

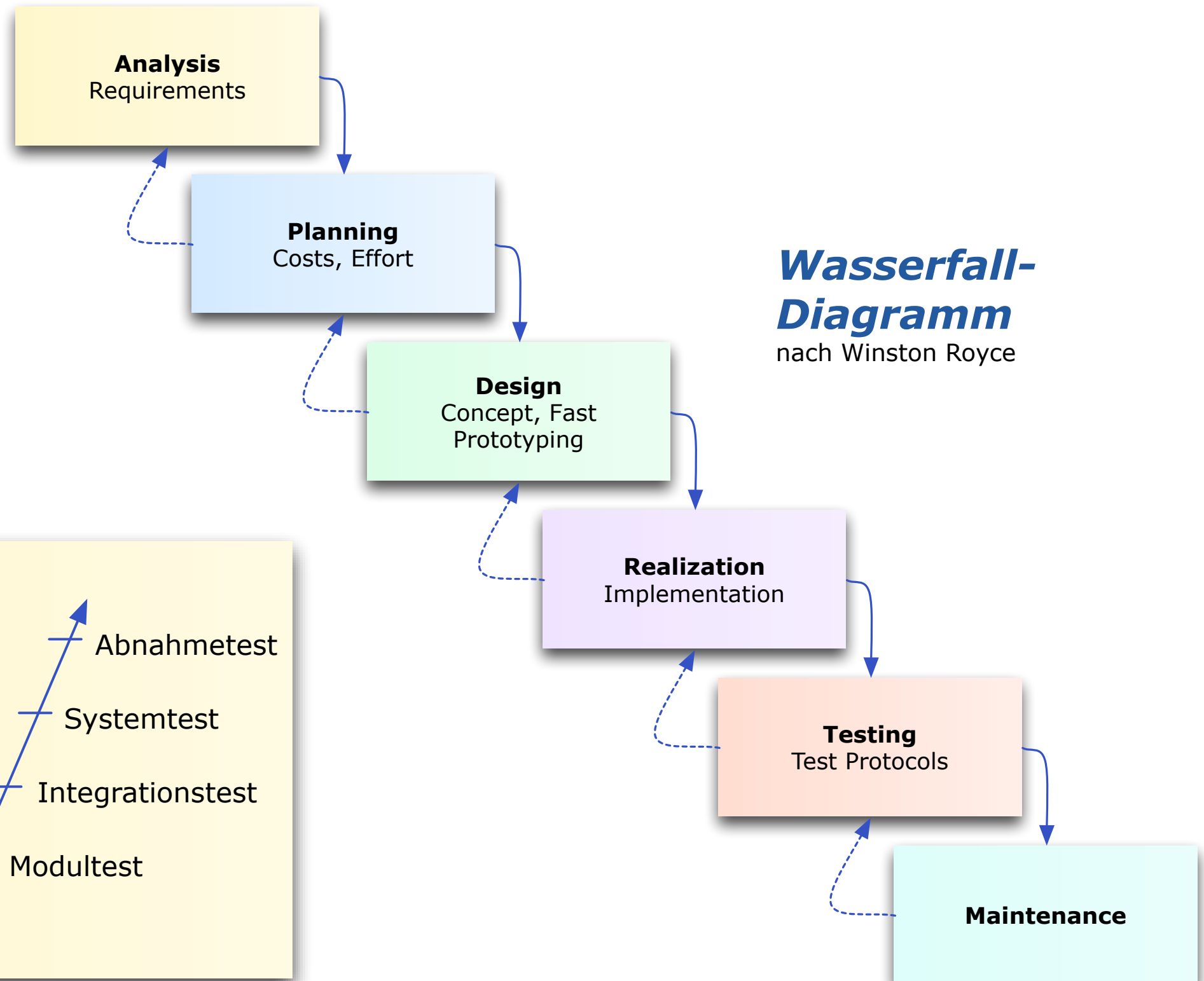
Schülerdossier Maturaarbeit

Projektphasen

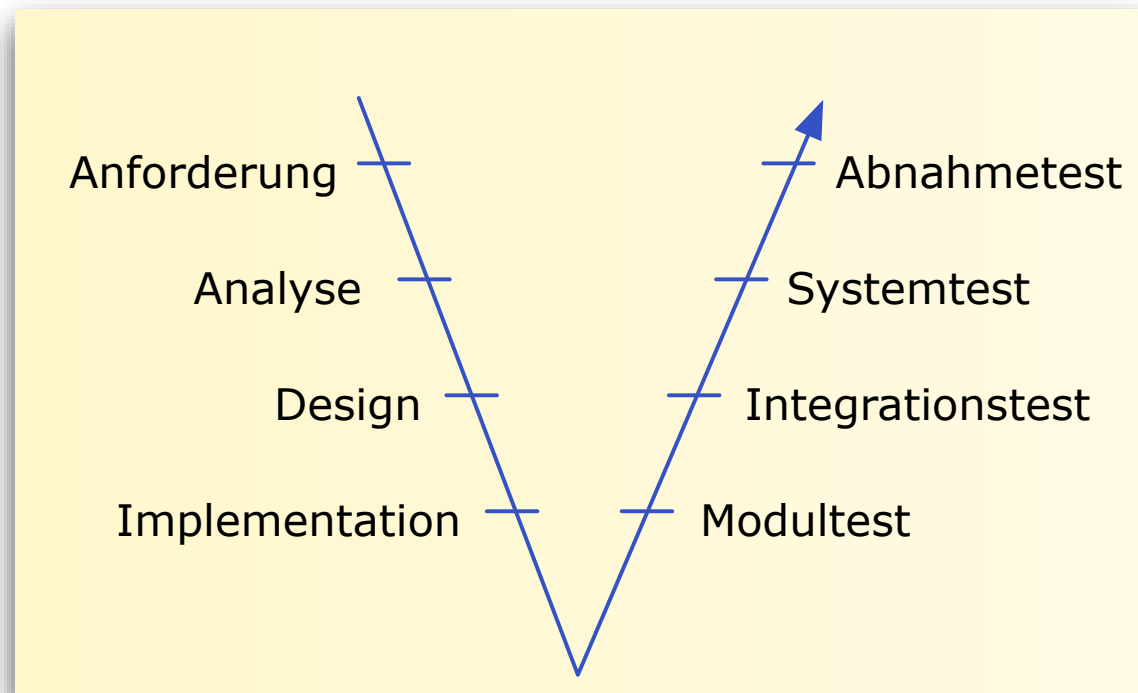
Für jedes Projekt müssen folgende Projektschritte behandelt und dokumentiert werden:

Analyse	Informationsbeschaffung, Interviews, Fragebogen, Zielgruppen, Use-Cases, Use-Case Diagramm, Entity Relationship Model (ERM), HW-, SW-Umgebung, Evaluationen (Verschiedene Varianten sollen aufgezeigt und bewertet werden).
Planung	Meilensteine, Projektrollenverteilung, Projektabschätzung, Aufwandabschätzung, (evtl. Gantt-Chart), Liste der Mindestziele (Must-Haves), Liste der zusätzlichen Ziele (Nice-To-Haves).
Design	GUI-Design (Layouts, Mockups, Fast-Prototypes), Datenbankschema, Klassendiagramm, Zustandsdiagramm, Sequenzdiagramm, Tests (Listen mit funktionalen und destruktiven Tests)
Realisation	Implementation des Programmes inkl. Helps Implementation der Datenbank Evtl. Einrichten des Servers Erstellen der Distribution
Test	Testen gemäss Testliste, Modultest, Integrationstest, Deployment-Test
Deployment	Installationsanleitung, Benutzeranleitung, Deployment-Scripts, Build-Pipes, Bulk-Dump

Projektphasen



V-Diagramm



Analyse

Bei der Analyse geht es darum, die Aufgabenstellung zu erfassen. Es kommt immer wieder vor, dass das Projekt nicht klar beschrieben ist oder ein Auftraggeber nicht genau weiss, was er will. Während der Analyse versuchen wir deshalb, die Anforderungen möglichst genau herauszuarbeiten und schriftlich festzuhalten.

Dabei konzentrieren wir uns auf die Frage nach dem "**Was**". **Was** soll das Programm können? **Was** soll dargestellt, gespeichert und ausgegeben werden? **Was** wird verlangt?

Weiterhin versuchen wir möglichst viel Informationen (Wikipediaeinträge, Code-Stücke, Tutorials, Videos usw.) zum Thema zusammenzutragen. Wir schreiben auch kleine Testprogramme um Codesegmente auszuprobieren, bei denen wir unsicher sind. In der späteren Coding-Phase sollten wir das Programm nur noch reinhacken können. Wenn wir dann noch recherchieren müssen, haben wir die Analyse schlecht gemacht oder zumindest schlecht dokumentiert.

Wir erstellen auch Fragenkataloge, führen Interviews mit Wissensträgern, machen Umfragen und bewerten Varianten.

Dann erstellen wir die Use-Cases und das Entity Relationship Model (ERM) und definieren eine Zielgruppe.

Als Ergebnis der Analyse erhalten wir das so genannte Anforderungsdokument. Dieses enthält eine klare Beschreibung der zu realisierenden Ziele, die Abgrenzung, das Use-Case Diagramm und das ERM.

Planung

Bevor wir mit dem eigentlichen Design beginnen können, planen wir das Projekt. Zunächst interessiert uns der Zeitaufwand. Diesen ermitteln wir, indem wir das Projekt in kleine Stücke, unterteilen und den Aufwand für jeden Projektschritt einzeln schätzen. Das Verteilen dieser Teilaufwände auf den Kalender ergibt dann die Zeitabschätzung.

Jeder Projektschritt schliesst mit einem Meilenstein ab. So ist z.B. "31.10.21 Analyse fertiggestellt" ein Meilenstein.

Auch hierhin gehört die Projektrollenverteilung, in der "Wer was macht" beschrieben wird.

Die Meilensteine, Aufwand- und Zeitabschätzungen sowie die Rollen können auch in einem Gantt-Diagramm (siehe S. 9) festgehalten werden.

Design

Beim Design (Programmwurf) geht es darum, wie die gestellte Aufgabe umgesetzt werden kann. Dabei fragen wir nach dem **"Wie"**. **Wie** soll die Applikation aussehen? **Wie** sollen die Daten strukturiert, angezeigt, gespeichert und ausgegeben werden.

Hier helfen uns Techniken wie Fast-Prototyping, das Klassen-, das Nassi-Shneiderman- und das Sequenzdiagramm.

Das Design sollte derart gut beschrieben werden, dass ein Programmierer genügend Informationen und Vorgaben hat, um das Programm anschliessend ohne Probleme realisieren zu können. Idealerweise erstellt der Programmierer das Design selber und lässt es von einem Kollegen reviewen (durchdenken, prüfen).

Ebenfalls zum Design gehört die Testvorbereitung. Dabei wird definiert, was getestet werden muss. Auf diese Weise wird die Qualität und die korrekte Funktionsweise des Programms von Beginn weg sichergestellt.

In einem Testdokument werden dazu in einer Liste die verschiedensten funktionalen und destruktiven Testfälle zusammen mit den erwarteten Ergebnissen zusammengestellt. Die Fälle sollten möglichst genau beschrieben werden: "1.2, 15, 'Hallo' eingeben" und nicht einfach "Einen Wert eingeben".

Coding (Implementation)

In dieser Phase wird der Programmcode geschrieben und gegebenenfalls mittels Compiler und Linker in ein ausführbares Programm konvertiert.

Programmieren ist ein Prozess, bei dem vor allem die Kenntnisse der Programmiersprache und ihrer Anwendung im Vordergrund stehen. Eine Entwicklungsumgebung und ein Versionierungs-Programm beschleunigen die Implementation und schaffen Sicherheit.

Hier sollten Kriterien wie Code for Robustness, Simplicity, Effectiveness und Reliability im Vordergrund stehen. Das Programm sollte bei Fehleingaben nicht abstürzen, einfach, schnell und zuverlässig sein.

Einfachheit und Übersichtlichkeit sind zwei besonders wichtige Gebote. Einfacher, gut strukturierter und übersichtlicher Code reduziert die Fehler und steigert somit die Qualität. Zudem sollte jedes Source-Code-File (ausgenommen Web) mit meinem Header versehen sein, welcher den Autor, eine kurze Beschreibung und die Geschichte festhält.

Am Schluss dieser Phase liegt das komplette Programm vor.

Test

Das Austesten des fertigen Programmcodes zeigt, ob alle geforderten Eigenschaften und Funktionen realisiert sind und ob sich das Programm tatsächlich so verhält, wie es erwartet wird.

Der Test ist eine der wichtigsten Phasen im Rahmen der Softwareentwicklung und kann sehr aufwändig sein. Als Basis für die Tests dient das Testdokument, welches während der Testvorbereitung in der Design-Phase erstellt wurde.

Nun werden alle festgehaltenen funktionalen und destruktiven Testfälle der Reihe nach durchgespielt und genau dokumentiert, welche Tests erfolgreich waren und welche nicht. Typischerweise wird dazu ein so genanntes Issue- oder Bug-Tracking-Tool verwendet, kann aber auch mit einer einfachen Tabelle geschehen.

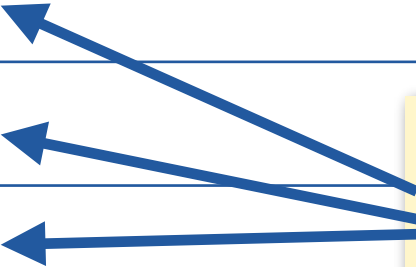
In einem Zyklus wird nun solange getestet und Probleme behoben, bis alle Testfälle erfolgreich waren.

Deployment

In dieser Phase erstellen wir eine Installationsanleitung und eine Benutzeranleitung. Die Installation sollte mithilfe eines von Ihnen erstellten Installers oder eines Installations-Scripts ganz einfach vonstatten gehen.

So nun ist das Projekt fertiggestellt!

Projektfortschrittsbericht

Projekt:		Arbeitspaket:	
Mitarbeiter:	Kalenderwoche:	Datum:	
Projektkurzbeschreibung:			
Projektfortschritt (in Prozent):			
Arbeiten, Ziele des Projektschrittes:			
Projektstand	Status	Begründung (nur bei gelb und rot)	
Termine	grün		
Kosten	gelb		
Produkt	rot		
Störungen/Probleme/Risiken:			
Notwendige Entscheidungen:			
Anlagen:			

Jedes dieser Felder
kann grün, gelb oder
rot sein!

Nützliche Programme / Werkzeuge

Gantt-Charts

- ProjectLibre (OpenProject)
- Open Workbench
- MS Project
- OmniProject
- GanttProject (<http://ganttproject.biz>, Java basierend)
- Binfire

Revision Control & Collaboration-Tools

- GitHub
- Binfire
- GitLab
- Subversion
- MS Team Foundation Server
- Dropbox
- OneDrive (Sharepoint)
- iCloud

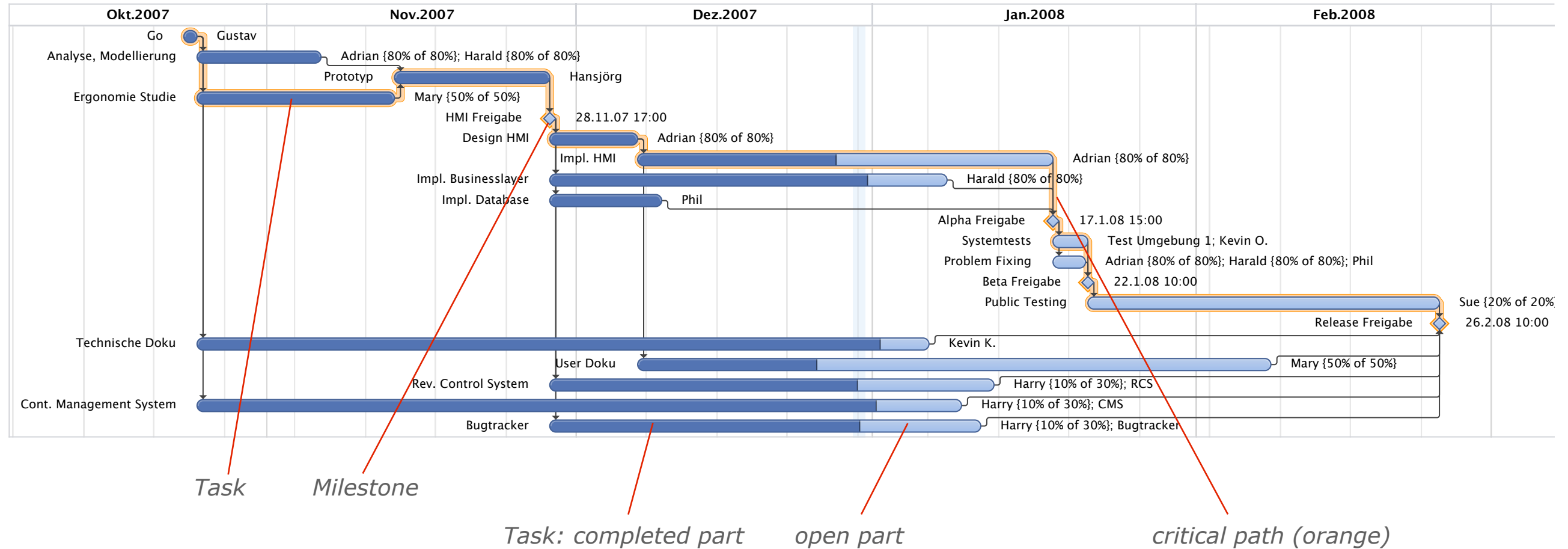
Fast-Prototyping

- Balsamiq Mockups www.balsamiq.com
- PowerPoint
- Keynote

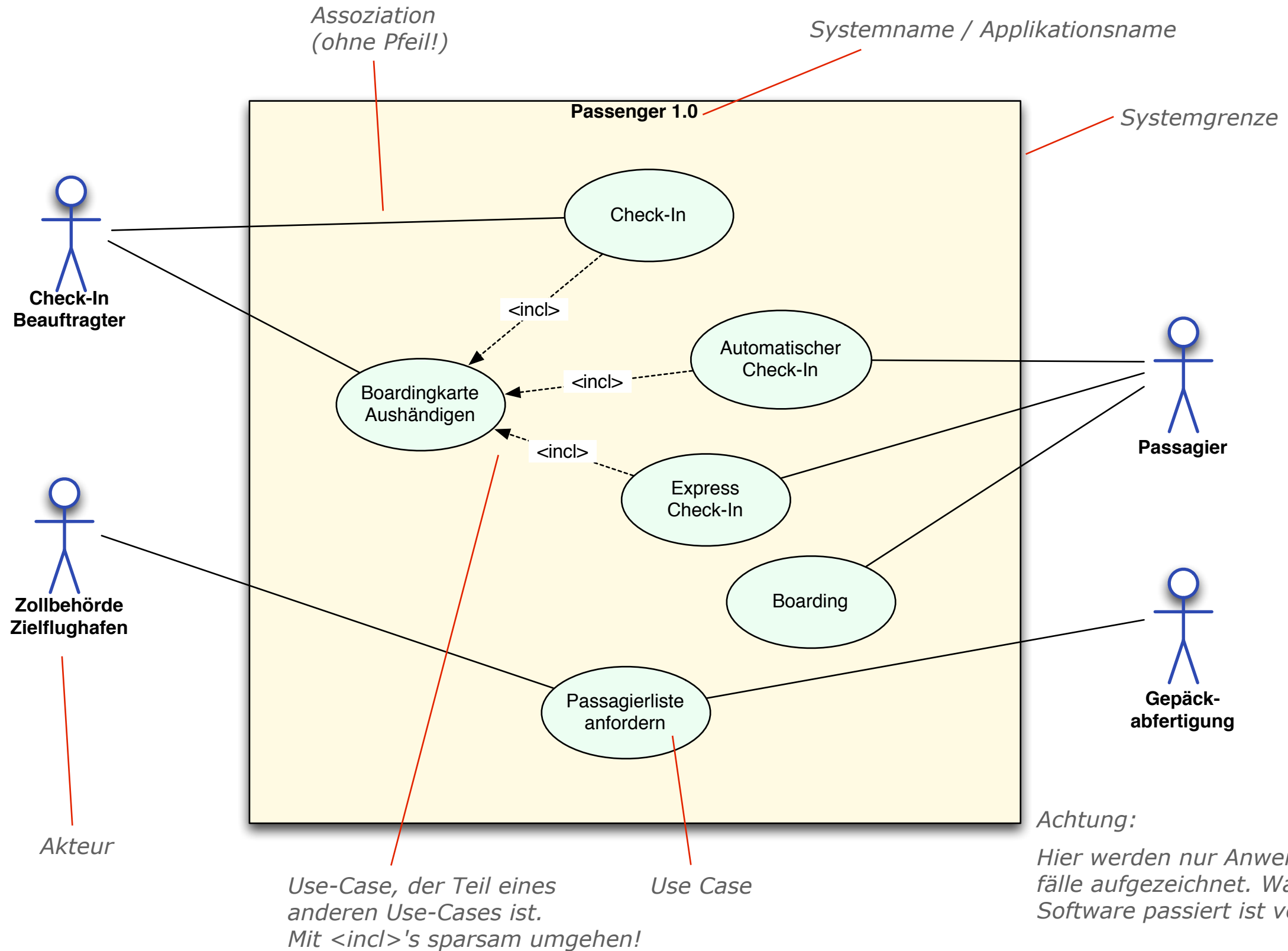
UML-Diagrams

- draw.io
- gliffy
- MySQL Workbench
- Dia
- Goon
- OminGraffle
- Visio

Gantt-Beispiel



Use-Case-Beispiel



Views erstellen

Typischerweise erstellt man vor dem Programm zuerst einen Prototypen, Fast-Prototype genannt, mit welchem man das GUI (Graphical User Interface) bzw. das HMI (Human Machine Interface) "trocken" durchspielen kann. Dazu stellt man jede View schematisch aber noch ohne Funktionalität dar. Erst wenn der Prototyp mit dem Auftraggeber besprochen wurde, beginnt man mit der Implementation.

Wir achten hierbei genau auf eine gute Software-Ergonomie (siehe nächste Seite). Achtung jeder Use-Case muss mithilfe mindestens einer View durchgeführt werden können.

Videothek

Kunden

Kunden Videos Ausleihen

Vorname	Nachname
Mohamad	Al Bezier
Klaus	Biedermann
Hans	Guten
Denise	Hösli
Susi	Werner
Ingeborg	Laux
Patrick	Senn

Kundenummer: 1003
 Anrede: Frau
 Vorname: Susi
 Nachname: Werner
 Strasse: Rebgasse 12
 PLZ: 8350
 Ort: Weinfelden
 Geburtsdatum: 15.05.1998
 Telefon: 071 / 664 20 77

Videonummer	Video Name	Ausleihdatum	Rückgabedatum
10006	10'000 B.C.	28.4.2015	
10005	Bob Aziz / Der Tanz des Windes	28.4.2015	
10000	American Pie 2	29.4.2015	30.4.2015



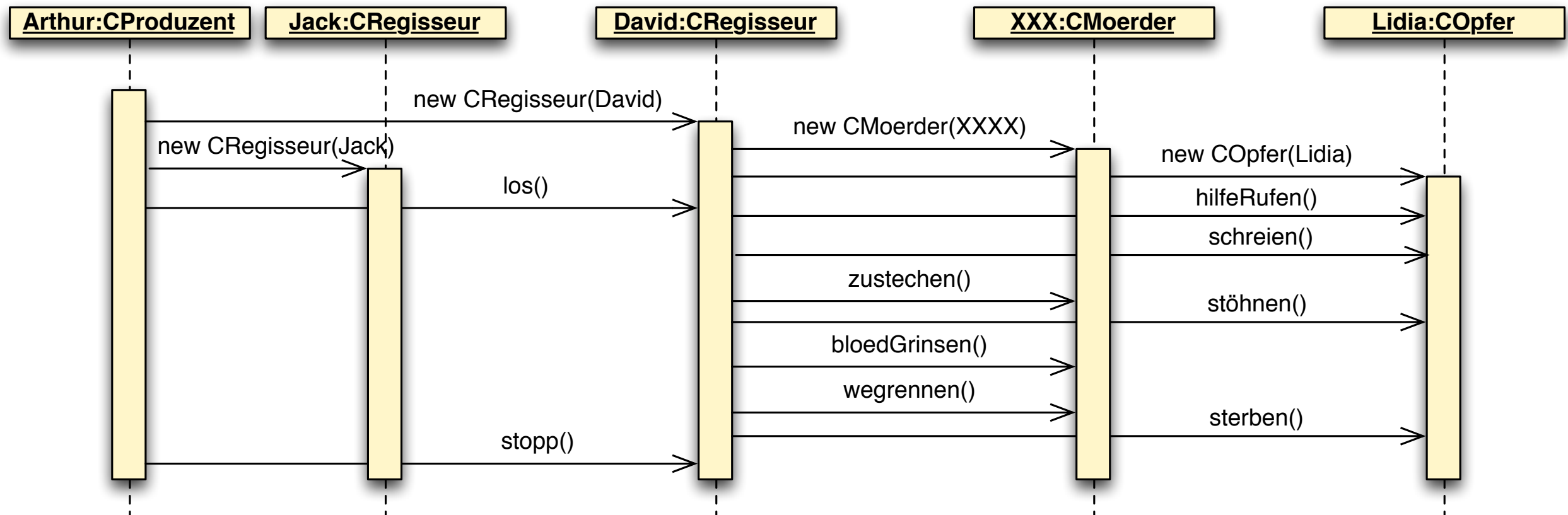
Beispiele erstellt mit: Balsamiq Mockups
www.balsamiq.com , Konto bei mir anfragen!

Software-Ergonomie

Damit ein Benutzer schnell und effizient mit einer Applikation arbeiten kann, und somit die Akzeptanz der Applikation hoch ist, gilt es einige Regeln zu beachten:

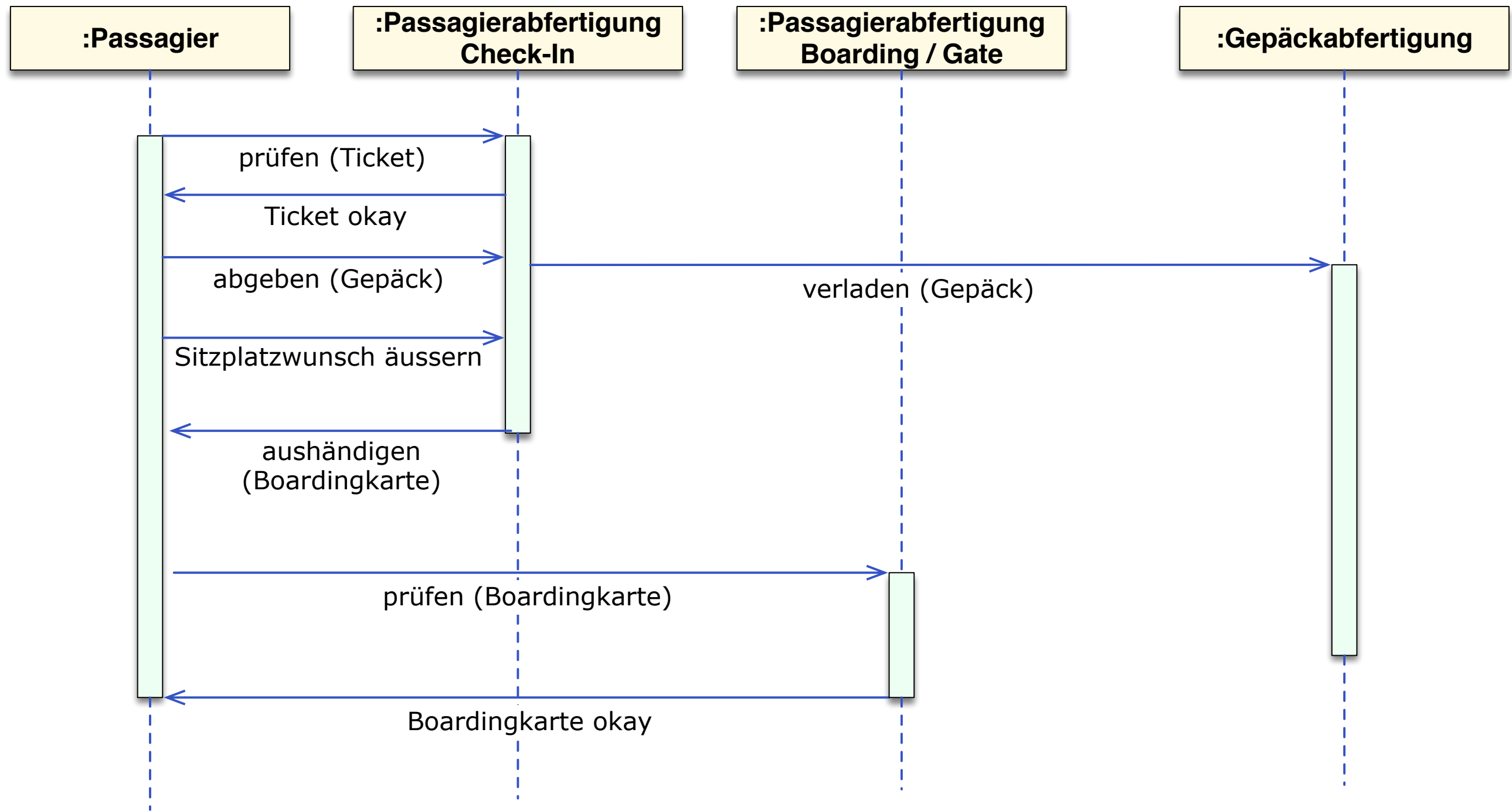
- Wo bin ich? Der Benutzer möchte jederzeit wissen wo er ist. z.B. Views mit Titel anschreiben
- Wo kann ich hin? Der Benutzer möchte jederzeit wissen wo er hin kann? Tasten, Reiter, Menus
- Höchstens 2-3 verschiedene Schriftarten verwenden.
- Schriftgrösse weder zu klein (ältere Leute) noch zu gross (unübersichtlich) wählen.
- Eingabemaske soll ca. 50% freie Fläche haben.
- Tasten, Buttons sollen in jeder View am gleichen Ort sein. Auch möglichst gleich wie OS.
- Zusammengehörende Felder sollen auch visuell gruppiert sein.
- Felder wenn immer möglich auf einander ausrichten.
- Mit Farben äusserst sparsam umgehen. Sich möglichst ans OS (Betriebssystem) halten. Keine grellen Farben und kein Blinken verwenden.
- Alle Tasten und Menus sollten für "Profi-"Benutzer auch per Tastatur bedienbar sein.

Beispiel für ein OOP-Sequenzdiagramm zur Verfilmung eines Krimis

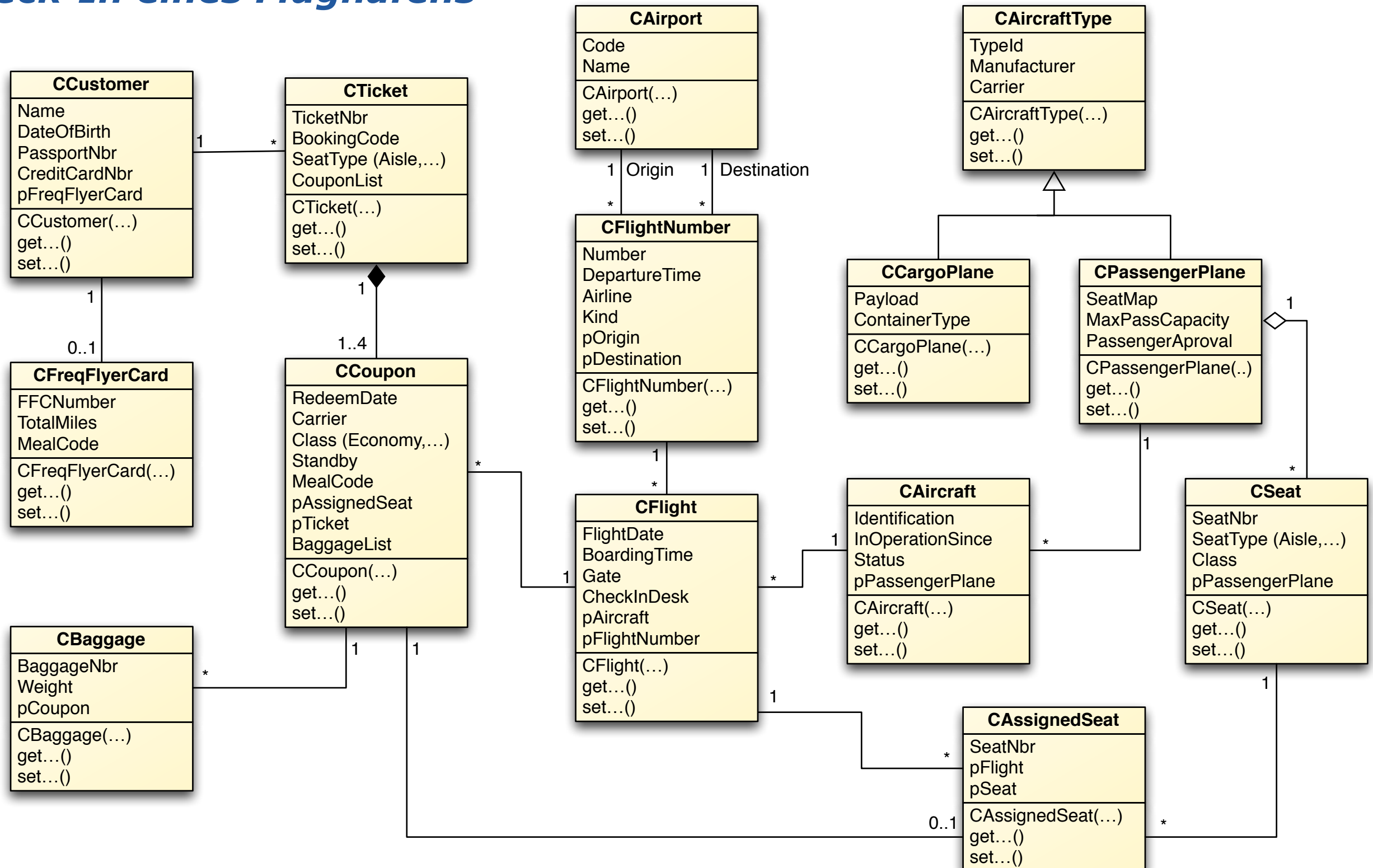


Beispiel für ein Betriebsablauf-Sequenzdiagramm

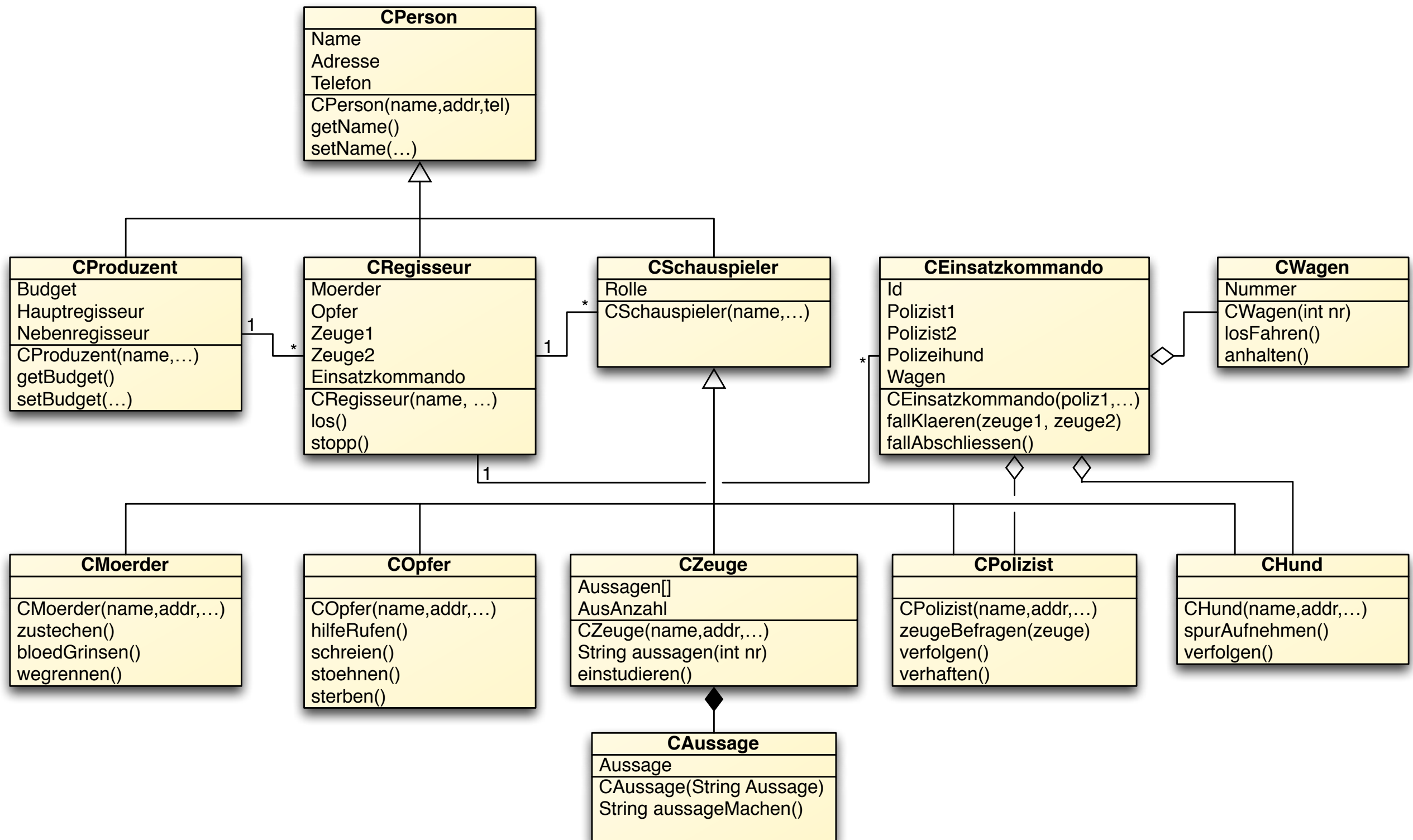
Passagierabfertigung Flughafen



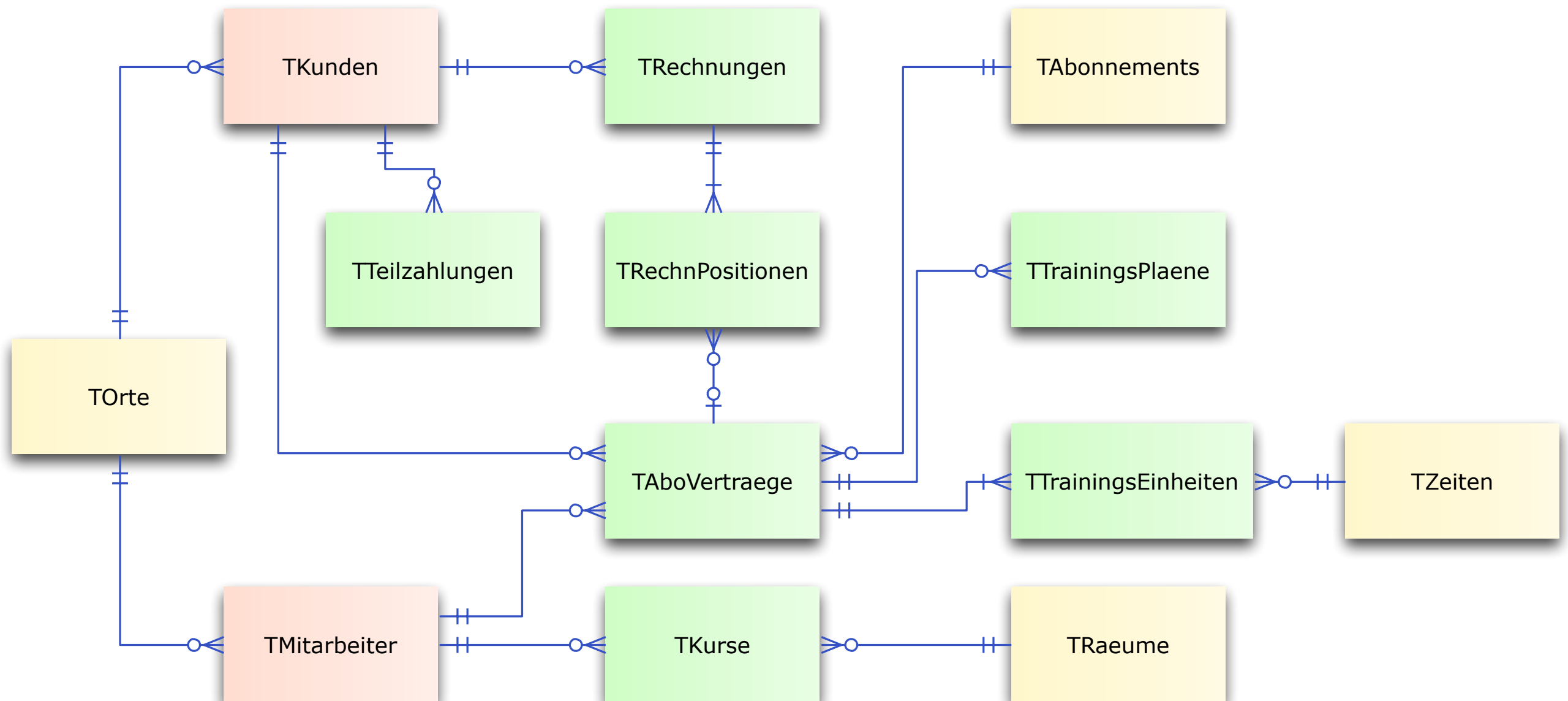
Beispiel für ein Klassendiagramm für den Check-In eines Flughafens



Beispiel für ein Klassendiagramm für die Verfilmung eines Krimis



Beispiel für ein ERM für ein Fitness-Studio



Testing

Um sicher zu gehen, dass sich eine Applikation auch so verhält, wie sie sich verhalten soll, müssen wir Tests durchführen. Bei den Tests ist es äusserst wichtig methodisch und strukturiert vorzugehen und somit wirklich alle Teilverhalten, Aspekte und Bedingungen genau zu prüfen und die Ergebnisse gut zu dokumentieren. Man muss sich jedoch bewusst sein, dass man nie alles testen kann ($2^{\text{Anzahl if's}}$!).

Man notiert sich dazu bereits in der Designphase so genannte Testfälle, die die wichtigsten Funktionalitäten abdecken sollten. Dabei unterscheidet man zwischen funktionalen und destruktiven Testfällen.

Funktionale Testfälle demonstrieren, dass die Applikation, wenn der Benutzer vernünftige, "gute" und wohlwollende Eingaben macht und korrekte Daten vorhanden sind, korrekte Ergebnisse liefert und gemäss Spezifikation funktioniert.

Destruktive Tests hingegen zeigen, dass die Applikation auch dann robust und vernünftig reagiert, wenn der Benutzer falsche und unmögliche Eingaben macht oder importierte Daten fehlerbehaftet sind.

Die Testfälle sollten dann möglichst von Anwendern, mindestens aber von "Nicht-Entwicklern" durchgespielt werden.

Wann werden Testfälle formuliert?

- Tests dann überlegen, wenn man noch die Übersicht hat, also in der Designphase

Wie werden Testfälle formuliert?

- Als Grundlage verwendet man die Use-Cases und überlegt sich für jeden Use-Case mehrere Testfälle
- Die Tests, die erwarteten Resultate und das Ergebnis müssen sehr präzise formuliert werden. Nur so können sie reproduzierbar immer gleich und bei jedem Release erneut getestet werden.
- Wir erstellen zwei Testtabellen, eine mit **funktionalen** Tests und eine mit **destruktiven** Tests.
- Die Tests sollen so formuliert sein, dass sie von jedermann durchgeführt werden können und absolut reproduzierbare Ergebnisse liefern.

Beispiele für Testfälle

Getestet von: **David Steger**
 Getestete Version: **1.12**
 Test-Datum: **4.2.2021**

Funktionale Testfälle

Test	Ergebnis	Bestanden
Layout Kunde: Button "Kunde Neu" anklicken	Neuer Datensatz mit neuer Kundennummer wird angelegt, alle anderen Kunden-Eingabefelder (Anrede, ..., Telefon) sind leer, Cursor steht im Feld Anrede	✓
Layout Kunde: Button "Kunde Neu" anklicken, Feld Geburtsdatum anklicken, Eingabe von 14.8.01 in das Eingabefeld Geburtsdatum, Tabulator Taste drücken	Geburtsdatum 14.08.01 erscheint im Eingabefeld. Curser steht im Feld Telefonnummer.	✗
Layout Kunde: Button "Videos" anklicken, 2. Video anklicken, Button "Kunde" anklicken, Button "Videos" anklicken	Videos Layout wird mit dem gleichen 2. Datensatz angezeigt wie dieses Layout das letzte Mal verlassen wurde	✓
Layout Kunde: In Kunden-Liste "Susanne Lèlange-Rüd.." anklicken	Der komplette Datensatz von Susanne Lèlange-Rüdisühli mit den Feldern Anrede, Knd. Nr. bis Telefon (siehe Anhang) alle korrekt ausgefüllt werden angezeigt.	✓

Destruktive Testfälle

Test	Ergebnis	Bestanden
3 Benutzer gleichzeitig: Layout Kunde: Button "Kunde Neu" anklicken, neuen Kunden Herr Klaus Biedermann, Hauptstrasse 17, 8521 Pfyn, 12.3.92, 071 432 56 47 eintragen, in den weissen Rand klicken	Bei einem Benutzer wird der Datensatz übernommen, bei den anderen erscheint die Meldung Kunde bereits vorhanden	✓
Layout Kunde: Button "Kunde Neu" anklicken, Feld Geburtsdatum anklicken, Eingabe von 14.8.21 in das Eingabefeld Geburtsdatum, Tabulator Taste drücken	Es erscheint die Meldung: Geburtsdatum liegt in der Zukunft. Nach OK steht Cursor auf Feld Geburtsdatum.	✗
Layout Kunde: Button "Kunde Löschen" so viele Male drücken bis kein Kunde mehr in der linken Liste. Dann noch ein weiteres Mal drücken.	Button nicht mehr aktiv. Es passiert nichts.	✗

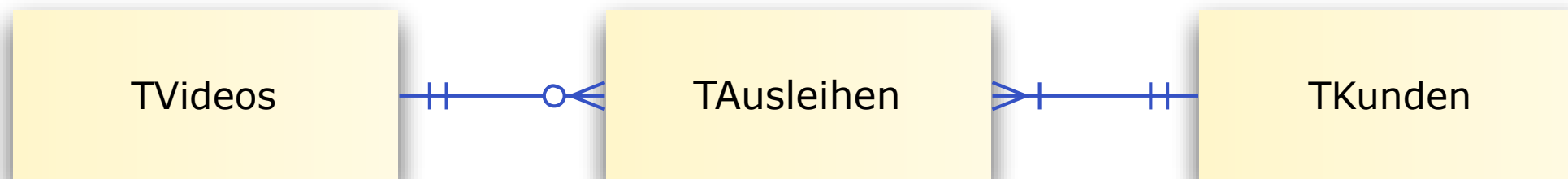
Beispiel Testfälle für die App SuperCalc

SuperCalc		Vers. 0.85			
Nr.	Testfall	Resultat	Test		Visum
			OK	Failed	
	Konstruktive Tests:				
1	Das Fenster wird durch click & drag an der rechten unteren Ecke aufs Maximum vergrössert.	Der Inhalt ist ansprechend angeordnet und gut über das Fenster verteilt.			
2	Das Fenster soll durch click & drag an der rechten unteren Ecke aufs Minimum verkleinert.	Der Inhalt ist ansprechend angeordnet. Nichts ist verdeckt oder abgeschnitten. Alles kann gelesen werden.			
3	Das Datum 7.2.1824 wird gefolgt von der Enter Taste per Tastatur ins Datumfeld eingegeben.	Im Ausgabefeld wird: "Dies ist ein Samstag." ausgegeben.			
4	Die Applikation WeekDayCalc wird mit Doppelclick auf WeekDayCalc.jar geöffnet.	Die Applikation öffnet sich. Man kann wie gewohnt damit arbeiten.			
	Destruktive Tests:				
5	Ins Ausgabefeld per Tastatur "Hallo" schreiben.	Das Ausgabefeld lässt sich nicht anklicken und auch nicht beschreiben.			
6	Das Datum 7.2.1824 wird gefolgt von 6 x Tabulatortaste per Tastatur ins Datumfeld eingegeben.	Fokus springt zwischen dem Berechnen-Knopf und dem Eingabefeld hin und her und ist am Schluss auf dem Eingabefeld.			
Tests ausgeführt am:		Unterschrift:			

Entwicklung einer Datenbankapplikation

1. ERM Entity Relationship Model

Als erstes benötigen wir ein bereinigtes ERM.
Hier am Beispiel einer Videothek.



Wichtig:

- Jede Relation (Entitätstyp) bezeichnet genau einen Sachverhalt!
- Relationen beginnen immer mit T (für Tabelle)
- Relation werden in der Mehrzahl gehalten (bestehen aus vielen Entitäten)

Jede Beziehung besteht aus 2 Assoziationen, welche wie folgt gelesen werden:

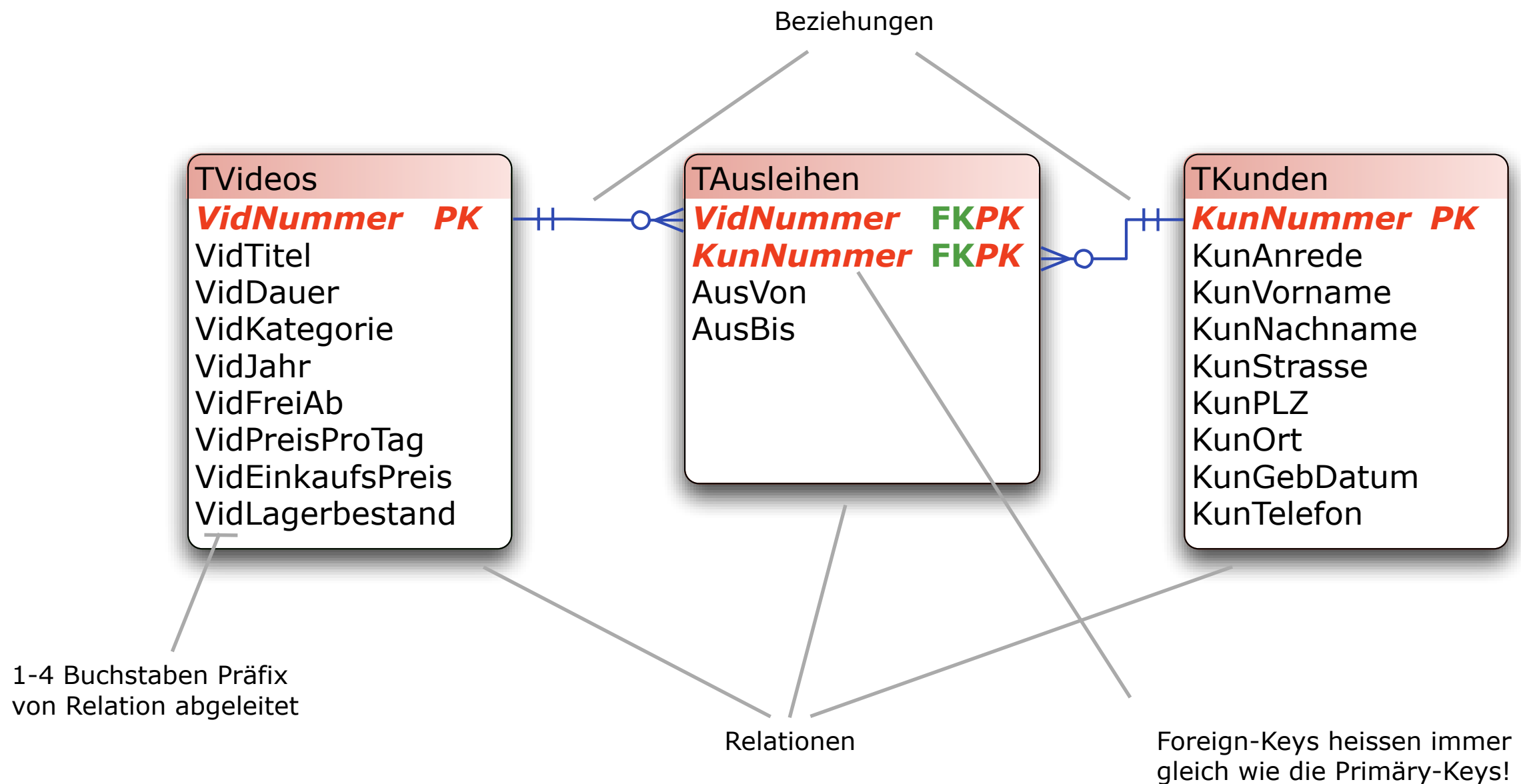
- **Ein** Video kann **nie bis mehrmals** ausgeliehen werden.
- **Eine** Ausleihe bezieht sich auf **genau ein** Video.

Immer EINS!

2. Datenbank-Schema

Dieses ERM bauen wir zu einem Datenbank-Schema aus.

Ein Datenbank-Schema ist ein ERM mit Attributen, Primär- und Fremdschlüsseln.



Die Entitätsmenge und ihre Elemente

Entitätstyp

Entitätsmenge Relation

Attribute/Eigenschaften

Primär-schlüssel

Entitäten Tupel

Wertebereich: Herr...Frau

Attributswerte

TKunden						
KunNummer	KunAnrede	KunVorname	KunNachname	KunStrasse	KunPLZ	KunOrt
10003	Herr	Silvan	Ritsch	Hofstr. 5	8580	Amriswil
10004	Herr	Christoph	Hauser	Seegasse 7	9320	Arbon
10005	Frau	Silvia	Agonet	Ottenberg	8570	Hard
10006	Herr	Christoph	Hauser	Alte Landstr. 4	8508	Homburg