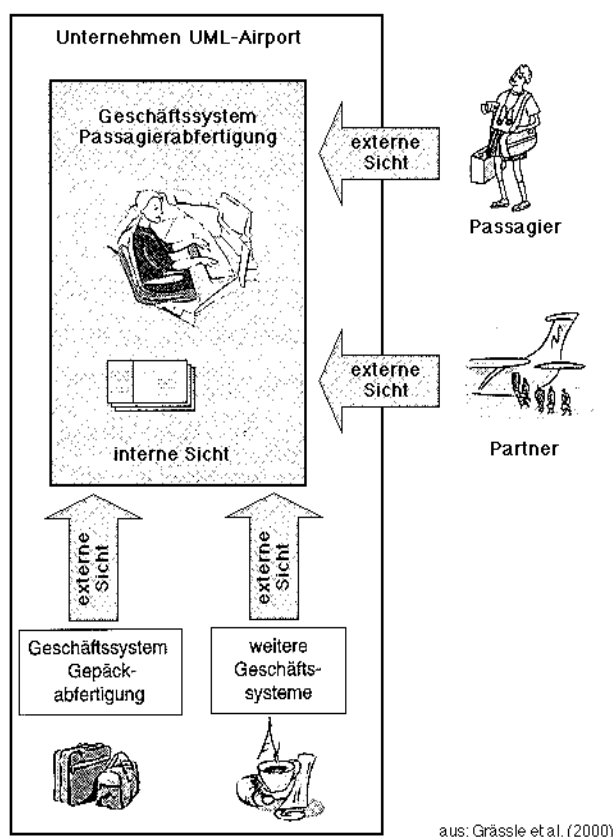


Abbildung 3 Externe und interne Sicht auf das Geschäftssystem

an denen Außenstehende beteiligt sind. In der UML sind dazu insbesondere drei Diagramme besonders geeignet: Anwendungsfalldiagramme, Aktivitätsdiagramme und Sequenzdiagramme.

### 3.4.1 Anwendungsfalldiagramme (*use case diagrams*)

Anwendungsfalldiagramme zeigen Akteure, Geschäfts-Anwendungsfälle und deren Beziehungen. In Anwendungsfalldiagrammen sind keine Abläufe beschrieben. Sie geben einen guten Überblick über die Funktionalität und den Kontext des Systems.

Im Anwendungsfalldiagramm arbeiten wir mit den folgenden Elementen:

**1.) Akteur (*actor*).** Ein Akteur repräsentiert eine Rolle, die ein Außenstehender



eines Geschäftssystems annimmt, wenn er mit dem Geschäftssystem interagiert. Ein Akteur kann z.B. ein Kunde, ein Geschäftspartner, ein Lieferant oder ein anderes Geschäftssystem sein.

**2.) Geschäfts-Anwendungsfall (*business use case*) oder Geschäftsprozess (*business process*).** Ein Geschäftsprozess beschreibt die Interaktion,



die zwischen einem Akteur und dem Geschäftssystem stattfindet. Er beschreibt also die *Dienstleistung* des Geschäftssystems, die der Akteur in Anspruch nimmt. Ein Geschäftsprozess wird aus der Sicht der Akteure beschrieben.

**3.) Assoziation (*association*).** Eine Assoziation ist die Beziehung zwischen ei-

\_\_\_\_\_ nem Akteur und einem Geschäftsprozess. Sie besagt, dass ein Akteur eine bestimmte Dienstleistung des Geschäftssystems, den Geschäftsprozess, nutzen kann. Die Assoziation gibt keine Auskunft darüber, *wie* die Dienstleistung genutzt wird. Die Assoziation bedeutet nur, *dass* der Akteur an der Ausführung des Anwendungsfalls beteiligt ist.

**4.) include-Beziehung (*include relation*).** Die include-Beziehung ist eine Be-



ziehung zwischen zwei Anwendungsfällen, die besagt, dass der Anwendungsfall auf der Seite mit dem Pfeil im anderen enthalten ist.

### 3.4.1.1 Erstellen eines Anwendungsfalldiagramms

Das Erstellen eines Anwendungsdiagramms erfolgt in vier Schritten: (i) Akteure finden - (ii) Geschäfts-Anwendungsfälle finden - (iii) Akteure und Anwendungsfälle verbinden - (iv) Erweiterungen suchen.

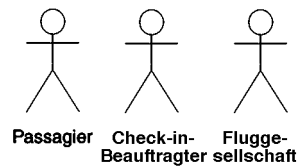
**(i) Akteure finden.** Zunächst gilt die Faustregel: Je mehr Akteure man findet, desto besser. Mit den Akteuren kann man in späteren Schritten weiterarbeiten. Sie können dann ggf. reduziert oder zusammengefasst werden.

Hilfreich bei der Diagrammerstellung sind die zuvor dokumentierten Szenarios. Sie liefern bereits die ersten Akteure

Durch die Beantwortung der folgenden Hilfsfragen, möglichst unter Einbeziehung der Wissensträger, findet man mögliche Akteure:

- o Wer sind die Kunden des Geschäftssystems bzw. der Geschäftsprozesse?
- o Wer sind die externen Geschäftspartner des Geschäftssystems? Welche Leistungen nutzen sie?
- o Welche unternehmensinternen Stellen und Organisationseinheiten nutzen die Leistungen des Geschäftssystems?
- o Mit welchen anderen Geschäftssystemen gibt es Verbindungen?

In unserem Fallbeispiel ergeben sich z.B. direkt folgende Akteure:

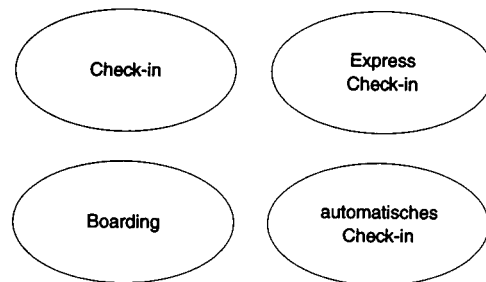


Neben dem *Passagier*, der den Reisenden bzw. den Kunden repräsentiert, gibt es die *Fluggesellschaft* und den *Check-in-Beauftragten*. Letzterer ist eine Person, die den Auftrag des Passagiers hat, mit seinem Ticket den Check-in durchzuführen.

**(ii) Geschäftsprozesse finden.** Auch hier gilt: Je mehr, desto besser. Folgende Hilfsfragen liefern mögliche Geschäftsprozesse:

- o Welche Leistungen (Sachgüter & Dienstleistungen) werden Kunden angeboten oder von ihnen genutzt?
- o Welche Leistungen werden unternehmensexternen Partnern zur Verfügung gestellt bzw. von diesen genutzt?
- o An welchen Leistungen sind Lieferanten beteiligt?
- o Was machen die einzelnen Akteure?
- o Wie und bei welchen Gelegenheiten wird mit anderen Geschäftssystemen kommuniziert?
- o Welche Ereignisse lösen welche Tätigkeiten aus?

In unserem Fallbeispiel ergeben sich nach ersten Überlegungen folgende Geschäfts-Anwendungsfälle:



Zunächst werden die Anwendungsfälle nur kurz und formlos beschrieben:

- Der *Check-in*-Vorgang umfasst das Vorlegen des Tickets, die *Gepäckaufgabe*, die Platzreservierung, die Übermittlung der Passagierliste an die Fluggesellschaft sowie die Erstellung und Aushändigung der Boarding-Karte.
- Der *Express-Check-in* kann von Passagieren genutzt werden, die nur Handgepäck haben. Es erfolgt keine *Gepäckaufgabe*, ansonsten entspricht es dem *Check-in*.
- Beim *Boarding* wird die Boardingkarte des Passagiers am Gate kontrolliert.
- Der *automatische Check-in* erfolgt ohne Mithilfe eines Check-in-Mitarbeiters direkt an einem Automaten (Bildschirm). Es kann kein Gepäck aufgegeben werden.

Auch hat sich die *Beobachtungstechnik* zum Auffinden von Geschäftsprozessen sehr bewährt: Durch das Beobachten der an den Geschäftsprozessen beteiligten Personen können Listen mit Tätigkeiten erstellt werden. Die Tätigkeiten können dann nach Ereignissen gruppiert werden und führen so zu Anwendungsfällen.

**(iii) Akteure und Anwendungsfälle verbinden.** Indem Geschäfts-Anwendungsfälle den Akteuren zugewiesen werden, entsteht ein erster Entwurf des Anwendungsfalldiagramms. Dies geschieht durch die Beantwortung der folgenden Frage:

- o Welchen Kunden oder Geschäftspartnern stehen welche Leistungen zur Verfügung?

Damit erhalten wir das Anwendungsfalldiagramm in *Abbildung 4*.

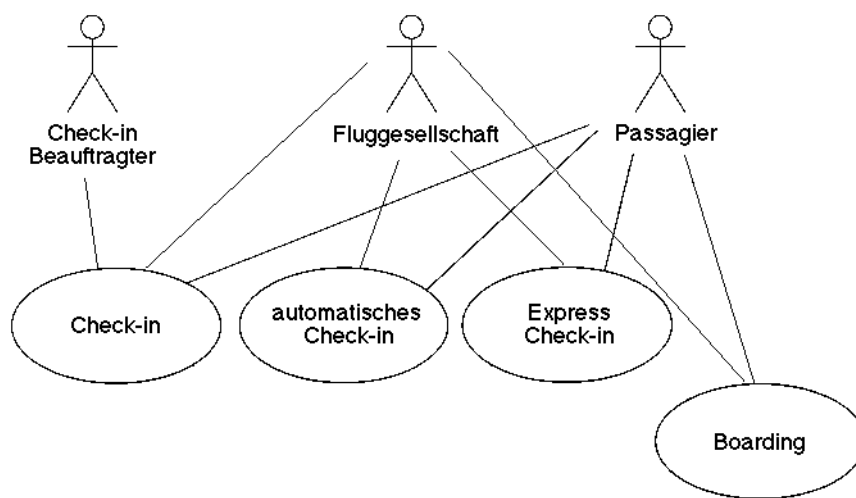


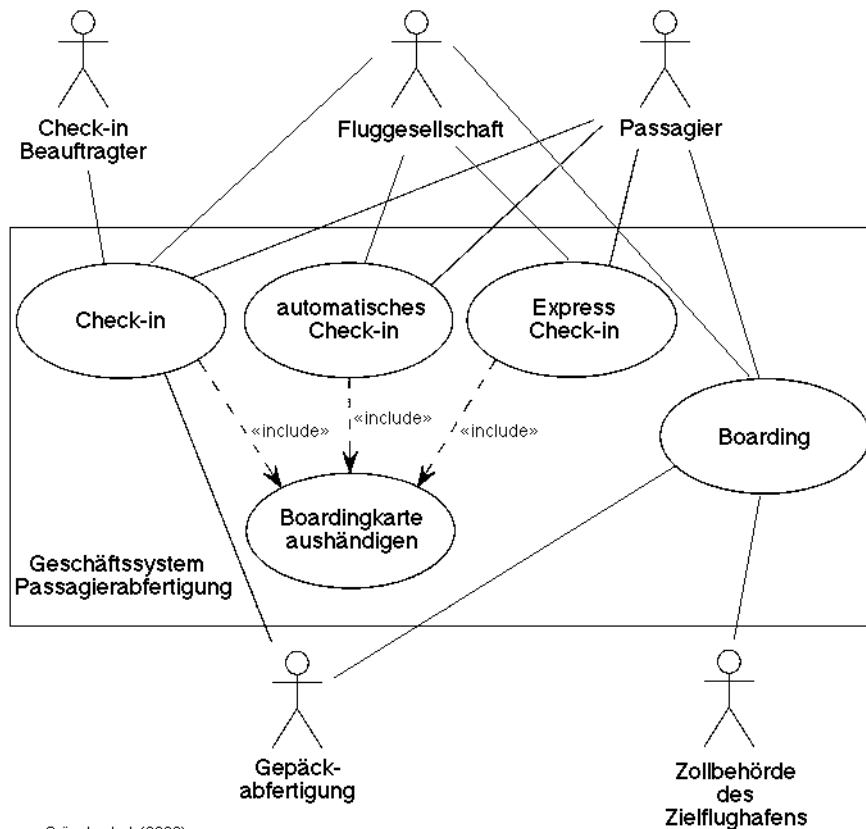
Abbildung 4 Erstes Anwendungsfalldiagramm

**(iv) Erweiterungen suchen.** Hat man einige Geschäfts-Anwendungsfälle gefunden, so lassen sich diese als Ausgangspunkt für weitere Fragen verwenden. Ausgehend von einem bestimmten Geschäftsprozess können folgende Fragen gestellt werden:

- o Gibt es etwas, was man tun muss, *bevor* eine bestimmte Leistung in Anspruch genommen werden kann?
- o Gibt es etwas, was man tun muss, *nachdem* ein bestimmter Geschäftsprozess ausgeführt ist?
- o Gibt es etwas, was man tun muss, falls *niemand* einen bestimmten Geschäftsprozess ausführt?

Dabei ist es sehr wichtig, das richtige Geschäftssystem vor Augen zu behalten. Vieles, was vor oder nach dem einem Anwendungsfall passieren muss, passiert auch *außerhalb* des betrachteten Geschäftssystems. In unserem Fallbeispiel gehört z.B. die Buchung der Reise oder die Fahrt zum Flughafen *nicht* zum betrachteten Geschäftssystem.

Das Geschäfts-Anwendungsfalldiagramm zu unserem Fallbeispiel könnte nach einigem Nachdenken also aussehen wie in *Abbildung 5* dargestellt.



aus : Grässle et al. (2000)

Abbildung 5 Anwendungsfalldiagramm des Geschäftssystems *Passagierabfertigung*

### 3.4.1.2 Verifikation

Sämtliche Diagramme und Aufzeichnungen müssen mit den Wissensträgern verifiziert werden. Die Frage, die man den Wissensträgern für *jedes* Diagramm stellen sollte, ist:

- o Ist alles richtig und vollständig, was in den Diagrammen steht?

Auch wenn die Wissensträger die Diagramme selbst lesen und verstehen können, sollte man die diagramme trotzdem vortragen. Erst so schließt sich der Kreis und es entsteht eine verifizierte Sicht, welche das gemeinsame aktuelle Verständnis der Geschäftssysteme und Geschäftsprozesse widerspiegelt.

Das fertige Anwendungsdiagramm sollte anhand der folgenden Checkliste geprüft werden:

#### Checkliste zur Verifikation der externen Sicht

- o *Vollständigkeit.* Das Anwendungsfalldiagramm ist vollständig, es gibt keine weiteren Geschäfts-Anwendungsfälle im System. Alle Leistungen, die Kunden oder Geschäftspartnern zur Verfügung stehen, sind durch Geschäfts-Anwendungsfälle dargestellt.
- o *Detailierungsgrad.* Jeder im Diagramm enthaltene Geschäftsprozess ist so detailliert, dass er den folgenden Anforderungen genügt:

- Er bildet eine fachlich zusammenhängende Folge von Interaktionen.
  - Er wird durch einen Akteur initiiert und hat nur wenige Akteure. (Geschäfts-Anwendungsfälle, die nicht durch einen Akteur angestoßen werden, sind interne Aktivitäten und werden nicht in der externen Sicht dargestellt; Anwendungsfälle mit zu vielen Akteuren müssen aufgeteilt werden.)
  - Er stellt eine Leistung dar, die ein messbares und geschäftsrelevantes Ergebnis liefert: z.B. unfertige Teilschritte wie *Gepäckstücke zählen* dürfen nicht vorkommen, sie müssen ggf. mit anderen Teilschritten zusammengelegt werden.
  - Er wird auch unabhängig von anderen Anwendungsfällen ausgeführt oder folgt zeitlich nicht direkt nach einem anderen Anwendungsfall. (Wird ein Anwendungsfall nie allein, sondern immer in einer festen Abfolge mit anderen Anwendungsfällen ausgeführt, muss er mit diesen zusammengelegt werden. Das Boarding ist zwar abhängig vom Check-in, da es nicht ohne es ablaufen kann, aber es ist keine feste Abfolge und es liegt eine gewisse Zeit zwischen den Geschäftsprozessen; außerdem könnte es nach dem Check-in eventuell gar nicht zum Boarding kommen, wenn z.B. der *Passagier* im Wartesaal verschläft...
- o *Beziehungen zwischen Anwendungsfällen.* Die include-Beziehungen sind richtig angewendet.
  - o *Namensgebung und Beschreibung.* Die Namen der Anwendungsfälle beschreiben die Leistungen, die das Geschäftssystem zur Verfügung stellt. Die Namensgebung entspricht dem im Geschäftssystem üblichen Wortschatz („Die Sprache der Anwender verwenden!“)
  - o *Akteure.* Die im Anwendungsfalldiagramm enthaltenen Akteure repräsentieren Rollen, die außenstehende Personen, Unternehmen oder andere Geschäfts-systeme dem Geschäftssystem gegenüber bei einer Interaktion einnehmen.

### 3.4.1.3 Wie liest man ein Anwendungsfalldiagramm?

Man liest ein Anwendungsfalldiagramm, indem man mit einem Akteur beginnt: z.B. findet man beim Akteur *Passagier* Assoziationen zu den Anwendungsfällen *Check-in*, *automatisches Check-in*, *Express-Check-in* und *Boarding*. Das bedeutet, dass Passagiere auf vier verschiedene Weise einchecken können. Dass der Akteur *Check-in-Beauftragter* nur eine Assoziation zum Anwendungsfall *Check-in* hat, bedeutet, dass ein Check-in-Beauftragter am UML Airport keinen Express-Check-in durchführen kann. Die *Fluggesellschaft* jedoch hat wie der *Passagier* Assoziationen zu allen Anwendungsfällen.

Da ein Anwendungsfall lediglich die Akteure und die Anwendungsfälle samt aller Beziehungen untereinander darstellt, dokumentiert es *nicht die Reihenfolge*, in der die Anwendungsfälle sinnvollerweise durchgeführt werden.

### 3.4.2 Aktivitätsdiagramme (*activity diagrams*)

Aktivitätsdiagramme sind eng verwandt mit den aus der strukturierten Analyse bekannten Datenflussdiagrammen oder Programmablaufplänen (PAP). Sie betonen die

*funktionale Sicht* und widersprechen damit der reinen Lehre der Objektorientierung. Dennoch sind sie für die Geschäftsprozessmodellierung offiziell erlaubt.<sup>1</sup>

Wir verwenden die Aktivitätsdiagramme in der externen Sicht für die Beschreibung derjenigen Geschäftsprozesse, die die Leistungen des Geschäftssystems beschreiben. Durch die Möglichkeit, parallele Prozesse explizit zu beschreiben, eignet sich das Aktivitätsdiagramm sehr gut für die Abbildung von Geschäftsprozessen, die in den seltensten Fällen rein linear ablaufen und meist Parallelitäten vorweisen.

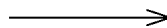
Im Aktivitätsdiagramm arbeiten wir mit den folgenden acht Elementen:

**1.) Aktivität.** Eine Aktivität ist ein einzelner Schritt eines Geschäftsprozesses



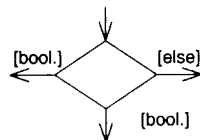
oder eines Anwendungsfalles, der eine wirtschaftliche Leistung des Geschäftssystems beschreibt.

**2.) Übergang (Kontrollfluss).** Dieser Pfeil stellt einen Übergang dar. Der ein-



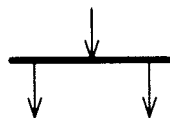
gehende Pfeil löst eine Aktivität aus. Durch den Abschluss einer Aktivität wird ein ausgehender Übergang ausgelöst, d.h. es wird entlang des ausgehenden Pfeils weiterverfahren. Salopp formuliert beschreibt der Pfeil den Kontrollfluss der Aktivitäten.

**3.) Verzweigung.** Die Raute stellt eine Verzweigung dar. Eine Verzweigung hat



einen Eingang und zwei oder mehr Ausgänge. Jeder Ausgang wird mit einer Bedingung (einem *Booleschen Ausdruck*) versehen, der in eckigen Klammern steht. Trifft eine Bedingung zu, wird am entsprechenden Ausgang weiterverfahren.

**4.) Aufspaltung.** Für das Aufspalten zweier oder mehrerer paralleler Übergänge

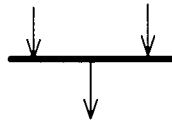


verwendet man einen Synchronisationsbalken, der als dicke horizontale (oder auch vertikale) Linie gezeichnet ist. Die Aufspaltung erlaubt die Parallelisierung von Aktivitäten.

---

1.siehe Grässle et al. (2000), S. 69: „Mit dem Segen der ‚drei Amigos‘“

**5.) Zusammenführung.** Für das Zusammenführen zweier oder mehrerer par-



alleler Übergänge verwendet man ebenfalls einen Synchronisationsbalken, der als dicke Linie gezeichnet wird. Auf diese Weise werden parallele Aktivitäten zusammengeführt. An der Zusammenführung findet eine Synchronisation statt, d.h. jeder wartet, bis alle eingehenden Übergänge die Zusammenführung erreicht haben, dann läuft der Kontrollfluss nach diesem Punkt weiter.

**6.) Startzustand.** Der Startzustand ist der Startpunkt des Aktivitätsdiagramms.



**7.) Endzustand.** Der Endzustand gibt an, dass die Ausführung der Aktivitäten



abgeschlossen ist. Es kann in einem Aktivitätsdiagramm mehrere Endpunkte geben.

**8.) Verantwortungsbereich.** Jeder Verantwortungsbereich repräsentiert



eine Verantwortlichkeit für einen Teil der Gesamtaktivität des Aktivitätsdiagramms. Jede Aktivität wird einem Verantwortungsbereich zugeordnet. In Geschäftsprozessmodellen entsprechen diese Bereiche meist Organisationseinheiten, oder es sind Akteure oder Geschäftssysteme. Jeder Verantwortungsbereich ist durch eine senkrechte durchgezogene Linie von seinem benachbarten Verantwortungsbereich abgeteilt; man spricht oft auch von „Schwimmbahnen“ (swim lanes).

### 3.4.2.1 Erstellen eines Aktivitätsdiagramms

In fünf Schritten werden wir ein Aktivitätsdiagramm erstellen: (i) Verantwortungsbereiche und Aktivitäten finden - (ii) Aktivitäten verbinden - (iii) Aktivitäten verfeinern - (iv) Sicht verifizieren.

**(i) Verantwortungsbereiche und Aktivitäten finden.** Was muss gemacht werden, wenn die Akteure die angebotenen Leistungen beanspruchen? Wir gehen vom Anwendungsfalldiagramm aus. Jedem Akteur eines Anwendungsfalldiagramms entspricht zunächst ein Verantwortungsbereich. Im nächsten Schritt leiten wir die Aktivitäten von den Geschäftsprozessen ab. Durch die folgenden Hilfsfragen kann man Aktivitäten finden:

- o Welche Arbeitsschritte sind notwendig für die Ausführung eines Geschäftsprozesses, d.h. für die Bereitstellung und Abwicklung einer Leistung, und wer ist dafür verantwortlich?
- o Was machen die einzelnen Akteure?

- o Welche Ereignisse lösen welche Arbeitsschritte aus?

Oft finden sich bereits vorhandene Dokumentationen von Abläufen, die durchaus als Basis für das Auffinden von Aktivitäten herangezogen werden können. Sehr hilfreich sind auch die anfangs erstellten Szenarien. Sie enthalten oft bereits einzelne Aktivitäten und deren Verantwortlichkeitsbereiche.

Die Aktivitäten werden mit kurzen Bezeichnungen (Verb+Objekt) versehen. Zweckmäßigerweise werden sie gleich in einer Tabelle notiert, deren Spalten jeweils die verantwortlichen Akteure bezeichnen. In *Abbildung 6* sind für den Geschäftssystem *Passa-*

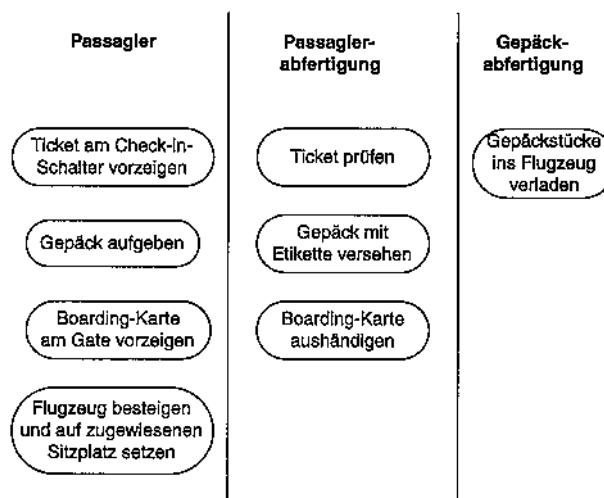


Abbildung 6 Aktivitäten und Verantwortlichkeiten

*gierabfertigung* die Aktivitäten mit ihren Verantwortlichkeitsbereichen aufgelistet.

Sehr komplexe Aktivitäten können gegebenenfalls formlos beschrieben werden.

Werden die Aktivitätsdiagramme verfeinert, können durchaus weitere Verantwortlichkeitsbereiche hinzukommen.

**(ii) Aktivitäten verbinden.** In welcher Reihenfolge laufen die Aktivitäten ab? Durch die Verbindung der einzelnen Aktivitäten zu einem Ablauf entsteht ein erstes Aktivitätsdiagramm. Dieser Ablauf wird als *Kontrollfluss* bezeichnet. Er verläuft von oben nach unten, wobei die einzelnen Aktivitäten ihre *swim lanes* nicht verlassen. Folgende Fragen helfen bei der Erstellung des Kontrollflusses:

- o In welcher Reihenfolge werden die Aktivitäten ausgeführt?
- o Welche Bedingungen müssen für die Ausführung von Aktivitäten erfüllt sein?
- o Wo muss verzweigt werden?
- o Welche Aktivitäten verlaufen parallel?
- o Muss auf die Beendigung von Aktivitäten gewartet werden, bevor mit weiteren Aktivitäten weiterverfahren werden kann?

Auf diese Weise gelangen wir in unserem Fallbeispiel auf das Aktivitätsdiagramm in *Abbildung 7*.

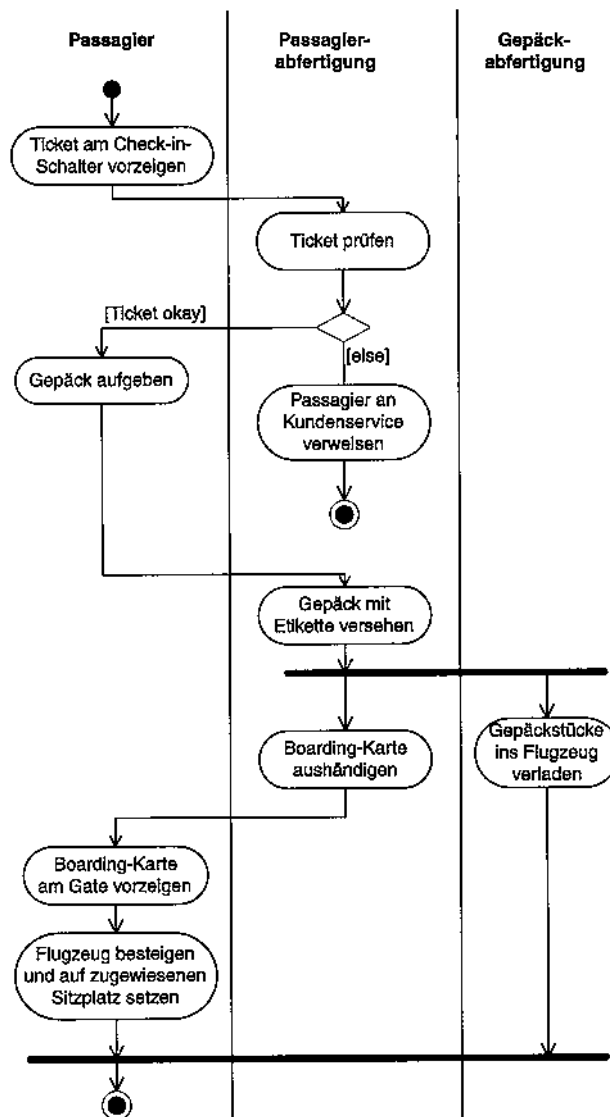


Abbildung 7 Aktivitätsdiagramm des Geschäftssystems *Passagierabfertigung*

**(iii) Aktivitäten verfeinern.** Kommen noch weitere Aktivitäten oder Aktivitätsdiagramme hinzu? Möglicherweise müssen einzelne Aktivitäten weiter aufgeteilt werden oder durch weitere Aktivitätsdiagramme verfeinert werden.

Aktivitäten müssen verfeinert werden, wenn beispielsweise

- eine Aktivität so umfangreich ist, dass sie nicht eindeutig einem Verantwortungsbereich zugeordnet werden kann;
- innerhalb einer Aktivität Verzweigungen existieren.

Es kann auch sein, dass weitere Aktivitäten hinzukommen, um den Kontrollfluss zu vervollständigen bzw. zu ergänzen.

Verschiedene Szenarien können in verschiedenen Aktivitätsdiagrammen dargestellt werden.

**(iv) Sicht verifizieren.** Wie beim Geschäfts-Anwendungsfalldiagramm müssen auch die Aktivitätsdiagramme gemeinsam mit den Wissensträgern auf ihre inhaltliche Korrektheit überprüft werden.

### 3.4.2.2 Wie liest man ein Aktivitätsdiagramm?

Zunächst betrachtet man die Verantwortlichkeitsbereiche. In unserem Fallbeispiel sind das der *Passagier*, die *Passagierabfertigung* und die *Gepäckabfertigung*. Der *Passagier* z.B. ist verantwortlich für die Aktivitäten *Ticket am Check-in-Schalter vorzeigen* und *Gepäck aufgeben*.

Man beginnt beim Lesen mit dem Startzustand und liest entlang der Pfeile. Aktivitäten zwischen Balken (Aufspaltungen und Zusammenführungen) werden parallel ausgeführt.

Es kann sinnvoll sein, ein detaillierteres Aktivitätsdiagramm zu erstellen, z.B. *Check-in* eines Passagiers in *Abbildung 8*.

### 3.4.3 Sequenzdiagramme (*sequence diagrams*)

Das erste objektorientierte UML-Diagramm, das wir kennen lernen, ist das Sequenzdiagramm. Es ist von fast genialer Einfachheit (es besteht im wesentlichen aus zwei Elementen!) und eignet sich für die Darstellung von Interaktionen. Es visualisiert den Nachrichtenaustausch zwischen ausgewählten Objekten und hebt den zeitlichen Verlauf der ausgetauschten Nachrichten hervor.

Zur Darstellung der externen Sicht erscheinen uns die Sequenzdiagramme am ehesten geeignet, aus folgendem Grund: Sequenzdiagramme sind für Ersteller und Leser wegen ihrer Einfachheit sehr verständlich. Insbesondere ist die Akzeptanz von Sequenzdiagrammen sehr hoch [vgl. Grässle *et al.* (2000), S. 84].

Im Sequenzdiagramm arbeiten wir mit den folgenden Elementen:

1. **Objekt.** Die an der Interaktion beteiligten Objekte werden am oberen Dia-



grammrand nebeneinander angeordnet. Man schreibt das Objekt unterstrichen und eingerahmt in ein Rechteck. Entweder schreibt man einfach den (unterstrichenen) Namen eines konkreten Objekts (z.B. Hugo Meier, der in die USA verreisen will), oder den (unterstrichenen) Namen der Klasse, zu der das Objekt gehört (hier: *Passagier*) mit einem Doppelpunkt davor. Im Modell des Geschäftssystems repräsentieren die Objekte die Akteure des Geschäftssystems und das Geschäftssystem selbst.

2. **Lebenslinie des Objekts.** Die Lebenslinie des Objekts verläuft vertikal nach



unten und beginnt bei dem Rechteck des Objekts.

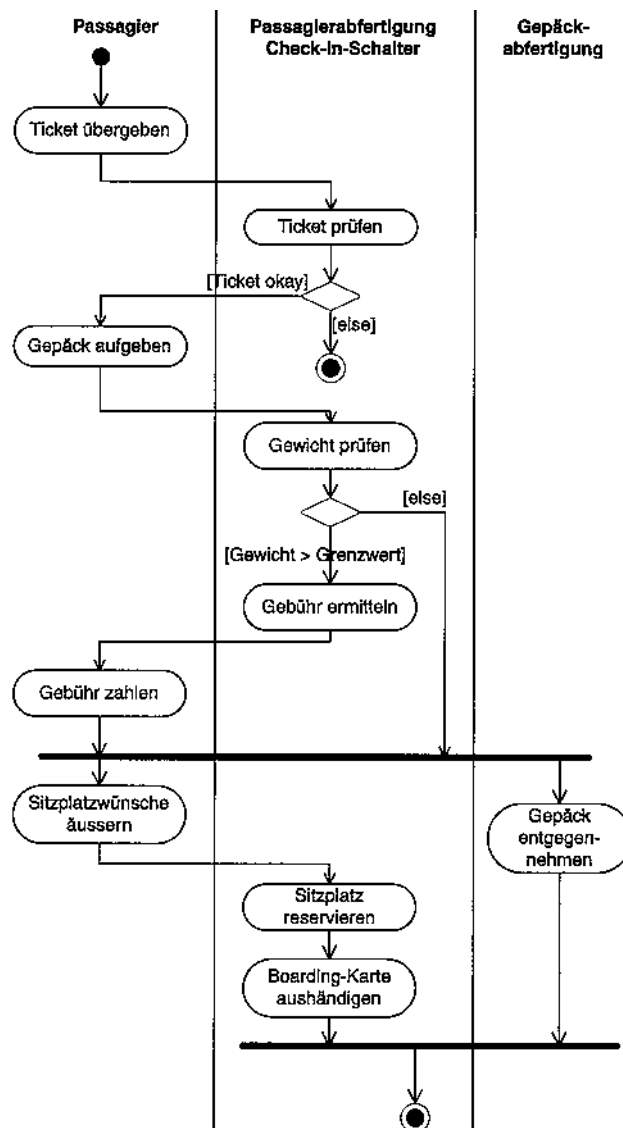
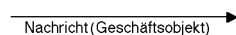


Abbildung 8 Aktivitätsdiagramm zum Anwendungsfall *Check-in*.

### 3. Nachricht und Geschäftsobjekt.



Nachrichten eingetragen, die die Objekte senden und empfangen. Sie werden in zeitlicher Reihenfolge von oben nach unten eingetragen. Der Pfeil deutet die Richtung, in der die Nachricht verschickt wird. In Klammern ist das Geschäftsobjekt verzeichnet, das zusammen mit der Nachricht übermittelt wird, z.B. *prüfen(Ticket)*.

**4. Aktivierungsleiste des Objekts.** Die Aktivierungsleiste unterbricht die Lebenslinie des Objekts und zeigt die Aktivierungsphase des Objekts an, in der es Nachrichten sendet und empfängt.



benslinie des Objekts und zeigt die Aktivierungsphase des Objekts an, in der es Nachrichten sendet und empfängt.

**5. Kommentar.** Das Sequenzdiagramm kann (soll!) durch Kommentare ergänzt

WENN Ticket nicht ok  
Passagier an Kundenservice  
verweisen  
...

werden. Es können beispielsweise Aktivitäten der Partner oder Bedingungen als Kommentare eingegeben werden. Kommentare werden am linken Rand des Sequenzdiagramms eingefügt.

### 3.4.3.1 Erstellen von Sequenzdiagrammen

**(i) Akteure und Geschäftssysteme bestimmen - Wer macht mit?** Sequenzdiagramme in der externen Sicht stellen die Interaktionen zwischen den Akteuren und den Geschäftssystemen dar. In der Regel hat man einen vollständigen Pool von Interaktionspartnern aus den Anwendungsfalldiagrammen, so dass sie abhängig vom Ablauf, der im Sequenzdiagramm dargestellt werden soll, daraus ausgewählt werden können.

In unserem Fallsbeispiel findet man für das Sequenzdiagramm *Check-in eines Passagiers* die Interaktionspartner *Passagier* und *Passagierabfertigung*.

**(ii) Initiator bestimmen - Wer beginnt die Interaktion?** Bei jeder Interaktionsfolge muss unter den Interaktionspartnern derjenige Akteur bestimmt werden, der die Interaktion anstößt - der Initiator.

In unserem Fallbeispiel beginnt der Passagier die Interaktion, indem er die Dienstleistung „Check-in“ der *Passagierabfertigung* in Anspruch nimmt.

**(iii) Nachrichtenaustausch beschreiben.** Nun wird die Abfolge der Interaktion festgelegt, d.h. für welchen Kommunikationsschritt welche Informationen ausgetauscht werden. Dabei werden die Nachrichten definiert: **Nachrichten** sind Aufforderungen an die jeweiligen Partner, etwas zu tun. Die Geschäftsobjekte, die mit ausgetauscht werden, werden ebenfalls dokumentiert.

**(iv) Den zeitlichen Ablauf der Interaktion festlegen.** Sämtliche Nachrichten werden in einer zeitlichen Reihenfolge ausgetauscht, die festgelegt und dargestellt werden muss. Die Nachrichten werden vertikal in der zeitlichen Reihenfolge von oben nach unten eingetragen.

**(v) Ergänzende Informationen eintragen.** Wichtige Aktivitäten der beteiligten Akteure und Geschäftssysteme können als Kommentare in das Diagramm eingetragen werden. Sie werden auf der Höhe der zugeordneten Nachrichten eingetragen.

**(vi) Sicht verifizieren.** Verifizieren Sie das Sequenzdiagramm der externen Sicht anhand der folgenden Checkliste:

- o Ist zu jedem Anwendungsfall ein Sequenzdiagramm erstellt?

- o Enthält jedes Sequenzdiagramm nur ein Objekt, das das Geschäftssystem repräsentiert, und höchstens so viele weitere Objekte wie dem Anwendungsfall Akteure zugeordnet sind?
- o Ist jeder Akteur, der im Anwendungsfalldiagramm aufgeführt ist, in mindestens einem der Sequenzdiagramme aufgeführt?
- o Sind alle wichtigen Kommentare eingefügt und ist das Diagramm trotzdem übersichtlich geblieben?

### 3.4.3.2 Lesen von Sequenzdiagrammen

Abbildung 9 zeigt ein Sequenzdiagramm mit den Objekten *Passagier* und *Passagierabfertigung Check-in* und *Gepäckabfertigung*. Beim Lesen des Diagramms beginnt

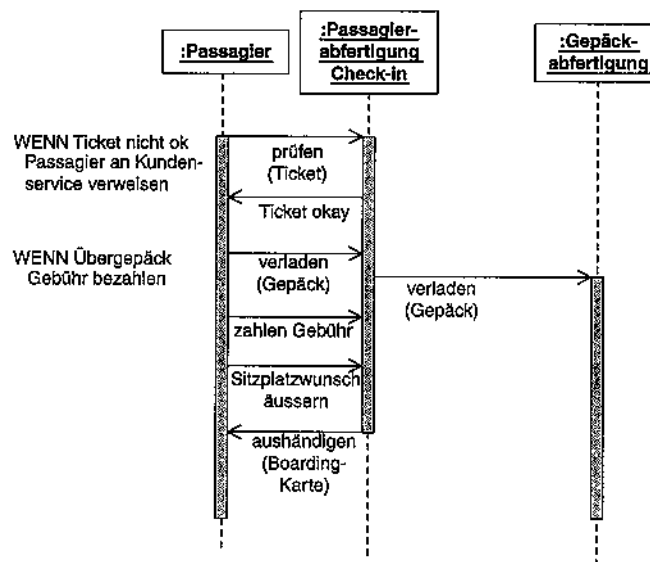


Abbildung 9 Sequenzdiagramm *Check-in eines Passagiers*

man mit der am obersten liegenden Nachricht; das ist die, die zuerst verschickt wird. Hier beginnt der Ablauf damit, dass der Passagier sein Ticket der Passagierabteilung zum Prüfen übergibt. Die Aufforderung *prüfen* ist die Nachricht, das Ticket ist ein Geschäftsobjekt. Anhand der Pfeilrichtung ist erkennbar, dass der Passagier der Sender der Nachricht ist und die Passagierabfertigung der Empfänger. Durch den Erhalt werden Aktivitäten angestoßen, was an der Aktivierungsleiste der Passagierabfertigung erkennbar ist. Im Diagramm ist nicht sichtbar, wie die Passagierabfertigung die Verarbeitung vornimmt, d.h. *welche* Aktivitäten durchgeführt werden. Einzig im Kommentar kann ein Hinweis enthalten sein. Die genaue Beschreibung befindet sich im Anwendungsfall-Aktivitätsdiagramm *Abbildung 8*.

Nachdem das Ticket als o.k. gemeldet wird, wird das Gepäck beim Check-in aufgegeben und kurz darauf die Gepäckabfertigung aufgefordert, es zu verladen, usw.

Beachten Sie, dass in *Abbildung 9* ein *Sequenzdiagramm* dargestellt ist. Sequenzdiagramme eignen sich allerdings auch gut für *anwendungsfallübergreifende* Darstellung von Geschäftsprozessen, s. *Abbildung 10*.

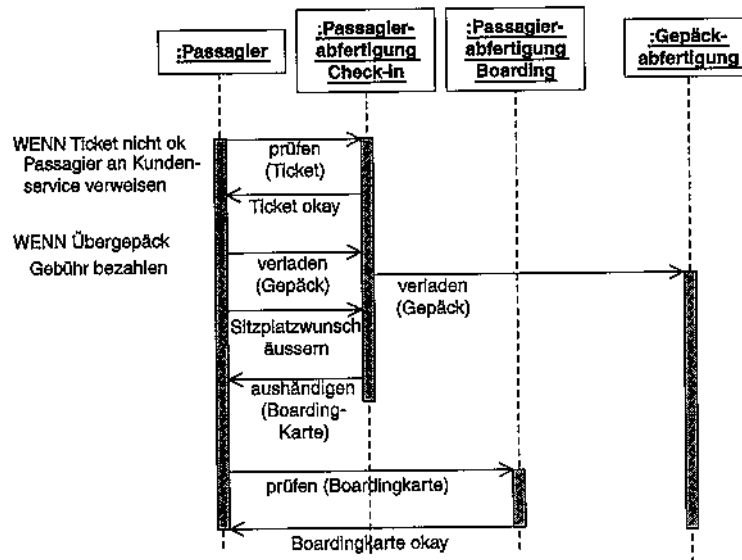


Abbildung 10 Anwendungsfallübergreifendes Sequenzdiagramm des Geschäftsprozesses *Passagierabfertigung*

**Beobachtung.** Im Sequenzdiagramm (*Abbildung 10*) ist ebenso wie im Aktivitätsdiagramm (*Abbildung 7*) erkennbar, ob Akteure gemeinsam oder unabhängig voneinander einen Geschäftsprozess ausüben können - was man im Anwendungsfall-Diagramm (*Abbildung 5*) *nicht* erkennen kann!

### 3.5 Die interne Sicht

Die interne Sicht beschreibt die internen Abläufe sowie die Aktivitäten, Beziehungen und Strukturen des Geschäftssystems. Die IT-Systeme und Menschen innerhalb des Geschäftssystems sind für die Bereitstellung der Leistungen des Geschäftssystems verantwortlich. Wir verlassen also das Umfeld des Geschäftssystems und tauchen in die *Black-box* ein, vgl. *Abbildung 3*.

Wir werden zur Darstellung der internen Sicht die folgenden Diagramme verwenden.

- **Paketdiagramme.** Sie beschreiben die Organisationseinheiten in Form von „Paketen“.
- **Klassendiagramme.** Sie beschreiben die Zusammenhänge und Beziehungen zwischen Mitarbeitern, Geschäftsobjekten und weiteren Organisationseinheiten.
- **Aktivitätsdiagramme.** Sie beschreiben die Geschäftsprozesse innerhalb des Geschäftssystems, also die Leistungserstellung durch die geschäftssysteminternen Ressourcen.

### 3.5.1 Organisationseinheiten als Pakete (*packages*)

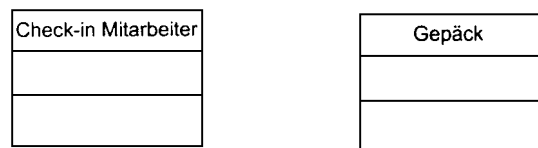
Bei der internen Sicht auf das Geschäftssystem spielt der Aufbau der Organisationseinheiten eine Rolle.

Für die Ausführungen von Geschäftsprozess-Aktivitäten können Organisationseinheiten verantwortlich sein. Eine **Organisationseinheit** ist eine unternehmeninterne Zusammenfassung einzelner Stellen oder Mitarbeiter. Organisationseinheiten werden in UML als Pakete dargestellt, die Mitarbeiter, Geschäftsobjekte und weitere Organisationseinheiten und deren Beziehungen enthalten können. Organisationseinheiten befinden sich grundsätzlich innerhalb des Geschäftssystems oder stellen das Geschäftssystem selbst dar. Organisationseinheiten, die sich außerhalb des Geschäftssystems befinden, sind Akteure.

Jedoch dürfen Pakete in der UML, die Organisationseinheiten darstellen, nicht mit Organigrammen verwechselt werden, denn sie enthalten neben den Mitarbeitern und Unterorganisationseinheiten eben auch die Geschäftsobjekte, also z.B. *Ticket* oder *Gepäck*.

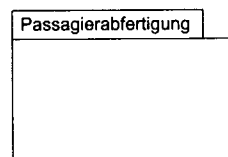
Folgende UML-Elemente werden wir für Pakete verwenden, die Organisationseinheiten darstellen:

**Klasse.** Eine Klasse repräsentiert ein fachlich relevantes Konzept, also eine Rolle,



eine Menge von Personen, Dingen oder Gegenständen, die für die jeweilige Sicht von Bedeutung ist. Klassen bilden das Vokabular des zu modellierenden Systems. Beispiele für Klassen sind Check-in-Mitarbeiter oder Gepäck, aber auch Passagier, Flugzeug, Ticket. Grafisch wird eine Klasse als Rechteck dargestellt, das in drei kleinere Rechtecke aufgeteilt ist. Das oberste Rechteck enthält den Namen der Klasse, das darunterliegende die Attribute (Eigenschaften) und das unterste die Methoden (Verhalten) der Klasse (soweit sie für die jeweilige Sicht relevant sind). Zur ersten Modellierung des Geschäftssystems werden wir Attribute und Methoden nicht beachten, daher bleiben die beiden unteren Rechtecke leer und werden erst im weiteren Vorgehen gefüllt.

**Paket.** Ein Paket wird in Form einer Akte mit Reiter (einem größeren Rechteck und ei-



nem kleineren links oben angeheftetem Rechteck) dargestellt. Pakete sind in UML die Möglichkeit zur Gruppierung. Den Inhalt des Pakets (z.B. Klassen, Objekte, weitere Pakete, ...) setzt man in den Hauptkasten.

In unserem Fallbeispiel betrachten wir die Organisationseinheit *Passagierabfertigung*.

### 3.5.1.1 Das Erstellen von Paketen

Folgende Schritte sind erforderlich, um Pakete zu erstellen.

**(i) Aus welchen Mitarbeitern und Geschäftsobjekten besteht das Geschäftssystem?** Zunächst bildet das gesamte Geschäftssystem die Organisationseinheit, die dargestellt werden soll. In unserem Fallbeispiel ist das die Passagierabfertigung *Abbildung 11*. Die relevanten Mitarbeiter-Rollen und Geschäftsobjekte sind dargestellt.

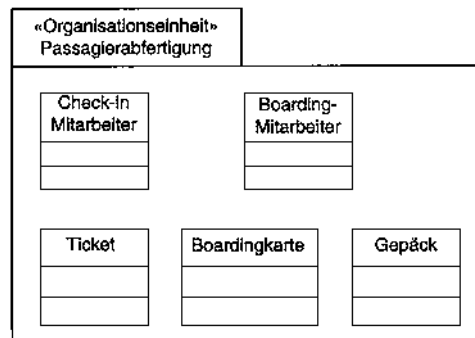


Abbildung 11 Das Paket *Organisationseinheit Passagierabfertigung*

Hilfreich sind i.a. bereits vorhandene Stellenbeschreibungen und Organigramme.

**(ii) Weitere Organisationseinheiten finden.** Kann die Organisationseinheit in weitere Organisationseinheiten (Abteilungen, Gruppen, Teams) unterteilt werden? Eine weitere Aufteilung ist allerdings nur dann sinnvoll, wenn sie für die Darstellung der Geschäftsprozesse wichtig ist.

In unserem Fallbeispiel unterteilen wir die *Passagierabfertigung* in die Organisationseinheiten *Check-in* und *Boarding* unterteilt.

**(iii) Zuordnung der Mitarbeiter und Geschäftsobjekte zu den Organisationseinheiten.** In *Abbildung 12* ist erkennbar, dass durch die Aufteilung der Geschäftsobjekte die Struktur klarer erkennbar wird.

**(iv) Verifikation.** Anhand der folgenden Checkliste verifiziert man abschließend das Paket einer Organisationseinheit in der internen Sicht:

- o Sind alle Mitarbeiter und Organisationseinheiten im Paket enthalten, die von den Leistungen des Geschäftssystems betroffen sind oder mit ihnen zu tun haben?
- o Sind alle Geschäftsobjekte im Paket enthalten, die für die Leistung des Geschäftssystems erforderlich sind oder mit ihnen zu tun haben?

### 3.5.2 Klassendiagramme (*class diagrams*)

Klassendiagramme gehören zu den wichtigsten Diagrammen der UML. Sie bilden letztendlich die kleinsten „Programmeinheiten“ des Softwaresystems. Beschränkt auf die interne Sicht, wie wir es hier tun, stellen sie die *strukturellen Teile* eines Geschäfts-

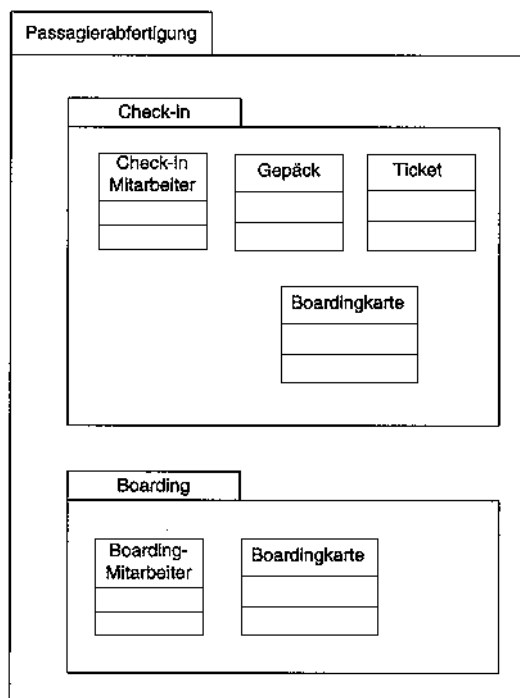


Abbildung 12 Organisationseinheit *Passagierabfertigung*

systems dar, also die Beziehungen, die Mitarbeiter, Geschäftsobjekte und außenstehende Parteien miteinander haben.

Wir werden zur Darstellung des Geschäftssystems in der internen Sicht stark vereinfachte Klassendiagramme verwenden. Einerseits gilt auch hier: *Weniger ist mehr*. Sobald man die vielfältigen Möglichkeiten der Klassendiagramme in der UML ausnutzt, sind sie nicht mehr unbedingt einfach lesbar. Andererseits muss man gerade auf der Ebene der Geschäftsprozessmodellierung davon ausgehen, dass nicht alle Beteiligten mit dem Klassenbegriff vertraut sind. Die UML hat jedoch den Anspruch, die Kommunikation unterschiedlicher Beteiligter zu erleichtern.

Wir werden Klassendiagramme also in mehreren Schritten einführen und zunächst nur diejenigen Aspekte einführen, die wir für die Geschäftsprozessmodellierung benötigen.

Klassendiagramme beinhalten folgende Elemente:

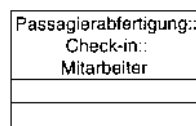
**1. Klasse.** Der Begriff der Klasse wurde bereits im Zusammenhang mit Paketen



erläutert. Da er zentral für die UML ist, wiederholen wir ihn hier noch einmal: Eine Klasse repräsentiert ein fachlich relevantes Konzept oder ein Schema, also eine Rolle, eine Menge von Personen, Dingen oder Gegenständen, die für die jeweilige Sicht von Bedeutung ist. Beispiele für Klassen sind Check-in Mit-

arbeiter oder Gepäck, aber auch Passagier, Flugzeug, Ticket. Grafisch wird eine Klasse als Rechteck dargestellt, das in drei kleinere Rechtecke aufgeteilt ist. Das oberste Rechteck enthält den Namen der Klasse, das darunterliegende die *Attribute* (Eigenschaften) und das unterste die *Methoden* (Verhalten) der Klasse (soweit sie für die jeweilige Sicht relevant sind). Zur Geschäftsprozessmodellierung werden Attribute und Methoden nicht beachtet, daher bleiben die beiden unteren Rechtecke leer und werden erst im weiteren Vorgehen gefüllt. Man kann sie aber auch weglassen, solange sie noch nicht gefüllt sind.

Es ist manchmal sinnvoll, das Paket anzugeben, in dem sich die Klasse befindet. Dies geschieht, indem man den Paketnamen mit zwei Doppelpunkten dem Klassennamen voranstellt:

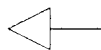


**2. Assoziation.** Eine Assoziation repräsentiert eine Beziehung, die eine genau



definierte fachliche Bedeutung hat. Die Assoziation kann mit einem Namen beschriftet sein. Will man dem Namen der Assoziation eine Richtung zuweisen, so lässt man ein gefülltes Dreieck in die Richtung zeigen, in der der Name gelesen werden soll.<sup>2</sup>

**3. Generalisierung.** Die Generalisierung ist eine Beziehung zwischen einer all-



gemeinen und einer speziellen Klasse. Der Pfeil weist auf die allgemeinere Klasse. Die Generalisierung dient der hierarchischen Strukturierung. Sollen Geschäftsobjekte unter einen übergreifenden Begriff zusammengefasst werden, ist die Generalisierung das richtige Instrument. **Obacht:** Für Mitarbeiter sollte man eher ein Paketdiagramm verwenden, wenn man die hierarchische Struktur darstellen will: In der Regel ist ein Vorgesetzter keine Generalisierung seiner Mitarbeiter!

### 3.5.2.1 Erstellen eines Klassendiagramms

**(i) Klassen finden.** Für das Klassendiagramm des Geschäftsmodells können die Klassen des Pakets verwendet werden, also die Mitarbeiter und Geschäftsobjekte. Auch die Akteure der Anwendungsfalldiagramme sind Klassen, die in das Klassendiagramm übernommen werden. Oft ergeben neue Akteure auch neue Geschäftsobjekte.

Für unser Fallbeispiel ergeben sich die Klassen aus *Abbildung 13*.

**(ii) Beziehungen zwischen den Klassen herstellen.** Die Zusammenhänge zwischen den gefundenen Klassen sowie Geschäftsregeln werden im Klassendiagramm als As-

<sup>2</sup>Ein Hinweis: Im ersten Schritt werden wir die sogenannten *Multiplizitäten* noch weglassen, die zu Assoziationen in der UML gehören. Sie werden für die Prozessmodellierung an dieser Stelle nicht gebraucht. Multiplizitäten werden wir in einem späteren Schritt einführen.

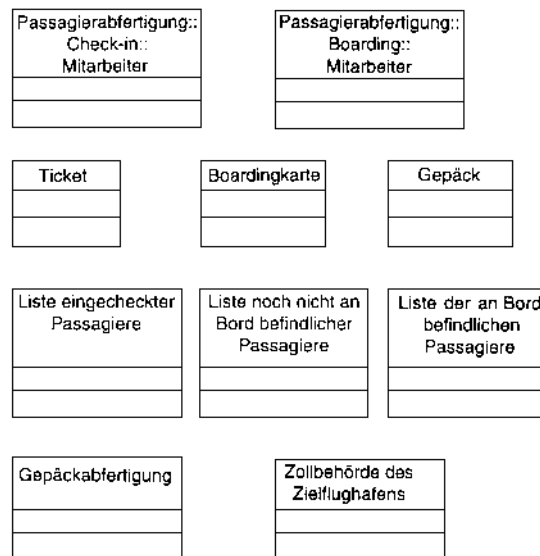


Abbildung 13 Klassen der internen Sicht des Geschäftsmodells

soziationen modelliert. Die strategische Frage lautet: Welche fachlichen Beziehungen bestehen unter den Mitarbeitern, Geschäftsobjekten und weiteren Objekten?

Obwohl mit den bereits gefundenen Klassen begonnen wird, findet man oft in diesem Arbeitsschritt auf Grund der fachlichen Diskussionen weitere Klassen.

**(iii) Die Beziehungen konkretisieren.** Die Beziehungen zwischen den einzelnen Klassen sollten mit aussagekräftigen Namen versehen werden. Damit wird das Klassendiagramm gut und intuitiv verständlich. In der Regel wird den Beziehungen eine Richtung mitgegeben, in der man die Beziehung lesen kann, siehe *Abbildung 14*.

**(iv) Generalisierungen einfügen - Können Geschäftsobjekte gruppiert werden?** Es kann sinnvoll sein, Geschäftsobjekte zu einer weiteren übergeordneten Klasse zusammenzufassen.

In unserem Fallbeispiel ist es hilfreich, die drei Passagierlisten zu einer Klasse *Passagierliste* zu verallgemeinern, *Abbildung 15*.

**(v) Verifikation.** Das fertige Klassendiagramm kann anhand der folgenden Checkliste geprüft werden:

- o Ist das Klassendiagramm vollständig? Sind insbesondere alle in den Paketdiagrammen vorhandenen Klassen im Klassendiagramm vorhanden?
- o Sind die Beziehungen aussagekräftig bezeichnet und sind die Pfeile in der richtigen Richtung?
- o Ist das Klassendiagramm korrekt? Das intensive, gemeinsame Lesen des Klassendiagramms mit Wissensträgern und das Durchspielen jeder Dienstleistung bringt die meisten Fehler zu Tage.

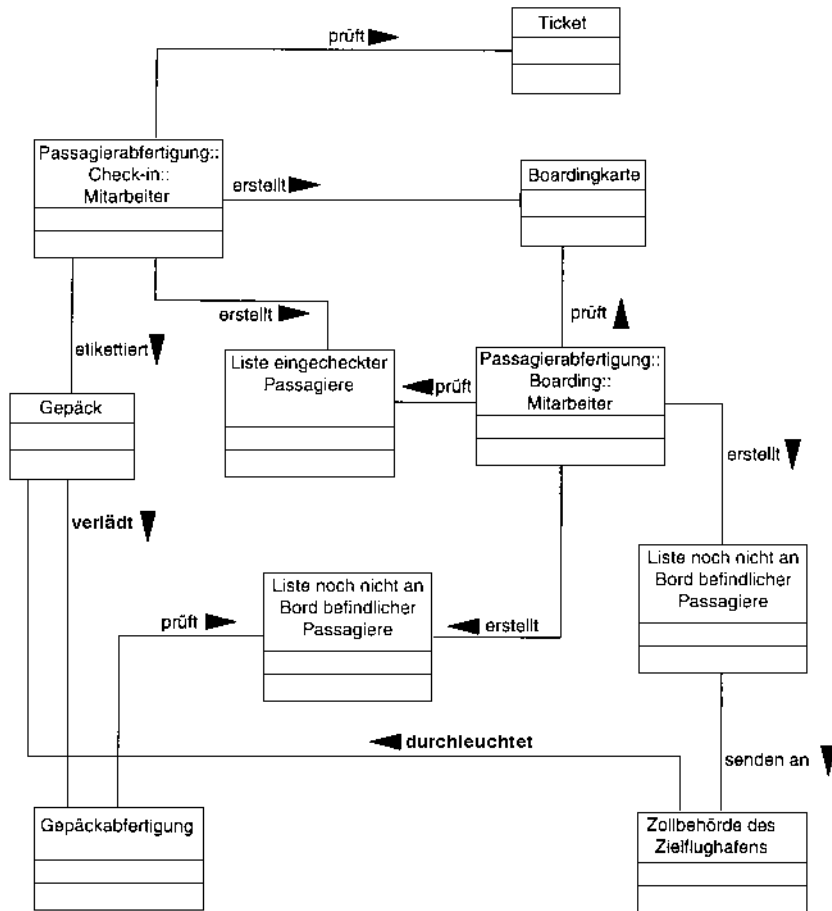


Abbildung 14 Klassendiagramm der Passagierabfertigung

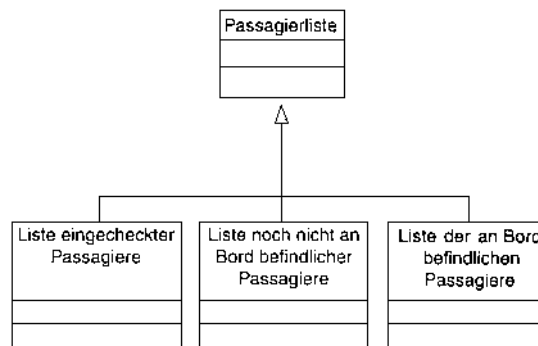


Abbildung 15 Generalisierung im Klassendiagramm

- o Ist der Detaillierungsgrad optimal? Ist es detailliert genug, um alles abzudecken? Ist es nicht zu detailliert und verschleiert durch Unübersichtlichkeit wichtige Aspekte?

### 3.5.2 Lesen von Klassendiagrammen

Das Lesen von Klassendiagrammen ist sehr intuitiv. In der internen Sicht sollte man mit den Mitarbeiterklassen beginnen. In *Abbildung 14* kann man z.B. an der Beschriftung der Klasse *Check-in-Mitarbeiter* erkennen, dass er der Organisationseinheit *Check-in* angehört, die wiederum der Organisationseinheit *Passagierabfertigung* untergliedert ist.

Dass die *Passagierabfertigung* und der *Check-in* Organisationseinheiten sind, sollte sich aus der vorher gemachten Paketerstellung ergeben: Sie sollten als Pakete dargestellt sein.

Der *Check-in-Mitarbeiter* ist mit vier Geschäftsobjekten assoziiert: Er prüft das Ticket, erstellt die Boardingkarte und die Liste der eingetragenen Passagiere und etikettiert das Gepäck.

### 3.5.3 Aktivitätsdiagramme

Aktivitätsdiagramme eignen sich auch zur Darstellung der internen Sicht. Allerdings sind jetzt nicht mehr die Akteure der externen Sicht im Mittelpunkt, sondern die Mitarbeiter.

Die Erstellung eines Aktivitätsdiagramms haben wir bereits kennen gelernt, sie geschieht hier analog. Für die interne Sicht ist jedoch zu beachten, dass nur die internen Abläufe und Geschäftsprozesse relevant sind.

Für unser Fallbeispiel ergibt sich das Aktivitätsdiagramm *Abbildung 16* für den Geschäftsprozess *Boarding eines Flugs* (nicht nur eines einzelnen Passagiers).

Dieses Aktivitätsdiagramm kann durch weitere Aktivitätsdiagramme verfeinert werden, beispielsweise durch den Geschäftsprozess *Boarding eines Passagiers* aus der Sicht des Boarding-Mitarbeiters (*Abbildung 17*).

## 3.6 Zusammenfassung: Modellierung von Geschäftsprozessen mit UML

Wir haben in diesem Abschnitt einige Diagramme der UML kennen gelernt, mit denen man Geschäftssysteme und Geschäftsprozesse modellieren kann.

Betriebswirtschaftlich bezeichnet ein Geschäftssystem die Wertschöpfungskette, die einen Wertschöpfungsprozess, d.h. eine Leistungserstellung, beschreibt. Ein Unternehmen kann ein oder mehrere Geschäftssysteme umfassen. Ein Beispiel eines Geschäftssystems ist die Passagierabfertigung des UML-Flughafens, die Gepäckabfertigung oder das Flughafenkontrollzentrum.

Ein Geschäftssystem beschreibt statische Aspekte, wie Organisationsstrukturen, und Geschäftsobjekte, wie Tickets oder Aufträge. Daneben umfasst es auch dynamische Abläufe, nämlich die Geschäftsprozesse. Ein Geschäftsprozess ist ein Vorgang, um ein Unternehmensziel oder ein betriebswirtschaftliches Ziel zu erreichen. Geschäftsprozesse können sein: Zeitungen verkaufen, Passagiere einchecken, Start- und Landeerlaubnisse verteilen.

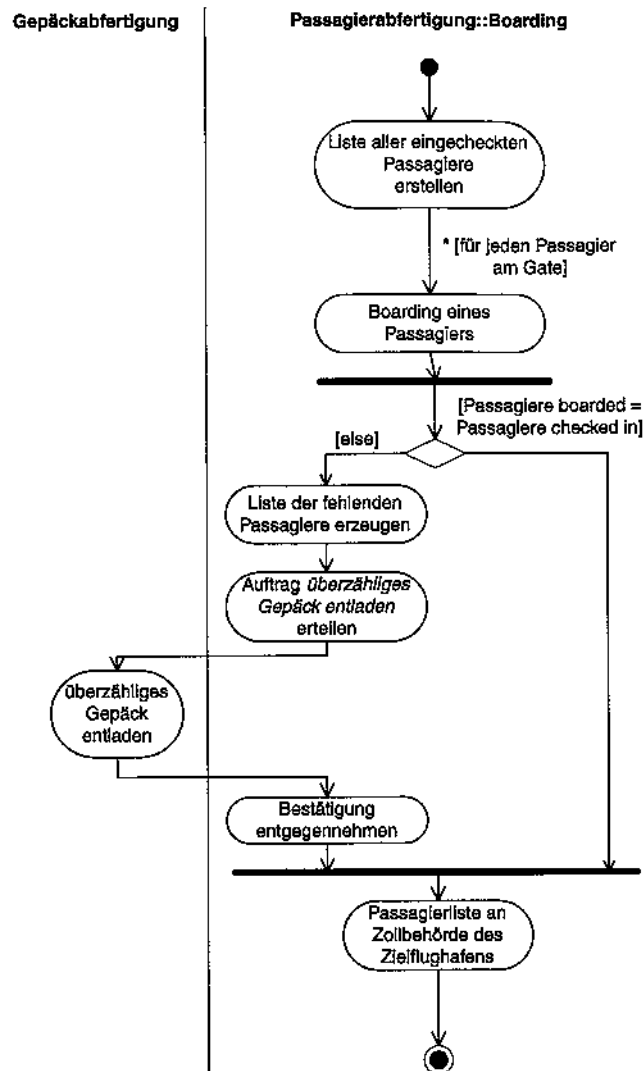


Abbildung 16 Aktivitätsdiagramm des internen Geschäftsprozesses *Boarding eines Flugs*

Ein Geschäftssystem kann aus zwei Sichten gesehen werden, der externen und der internen Sicht.

- Die externe Sicht beschäftigt sich nur mit den äußeren Beziehungen des Geschäftssystems, also den Akteuren wie Kunden, Geschäftspartner oder andere Geschäftssysteme innerhalb des Unternehmens. Sie alle sehen das Geschäftssystem als *Black-box*. Zur ersten Strukturierung eignen sich Anwendungsfalldiagramme, Aktivitätsdiagramme und Sequenzdiagramme. Aktivitäts- und Sequenzdiagramme kann man sowohl zur Darstellung des gesamten Geschäftssystems mit den jeweiligen Akteuren verwenden, als auch für die einzelnen Anwendungsfälle.
- Die interne Sicht betrachtet die Mitarbeiter, die Geschäftsobjekte und ihre Beziehungen untereinander, sowie die Aktivitäten der Mitarbeiter. Zur ersten

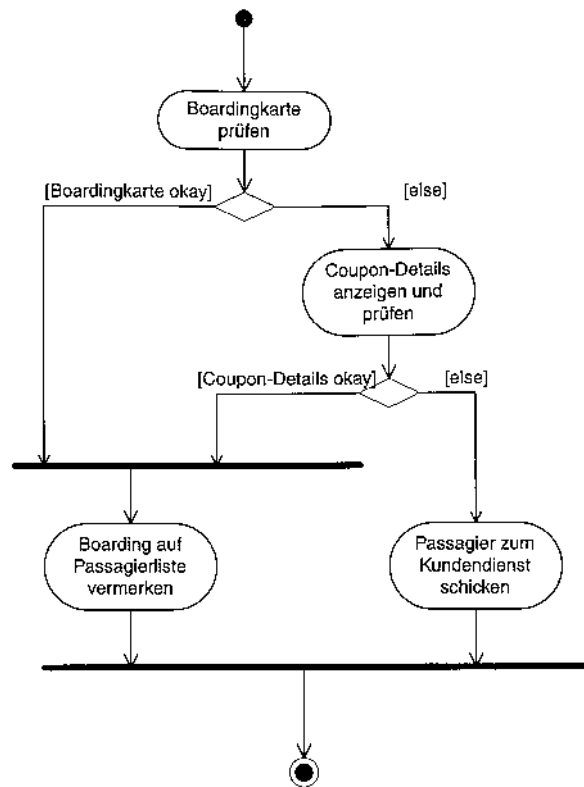


Abbildung 17 Aktivitätsdiagramm des Geschäftsprozesses *Boarding eines Passagiers* aus der internen Sicht eines Boarding Mitarbeiters

Strukturierung können Pakete erstellt werden, mit deren Hilfe man die Klassen ordnet und danach Klassendiagramme und Aktivitätsdiagramme erstellt.

## 4 Das Modell des IT-Systems

In diesem Kapitel wird gezeigt, wie mit Hilfe der UML ein konzeptionelles Modell eines IT-Systems erstellt werden kann. Das Modell werden wir aus vier verschiedenen Sichten sehen, die jeweils bestimmte Aspekte des Systems hervorheben und eng miteinander verknüpft sind. Wir werden - wie schon im Fall der Geschäfts-systemmodellierung - Anwendungsfalldiagramme verwenden, unsere Kenntnisse über Sequenzdiagramme und Klassendiagramme ergänzen, und zwei neue Diagrammtypen der UML kennen lernen, das Zustandsdiagramm und das Kollaborationsdiagramm.

### 4.1 Die Sicht von außen

Benutzt jemand ein Gerät, z.B. ein Auto, einen Geldautomat oder ein Mobiltelefon, so interessiert kaum, wie das Gerät im Innern aussieht oder wie es genau funktioniert. „Es ist mir egal, wie es funktioniert. Hauptsache es funktioniert!“ Dem Benutzer ist es wichtig, was er mit einem Gerät machen kann, d.h. welche Funktionalitäten es ihm bietet.

Diese Anwendersicht ist die (bereits oben erwähnte) **Black-box-Sicht**. Das System oder das Gerät ist wie eine schwarze Schachtel, in die man nicht hineinschauen kann und von der man nicht weiß, wie sie funktioniert, aber man weiß, was sie tun kann und wofür man sie braucht. Genau diese Sichtweise sollte der Anwender eines IT-Systems im Idealfall haben. Er benutzt das IT-System zur Erreichung seiner Ziele genau wie einen Kopierer, ein Auto oder einen Fernseher.

Daher ist die Sicht von außen wesentlich für ein IT-Systemmodell. Hier wird festgelegt, was der zukünftige Anwender von dem System erwartet. Mit der hier festgelegten Funktionalität wird letztlich getestet werden, ob das System die Vorgaben erfüllt.

Die Sicht von außen besteht aus den Elementen Anwendungsfalldiagramm, Sequenzdiagramm und Oberflächenprototyp.

#### 4.1.1 Die Benutzeroberfläche und die Gebrauchsanweisung

Ein wesentliches Element eines IT-Systems ist seine **Benutzeroberfläche (user interface)**. Bei einem Geldautomaten besteht diese Oberfläche aus einem kleinen Bildschirm, den Tasten, den Öffnungen für Karte, Geldscheine und Quittungen, sowie dem Piepston.

Die Benutzeroberfläche ist der einzige Zugang, den ein Anwender auf die Funktionalität des Systems hat. Fehlt bei einem Videorecorder die Aufnahmetaste, so ist es nicht möglich, etwas aufzunehmen, auch wenn das Gerät intern dafür ausgerüstet ist. *Die Benutzeroberfläche ist eine Sicht auf die Funktionalität des Systems. Was in dieser Sicht fehlt, ist nicht ansprechbar.*

Die Benutzeroberfläche zeigt jedoch eine statische Sicht auf das System. Sie zeigt nicht, wie das System benutzt wird, also welche Bedienungselemente in welcher Reihenfolge betätigt werden müssen, um eine bestimmte Aufgabe zu erledigen. Zu einer Benutzeroberfläche gehört daher eine **Gebrauchsanweisung**, also eine Beschrei-

bung darüber, welche Aktionen überhaupt möglich sind und welche Reihenfolge eingehalten werden muss, um das System sinnvoll zu verwenden. Ein sinnvoller Ablauf für die Benutzung eines Telefons ist beispielsweise: Hörer abnehmen, auf den Summton warten, eine gültige Telefonnummer tippen, warten bis sich jemand meldet, sprechen, Hörer auflegen.

Aus Sicht des IT-Systems entsprechen solche Abläufe den **Anwendungsfällen** der UML. Sie sind die Gebrauchsanweisung für die Benutzeroberfläche. Nur was in der Gebrauchsanweisung definiert ist, ist ein sinnvoller Ablauf. Wir werden zur Darstellung der Gebrauchsanweisung Anwendungsfalldiagramme und Sequenzdiagramme verwenden.

Das beste Vorgehen, um Anwendungsfälle eines IT-Systems zu modellieren, ist sich konkret einen Anwender vorstellen, der vor dem Bildschirm sitzt und mit dem IT-System arbeitet, siehe *Abbildung 18*. Der Anwender wird zum Akteur, und der Anwen-



Abbildung 18 Akteur, „der“ einen Anwendungsfall ausführt

denungsfall ist nichts anderes als die Beschreibung seiner Tätigkeiten. Ein Akteur

- interagiert direkt mit dem System
- steht selbst aber außerhalb des Systems.

Der Akteur bedient das IT-System also immer direkt. Auch beim übergeordneten Geschäftssystem steht der Akteur außerhalb, ist also z.B. Kunde oder Geschäftspartner. Der Mitarbeiter dagegen ist Bestandteil des Geschäftssystems und kann daher niemals Akteur des Geschäftssystems sein, siehe *Abbildung 19*.

Die Sicht von außen auf das IT-System besteht aus den folgenden drei Elementen:

- Anwendungsfalldiagramme, die als Akteure alle Anwender zeigen und als Anwendungsfälle alle Aufgaben, welche die Anwender mit dem IT-System ausführen können;
- Sequenzdiagramme, die für einzelne Anwendungsfälle den Ablauf der Interaktion des Anwenders mit dem IT-System zeigen;

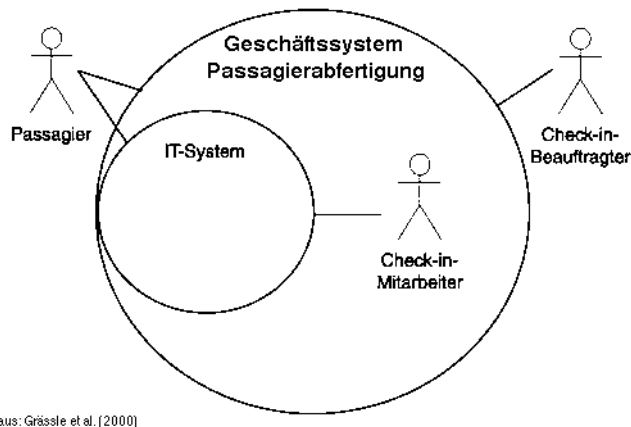


Abbildung 19 Akteure des Geschäftssystems und des IT-Systems

- Prototypen, die die Benutzeroberfläche zeigen.

Alle drei Elemente zusammen ergeben einen guten Überblick über das IT-System aus Benutzersicht.

#### 4.1.2 Das Anwendungsfalldiagramm des IT-Systems

Abbildung 20 zeigt das Anwendungsfalldiagramm mit den Akteuren *Check-in-Mitarbeiter*

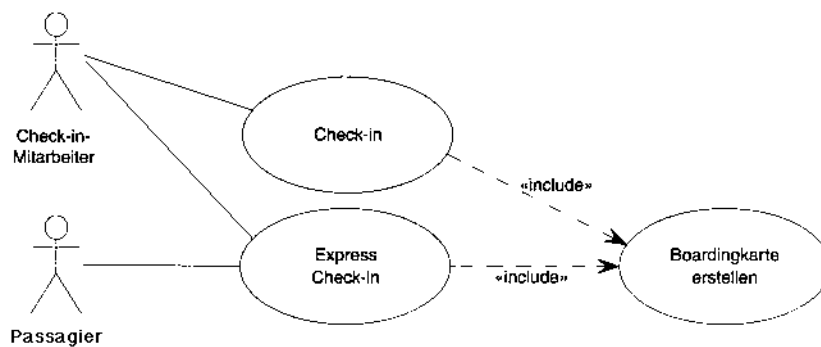


Abbildung 20 Anwendungsfalldiagramm

ter und *Passagier* sowie den Anwendungsfällen *Check-in*, *Express-Check-in* und *Boarding-Karte erstellen*. Beim Lesen des Diagramms beginnt man je nach Interesse bei einem Akteur oder einem Anwendungsfall.

Beginnend beim Akteur *Check-in-Mitarbeiter* findet man Assoziationen zu den Anwendungsfällen *Check-in* und *Express-Check-in*. Das bedeutet, dass Personen, die dem IT-System in der Rolle des Check-in-Mitarbeiters auftreten, im IT-System die Anwendungsfälle *Check-in* und *Express-Check-in* ausführen können. In dem Falle, dass das Diagramm nicht mehr ergänzt werden kann, sind diese beiden Anwendungsfälle *alles*, was ein Check-in-Mitarbeiter mit dem System machen kann.

Der Akteur *Passagier* hat eine Assoziation zum Anwendungsfall *Express-Check-in*, was bedeutet, dass Personen in der Rolle des Passagiers direkt mit dem IT-System den Anwendungsfall *Express-Check-in* ausführen können. Auch der Akteur *Check-in-Mitarbeiter* hat eine Assoziation zum Anwendungsfall *Express-Check-in*, d.h. auch er kann ihn ausführen. Es bedeutet aber *nicht*, dass Passagier und Check-in-Mitarbeiter beim Express-Check-in zusammenarbeiten.

Natürlich checkt sich auch beim Anwendungsfall *Check-in* ein Passagier ein, aber Akteur für das IT-System ist immer nur jemand, der mit dem IT-System direkt interagiert. Beim Anwendungsfall *Express-Check-in* ist das entweder der Passagier, der mit seinem Ticket an einem Automaten eine Boardingkarte löst, oder ein Check-in-Mitarbeiter, der ihm diese Arbeit abnimmt.

Für das Geschäftssystem ist jedoch in jeden Fall der Passagier der Akteur, denn er steht außerhalb des Geschäftssystems, während der Check-in-Mitarbeiter kein Akteur ist, da er im Geschäftssystem arbeitet.

Das Anwendungsfalldiagramm zeigt zudem noch eine include-Beziehung, die die Anwendungsfälle *Check-in* und *Express-Check-in* beide zum Anwendungsfall *Boardingkarte erstellen* haben, d.h. in beiden Fällen wird eine Boardingkarte erstellt. Beachten Sie, dass das eine einfache Art ist, um Teile von Anwendungsfällen wiederzuverwenden.

### 4.1.3 Das Sequenzdiagramm des IT-Systems

Abbildung 21 zeigt das Sequenzdiagramm des Anwendungsfalles *Boarding*. Ein Se-

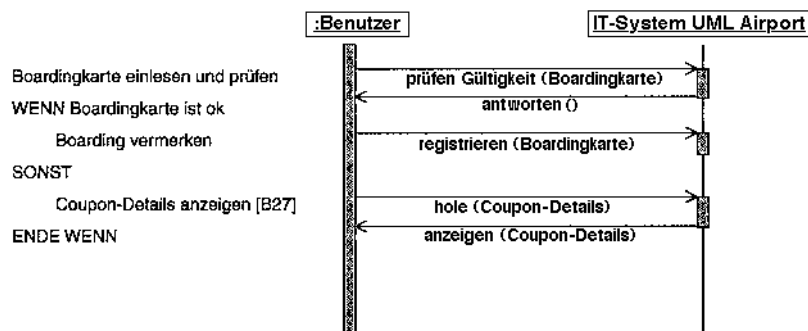


Abbildung 21 Das Sequenzdiagramm des IT-Systems

quenzdiagramm beschreibt meist am einfachsten den Ablauf eines Anwendungsfalles.

Der Ablauf unseres Beispiels ist am Kommentar am linken Rand ersichtlich. Als erstes wird die Boardingkarte eingelesen und geprüft, indem die Nachricht *prüfen Gültigkeit(Boardingkarte)* von dem Benutzer an das IT-System geschickt wird.

Wie diese Nachricht intern behandelt wird, ist in diesem Diagramm nicht ersichtlich, das IT-System ist eine *Black-box*. Die Gültigkeit der Boardingkarte wird geprüft, wahrscheinlich anhand der eingelesenen Werte der Karte und der gespeicherten Informationen aus vorhergehenden Prozessen. Aufgrund des Resultats, welches das IT-Sy-

stem zurückliefert, kann der Anwender erkennen, ob die Boardingkarte gültig ist oder nicht.

Ist die Karte in Ordnung, so wird die Nachricht an das IT-System geschickt, dass sie registriert wird, und im IT-System kann vermerkt werden, dass der Passagier eingestiegen ist. Ist die Boardingkarte nicht in Ordnung, so sollen am Bildschirm Details angezeigt werden. Die Anzeige der Details soll mit der Referenz [B27] geschehen, die auf eine Bildschirmmaske in einem Prototypen Bezug nimmt, siehe *Abbildung 22*.

The image shows a screenshot of a software dialog box titled "Coupon-Details". It contains several input fields for flight information:

|            |                  |            |        |
|------------|------------------|------------|--------|
| Ticket#    | G97AH-8825.3/2   | Sitzplatz: | 17A    |
| Passagier  | Huber, MRS       | Flug:      | SR686  |
| Von:       | Zurich           | Nach:      | Malaga |
| Datum/Zeit | 21.05.2001 11:55 | Raucher:   | Nein   |
| Klasse:    | H                |            |        |

An "OK" button is located at the bottom right of the dialog box. Below the dialog box, there is a small text reference: "aus Grässle et al. [2000]"

Abbildung 22 Dialogfenster [B27] aus dem Anwendungsfall *Boarding*

Zu bemerken ist, dass bei diesem Sequenzdiagramm die Entscheidung, ob registriert wird oder nicht, bei dem Benutzer liegt, und nicht vom System automatisch durchgeführt wird.

## 4.2 Die strukturelle Sicht

### 4.2.1 Objekte und Klassen

Eine Grundidee der Objektorientierung ist die möglichst gute Abbildung eines realen Objekts der Welt in einem IT-System. Man benötigt dafür ein Modell des Objekts, d.h. man muss „abstrahieren“: das für den beabsichtigten Zweck Wesentliche hervorheben und alle anderen Aspekte weglassen.

Um also Objekte erfolgreich modellieren und im IT-System abbilden zu können, muss man wissen, wozu sie benötigt werden. Das Objekt „Herr Meier“ wird in einem Kundenverwaltungssystem ganz anders aussehen als im Steuerregister oder in einem medizinischen Verwaltungssystem.

Bei der Modellierung der realen Welt wird man in zwei Schritten vorgehen. Zunächst wird von einzelnen realen Individuen hin zu Objekten abstrahiert, in einem zweiten Schritt werden ähnliche Objekte zu Klassen zusammengefasst, siehe *Abbildung 23*.

- 1.) Die direkte Abbildung eines Objekts der realen Welt (einem Gegenstand, einer Person, einer Rolle, einem Vertrag, einem Konzept, einem Begriff o.ä.) in ein Modell führt zu einem *Objekt des Modells*. Dabei besteht eine 1:1-Beziehung zwischen dem Objekt der realen Welt und dem Objekt des Modells, d.h. das Objekt des Modells repräsentiert genau ein Exemplar der realen Welt. Die Definition eines Objekts im Modell ist ein erster Abstraktionsschritt, da hier nur noch die relevanten Attribute des Exemplars aus der realen Welt beachtet werden. Z.B. wird der Mensch Herr Meier in dem Objekt **Herr Meier** auf diejenigen

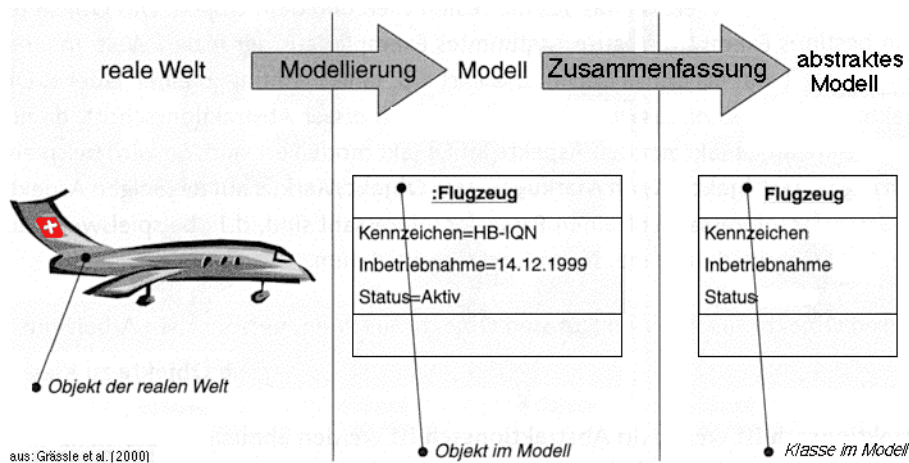


Abbildung 23 Modellierung

Aspekte reduziert, die für die Passagierabfertigung relevant sind, also Anrede, Vorname, Name und Geburtsdatum.

- 2.) In einem zweiten Abstraktionsschritt werden ähnliche Objekte zu *Klassen* zusammengefasst. „Ähnlich“ bedeutet hier,
- dass die Rolle im betrachteten System dieselbe ist,
  - dass die gleichen Attribute (Merkmale) relevant sind und
  - dass die Objekte sich in ihrem Verhalten gleichen.

Oft werden die beiden Abstraktionsschritte zusammengefasst, d.h. man bildet direkt Klassen.

Eine Klasse hat also im wesentlichen zwei Bedeutungen, siehe *Abbildung 24*:

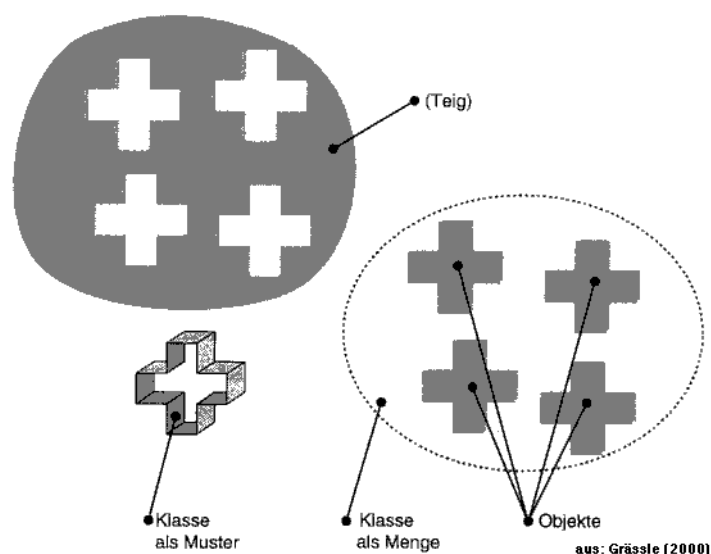


Abbildung 24 Klassen und Objekte

- Einerseits ist die Klasse das *Muster*, nach dem ihre Objekte gebildet werden,
- andererseits ist die Klasse die *Menge* der Objekte, die nach ihrem Muster gebildet wurden.

#### 4.2.2 Generalisierung, Spezialisierung und Vererbung

Man kann aus mehreren ähnlichen Klassen gemeinsame Charakteristika extrahieren und sie zu einer *Oberklasse* (*super class*) zusammenfassen. Dieser Vorgang heißt *Generalisierung* oder *Verallgemeinerung*. In *Abbildung 25* haben die Klassen *Gepäckstück*

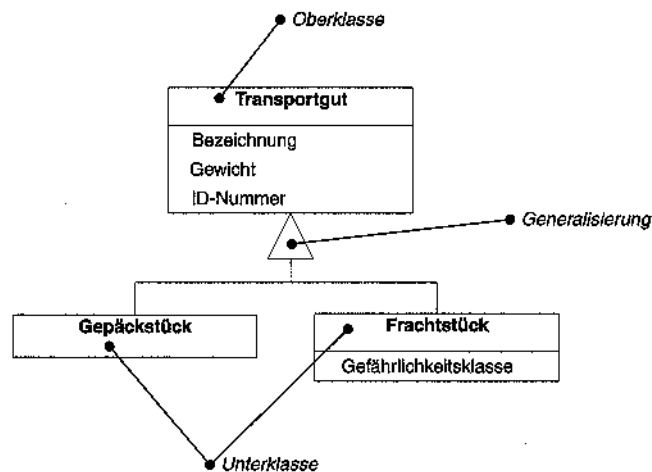


Abbildung 25 Generalisierung

*stück* und *Frachtstück* teilweise dieselben Attribute. Auch fachlich sind sich die beiden Klassen ähnlich. Bei der Generalisierung wird das Gemeinsame zusammengenommen und eine Oberklasse *Transportgut* gebildet. Damit sind *Gepäckstück* und *Transportgut* Unterklassen der Klasse *Transportgut*. Beachten Sie, dass die gemeinsamen Attribute meist nur noch in der Oberklasse aufgeführt werden. Obwohl sie in den Unterklassen nicht aufgeführt sind, gelten sie dort trotzdem: Sie haben die Attribute von ihrer Oberklasse *geerbt*.

Sehr wichtig ist die fachliche Beziehung zwischen Ober- und Unterklassen:

- Alle Aussagen, die über eine Oberklasse gemacht werden, müssen auch für die Unterklassen gelten! Hat z.B. die Oberklasse *Transportgut* ein Attribut *Gewicht*, dann haben auch alle Unterklassen das Attribut *Gewicht* geerbt, auch wenn es nicht extra aufgeführt wird.
- Alles, was mit einem Objekt der Oberklasse getan werden kann, kann auch mit einem Objekt der Unterklasse getan werden! Beispiel: Wenn *Transportgut* verladen werden kann, müssen auch *Gepäckstücke* verladen werden können.
- In der Fachsprache des Systems, das man modelliert, muss die Unterklasse eine spezielle Art der Oberklasse sein. Beispiel: Das *Gepäckstück* ist ein spezieller Fall eines *Transportgutes*. Gegenbeispiel: Der *Flug* ist *kein* spezieller Fall einer *Flugnummer*.

**Warnung.** Das Konzept der Generalisierung und Vererbung ist ein sehr mächtiges Verfahren der Objektorientierung. Vererbung ist hervorragend geeignet, um Eigenschaften und Methoden wiederzuverwenden. Man sollte jedoch sehr sorgfältig damit umgehen. Die strategische Frage zur Verwendung ist:

- Sind die Objekte der geplanten Klasse stets eine spezielle *Art* von Objekten der Oberklasse, oder sind sie nur eine *Rolle* von Objekten der Oberklasse?

Nur bei „ist eine spezielle Art von“ darf generalisiert und vererbt werden, bei „eine Rolle von“ sollte die „*Komposition*“ (eine spezielle Art der Assoziation, s.u.) eingesetzt werden.

**Merkregel:** Die Komposition ist die Regel, Vererbung ist die Ausnahme (Coad & Mayfield 2000).

**Beispiel.** Eine Klasse *Person* kann zwar als Unterklassen *Angestellter* und *Student* haben, denn *Angestellter* „ist eine Art von“ *Person*, genau wie *Student*. Jedoch wird es schwierig, wenn *Student* noch während seines Studiums *Angestellter* wird! Viel einfacher ist die Situation, wenn man *Angestellter* und *Student* „als eine Rolle“ von *Person* auffasst.

#### 4.2.3 Zusammenfassung Klassen und Objekte

Ein Objekt hat stets drei Charakteristika:

- Zustand
- Verhalten
- Identität

|                |
|----------------|
| <b>:Objekt</b> |
| Attribute      |
| Methoden       |

(i) Der Zustand ist gegeben durch die Attribute.

(ii) Das Verhalten wird bestimmt durch die Methoden.

(iii) Identität bedeutet, dass ein Objekt einzigartig ist, selbst wenn sein Zustand und sein Verhalten gleich mit einem anderen Objekt sind.

Eine Klasse dagegen ist das Schema, das Muster oder eine Menge von Objekten.

**Beispiel:**

1.) Ein Kunde eröffnet ein Konto in einer Filiale.

-> 3 Klassen

2.) Ein Kunde eröffnet ein Konto in der Filiale Köln.

-> 2 Klassen und 1 Objekt

- 3.) Gerda Müller eröffnet ein Konto in der Filiale Köln.  
-> 1 Klasse und 2 Objekte
- 4.) Gerda Müller eröffnet das Konto 0815 in der Filiale Köln.  
-> 3 Objekte.

#### 4.2.3.1 Statische und dynamische Geschäftsregeln

**Geschäftsregeln** sind im IT-System abgebildete fachliche Regeln, die aus Geschäftsstrategien, Anforderungen und fachlichen Vorgaben hergeleitet werden können. Beispiele für Geschäftsregeln sind:

- Beim Check-in muss jedem Passagier ein Sitzplatz zugewiesen werden.
- Ein Sitzplatz darf für eine Flugstrecke nur einem Passagier zugewiesen werden.
- Ein Flug darf nicht gestrichen werden, wenn er bereits gestartet ist.

Die Geschäftsregeln lassen sich in zwei Gruppen einteilen:

- 1.) **Statische Geschäftsregeln.** Das sind Geschäftsregeln, die *jederzeit* überprüfbar sind. Sie haben zu tun mit den statischen Strukturen der Klassen und können daher in Klassendiagrammen dargestellt werden.
- 2.) **Dynamische Geschäftsregeln.** Das sind Geschäftsregeln, die *nur zu einem bestimmten Zeitpunkt* überprüfbar sind, nämlich wenn etwas passiert, d.h. bei Ereignissen. Sie können in Zustandsdiagrammen dokumentiert werden.

#### 4.2.4 Klassendiagramme und statische Geschäftsregeln

Wir haben Klassendiagramme bereits kennen gelernt in Abschnitt 3.5.2. Wir werden nun Erweiterungen der Assoziation einführen:

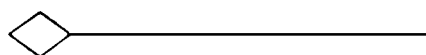
**Multiplizität oder Kardinalität.** Die Multiplizität erlaubt Aussagen über die *Anzahl*

\* 0..1

---

der Objekte, die an einer Assoziation beteiligt sind. Es gelten dabei die Konventionen aus *Abbildung 26*. Zu beachten ist, dass es in der UML also auch möglich ist, für die Ober- und Untergrenze beliebige Werte einzusetzen.

**Aggregation.** Die Aggregation ist eine speziell Assoziation mit der Bedeutung „be-



steht aus“ oder (die Gegenrichtung) „ist Teil von“. Die Raute deutet auf das Ganze, die andere Seite auf den Teil. Man kann folgende drei Aggregationstypen unterscheiden:

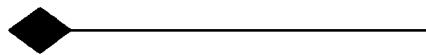
- das Ganze und seine Teile, z.B. das Auto (Ganzes) und sein Motor (Teil)
- der Behälter und sein Inhalt, z.B. das Auto (Behälter) und sein Fahrer (Inhalt)
- die Kollektion und ihre Mitglieder, z.B. die Firma (Kollektion) und die Angestellte (Mitglied)

|                |                   |
|----------------|-------------------|
| 1              | genau 1           |
| 0..1           | 0 bis 1           |
| *              | 0 bis viele       |
| 3..*           | 3 bis viele       |
| 0..2           | 0 bis 2           |
| 2              | genau 2           |
| 2, 4, 6        | 2, 4 oder 6       |
| 1..5, 8, 10..* | nicht 6, 7 oder 9 |

Abbildung 26 Multiplizitäten

Insbesondere kann ein Objekt gleichzeitig Teil von mehreren Ganzen sein. Z.B. kann die Angestellte der Firma auch Mitglied eines Sportvereins sein. Man bezeichnet solche Aggregationen auch als *shared aggregation* oder *weak ownership*.

**Komposition.** Die Komposition ist eine starke Form der Aggregation, also wieder eine



Ganzes-Teil-Beziehung. Nur ist hier jedes Objekt - zu einem Zeitpunkt - Teil von *höchstens einem* Ganzen (*unshared aggregation*). Wird das Ganze gelöscht, so werden auch seine Teile gelöscht: Die Teile leben und sterben mit dem Ganzen. Teile können aber nach dem Ganzen erzeugt werden und vor dem Ganzen sterben. Und schließlich: Wird das Ganze kopiert, werden alle lebenden Teile mit kopiert. Bei einer Komposition ist die Raute schwarz.

Zur Abgrenzung zwischen Aggregation und Komposition siehe *Abbildung 27*. Ferner sei noch auf die interessante Darstellung in Balzert (1999) [S. 46ff] verwiesen.

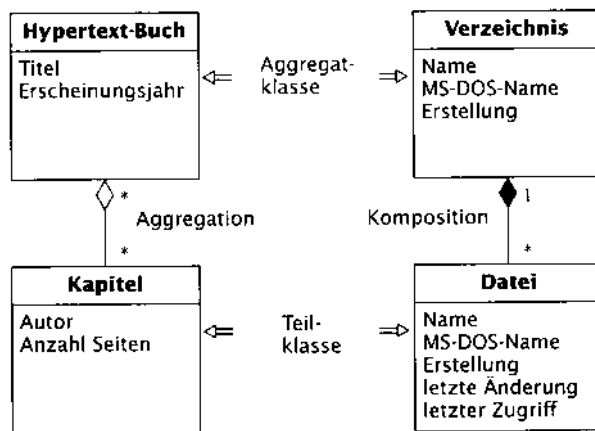


Abbildung 27 Aggregation <-> Komposition

#### 4.2.4.1 Erstellen eines Klassendiagramms

Das Hauptproblem beim Erstellen von Klassendiagrammen ist es, die „richtigen“ Klassen zu finden. Am besten geht man dieses Problem von zwei Seiten an und erstellt das Klassendiagramm in zwei Arbeitsgängen, Top-down und Bottom-up. Bei der Top-down-Analyse werden die Klassen vor allem auf Basis des allgemeinen Verständnisses des Fachgebiets gefunden. Es geht darum, ein Grundgerüst von Klassen zu finden, auf dem bei der anschließenden Bottom-up-Analyse aufgebaut werden kann. Bei der Bottom-up-Analyse werden die Klassen vor allem auf Basis der Ein- und Ausgaben des IT-Systems gefunden.

##### Top-Down

**1. Klassen identifizieren und modellieren.** Auf Basis von allgemeinem Fachwissen, Gesprächen mit Fachexperten und Dokumenten, wird eine Analyse der fachlichen Zusammenhänge, Geschäftsaktivitäten, Informationsbedürfnisse, Akteure und Prototypen durchgeführt. Die zu stellenden Fragen lauten:

- Welche statischen Geschäftsregeln gibt es?
- Welches sind die wichtigsten Dinge, mit denen das IT-System arbeiten wird?
- Welche Klassen lassen sich daraus bilden?

Die Antworten auf diese Fragen liefern eine Anzahl möglicher Klassen, die in einem ersten Entwurf des Klassendiagramms modelliert werden. So beinhaltet eine statische

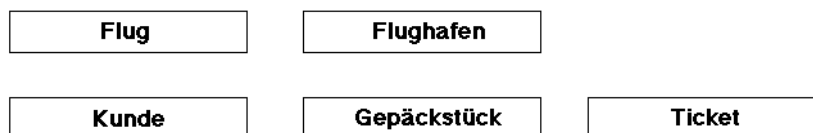


Abbildung 28 Mögliche Klassen

Geschäftsregel wie „*Ein Ticket gehört zu genau einem Kunden*“ die beiden Klassen **Kunde** und **Ticket**.

**2. Assoziationen identifizieren und modellieren.** Die Zusammenhänge der gefundenen Klassen sowie der statischen Geschäftsregeln werden im Klassendiagramm als Assoziationen mit aussagekräftigen Namen und Multiplizitäten modelliert. Die hier zu stellenden Fragen lauten:

- Welche statischen Geschäftsregeln gibt es?
- Welche fachlichen Beziehungen bestehen unter den Objekten?
- Wie viele Objekte einer Klasse sind an einer Beziehung beteiligt?

Die erste Frage kennen wir bereits aus dem ersten Arbeitsschritt der Klassenidentifikation. Hier leiten wir aus den Geschäftsregeln zusätzlich Assoziationen ab. Die Geschäftsregel „*Ein Ticket gehört zu genau einem Kunden*“ beispielsweise liefert eine Assoziation zwischen **Kunde** und **Ticket** mit der Multiplizität 1 : \*.

Die zweite Frage stellt man sich jeweils für Objekte von zwei verschiedenen Klassen, also beispielsweise für die Klassen **Flug** und **Kunde** aus unserem Fallbeispiel. Dabei ist wichtig zu erkennen, ob eine direkte fachliche Beziehung besteht oder ob eine Beziehung nur indirekt über andere Objekte existiert. In unserem Fallbeispiel stellt sich heraus, dass der **Kunde** ein **Ticket** besitzt, das wiederum aus **Coupons** besteht, welche für einen **Flug** gelten. Die zweite Frage zielt darauf ab, die Multiplizitäten der Beziehung zu ermitteln, also beispielsweise, wie viele **Tickets** ein **Kunde** haben kann und wie vielen **Kunden** ein **Ticket** gehört, siehe *Abbildung 29*. Obwohl zu Anfang mit

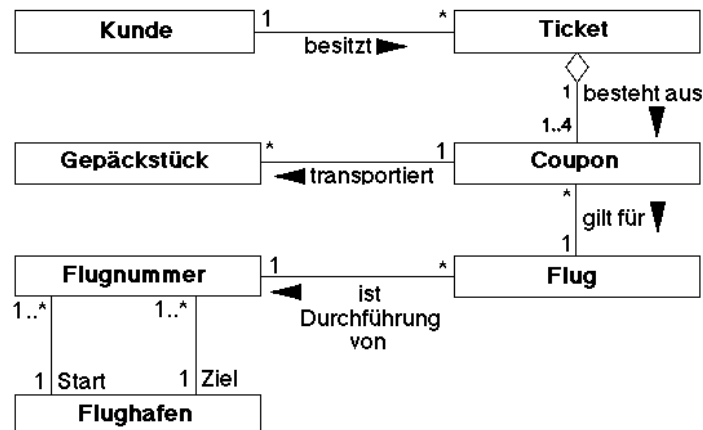


Abbildung 29 Klassen und Assoziationen

den bereits gefundenen Klassen begonnen wird, findet man in diesem Arbeitsschritt aufgrund der fachlichen Diskussionen in der Regel weitere Klassen.

**3. Attribute definieren.** Die Informationen, die über eine Klasse benötigt werden, müssen ermittelt und in Form von Attributen modelliert werden. Die Frage, die man sich stellen muss, lautet:

- Welche Informationen interessieren mich an einer bestimmten Klasse?

Bei dieser Frage geht es darum, offensichtlich benötigte Attribute der einzelnen Klassen zu finden (*Abbildung 30*). In der Regel wird die hier erstellte Liste von Attributen je

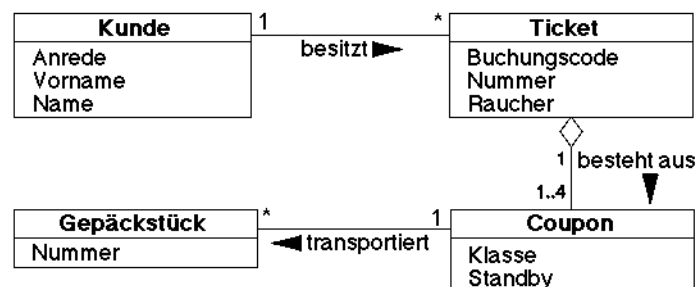


Abbildung 30 Klassen und Attribute

Klasse noch bei der detaillierteren Bottom-up-Analyse ergänzt, wenn es um die kon-

kreten Eingaben oder Abfragen des IT-Systems geht. Man sollte zu diesem Zeitpunkt also nicht zu viel Zeit in die Beantwortung der Frage investieren.

### Bottom-up-Analyse

**4. Geforderte Abfragen und Eingaben auflisten.** Zunächst stehen die Abfragen im Vordergrund, denn sie hängen mit dem eigentlichen Zweck des IT-Systems zusammen. Die Fragen, die man sich stellen muss, lauten:

- Welche Informationen muss das IT-System bereitstellen können?
- Welche Informationen muss das IT-System entgegen nehmen können?

Bei der Beantwortung dieser Fragen kann auf die bereits gefundenen Anwendungsfälle aufgebaut werden. In den Sequenzdiagrammen ist bereits skizziert, welche Abfragen und welche Nachrichten in einem Anwendungsfall auftreten. Eine weitere wichtige Informationsquelle sind die Geschäftsprozesse des Geschäftssystems. Das Ergebnis dieses Arbeitsschritts ist eine Liste von Informationsanforderungen:

| Anforderung    | Art     | Anwendungsfall             |
|----------------|---------|----------------------------|
| Boardingkarte  | Ausgabe | Boardingkarte erstellen    |
| Passagierliste | Ausgabe | Boarding abschließen       |
| Ticketdetails  | Ausgabe | Check-in, Express-Check-in |
| Kundendetails  | Ausgabe | Check-in, Express-Check-in |
| Coupon         | Eingabe | Check-in, Express-Check-in |
| ...            |         |                            |

**5. Abfragen und Eingaben formalisieren.** Damit für die einzelnen Abfragen und Eingaben Klassendiagramme erstellt werden können, muss zuerst festgelegt werden, wie sie aussehen. Komplexe Abfrageresultate oder Eingabe werden gesammelt oder musterhaft skizziert. *Abbildung 31* zeigt eine z.B. Passagierliste.

Die Fragestellung:

- Wie sieht die Darstellung der Abfrage oder Eingabe konkret aus?

Gute Informationsquellen sind bereits vorhandene Formulare und Bildschirmmasken aus dem Prototypen.

**6. Informationen analysieren.** Dieser Schritt ist die Hauptarbeit der Bottom-Up-Analyse. Welche Klassen, Assoziationen und Attribute werden benötigt? Für jede Abfrage oder Eingabe wird auf Basis der bereits vorhandenen Klassen ein eigenes kleines Klassendiagramm erstellt, indem die skizzierten Ein- und Ausgaben des IT-Systems modelliert werden („Klassenmodellierung im Kleinen“).

Die strategischen Fragen lauten:

- Welche Datenelemente sind auf der Ein- oder Ausgabe vorhanden?
- Welche Objekte verstecken sich hinter den Datenelementen?
- Welche Beziehungen bestehen zwischen den gefundenen Objekten?

| Passagierliste für Flug SR9011<br>vom 17.6.2000, Stand 1240 Uhr |           |          |
|---|-----------|----------|
| Ms  | Achermann | Irmgard  |
| Mr  | Adams     | Douglas  |
| Ms  | Baumann   | Hennette |
| Mr  | Baumann   | Philippe |
| Mr  | Born      | André    |
| Mr  | Cleese    | John     |
| Ms  | Dunnnett  | Dorothy  |
| Mr  | Flachmann | Ferdi    |
| Mr  | Grässle   | Caroline |
| Mr  | Grässle   | Kurt     |
| Mr  | Grässle   | Milla    |
| Mr  | Gubser    | Rolf     |
| Mr  | Hirsch    | Harry    |
| Mr  | Jarchow   | Thomas   |
| Mr  | Milligan  | Spike    |
| Mr  | Schacher  | Markus   |
| Mr  | Stahe!    | Reto     |

Seite 1

Abbildung 31 Passagierliste

- Welche bereits modellierten Objekte können verwendet werden?

Für die Passagierliste aus *Abbildung 31* lässt sich das Klassendiagramm in *Abbildung 32* erstellen. Mit Berücksichtigung der bereits gefundenen Klassen aus der Top-Down-

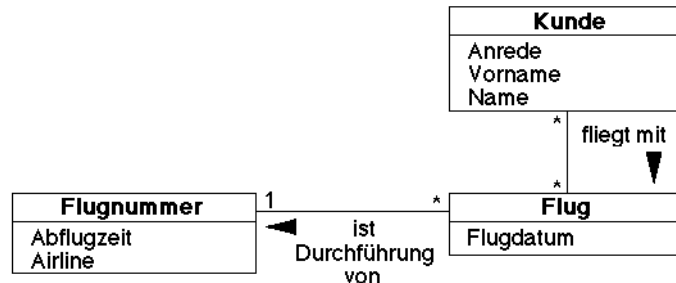


Abbildung 32 Klassendiagramm für die Passagierliste

Analyse entsteht das Klassendiagramm in *Abbildung 33*.

**7. Klassendiagramme konsolidieren.** Im letzten Arbeitsschritt müssen alle einzelnen Klassendiagramme, soweit dies nicht bereits geschehen ist, zu einem gesamten Klassendiagramm konsolidiert werden. Dabei müssen Unstimmigkeiten entdeckt und beseitigt werden. Geeignete Fragen sind:

- Gibt es in den einzelnen Klassendiagrammen Klassen, die unterschiedliche Namen haben, aber das gleiche abbilden?
- Gibt es in den einzelnen Klassendiagrammen mehrere Beziehungen, die die gleiche Bedeutung haben?
- Gibt es in den Klassen Attribute, die unterschiedlich heißen, aber die gleiche Bedeutung haben?

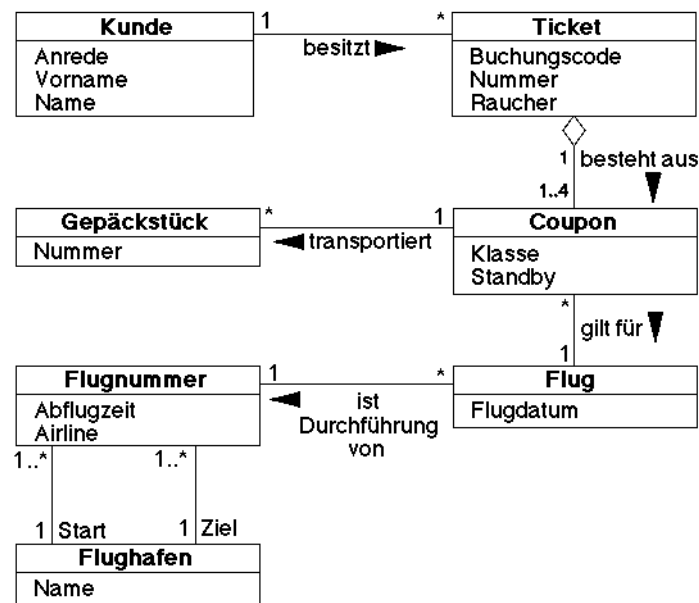


Abbildung 33 Überarbeitetes Klassendiagramm für die Passagierliste

Werden alle einzelnen Klassendiagramme zu einem gemeinsamen Diagramm vereinigt, so stellen sich diese Fragen eigentlich von selbst. Sind die Unstimmigkeiten einmal erkannt, lassen sie sich in der Regel leicht beheben.

#### 4.2.4.2 Lesen von Klassendiagrammen

Das Klassendiagramm in *Abbildung 33* besteht aus den Klassen **Kunde**, **Ticket**, **Gepäckstück**, **Coupon**, **Flug**, **Flugnummer** und **Flughafen**, ihren Attributen und ihren Assoziationen. Eine Assoziation liest man wie folgt:

- „Ein (der Satz beginnt immer mit „ein“!) Objekt der einen Klasse hat eine Assoziation zu einer Anzahl von Objekten der anderen Klasse.“

In diese allgemein gültige Formulierung muss man die entsprechenden Werte aus dem Diagramm einfügen. Z.B. liest man die Assoziation zwischen **Kunde** und **Ticket**: *Ein Kunde besitzt null, 1 oder mehrere Ticket(s)*. Assoziationen zwischen zwei Klassen kann man stets in beiden Richtungen lesen: *Ein Ticket ist im Besitz von genau einem Kunden*.

Auf diese Art und Weise lassen sich alle Assoziationen lesen.

#### 4.2.5 Zustandsdiagramme und die Verhaltenssicht

Anwendungsfälle und Szenarios bieten eine Möglichkeit, das Verhalten von Systemen, seien es Geschäftssysteme oder IT-Systeme, zu beschreiben. Mit Verhalten ist die Wechselwirkung und Interaktion zwischen Objekten des Systems gemeint. Oft ist es notwendig, das Verhalten im Innern eines Objekts zu betrachten. Dazu werden Zustandsdiagramme verwendet, die die verschiedenen Zustände eines Objekts zeigen sowie die Ereignisse oder Nachrichten, die einen Übergang von einem Zustand in den nächsten verursachen, und die Aktionen, die durch einen Zustandswechsel entstehen.

Ein Zustandsdiagramm wird im allgemeinen nicht für alle Klassen des Systems erstellt, sondern nur für diejenigen mit einem besonderen dynamischen Verhalten, also mit einem ereignisreichem Leben und regem Nachrichtenaustausch.

#### 4.2.5.1 Das Leben von Objekten und dynamische Geschäftsregeln

Auch die Objekte eines IT-Systems haben ein Leben. Ein Objekt wird irgendwann erzeugt, im Laufe der Zeit gelesen und verändert, und schließlich zerstört. Ein Objekt befindet sich im Laufe seines Lebens stets in einem wohldefinierten *Zustand*, eine Veränderung entspricht einer Zustandsänderung, ausgelöst durch bestimmte *Ereignisse*.

Die Veränderungen eines Objekts entsprechen gewissen Regeln, den dynamischen Geschäftsregeln, die nur zu bestimmten Zeitpunkten, d.h. bei Auftreten bestimmte Ereignisse, überprüfbar sind. Kurz gesagt: *Das Verhalten der Objekte, also ihre Zustandsänderung durch Ereignisse, wird wesentlich durch die dynamischen Geschäftsregeln bestimmt.* Beispiele dynamischer Geschäftsregeln sind:

- Ein Flugzeug darf keinem Flug zugeordnet werden, wenn es sich in der Wartung befindet
- Das Flugzeug darf nicht ausgemustert werden, solange es noch für Flüge eingeplant ist.

Aus Sicht der Objektorientierung ausgedrückt in Objekten, Zuständen und Ereignissen lauten sie wie folgt:

- Das Ereignis **Flugzeug einem Flug zuordnen** ist im Zustand **in Wartung** des Objekts **Flugzeug** nicht zulässig.
- Das Ereignis **Flugzeug ausmustern** ist im Zustand **für Flüge geplant** des Objekts **Flugzeug** nicht zulässig.

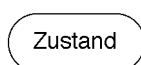
#### 4.2.5.2 Zustandsdiagramm (*statechart diagram*)

Das Verhalten eines Objekts wird mit einem Zustandsdiagramm dargestellt. Mit diesem Diagramm wird für die entsprechende Klasse, zu der das Objekt gehört, dokumentiert, welche dynamischen Geschäftsregeln eingehalten werden müssen. Anders gesagt: Welche Ereignisse sind in welchen Zuständen zulässig?

Die Verhaltenssicht besteht aus einer Menge von Zustandsdiagrammen, die jeweils das Verhalten eines einzelnen Objekts zeigen. Alle Zustandsdiagramme zusammen zeigen also das Verhalten aller Objekte des IT-Systems. Normalerweise werden jedoch nicht alle Zustandsdiagramme erstellt, sondern nur diejenigen, die wichtige Geschäftsregeln enthalten oder wichtige Objekte beschreiben.

Im Zustandsdiagramm arbeiten wir mit den folgenden Elementen:

1. **Zustand (*state*)**. Ein *Zustand* ist ein Umstand oder eine Situation im Leben



eines Objekts, während dem es einer Bedingung genügt, eine Aktivität ausführt oder auf ein Ereignis wartet. Der Zustand eines Objektes wird durch die Werte seiner Attribute und Assoziationen bestimmt. Ein Zustand ist also eine Menge

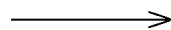
von Attributwerten, in denen sich das Objekt gleich verhält. Ein Objekt bleibt eine endliche Dauer in einem Zustand. Zum Beispiel kann ein Objekt **Heizung** eines Hauses in einem von vier Zuständen sein: **Leerlauf** (wartend auf einen Befehl zum Heizen), **Hochfahren** (Gas ist eingeschaltet, wartet bis Betriebstemperatur erreicht), **Aktiv** (Brenner und Gebläse sind eingeschaltet) und **Herunterfahren** (Brenner ausgeschaltet, Gebläse entfernt Resthitze).

**2. Startzustand.** Der Startzustand repräsentiert die Quelle aller Objekte des



Diagramms. Er ist kein normaler Zustand (sondern ein „Pseudozustand“), denn in diesem Zustand existieren die Objekte noch gar nicht.

**3. Übergang.** Ein Übergang repräsentiert den Wechsel von einem Zustand in



einen anderen.

**4. Endzustand.** Der Endzustand repräsentiert das Ende eines Objekts. Er ist -



wie der Anfangszustand - kein echter Zustand, denn in diesem Zustand existieren die Objekte nicht mehr.

**5. Wächterbedingung.** Eine *Wächterbedingung* ist eine Bedingung, die erfüllt

[Wächterbedingung]

sein muss, damit der zugehörige Übergang stattfinden („feuern“) kann. Sie wird als Boolescher Ausdruck in eckigen Klammern geschrieben.

**6. Ereignis und Aktion.** Ein *Ereignis* ist ein (relevantes) Vorkommnis zu einem

Ereignis / Aktion

bestimmten Zeitpunkt an einem bestimmten Ort, ein Stimulus, der eine Zustandsänderung auslösen kann. Eine *Aktion* oder *Methode* ist eine ausführbare und atomare (= durch kein Ereignis unterbrechbare) Verarbeitung,<sup>3</sup> die eine Zustandsänderung bewirkt oder einen Wert zurückgibt. Die Aktion (einschließlich dem /-Zeichen) kann weggelassen werden, wenn sie sinngemäß nur aussagt: „Zustand wechseln“ (*implizite* Aktion), oder wenn sie noch nicht feststeht. Folgen mehrere Aktionen, so werden sie durch Semikolons getrennt. Aktionen können in der UML ohne weiteres informell in normalem Deutsch beschrieben werden. Für häufig verwendete Aktionen werden jedoch folgende Schlüsselwörter genutzt.

- create - destroy: erzeugt oder löscht ein Objekt (kann auch weggelassen werden, da es implizit ist)

**7. Zustandsdetails oder „Automatenmerkmale“.** Details der Zustände sind Aktionen oder Aktivitäten, die bei Eintreten oder Beendigung des Zustands auftreten oder während des Zustands geschehen. Ebenso zählen so genannte „interne Zustandsübergänge“ dazu, also Ereignisse, die nicht zu einem Zustands-

---

3. Nicht-atomare Verarbeitungen, d.h. solche, die durch Ereignisse unterbrochen werden, sind Aktivitäten!

wechsel führen. Sie alle werden im unteren Teil des Zustandssymbols aufgelistet.

**Beispiel.** Das Objekt **Flugzeug** hat im Laufe seines Lebens drei Zustände. Das Dia-

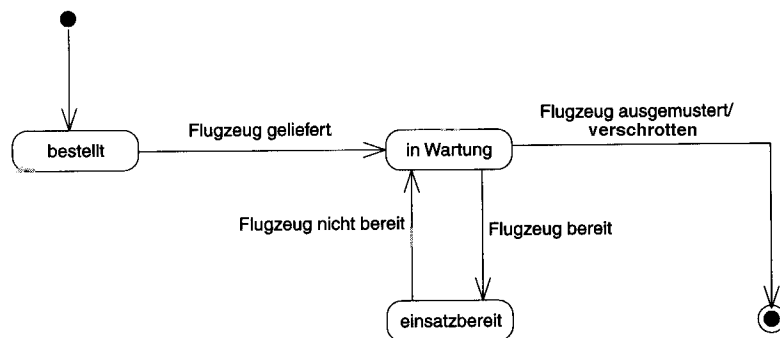


Abbildung 34 Zustandsdiagramm mit Ereignissen

gramm in *Abbildung 34* zeigt sie, als auch die möglichen Übergänge zwischen ihnen und die Ereignisse, die diese Übergänge auslösen.

#### 4.2.5.3 Das Erstellen eines Zustandsdiagramms

Die folgende Checkliste zeigt die Schritte, die für das Erstellen eines Zustandsdiagramms einer Klasse erforderlich sind.

**(i) Identifikation der für das Objekt relevanten Ereignisse.** Von was ist das Objekt betroffen? Zuerst muss herausgefunden werden, welche Ereignisse für ein Objekt relevant sind, d.h. welche Ereignisse eine Aktion, einen Zustandsübergang oder beides auslösen. Mit den folgenden Fragen können relevante Ereignisse gefunden werden:

- Welche Ereignisse führen dazu, dass ein Objekt erzeugt oder zerstört wird?
- Welche Ereignisse definieren oder ändern Attributwerte?
- Welche Ereignisse stellen Beziehungen zu anderen Objekten her oder lösen die Beziehungen?
- Welche Ereignisse bewirken eine Zustandsänderung des Objekts?
- Welche Ereignisse aus dem Sequenzdiagramm der Sicht von außen (s. Geschäftsprozessmodellierung) betreffen das Objekt?

Die Beantwortung dieser Fragen führt zu einer Liste von Ereignissen, die für das Objekt relevant sind. Da alle Ereignisse ursprünglich aus einem Anwendungsfall stammen, muss für neue Ereignisse, die nicht in Sequenzdiagrammen enthalten sind, ein Anwendungsfall gefunden werden. Ein Ereignis, das nicht im Rahmen eines Anwendungsfalls an das IT-System geschickt wird, wird *nie* an das IT-System geschickt! Gegebenenfalls müssen neue Anwendungsfälle modelliert werden.

In unserem Fallbeispiel findet man für das Objekt **:Flug** beispielsweise folgende relevante Ereignisse:

- Flug definiert
- Flug gestartet
- Flug gelandet
- Flug annulliert
- neues Flugdatum
- Flug irrelevant
- Flugnummer irrelevant

**(ii) Zeitliche Gruppierung der Ereignisse.** Die gefundenen Ereignisse werden in drei Gruppen gegliedert: objekterzeugende Ereignisse, Ereignisse während des Lebens eines Objekts und objektzerstörende Ereignisse. In unserem Beispiel sind das:

| erzeugendes Ereignis<br>(Geburt) | Ereignis während des<br>Lebens eines Objekts                          | zerstörende Ereignisse<br>(Tod)          |
|----------------------------------|---|--|
| Flug definiert                   | Flug gestartet<br>Flug gelandet<br>Flug annulliert<br>neues Flugdatum | Flug irrelevant<br>Flugnummer irrelevant |

**(iii) Identifizierung der Zustände.** Die nicht immer leichteste (aber wichtigste!) Aufgabe bei der Erstellung des Zustandsdiagramms ist die Bestimmung der Zustände des Objekts. Man kann zunächst anhand der im ersten Schritt bestimmten Ereignisse versuchen, die entsprechenden Zustände, indem man folgende Hilfsfragen stellt:

- Ist ein gegebenes Ereignis gemäß den dynamischen Geschäftsregeln für alle Fälle im Leben des Objektes zulässig? Die verschiedenen Fälle entsprechen genau den verschiedenen Zuständen des Objekts.
- In welchem Zustand ist das Objekt, nachdem es erzeugt wurde?
- In welchen Zuständen kann sich das Objekt befinden, bevor es zerstört wird?
- In welchem Zustand ist das Objekt jeweils vor und nach Eintreffen eines der Ereignisse?
- Ist der Wechsel eines Zustand in einen anderen abhängig von Bedingungen? So erkennt man die entsprechenden Wächterbedingungen.

In unserem Fallbeispiel ist das Ereignis *Flug gestartet* nur zulässig, wenn der Flug nicht bereits im Zustand **gestartet** ist. Beantwortet man die Hilfsfragen für alle Ereignisse, so entsteht das Zustandsdiagramm aus *Abbildung 35*.

**(iv) Das Zustandsdiagramm um Aktionen ergänzen.** Nachdem die Ereignisse eines Objekts gefunden und modelliert sind, werden deren Folgen in Form von Aktionen spezifiziert. Folgende Hilfsfragen sind zu beantworten:

- Wo werden Aktionen zur Behandlung der Attributwerte benötigt?
- Wo werden sonstige Aktionen (Abfragen, Berechnungen) benötigt?

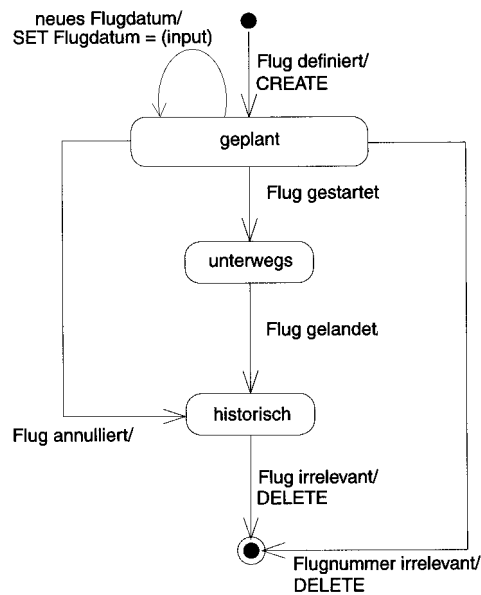


Abbildung 35 Zustandsdiagramm der Klasse **Flug**

So werden Aktionen gefunden, die die Zustandsübergänge spezifizieren oder zu Zustandsdetails führen.

#### 4.2.5.4 Verifizieren des Zustandsdiagramms

Ein fertiggestelltes Zustandsdiagramm kann anhand der folgenden Checkliste geprüft werden:

- o Ist ein Endzustand modelliert? Oder wird ein Objekt tatsächlich nie zerstört („zirkuläres Leben“)?
- o Falls mehrere Übergangspfeile einen Zustand verlassen und durch dasselbe Ereignis ausgelöst werden, so dürfen ihre Wächterbedingungen nicht gleichzeitig wahr sein können, sondern müssen sich gegenseitig ausschließen.

### 4.3 Die Ablaufsicht - Kollaborations- und Sequenzdiagramme

Das bloße Aufzeigen von Strukturen reicht meist nicht aus, um ein System zu verstehen. Um mit einem Bild von Grässle *et al.* (2000) [S.181] zu sprechen: Selbst mit einem genauen Plan einer Ö raffinerie ist es nicht oder nur schwer zu verstehen, wie aus Öl Benzin erzeugt wird. Erst wenn man den Ablauf des Raffinierens kennt, wird das ganze verständlich.

Gleiches gilt natürlich für Modelle in der UML. Die statische Sicht der Klassen ist zwar ein sehr wichtiger Schritt, um ein System zu verstehen und zu modellieren. Aber erst die Betrachtung der Abläufe macht das System verständlich.

Zudem wird durch die Modellierung der Abläufe die statische Sicht noch einmal überprüft und gegebenenfalls vervollständigt.

Während Anwendungsfälle stets die Sicht von außen auf ein System zeigen und es als *Black-box* betrachten, so wird durch die Betrachtung der Abläufe diese *Black-box* geöffnet und das System von innen gezeigt.

Für die Modellierung der Abläufe sieht die UML zwei Diagramme vor, das Kollaborationsdiagramm und das Sequenzdiagramm. Beide Diagramme werden als *Interaktionsdiagramme* bezeichnet.

In beiden Diagrammen wird gezeigt, wie die einzelnen Objekte zusammenarbeiten.

#### 4.3.1 Kollaborationsdiagramm (*collaboration diagram*)

Das Kollaborationsdiagramm beschreibt die strukturelle Organisation der Objekte, die Nachrichten senden und empfangen. Ein Kollaborationsdiagramm besteht aus den folgenden Elementen:

**1. Akteur.** Ein Akteur des Kollaborationsdiagramms repräsentiert einen beliebigen



aus dem Anwendungsfalldiagramm. Falls ein bestimmtes Ereignis bzw. eine bestimmte Nachricht in mehreren Anwendungsfällen enthalten ist, die verschiedene Akteure haben, so verwenden wir den Akteur „Irgendwer“, um uns nicht auf einen richtigen Akteur festlegen zu müssen.<sup>4</sup>

**2. Objekt und Eintrittsobjekt.** Ein Objekt repräsentiert ein Objekt einer Klasse

**Objekt:Klasse**

aus der statischen Sicht, z.B. Hans Meier, der ein Objekt der Klasse Passagier ist. Das Eintrittsobjekt ist das erste Objekt, das eine Nachricht von einem Akteur erhält. Beim Eintrittsobjekt beginnt der Interaktionspfad.

**3. Assoziation.** Entsprechend den Klassendiagrammen existieren zwischen

den Objekten des Kollaborationsdiagramms Assoziationen. Sie zeigen Verbindungen zwischen Objekten an.

**4. Nachrichten.** Die Nachrichten sind im Regelfall Abfragen an das IT-System



aus einem Anwendungsfall. In den Klammern steht der Parameter, der der Nachricht mitgegeben wird, z.B. die Nummer des Tickets, damit das richtige Ticket gelesen werden kann. Wird kein Parameter übergeben, so werden oft leere Klammern geschrieben, aber auch diese können in UML weggelassen werden.

---

<sup>4</sup>in Anlehnung an Grässle *et al.* (2000)

### 4.3.2 Erstellen eines Kollaborationsdiagramms

In unserem Modell des IT-Systems sind die Anwendungsfälle die Quelle der Ereignisse. Was in einem Kollaborationsdiagramm dokumentiert ist, findet im Kontext eines Anwendungsfalldiagramms statt. Manchmal kann der im Kollaborationsdiagramm beschriebene Ablauf in mehreren Anwendungsfalldiagrammen vorkommen. Dann kann der Akteur „Irgendwer“ stellvertretend für die jeweiligen Akteure der Anwendungsfälle stehen, durch die die Nachrichten (i.d.R. als Abfragen) kommen.

In *Abbildung 36* ist exemplarisch ein Kollaborationsdiagramm dargestellt. Es doku-

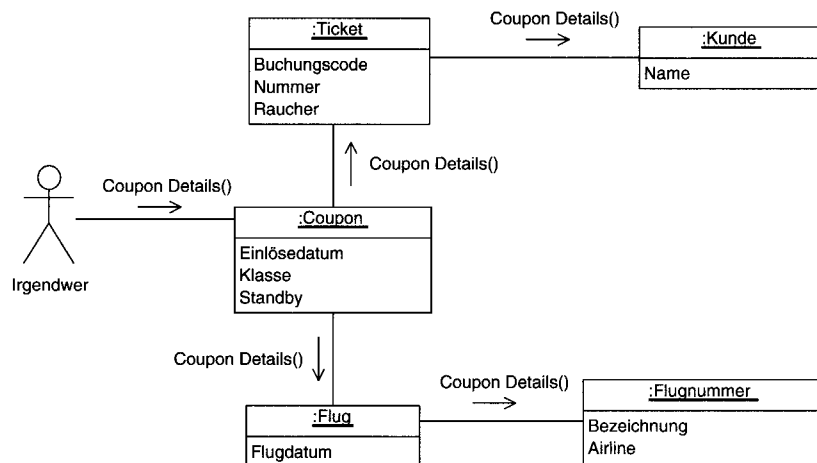


Abbildung 36 Kollaborationsdiagramm

mentiert den Ablauf der Abfrage (Nachricht) **Coupon Details**.

Im Gegensatz zum Sequenzdiagramm gibt es im Kollaborationsdiagramm keine Zeitdimension. Die Objekte können beliebig über das Diagramm verteilt werden. Daher kann eine Reihenfolge, in der die Nachrichten verarbeitet werden, nur bedingt dargestellt werden. Der Akteur hilft dabei, den Ausgangspunkt der Nachrichten zu erkennen.

Die folgenden Schritte helfen, ein Kollaborationsdiagramm zu erstellen.

**1. Abfrageresultat skizzieren.** Ausgangspunkt für die Modellierung von Nachrichten ist das erwartete Resultat. In der Regel handelt es sich um eine Anzeige am Bildschirm oder um ein gedrucktes Dokument, z.B. eine Liste oder einen Beleg. *Abbildung 37* zeigt z.B. ein Fenster aus dem Prototypen des IT-Systems.

## 4.4 Zusammenfassung der Modellierung eines IT-Systems

Wir haben in diesem Abschnitt gesehen, wie mit Hilfe der UML ein konzeptionelles Modell eines IT-Systems erstellt werden kann. Dabei sollte darauf geachtet werden, nicht alles in voller Tiefe mit der UML zu modellieren, da in der Realisierungsphase meist Erkenntnisse gewonnen werden, die in der Entwurfsphase nicht absehbar waren. Zudem sollte das Modell mit möglichst geringem Aufwand erstellt werden.

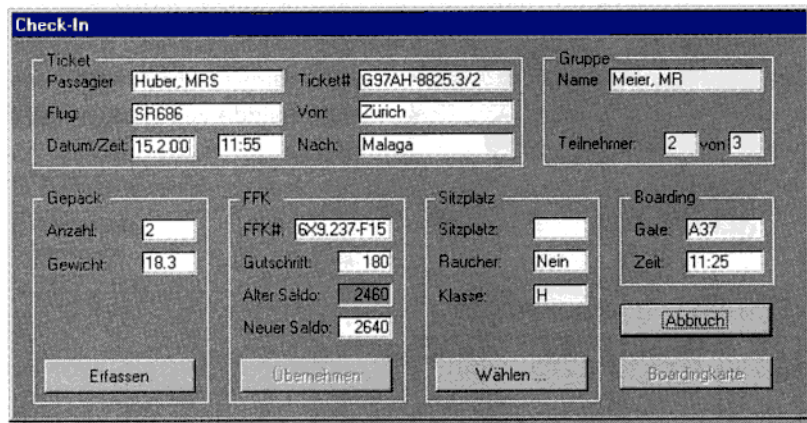


Abbildung 37 Bildschirmansicht aus dem Passagierabfertigungssystem

Das Modell des IT-Systems besteht aus vier Sichten, die jeweils bestimmte Aspekte hervorheben und eng miteinander verknüpft sind. Die einzelnen Sichten und die darin verwendeten UML-Diagramme sind in folgender Tabelle dargestellt:

| Sicht              | Aspekt   | verwendete UML-Diagramme                   |
|--------------------|--|--|
| Sicht von außen    | Funktionalitäten des IT-Systems für die Anwender                         | Anwendungsfalldiagramm und Sequenzdiagramm |
| strukturelle Sicht | Strukturen, in denen die Informationen abgelegt sind (statischer Aspekt) | Klassendiagramm                            |
| Verhaltenssicht    | Verhalten der Objekte  | Zustandsdiagramm                           |
| Ablaufsicht        | Abläufe innerhalb des IT-Systems   | Sequenzdiagramm und Kollaborationsdiagramm |

Die Ereignisse und Nachrichten sind das Bindeglied, das die verschiedenen Sichten zusammenhält. Sie sind in allen Sichten enthalten bis auf die strukturelle Sicht, die nur statische Aspekte zeigt:

- In der Sicht von außen werden die Anwendungsfälle als Abfolge von Ereignissen und Nachrichten beschrieben, die an das IT-System geschickt werden.
- In der Verhaltenssicht wird für jede Klasse gezeigt, wie ihre Objekte auf Ereignisse und Nachrichten reagieren, die bei ihnen eintreffen.
- In der Ablaufsicht wird gezeigt, wie die einzelnen Ereignisse an die entsprechenden Objekte weitergegeben werden.

Wir haben zur Modellierung des IT-Systems längst nicht alle Diagramme der UML verwendet. In der Praxis hat sich die beschriebene Diagrammkombination für die Modellierung von IT-Systemen bewährt. Für andere Modelle sieht die Zusammenstellung der Diagramme anders aus. Wir haben im vorigen Abschnitt gesehen, dass sich im Modell des Geschäftssystems beispielsweise der Einsatz von Aktivitätsdiagrammen durchaus anbot, den wir hier vermieden haben.

## A Anhang

### A.1 Software zu UML

#### A.1.1 ArgoUML

ArgoUML basiert auf Java 2 und ist frei herunterladbar aus dem Web unter

<http://www.argouml.org>.

#### A.1.2 Rational Rose

Kommerzielles CASE-Tool, (*CASE = Computer aided software engineering*)

URL: <http://www.rational.com>

Ermöglicht Transformation der UML-Modelle in Java und C++; umgekehrt können aus Programmen UML-Modelle zurückgewonnen werden (*round-trip engineering*)

#### A.1.3 TogetherSoft

Kommerzielles CASE-Tool von Peter Coad, einem der Pioniere des Objektorientierten Software-Engineering

URL: <http://www.togethersoft.com>

Ermöglicht Transformation der UML-Modelle in Java und C++; umgekehrt können aus Programmen UML-Modelle zurückgewonnen werden (*round-trip engineering*)

## Literatur

- Heide Balzert (1999): *Lehrbuch der Objektmodellierung. Analyse und Entwurf*. Spektrum Akademischer Verlag , Heidelberg und Berlin
- Heide Balzert (2001): *UML kompakt*. Spektrum Akademischer Verlag , Heidelberg und Berlin
- Grady Booch, Jim Rumbaugh und Ivar Jacobson (1998): *Das UML Benutzerhandbuch*. Addison Wesley Longman Bonn
- Rainer Burkhardt (1997): *UML - Unified Modeling Language: Objektorientierte Modellierung für die Praxis*. Addison Wesley Longman, Bonn
- Peter Coad und Mark Mayfield (1999): *Design mit Java. Bessere Applets und Anwendungen. 2. Auflage*. Prentice Hall, München
- Peter Forbig (2001): *Objektorientierte Softwareentwicklung mit UML*. Fachbuchverlag Leipzig im Carl Hanser Verlag, München Wien
- Patrick Grässle, Henriette Baumann und Philippe Baumann (2000): *UML projektorientiert. Geschäftsprozessmodellierung, IT-System-Spezifikation und Systemintegration mit der UML*. Galileo Press GmbH, Bonn
- IvarJacobson, Grady Booch und James Rumbaugh (1999): *The Unified Software Development Process*. Addison Wesley, Reading
- Ari Jaaksi, Juha-Markus Aalto, Ari Aalto und Kimmo Vättö (1999): *Tries and True Object development. Industry-Proven Approaches with UML*. Cambridge University Press, Cambridge
- Philippe Kruchten (1999): *Der Rational Unified Process: Eine Einführung*. Addison Wesley Longman, München
- Object Management Group (OMG): *UML Specification v1.3*. <http://www.omg.org>.

# Index

## A

Aggregation 42  
aggregation, shared 43  
Aktion 50  
Aktivitäten 7  
Aktivitätsdiagramm 31  
Aktivitätsdiagramme 15  
Anwendungsfall 35, 54  
Anwendungsfalldiagramme 10  
Assoziation 28  
Attribut 28

## B

Benutzeroberfläche 34  
Blackbox 34

## E

Ereignis 42, 49  
Ereignis, Definition 50

## G

Gebrauchsanweisung 34  
Generalisierung 28  
Geschäftsobjekte 7  
Geschäftsprozess 7  
Geschäftsregeln 42  
Geschäftsregeln, dynamische 49  
Geschäftssystem 7

## K

Klasse 27, 39

Klassendiagramm 26, 42  
Kollaborationsdiagramm 54  
Komposition 43

## M

Methode 28, 50  
Mitarbeitern 7

## N

Nachricht 22

## O

Objekt 38, 49  
Organisationseinheit 25  
Organisationsstruktur 7

## P

Paketdiagramme 25

## S

Sequenzdiagramm 20  
swim lanes 17  
Szenarien 8  
Szenario 11

## U

Unternehmen 7

## V

Vererbung 40  
Verhalten 48  
Verhaltenssicht 49

## Z

Zustand 49