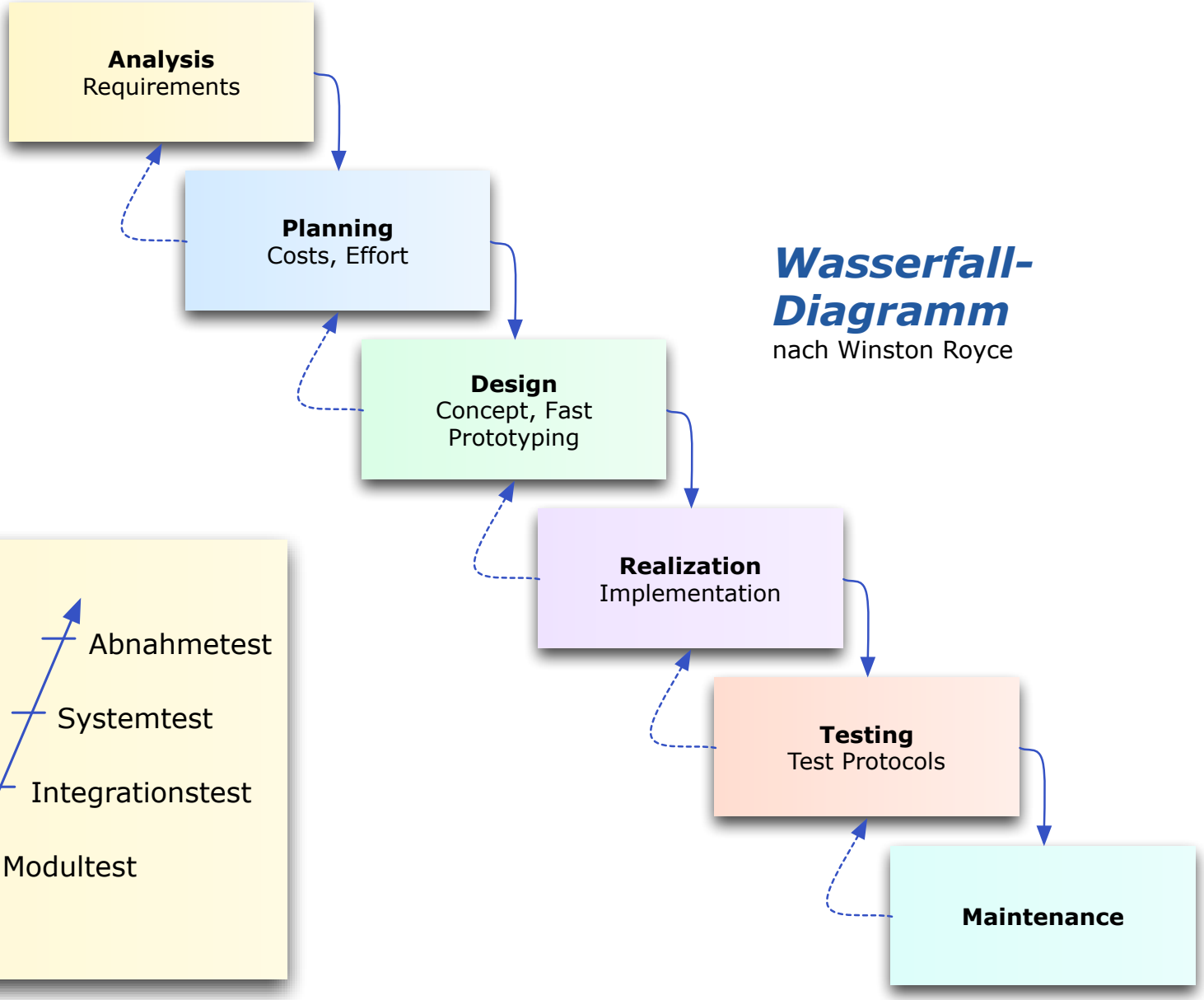
Schülerdossier
Selbständige Arbeit
Maturaarbeit

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, 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 oder Fast-Prototype, Datenbankschema, Klassendiagramm, Zustandsdiagramm, Sequenzdiagramm, Test-Liste (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, Bulk-Dump

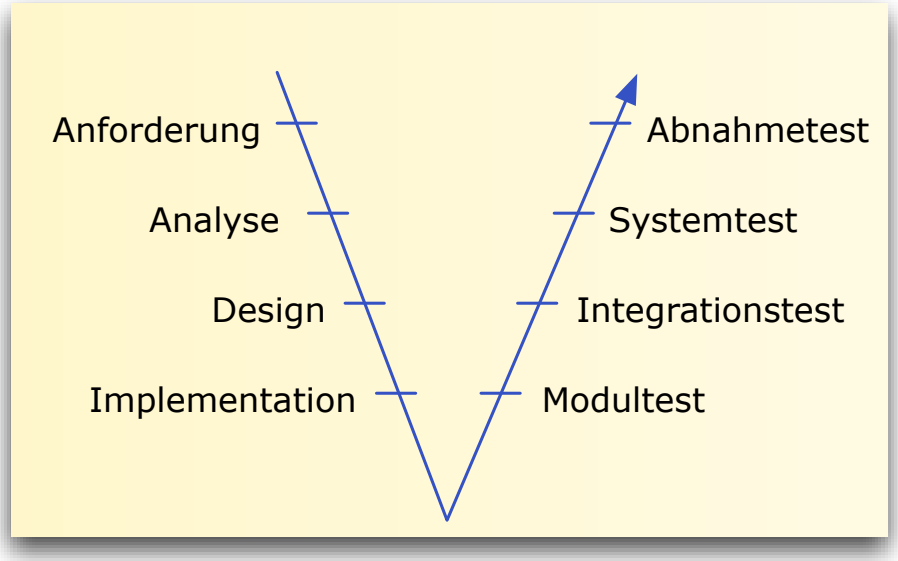
Projektphasen



Wasserfall-Diagramm

nach Winston Royce

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?

Wir erstellen Fragenkataloge, führen Interviews mit Wissensträgern, machen Umfragen, beschreiben 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 (was wird nicht realisiert), das Use-Case Diagramm und das ERM (siehe Beispiele).

Planung

Bevor wir mit dem eigentlichen Design beginnen können, planen wir das Projekt. Zunächst interessiert uns der Aufwand. Diesen ermitteln wir, indem wir das Projekt in kleine Stücke unterteilen und den Aufwand für jedes einzeln schätzen. Das Verteilen dieser Teilaufwände auf den Kalender ergibt dann die Zeitabschätzung. Auch hierhin gehört die Projektrollenverteilung in der wer was macht beschrieben wird.

Die Aufwand- und die Zeitabschätzung sowie die Rollen können auch in einem Gantt-Diagramm festgehalten werden.

Design

Beim Design (Programmentwurf) 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 (druchdenken, 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

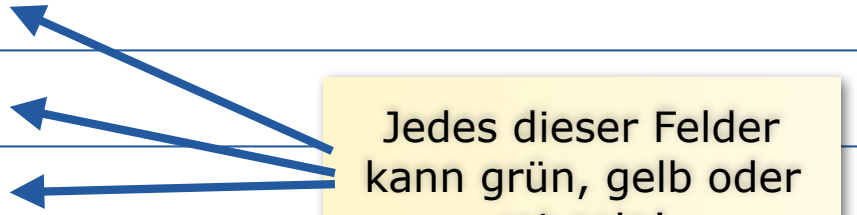
In dieser Phase erstellen wir eine Installationsanleitung und eine Benutzeranleitung. Die Installation sollte mithilfe eines von Ihnen erstellten Installers oder eines Installations-Scripts ganz einfach vonstatten gehen. Die Benutzeranleitung sollte von jeglichen technischen Schnick-Schnack befreit und so von irgend einem Benutzer gelesen werden können

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:

Nützliche Programme / Werkzeuge

Gantt-Charts

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

Collaboration-Tools

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

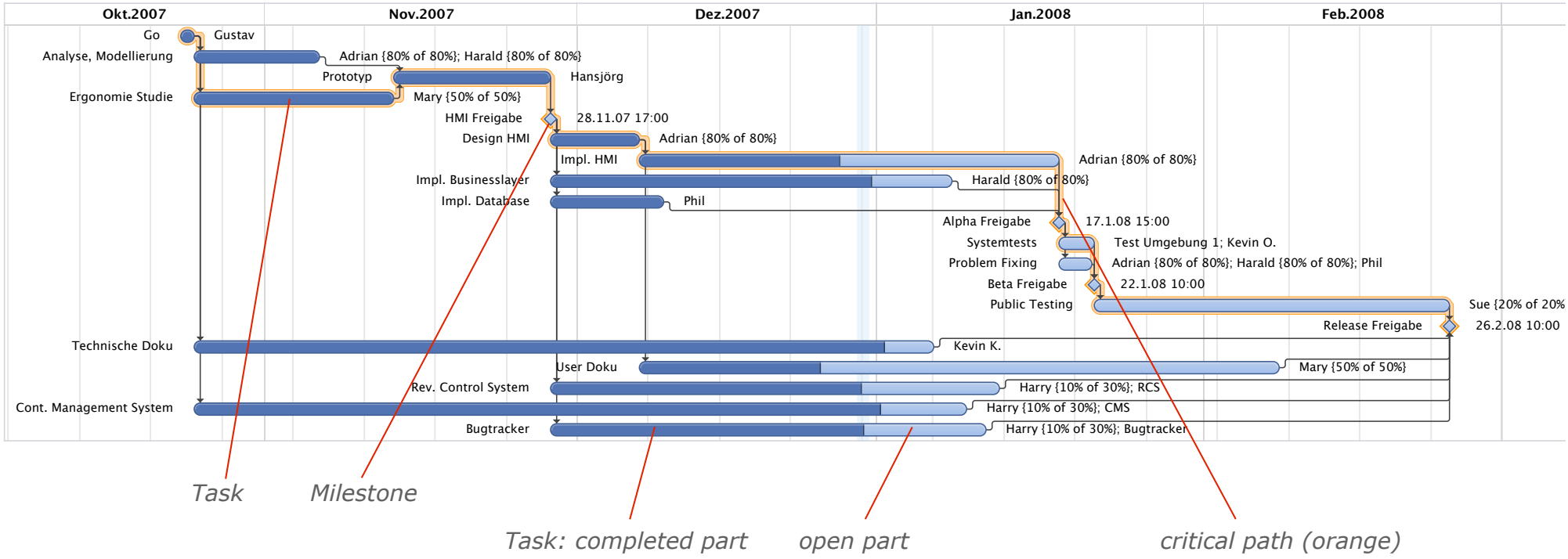
Fast-Prototyping

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

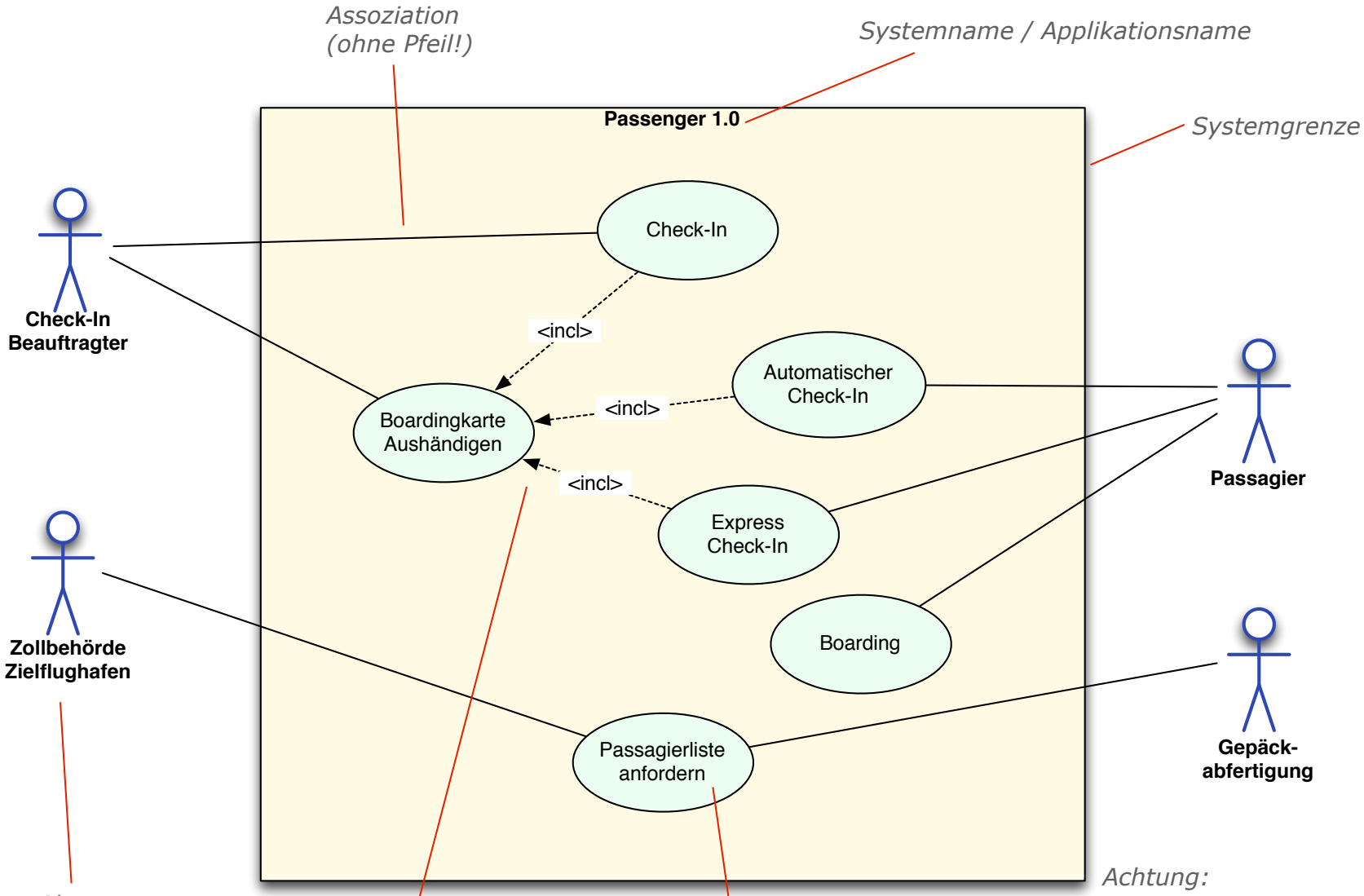
UML-Diagrams

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

Gantt-Beispiel



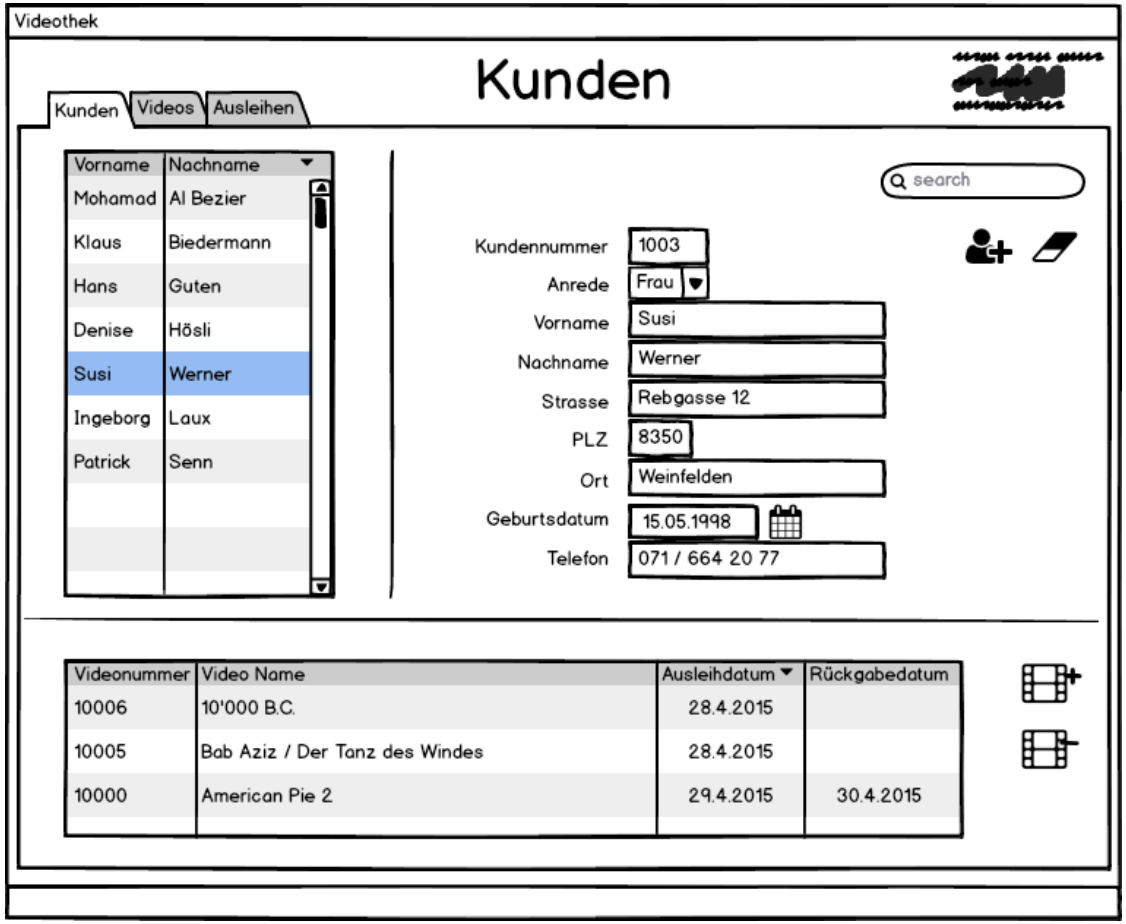
Use-Case-Beispiel



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.



Beispiele erstellt mit:

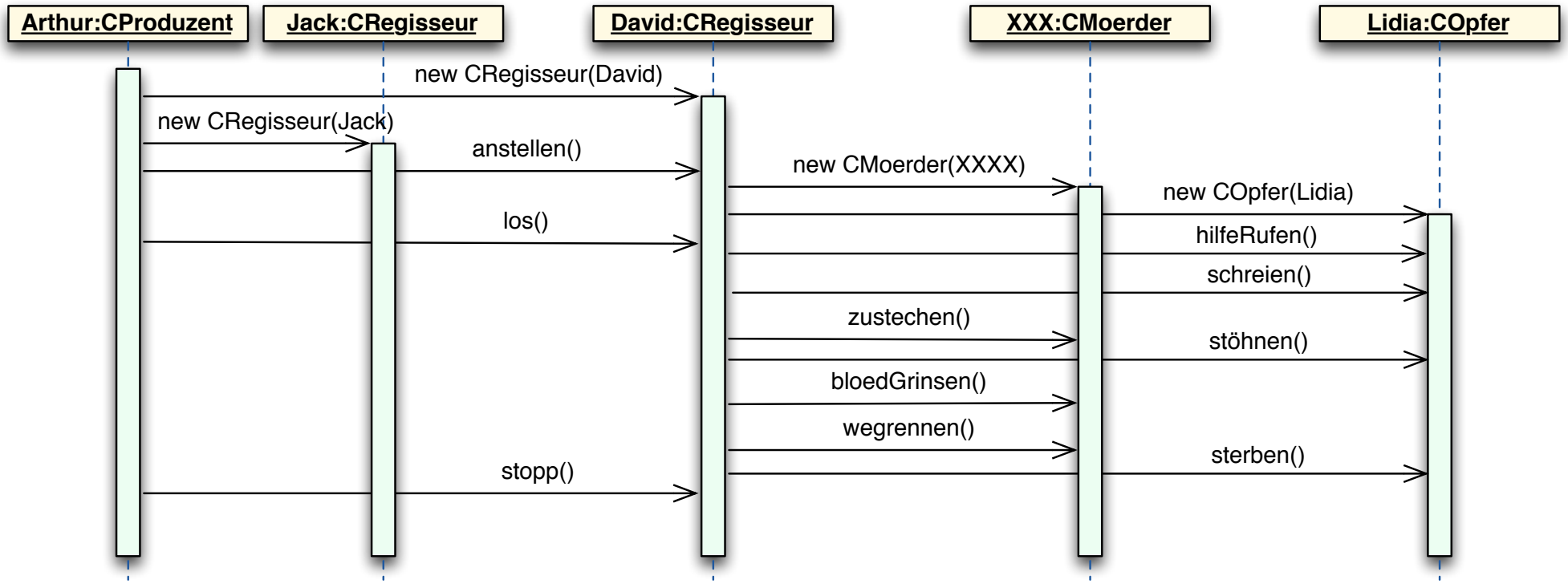
Balsamiq Mockups (www.balsamiq.com)

Software-Ergonomie

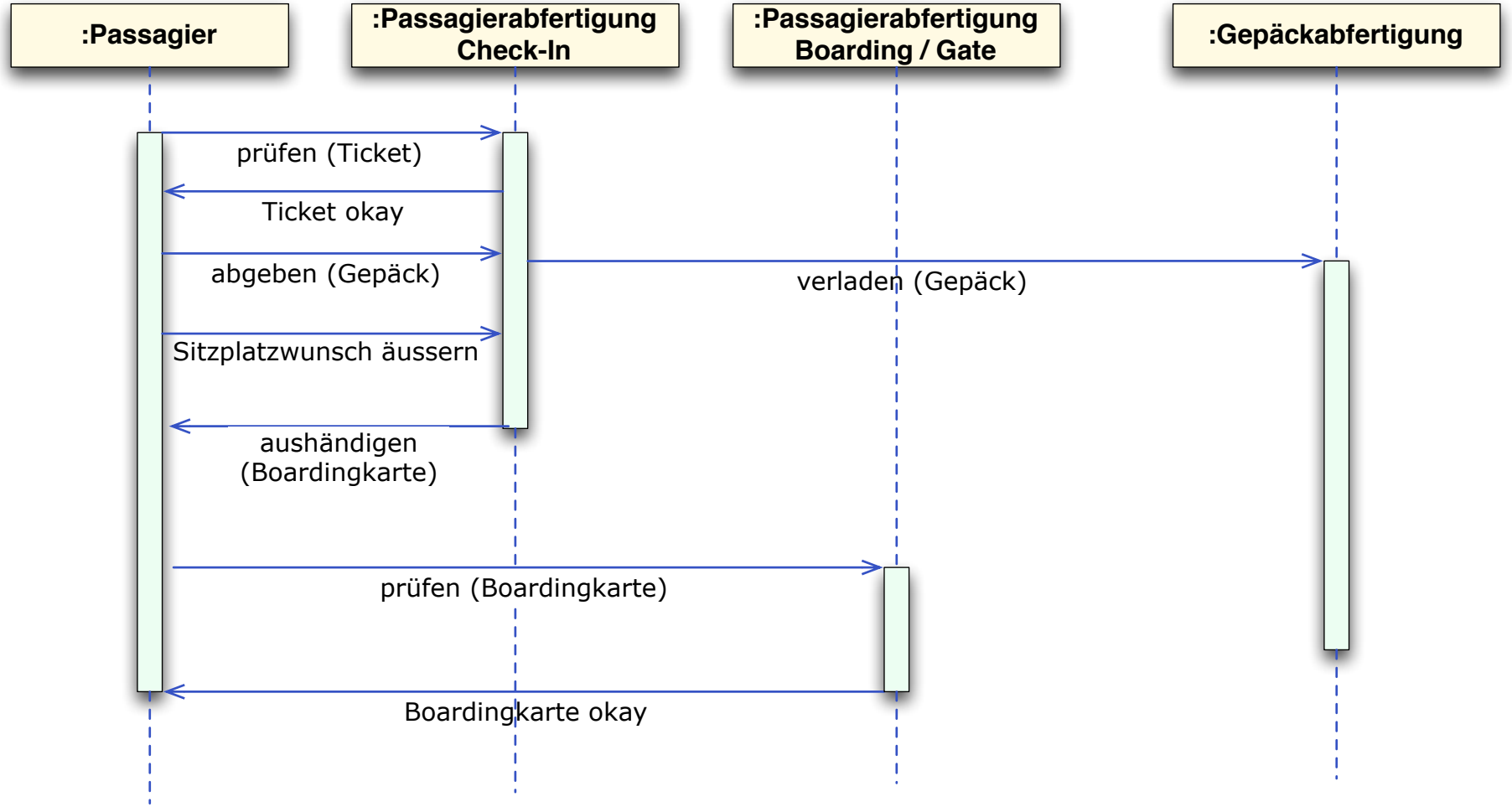
Damit ein Benutzer schnell und effizient mit einer Applikation arbeiten kann, und somit die Akzeptanz der Applikation hoch ist, gilt es einige Regeln zu beachten:

- 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.

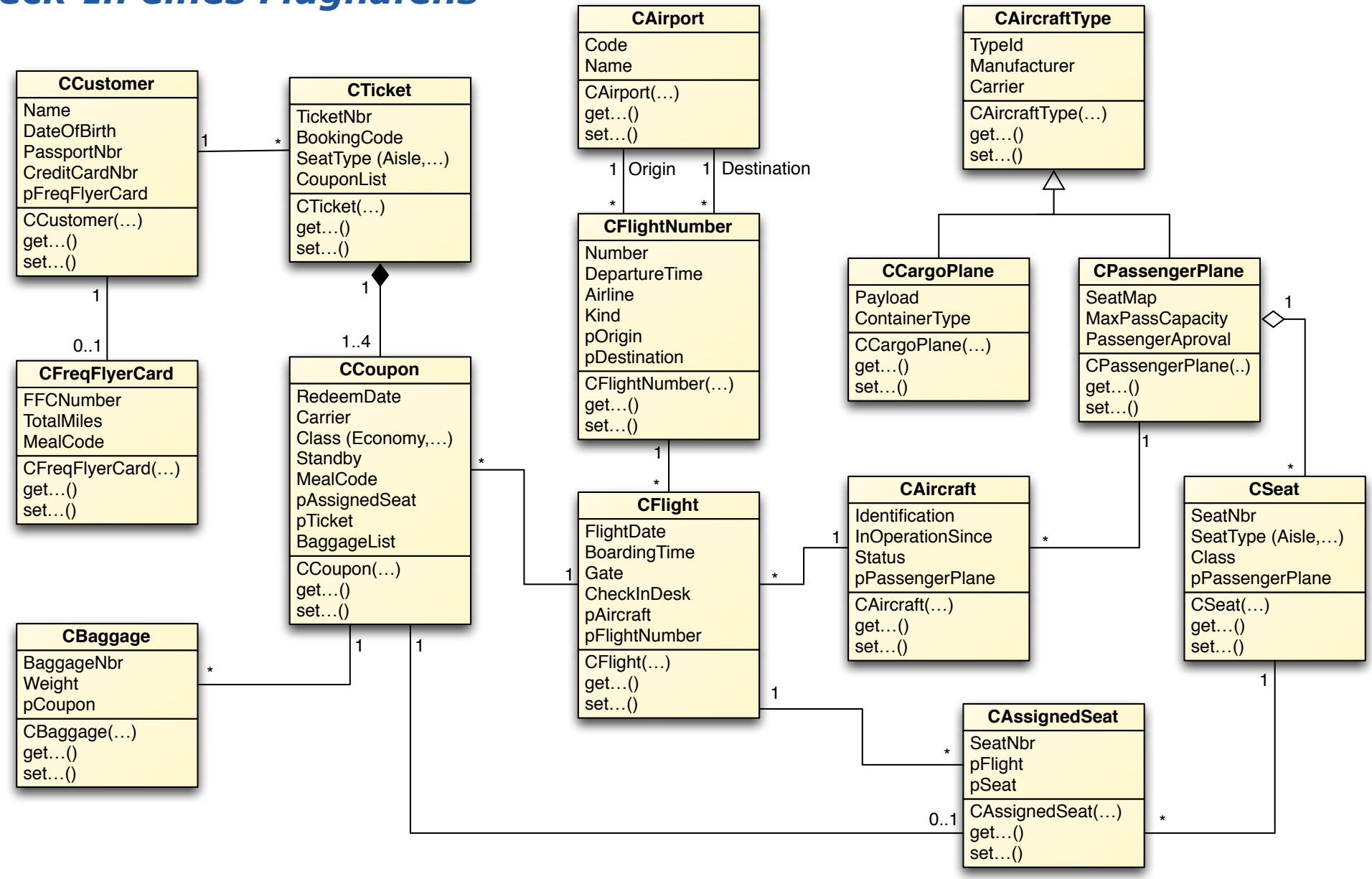
Beispiel für ein Sequenzdiagramm im OOP zur Verfilmung eines Krimis



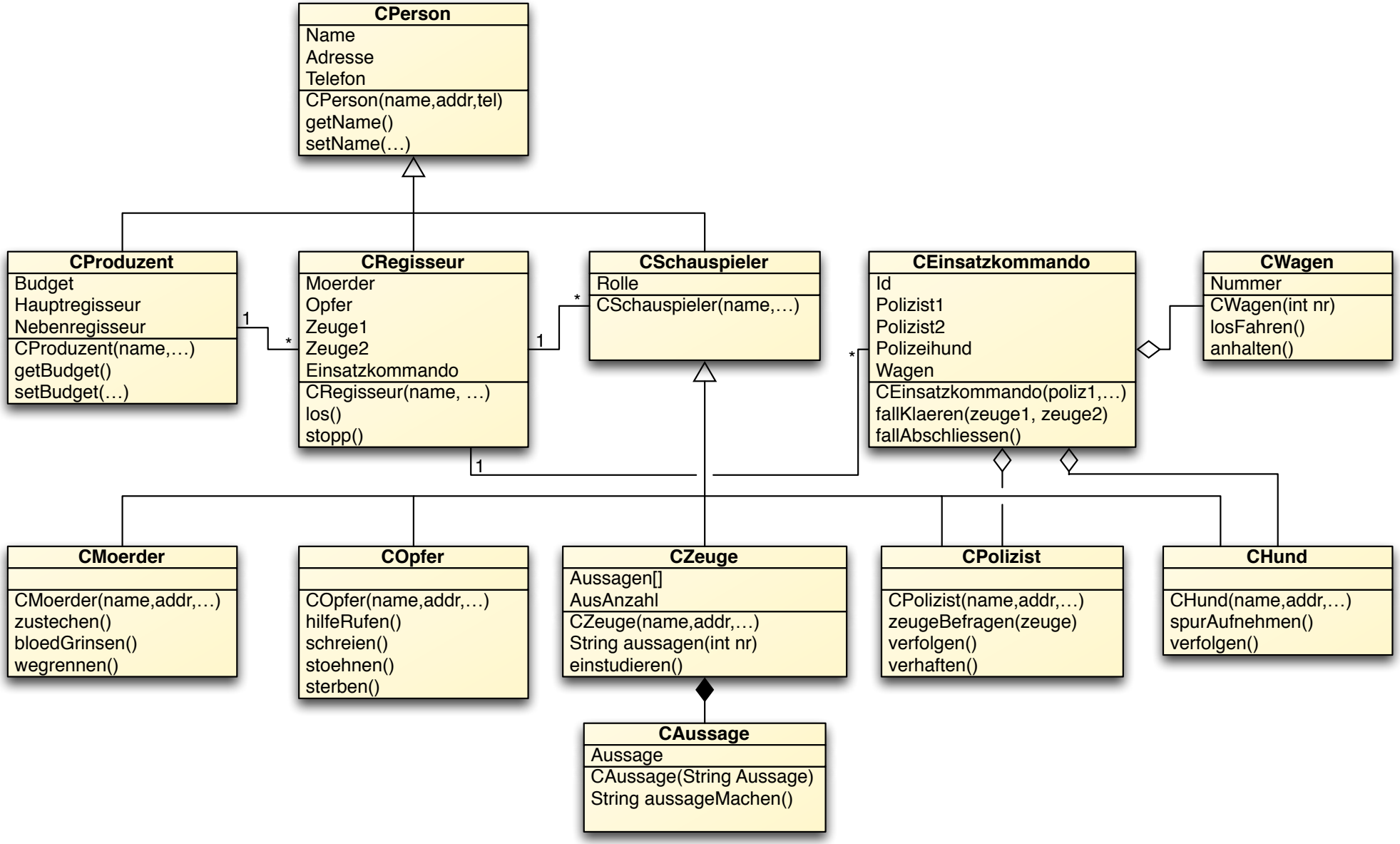
Beispiel für ein Sequenzdiagramm (Betriebsablauf/versch. Systeme) Passagierabfertigung Flughafen



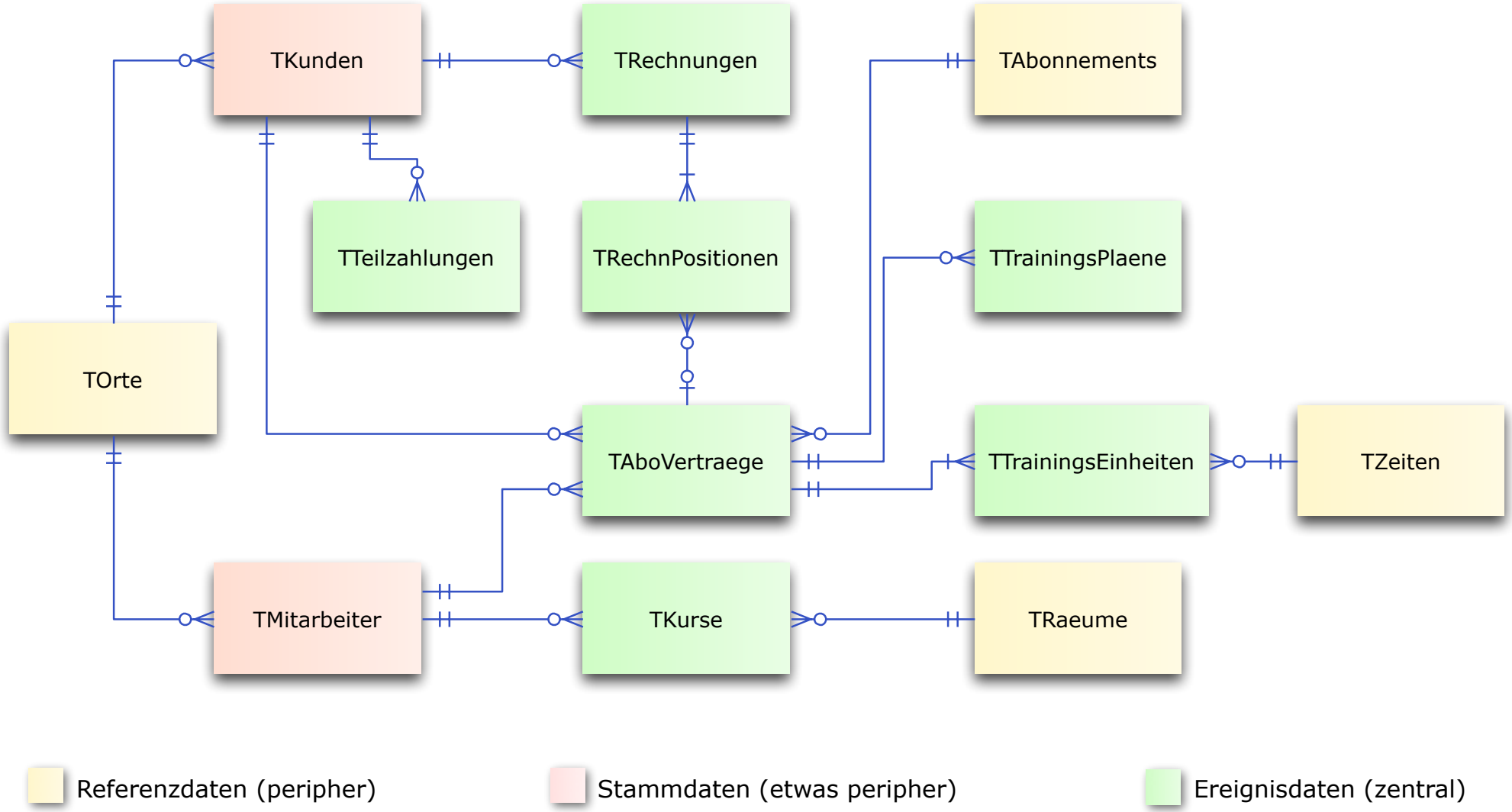
Beispiel für ein Klassendiagramm für den Check-In eines Flughafens



Beispiel für ein Klassendiagramm für die Verfilmung eines Krimis



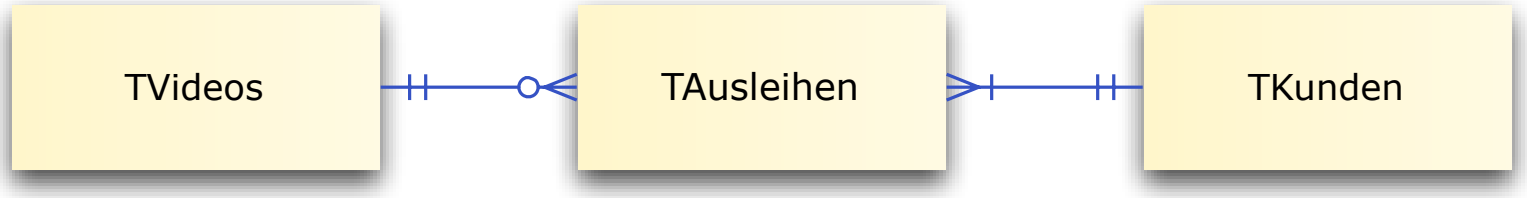
Beispiel für ein ERM für ein Fitness-Studio



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)

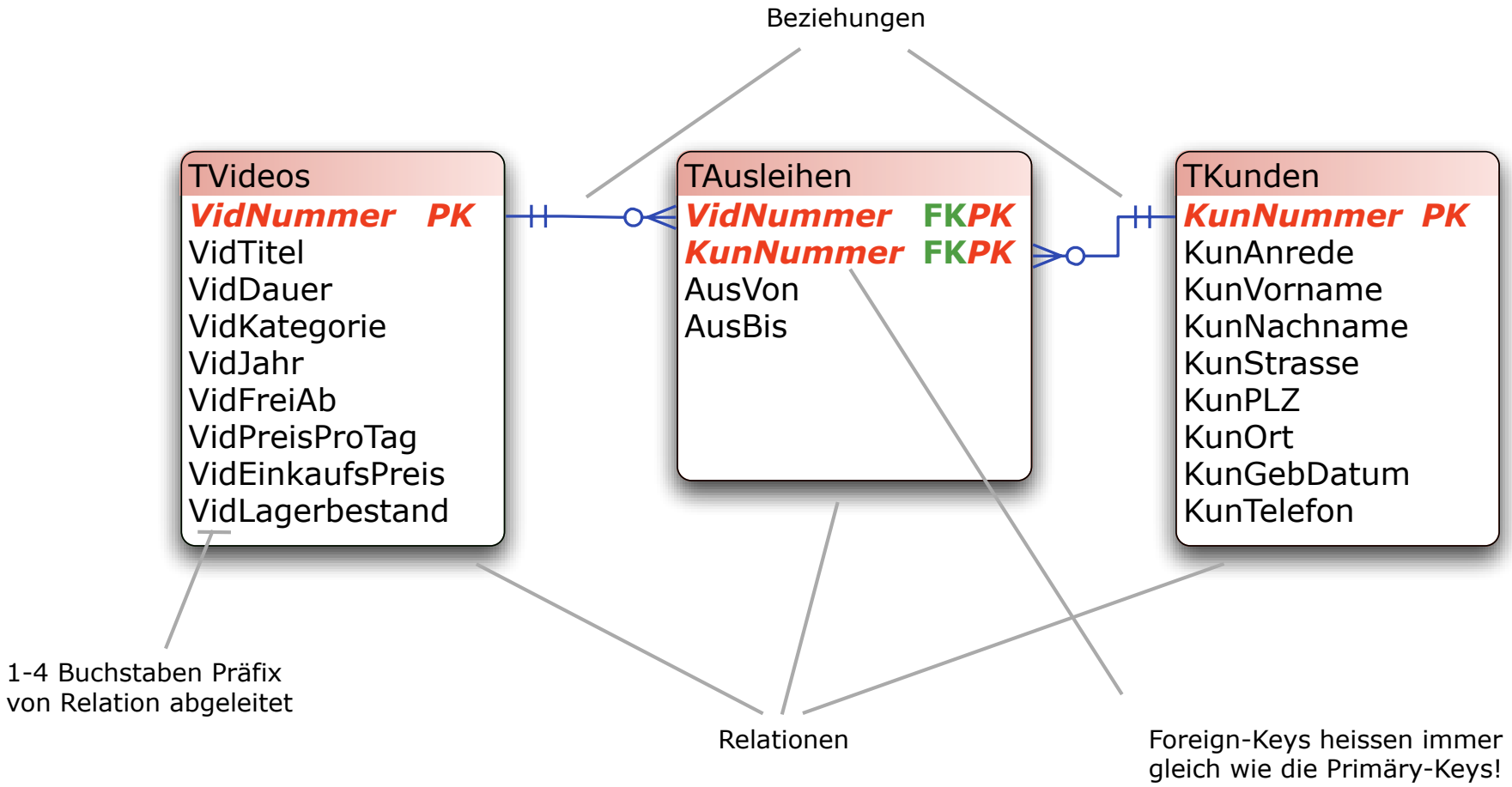
Jede Beziehung besteht aus 2 Assoziationen, welche wie folgt gelesen werden:
- **Ein** Video kann **nie bis mehrmals** ausgeliehen werden.
- **Eine** Ausleihe bezieht sich auf **genau ein** Video.

Immer EINS!

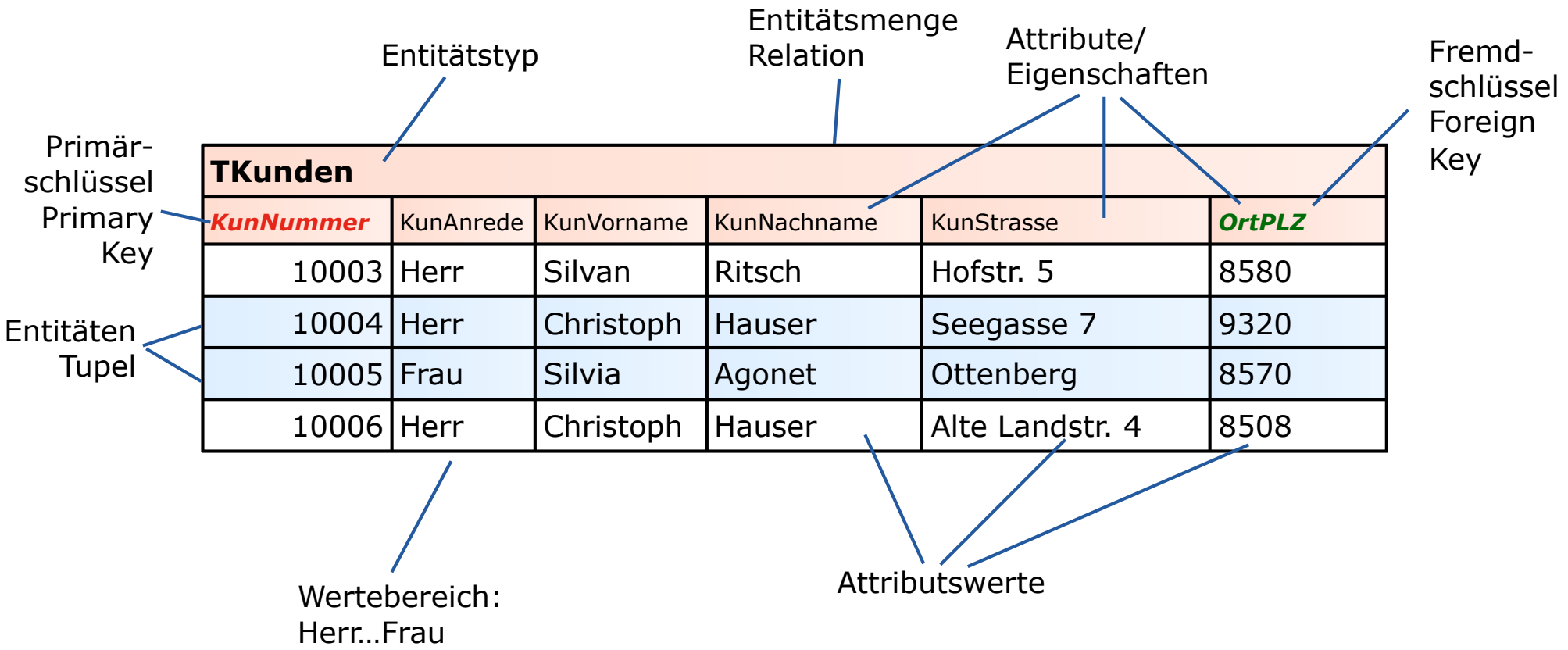
2. Datenbank-Schema

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



TKunden

KunNummer	KunAnrede	KunVorname	KunNachname	KunStrasse	*OrtPLZ*
10003	Herr	Silvan	Ritsch	Hofstr. 5	8580
10004	Herr	Christoph	Hauser	Seegasse 7	9320
10005	Frau	Silvia	Agonet	Ottenberg	8570
10006	Herr	Christoph	Hauser	Alte Landstr. 4	8508