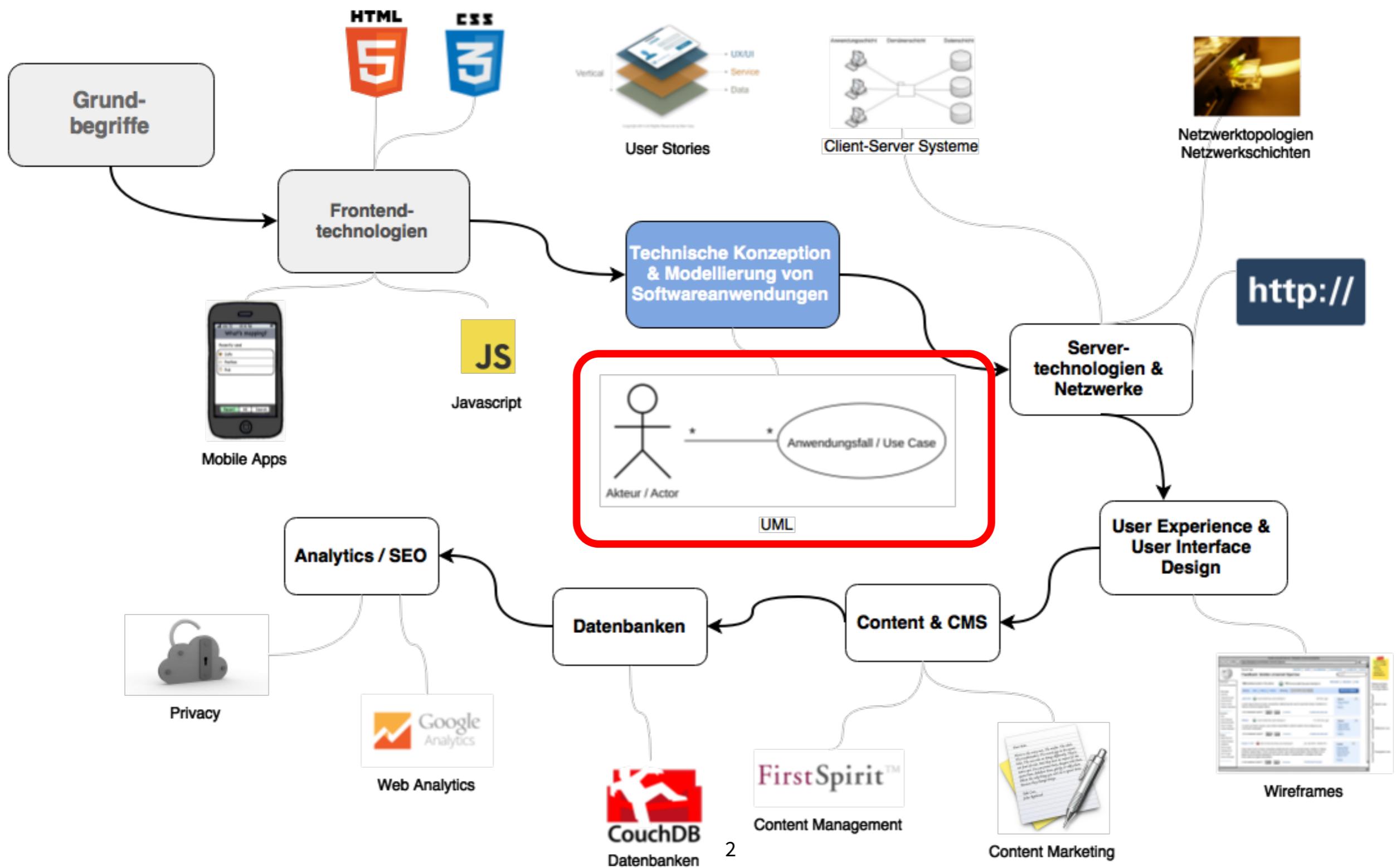


Modellierung von Softwareanwendungen

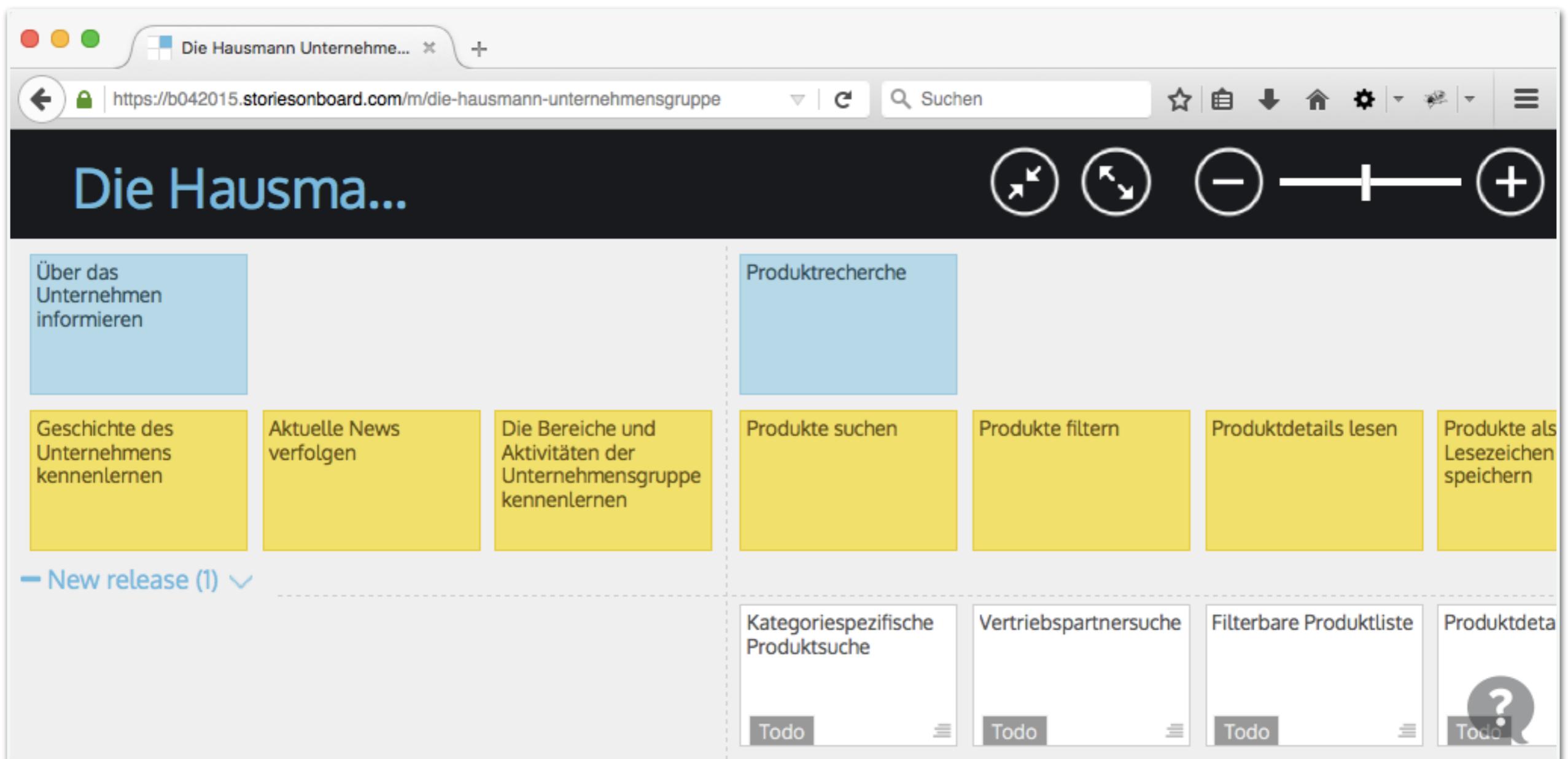
Einführung in Softwaretechnologien

Alexander Thomas
me@alexander-thomas.net

Überblick



Story Maps



<https://b042015.storiesonboard.com/m/die-hausmann-unternehmensgruppe>

Modellierung von Software-Systemen

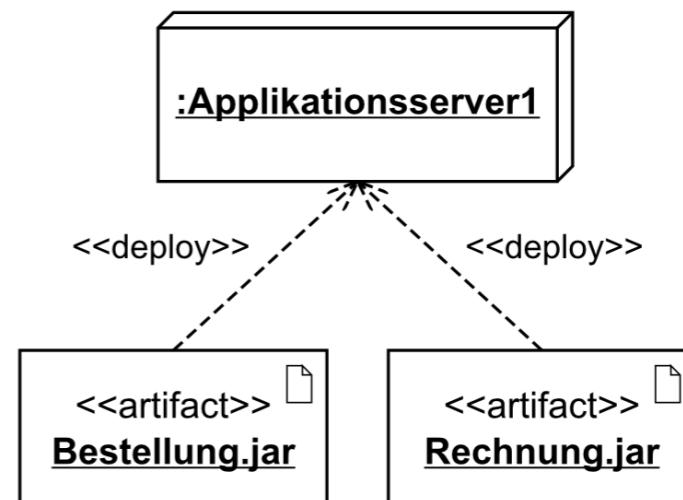
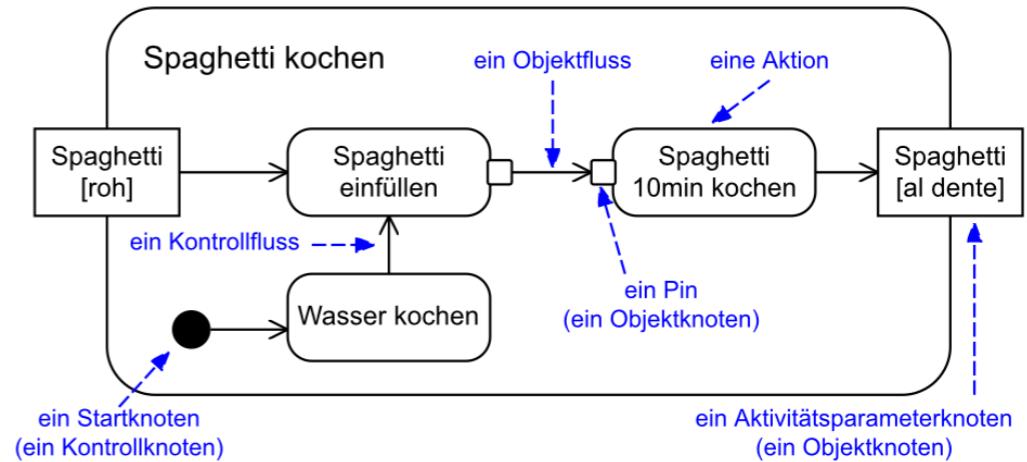
- Gesamtkomplexität eines Softwaresystems ist schwer zu erfassen
 - Fachliche Anforderungen (Kunde muss verifiziert sein, um zu...)
 - Technische Anforderungen (Authentifizierung, Double Opt-In)
 - Modelle helfen daher den Projektbeteiligten, **schrittweise ein System zu erarbeiten** und immer nur so viel Komplexität zu betrachten, wie gerade nötig.
 - Als Abbild von etwas erfasst ein Modell **nie alle Attribute des Originals**, sondern nur diejenigen, die dem Modellschaffer bzw. Modellnutzer **relevant erscheinen**.
- > Ein Modell wird also **immer interpretiert**.

Modellierung von Software-Systemen

- Ein Modell zeichnet sich also durch **Abstraktion** aus, also die bewusste Vernachlässigung bestimmter Merkmale, um die für den Modellierer oder den Modellierungszweck **wesentlichen Modelleigenschaften hervorzuheben**.
- Im besten Fall beleuchtet **ein Modell die verschiedenen Aspekte eines Systems gleichzeitig für verschiedene Anforderer**.
- UML-Modelle erlauben so **Kommunikation zwischen Fachleuten und Umsetzern**

Modellierung von Software-Systemen

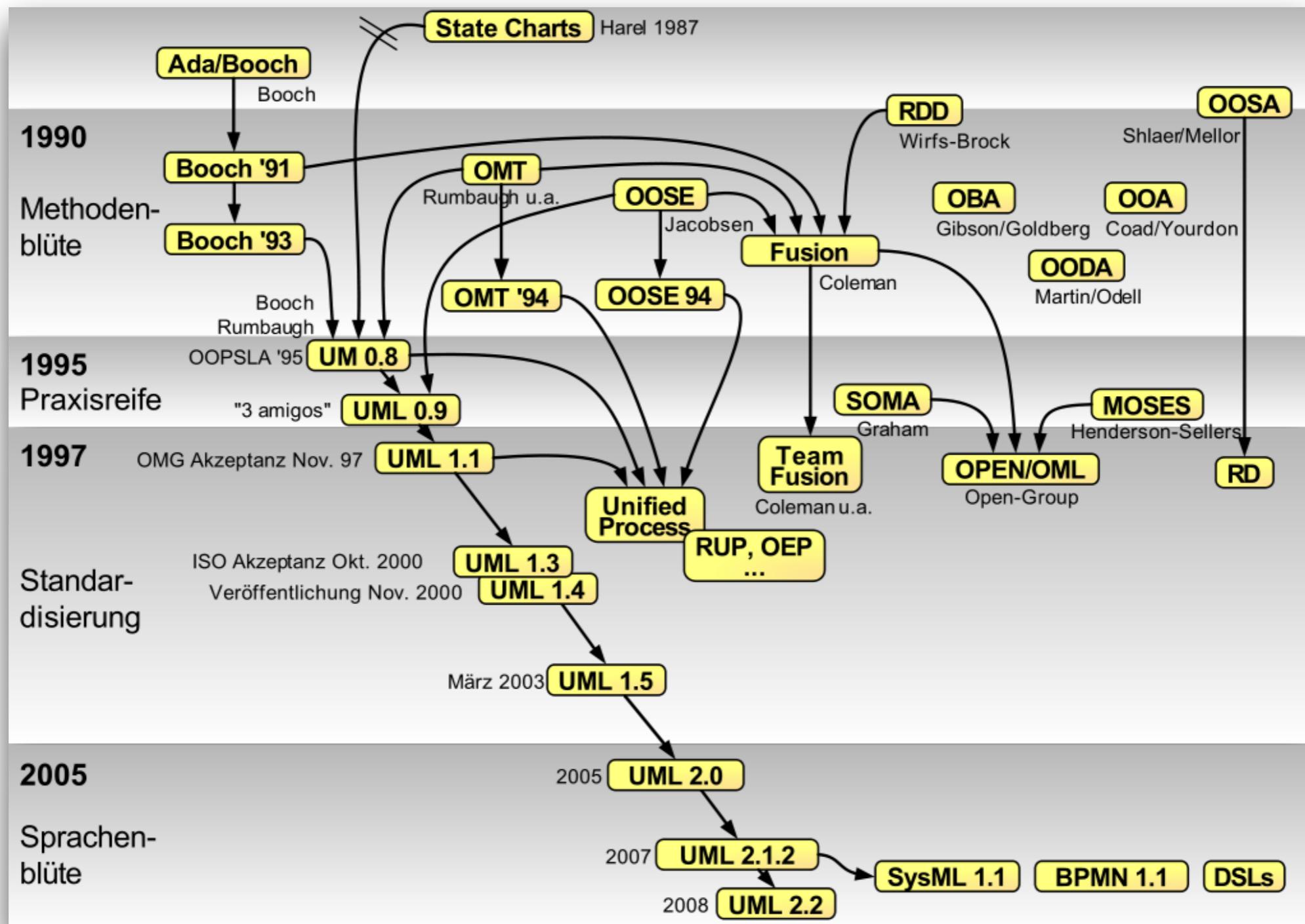
- Softwareentwickler realisieren Arbeitsabläufe, die Business Analysten in Zusammenarbeit mit Fachvertretern in **Aktivitätsdiagrammen** beschrieben haben
- Systemingenieure installieren und betreiben Softwaresysteme basierend auf einem Installationsplan, der als **Verteilungsdiagramm** vorliegt.



UML Entstehungsgeschichte

- UML ist die **Konsolidierung verschiedener Modellierungsnotationen**, die zwischen Ende der achtziger bis Anfang der neunziger Jahre entstanden waren.
- Ausgangspunkte dieser Notationen waren **objektorientierte Programmiersprachen**, wie Smalltalk oder C++ und stetig **wachsende Anforderungen** an die Strukturierung von Software-Systemen.

UML Entstehungsgeschichte



UML Entstehungsgeschichte

- Seit 1994 arbeiteten Grady Booch, James Rumbaugh und Ivar Jacobson in ihrer gemeinsamen Firma Rational Rose an einer **umfassenden Methode zur Vereinheitlichung der Modellierung.**
- In Zusammenarbeit mit Firmen aus der Software-Industrie entstand daraus die **Unified Method 0.8**, die bis 1997 zur **Unified Modeling Language 1.1** weiterentwickelt wurde.
- Die Notation der UML 1.1 war so umfangreich, dass der Anspruch eine Methode zu definieren auf die **Definition einer Modellierungssprache** reduziert wurde.

UML Entstehungsgeschichte

- UML 1.1 wurde **1997** bei der Object Management Group (OMG) eingereicht und wurde ein weltweiter Standard im Bereich der Objekt-Orientierten Entwicklung.
- Die erste Version der UML wurde bis zur UML 1.5 weiterentwickelt.
- Um die UML an neue Technologien anzupassen, wurde die Struktur mit UML 2.x noch einmal weitreichend überarbeitet.
- Zur Zeit steht UML bei **Version 2.5** (<http://www.omg.org/spec/UML/>).

Was ist UML

- **Unified Modeling Language (UML)** ist eine **Modellierungssprache** zur Beschreibung von Software-Systemen.
- Sie wird von der **Object Management Group (OMG)** entwickelt und ist sowohl von ihr als auch von der ISO (ISO/IEC 19505 für Version 2.4) standardisiert.
- UML enthält verschiedene Arten von **Diagrammen** und **Bezeichner** für die statische und dynamische **Modellierung** von Analyse, Design und Systemarchitektur.

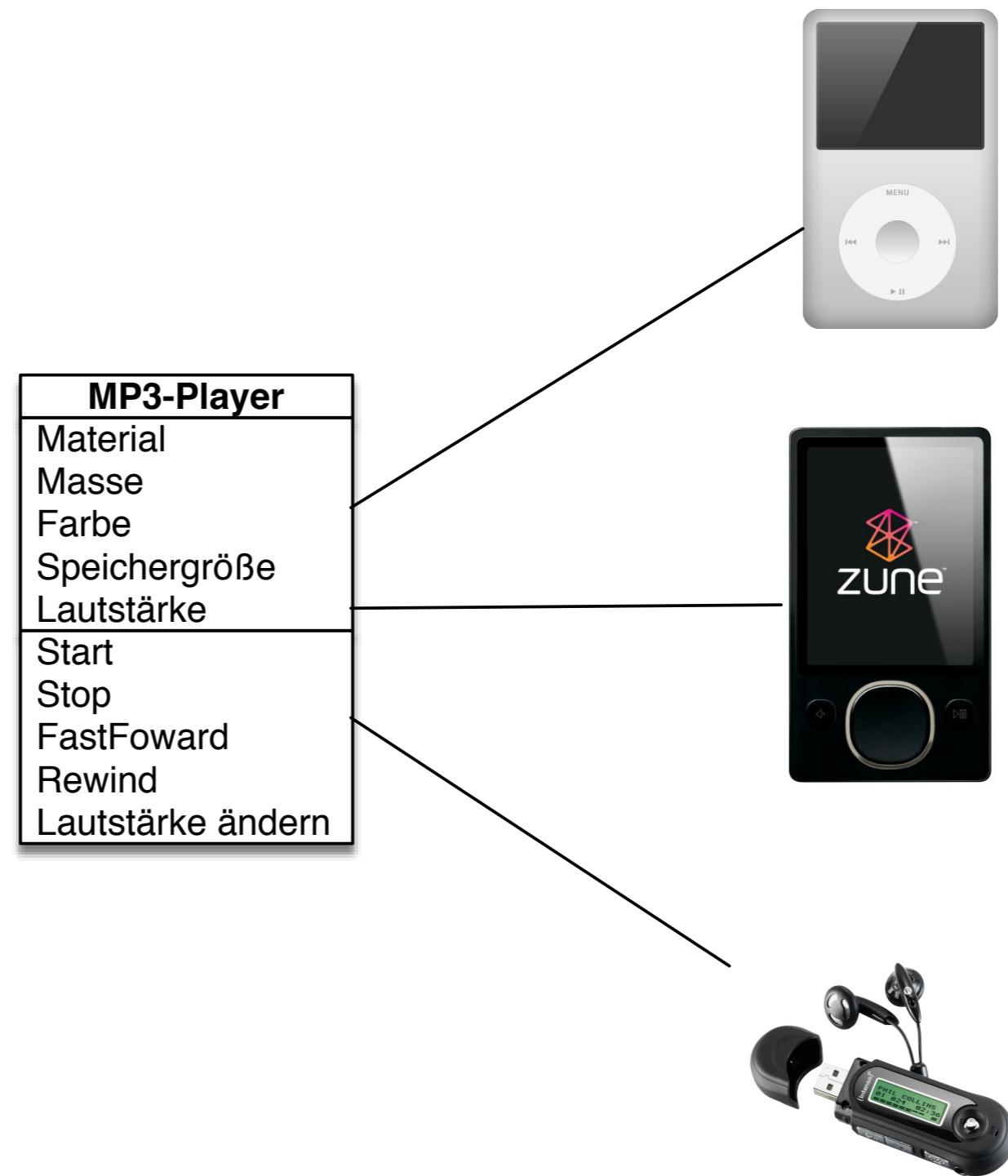
Was ist UML

- Man kann UML daher als **Notationssystem** beschreiben, das einen **Zeichenvorrat** aus Symbolen Zahlen, Sonderzeichen und Buchstaben vorhält, um **Elemente** eines Software-Systems und deren **Relationen** zu beschreiben.
- Obwohl nicht zwingend, ist der Hauptanwendungsfall von UML die Beschreibung **objektorientierter Systeme**.

Objektorientierung

- **Objektorientierte Programmierung (OOP)** ist eine Methode zur **Modularisierung von Programmen**.
- Hierbei werden Programme in **Einheiten** unterteilt, die **Objekte** genannt werden.
- Jedes **Objekt besitzt einen Zustand**, der durch dessen **Eigenschaften** (Objektattribute) beschrieben wird.
- Nur die im Objekt selbst vorhandenen **Funktionen** (Methoden genannt), können dessen **Daten manipulieren** und so den Zustand verändern.

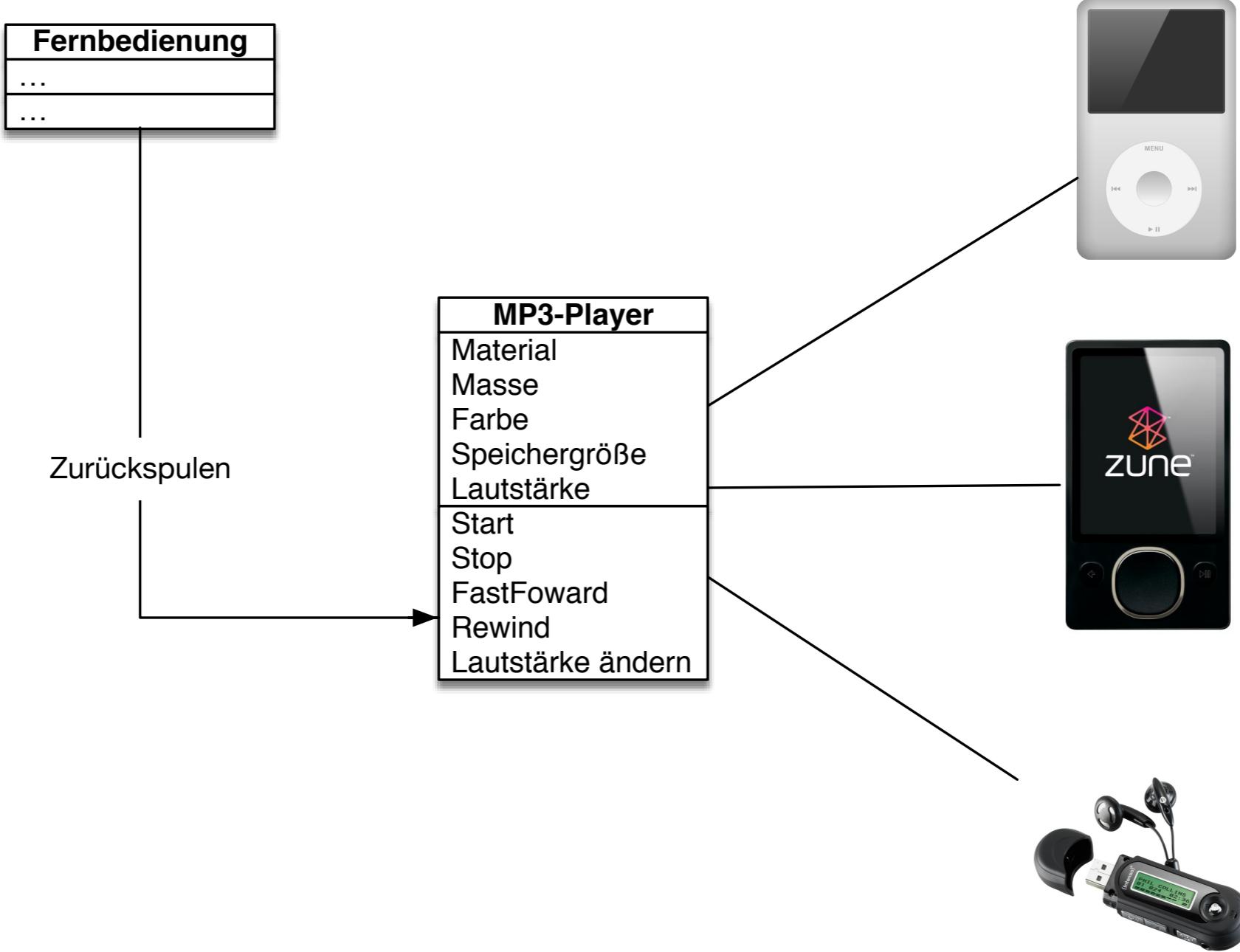
Objektorientierung



Objektorientierung

- Objekte können anderen Objekten **Botschaften** senden (indem sie deren Methoden aufrufen) und sie damit auffordern, ihren Zustand zu ändern.
- Letztendlich bleibt es aber dem Objekt selbst überlassen, ob es der Aufforderung nachkommt. Somit befindet sich das **Objekt** immer in einem **wohldefinierten, selbstkontrollierten Zustand**.

Objektorientierung



UML

Diagrammtypen

- UML 2.x definiert verschiedenste **Diagrammtypen**. Diese lassen sich unterteilen in:
 - **Strukturdiagramme**: wie sieht das System aus, aus welchen Komponenten besteht es, ...
 - **Verhaltensdiagramme**: wie verhält sich das System, welche Zustände gibt es, wie interagieren die Bestandteile des Systems
- Wir betrachten heute die drei Diagrammtypen **Klassendiagramm**, **Sequenzdiagramm** und **UseCase Diagramm**. Aber es gibt viel mehr...

UML

Diagrammtypen

Verhaltensdiagramme

Aktivitätsdiagramm

Use Case Diagram

Zustandsdiagramm

Interaktionsdiagramme

Sequenzdiagramm

Kommunikationsdiagramm

Zeitverlaufsdigramm

Strukturdiagramme

Klassendiagramm

Verteilungsdiagramm

Objektdiagramm

Komponentendiagramm

Paketdiagramm

Kompositionssstrukturdiagramm

Profildiagramm

UML

Diagrammtypen

- In UML können **Diagramme geschachtelt** werden.
- So entsteht eine **Hierarchie von Diagrammen**, bei der Details eines Diagramms in weiteren Diagrammen näher betrachtet werden.
 - Ein Diagramm kann Verhalten, das in einem anderen Diagramm modelliert ist, aufrufen.
 - Es ist möglich **Vererbungsbeziehungen** zwischen Diagrammen aufzubauen.
 - Diagramme, die Verhalten modellieren, können mit **Ein- und Ausgabeparametern** versehen werden.

UML

Begrifflichkeiten

Artefakt

- In der UML Spezifikation werden **Ergebnisse oder Produkte eines Entwicklungsprozesses** als Artefakt (artifact) bezeichnet. Ein Artefakt ist also z. B. eine Klasse, ein Diagramm, eine Beschreibung eines Elementes.

Notiz, Anmerkung

- Jedes UML Element kann mit einer Notiz versehen werden. Die Notiz wird über eine gestrichelte Linie mit dem Element verbunden, das beschrieben wird.

UML

Begrifflichkeiten

Szenario

- Ein Szenario ist eine Szene eines Modells, also **ein Ausschnitt, für den nur bestimmte Aspekte des Gesamtsystems dargestellt werden.**

Constraint - Einschränkung

- Eine Einschränkung ist eine **Bedingung, die bei der Implementierung eines Systems erfüllt sein muss.** Constraints werden in geschweiften Klammern geschrieben und mit einer gestrichelten Linie mit dem Element verbunden, auf das sie sich beziehen.

UML

Anwendungsfalldiagramme

Anwendungsfälle

- Ein alternativer Ansatz zur **Ermittlung von Systemanforderungen** sind **Anwendungsfälle (Use Cases)**
- Use Cases geben Auskunft darüber, was ein geplantes **System aus Sichtweise der Benutzer** leisten soll und stellen damit die **Menge der Systemfunktionen** dar.
- Diese Funktionen müssen **im weiteren Projektverlauf genauer zu beschreiben** werden.

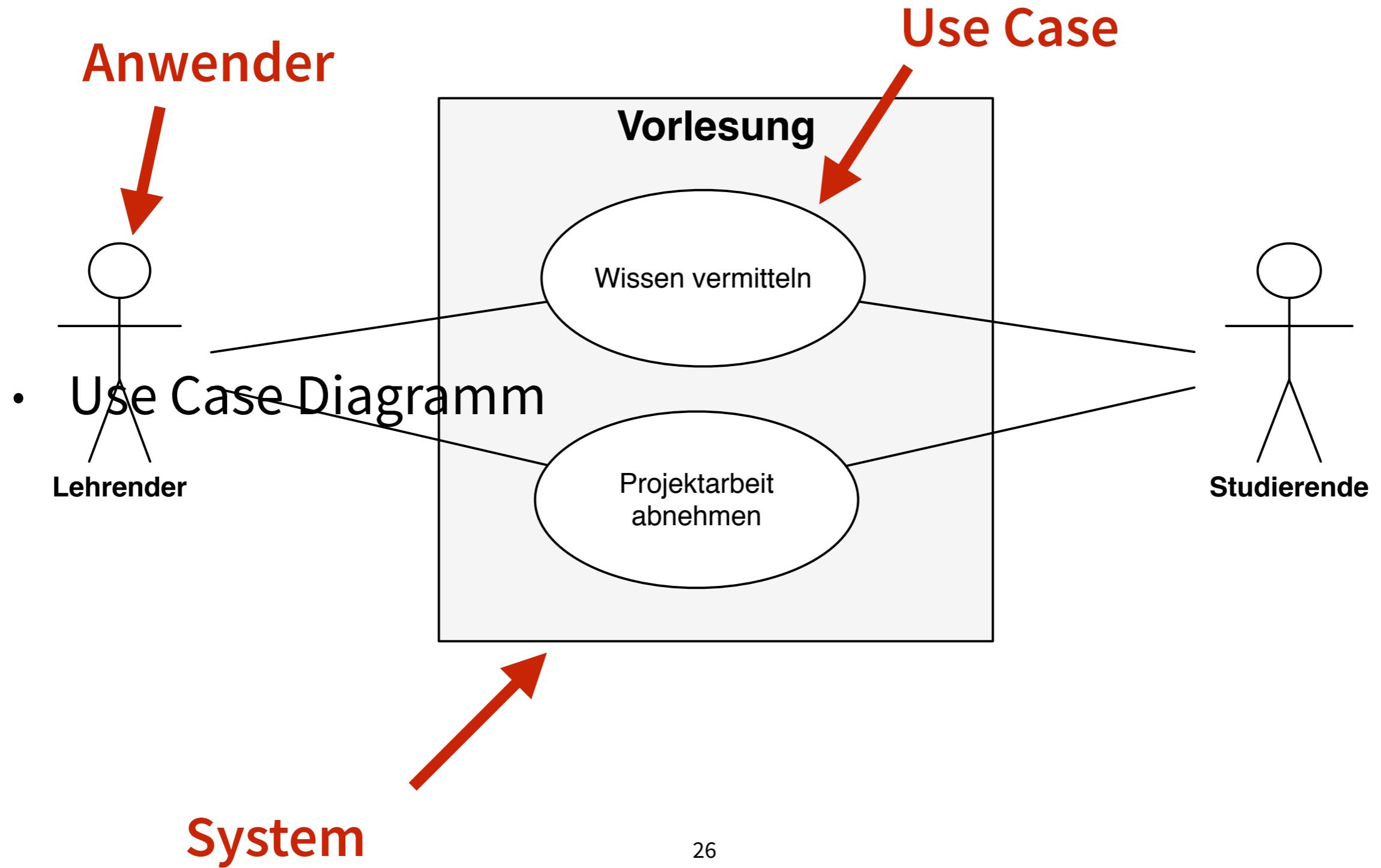
Use Case Identifikation

- Folgende **Fragen** sind zur Identifikation von Use Cases hilfreich:
 - Für **welchen Zweck** soll das System eingesetzt werden?
 - **Wofür** will der spätere Benutzer es einsetzen?
 - Durch welches **externe Ereignis** wird ein Use Case angestoßen?
 - Welches **Ergebnis** liefert ein Use Case dem späteren Nutzer?

Use Case Diagramm in UML

- **Sammlung und Visualisierung von Anwendungsfällen erfolgt mit Hilfe des UML Use Case Diagramms**
- Hierbei werden Anwendungsfälle und Akteure mit hohem Abstrahierungsgrad durch Linien einander zugeordnet

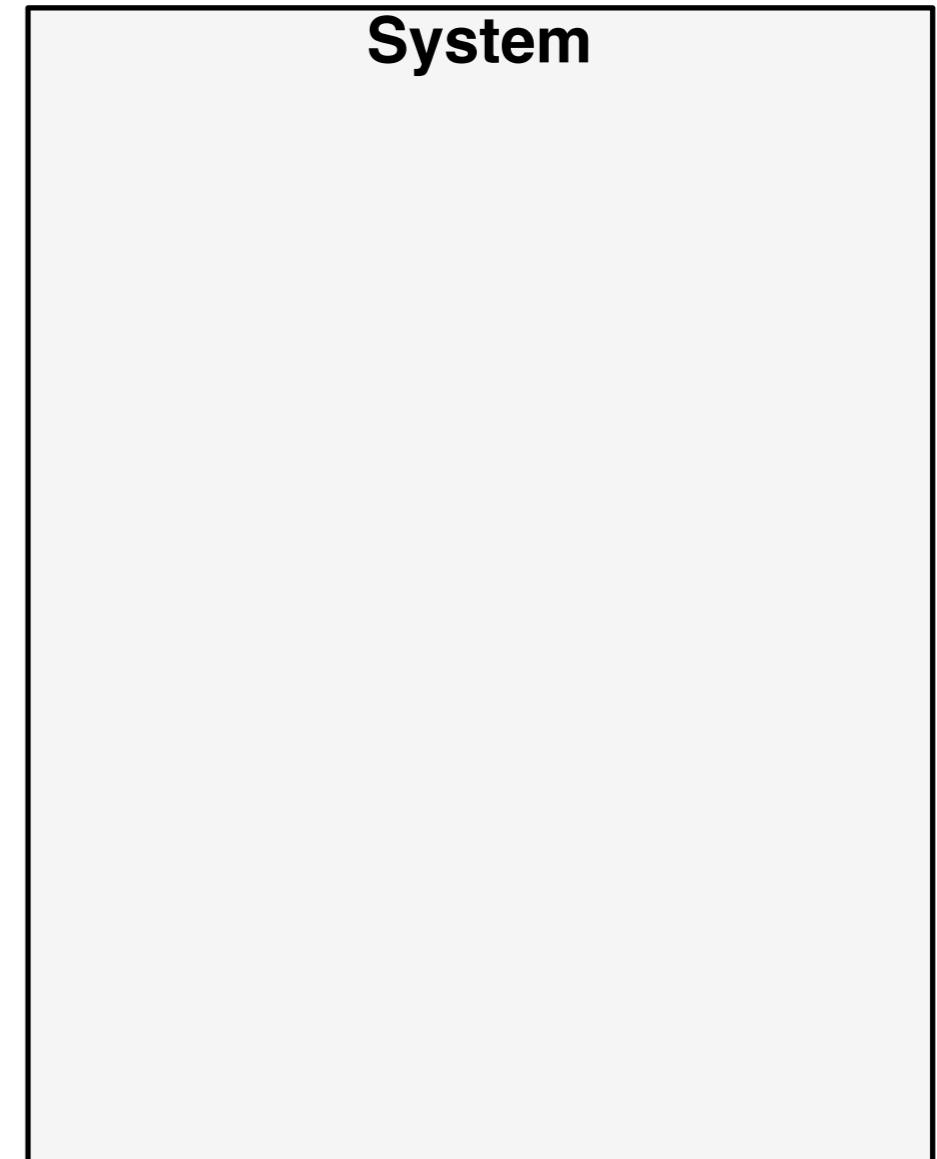
Use Case Diagramm in UML



Use Case Diagramm in UML

System

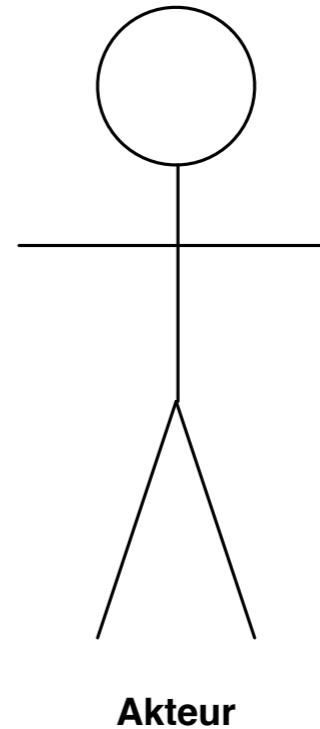
- Das **Rechteck** stellt das **geplante System** dar. Das System erhält einen Namen.
- Ein Use Case Diagramm kann auch **mehrere Systeme** enthalten, die ineinander geschachtelt sein dürfen. Dadurch kann ein System in Teilsysteme gegliedert werden.



Use Case Diagramm in UML

Akteur

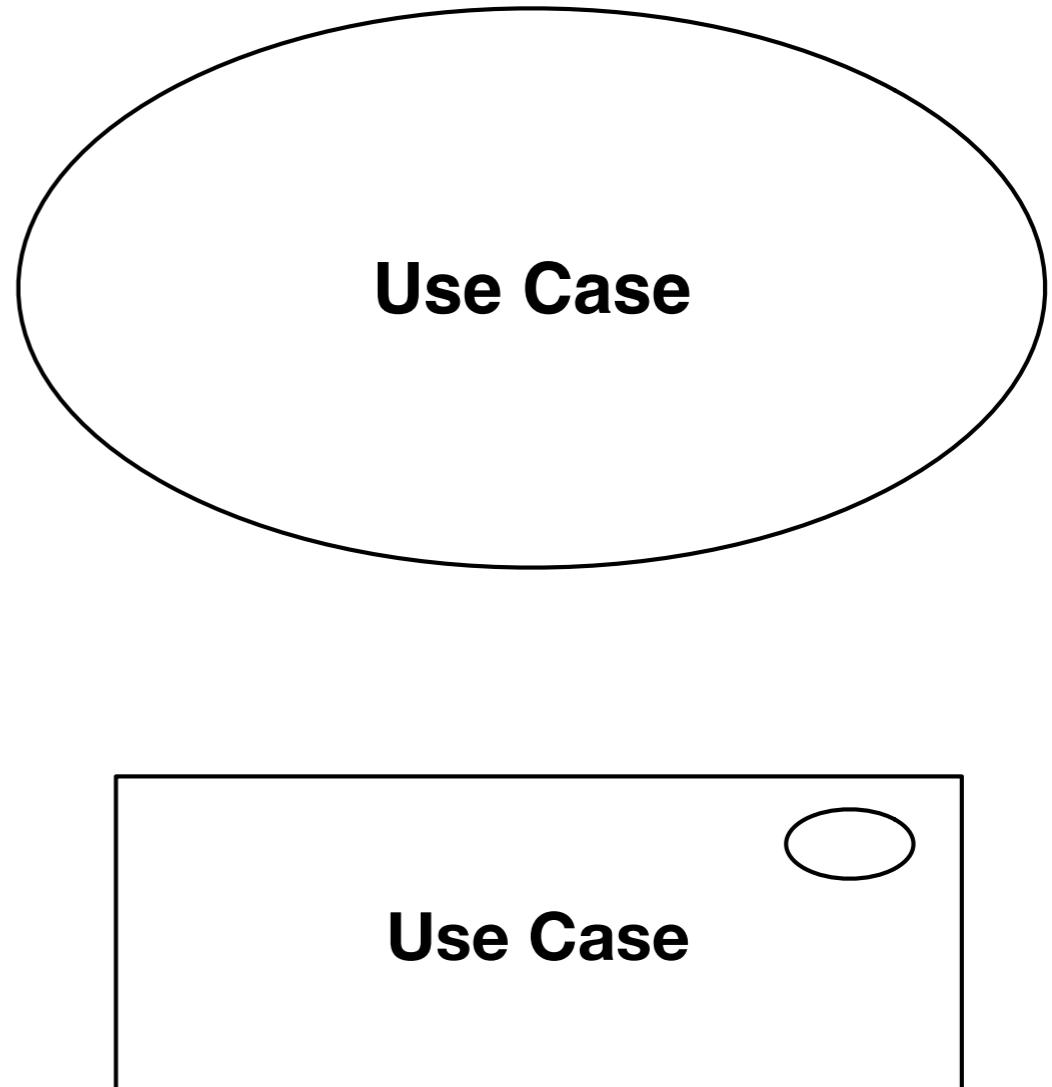
- **Person**, die auf das System zugreift oder ein anderes **System**, das mit dem geplanten System kommuniziert.
- **Gehört nicht zum geplanten System!**
- Der Akteur kann als **Strichmännchen** oder als **Kästchen mit dem Stereotyp <<actor>>** dargestellt werden.



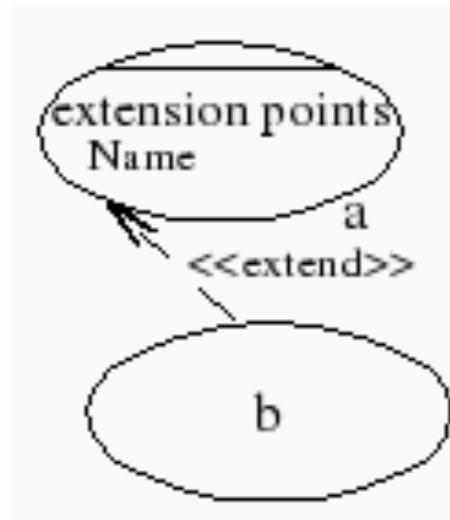
Use Case Diagramm in UML

Use Case

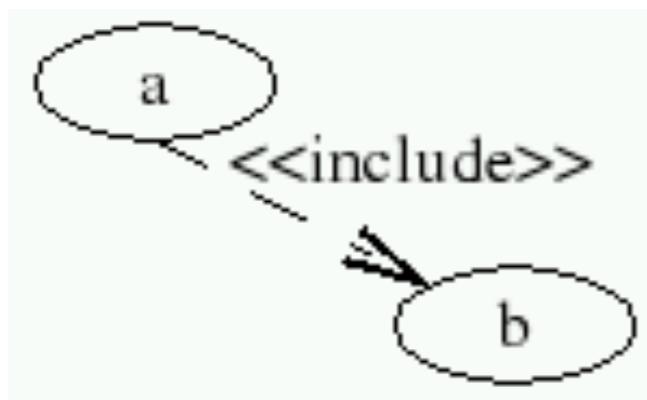
- Ein Anwendungsfall ist ein in sich **abgeschlossener Vorgang**, der für einen oder mehrere Akteure ein **beobachtbares Ergebnis** liefert.
- Er beschreibt aus Sicht der Akteure welche Leistungen das System für den Anwender zur Verfügung stellt.
- Kann auch als Rechteck mit einer Ellipse dargestellt werden.



Use Case Diagramm in UML

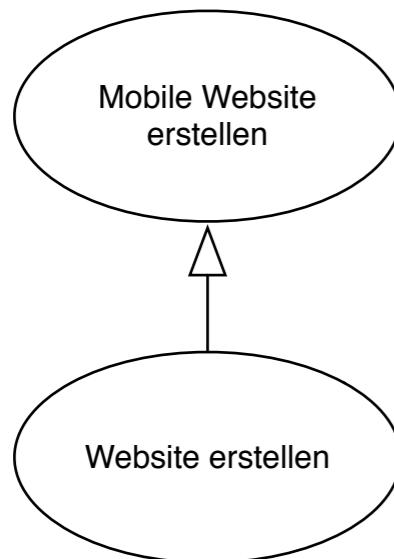


- Durch die **extend Beziehung** beschreibt die Erweiterung des Verhaltens eines Use Cases durch einen anderen.

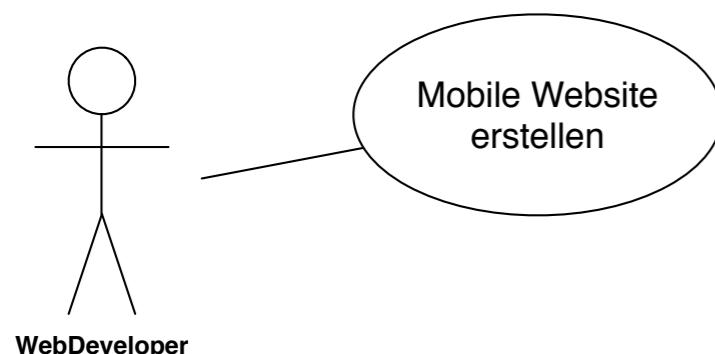


- Die **include Beziehung** definiert einen Use Case, der die Funktionalität, die ein anderer Use Case zur Verfügung stellt, importiert. Der Use Case a importiert die Funktionalität des Use Case b.

Use Case Diagramm in UML



- Die **Verebungsbeziehung** kann verwendet werden um Verhalten zwischen Use Cases zu vererben oder Vererbungen zwischen Akteuren aufzubauen.



- Eine **Assoziation** zwischen einem Akteur und einem Use Case beschreibt den **Zugriff des Akteurs auf die Funktionalität** des Systems bzw. die **Antwort des Systems an einen Akteur**.

Use Case Spezifikation

- Die **Textstruktur** eines Use Cases enthält üblicherweise folgende Abschnitte:
 - **Name** des Use Case
 - **Auslösendes Ereignis**
 - **Verhalten im Normalfall**
 - **Verhalten im Fehlerfall**
 - **Ergebnis**

Use Case

Textstruktur

- **Name des Use Case:** Wissen vermitteln
- **Auslösendes Ereignis:** Lehrender und Studierende treffen sich im Rahmen einer Vorlesung
- **Verhalten im Normalfall:** Nachdem der Lehrende einen thematischen Kontext hergestellt hat, gibt er Impulse, um die Studierenden zu aktivieren und die Vorlesung zunehmend zu einer Diskussion werden zu lassen.
- **Verhalten im Fehlerfall:** Gelingt es dem Lehrenden nicht mit den Studierenden in einen lebendigen Austausch zu treten, langweilen sich die Studierenden 90 Minuten lang zu Tode und der Lehrende ist frustriert.
- **Ergebnis:** Die Studierenden erwerben neue Kompetenzen und werden zum Denken angeregt.

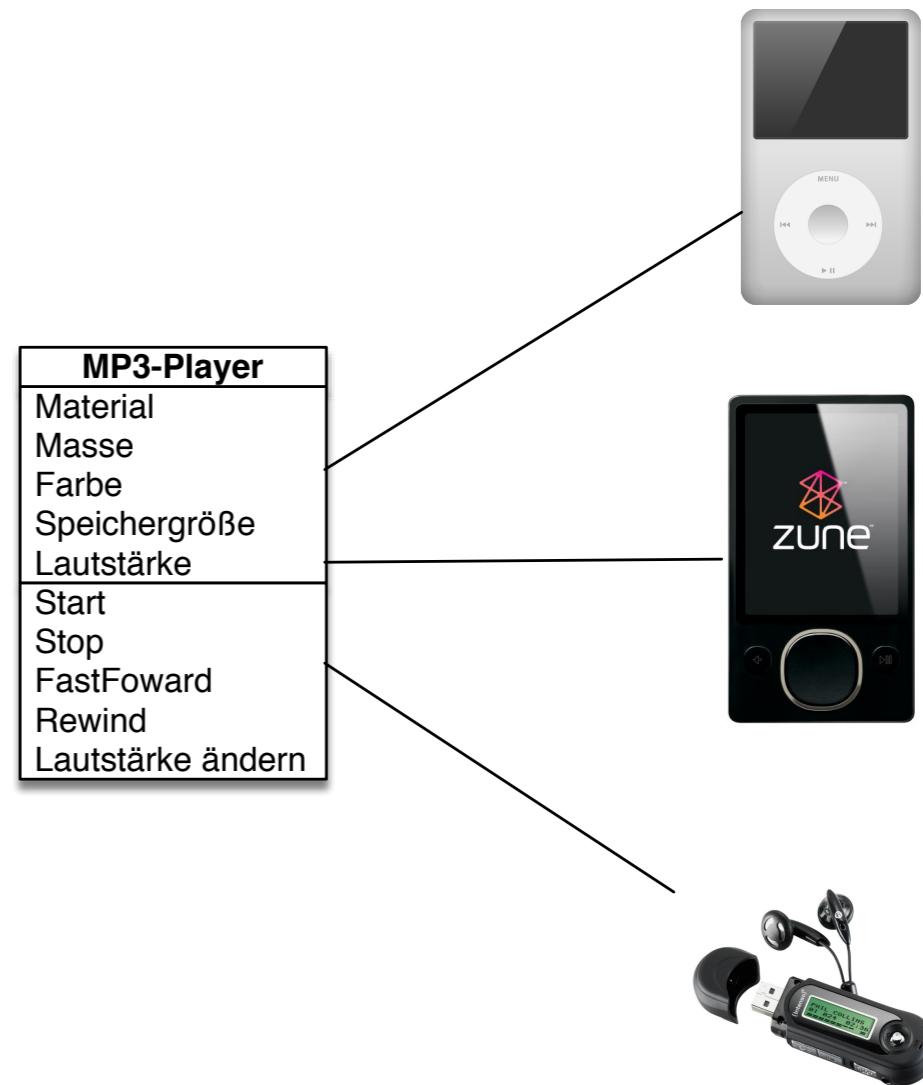
UML

Klassendiagramme

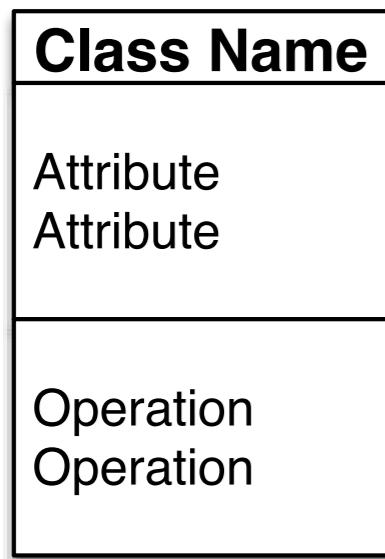
UML

Klassendiagramm

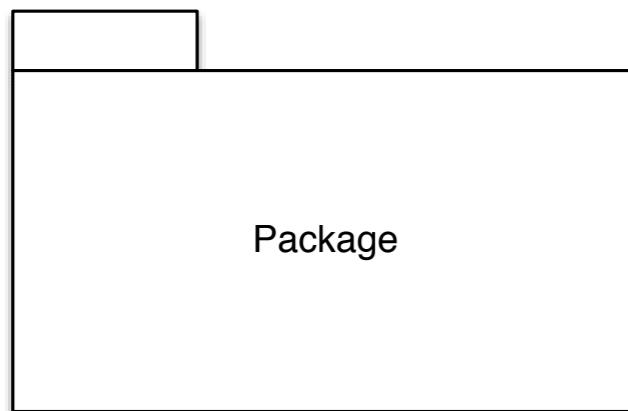
- Eine Klasse ist eine **Zusammenfassung gleichartiger Objekte mit unterschiedlicher Ausprägung.**
- Ein Klassendiagramm dient dazu diese gleichartigen Objekte zu identifizieren und ihre Beziehung zueinander darzustellen.



Bestandteile des Klassendiagramms

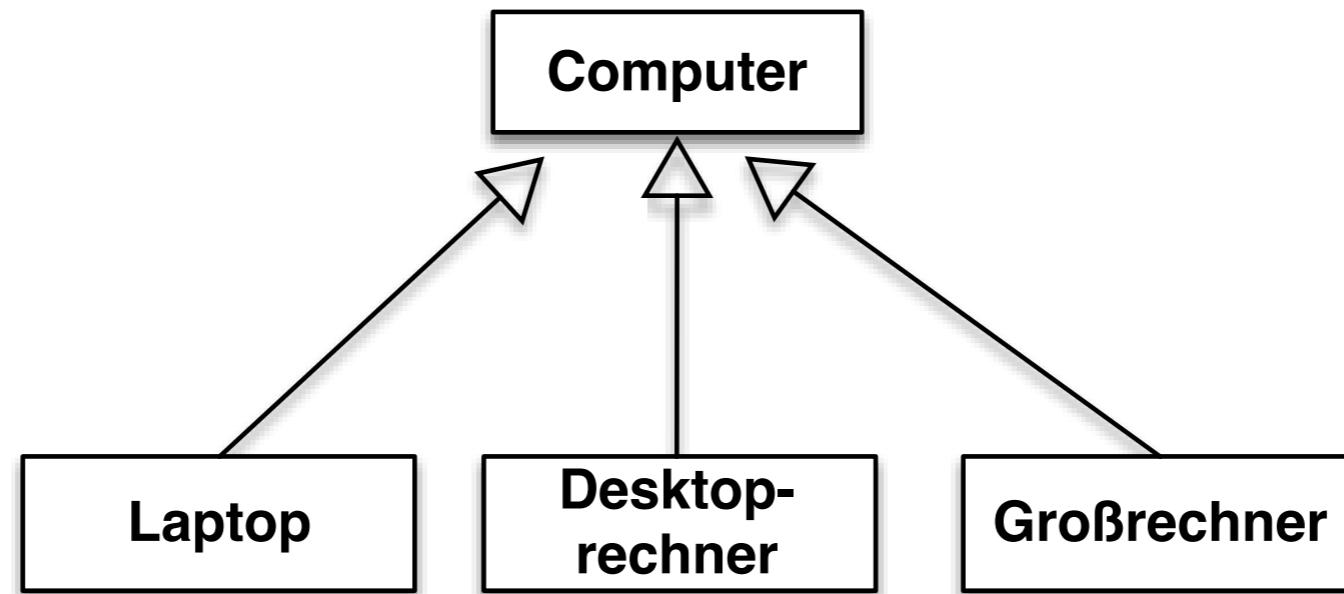


- Eine **Klasse** besteht mindestens aus einem **Bezeichner** und kann zusätzlich die **Eigenschaften**, die diese Klasse auszeichnen sowie die **Operationen** (Funktionen), die diese Klasse bereitstellt, enthalten.



- Ein **Paket** (Package) dient als **Container** von **Modellelementen** mit denen das Gesamtmodell in überschaubarere Einheiten untergliedert werden kann.

Bestandteile des Klassendiagramms



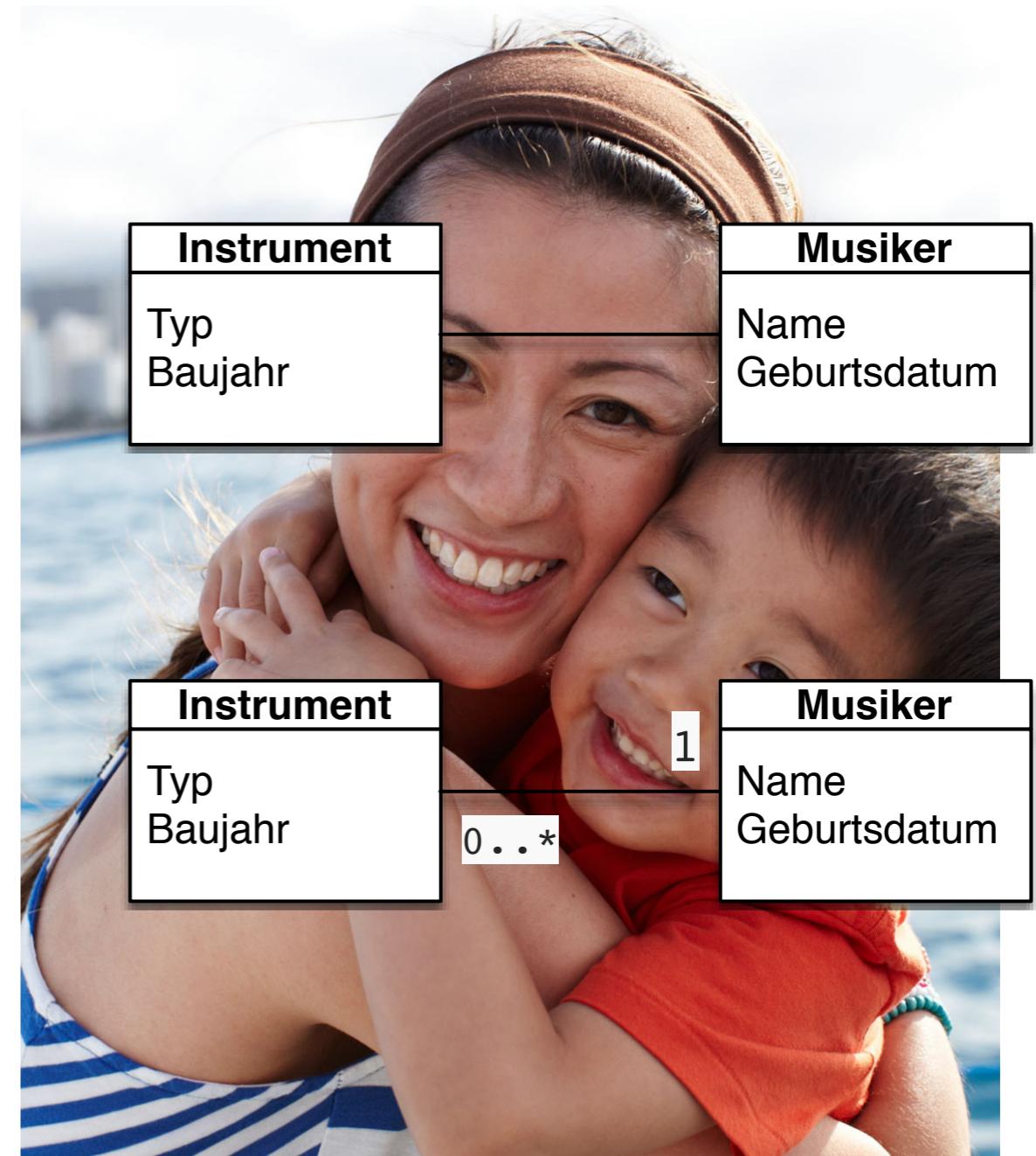
Vererbung

- Wichtiges **Abstraktionsprinzip** zur hierarchischen Gliederung von Klassen (Generalisierung und Spezialisierung).
- Attribute und Operationen der Oberklasse sind auch den Unterklassen zugänglich.

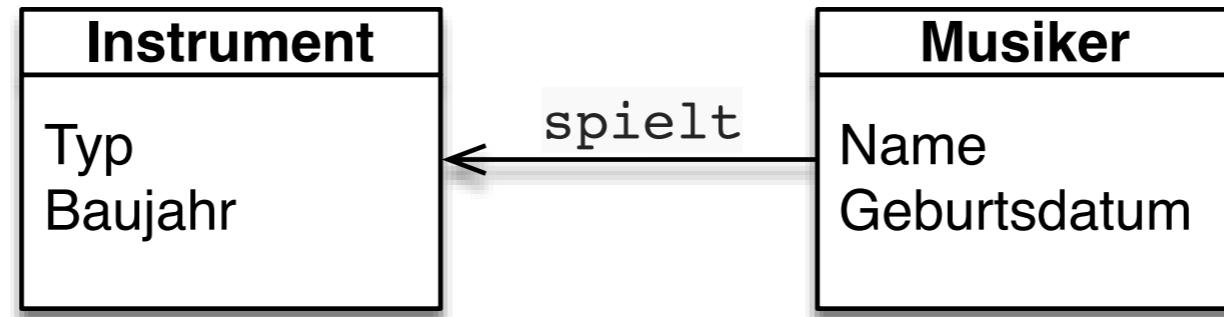
Bestandteile des Klassendiagramms

Assoziationen

- Eine Relation beschreibt die **Beziehung zwischen** zwei oder mehr **Klassen**.
- Die **Multiplizität einer Assoziation** gibt an, mit wieviel Objekten der gegenüberliegenden Klasse ein Objekt assoziiert sein kann.



Bestandteile des Klassendiagramms



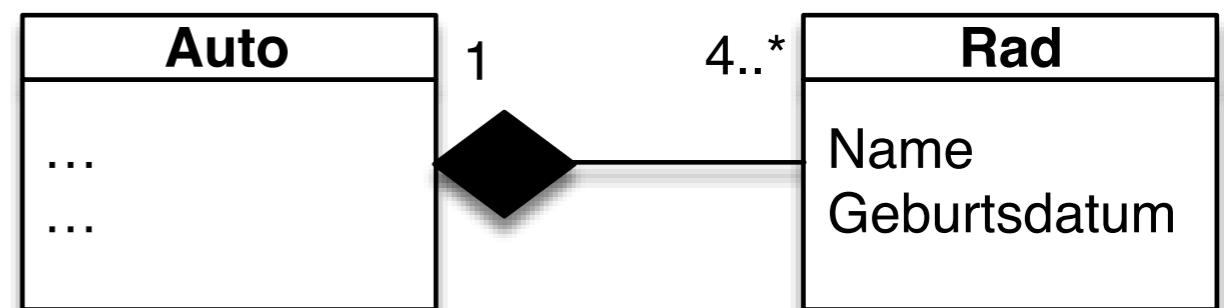
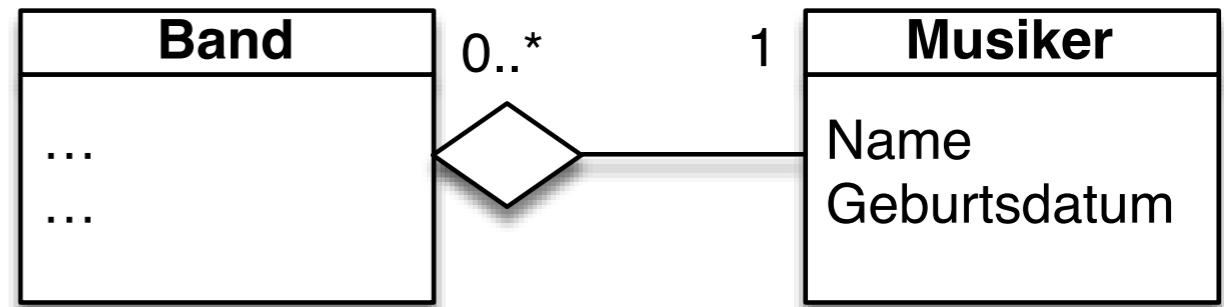
Gerichtete Assoziationen

- Gerichtete Assoziationen sind Beziehungen, die eine **bestimmte Zugriffsrichtung** ausdrücken.
- Dargestellt wird die Navigation durch eine **offene Pfeilspitze**, die die zugelassene Navigationsrichtung angibt.
- Die Assoziation sollte einen **Namen** enthalten.

Bestandteile des Klassendiagramms

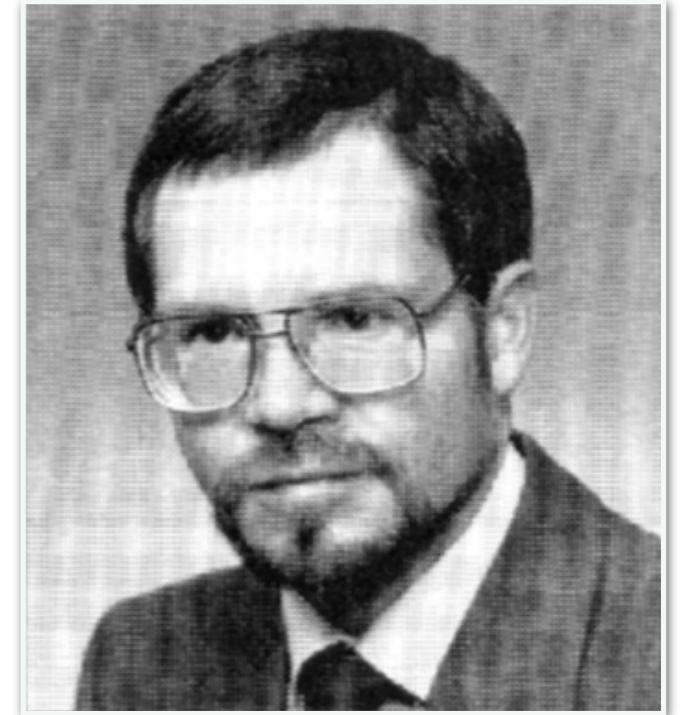
Komposition und Aggregation

- beschreibt die **Beziehung** zwischen einem **Ganzen** und seinen **Teilen**.
- Komposition ist ein **Spezialfall** der Aggregation: die Teile stehen in Existenzabhängigkeit zueinander.



Identifikation von Klassen

- Substantivmethode
 - Alle Substantive aus den Anforderungen werden als potenzielle Klassenkandidaten markiert.
 - Redundante Begriffe werden eliminiert.
 - Begriffe der technische Realisierung werden ausgeklammert.



James Rumbaugh

Das Ergebnis ist eine Menge von Begriffen, die in der Projektdomäne als Klassen eine Bedeutung haben.

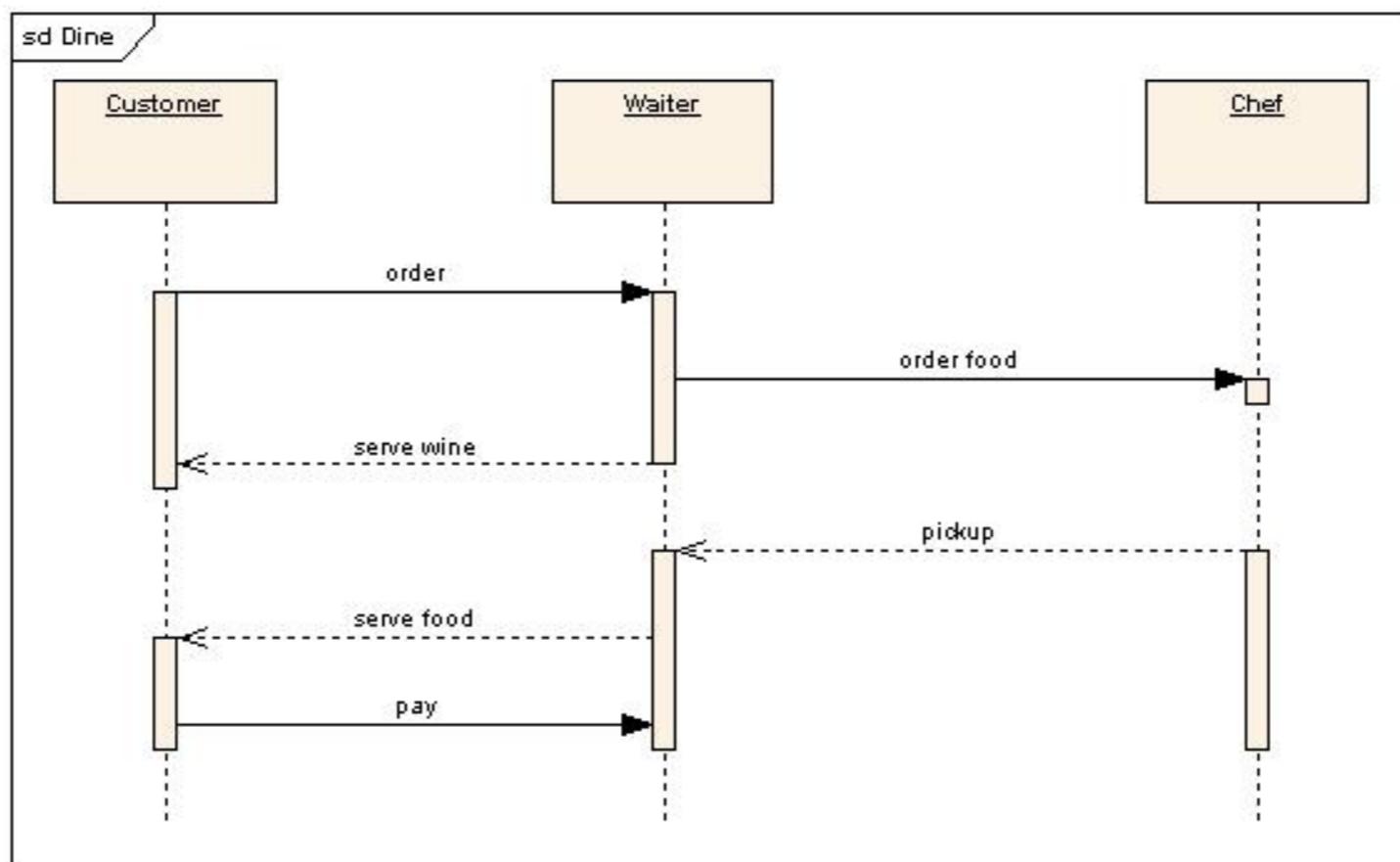
Identifikation von Klassen

- **CRC-Methode**
- für jede potenzielle Klasse werden drei Fragestellungen beantwortet:
 - class:
Name der Klasse
 - responsibility:
Zu welchem Zweck dient sie im Gesamtsystem
 - collaboration:
Mit welchen Klassen soll diese Klasse zusammenarbeiten

UML

Sequenzdiagramme

UML Sequenzdiagramm

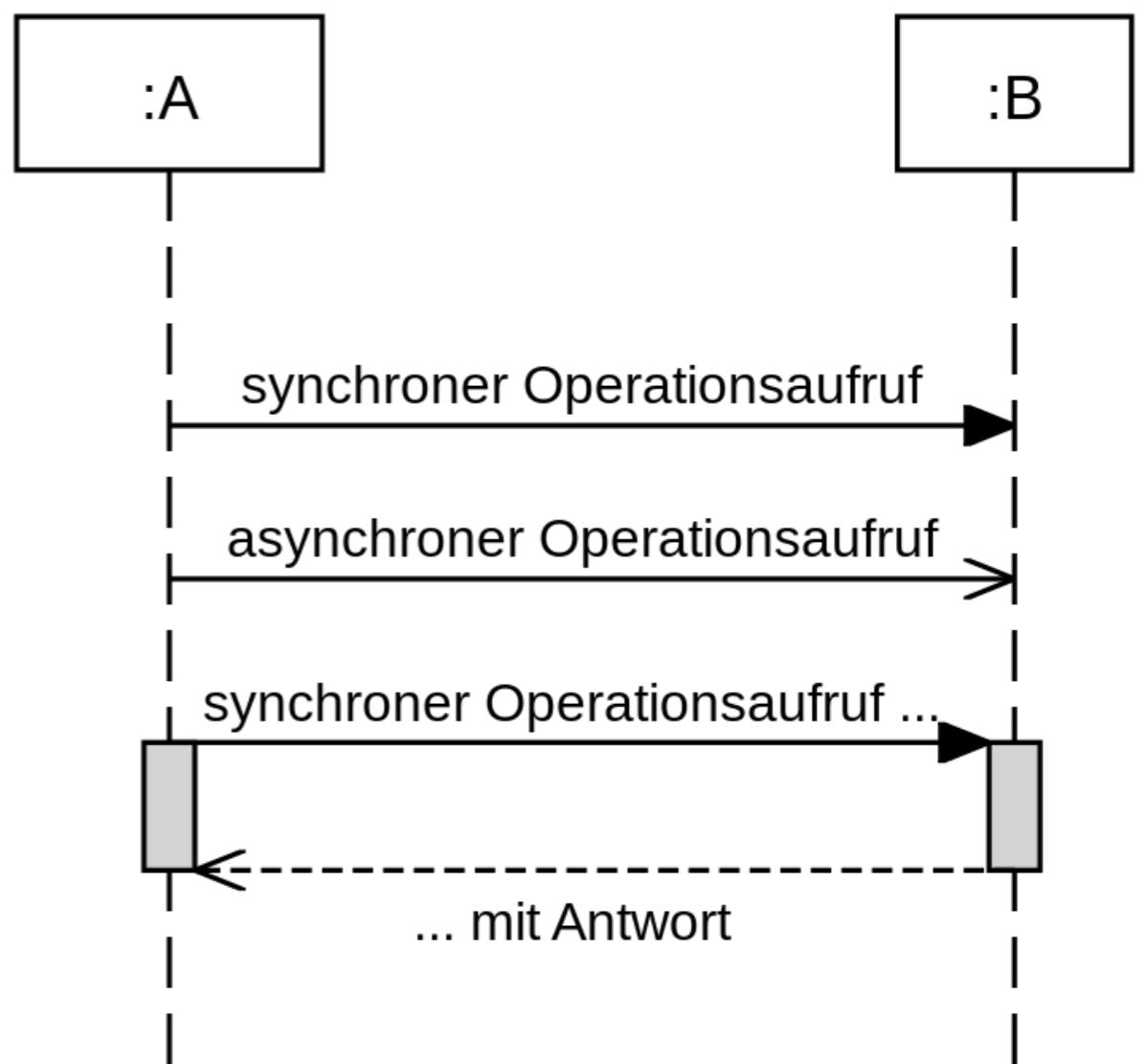


*„[A] Sequence diagram is a **time dependent view** of the interaction between objects to accomplish a behavioral goal of the system.“*

UML Sequenzdiagramm

Bestandteile

- **Objekte**
(Kommunikationspartner)
- **Lebenslinien**
- **Nachrichten (Pfeil)**
 - Synchron (gefüllte Pfeilspitze)
 - Asynchron (offene Pfeilspitze)
- **Aktivierungsbalken**



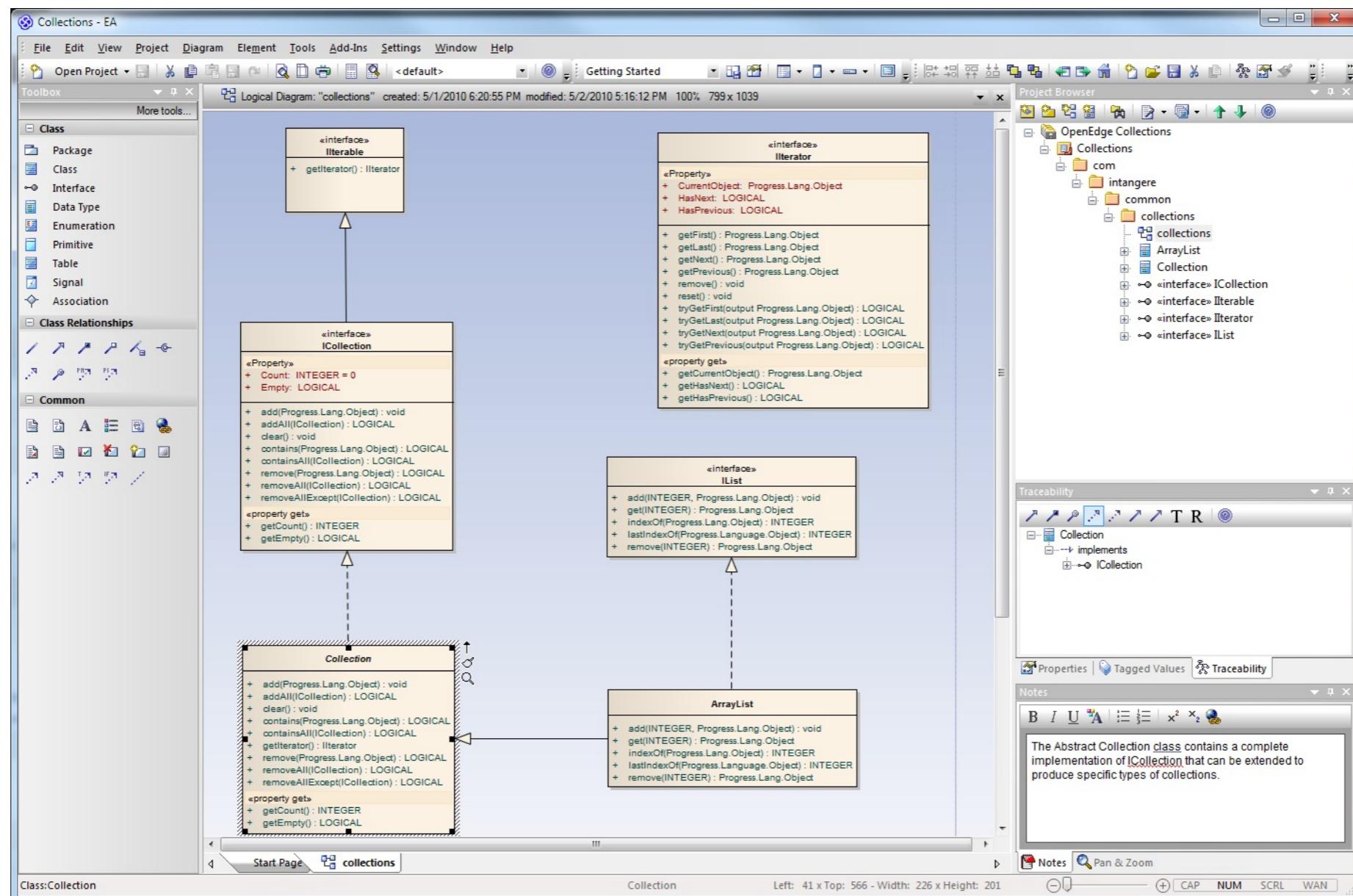
UML Sequenzdiagramm

- Ein Sequenzdiagramm stellt **einen Weg durch einen Entscheidungsbaum** innerhalb eines Systemablaufes dar.
- Dabei spezifiziert es die **zeitliche Ordnung von Ereignisauftritten**:
 - Welche Ereignisse müssen vor und welche nach einem bestimmten Ereignisauftritt auftreten?
 - Auch hier gilt es einen dem zu beschreibenden **Szenario** adäquaten **Detaillierungsgrad** zu finden.

UML Werkzeuge

UML

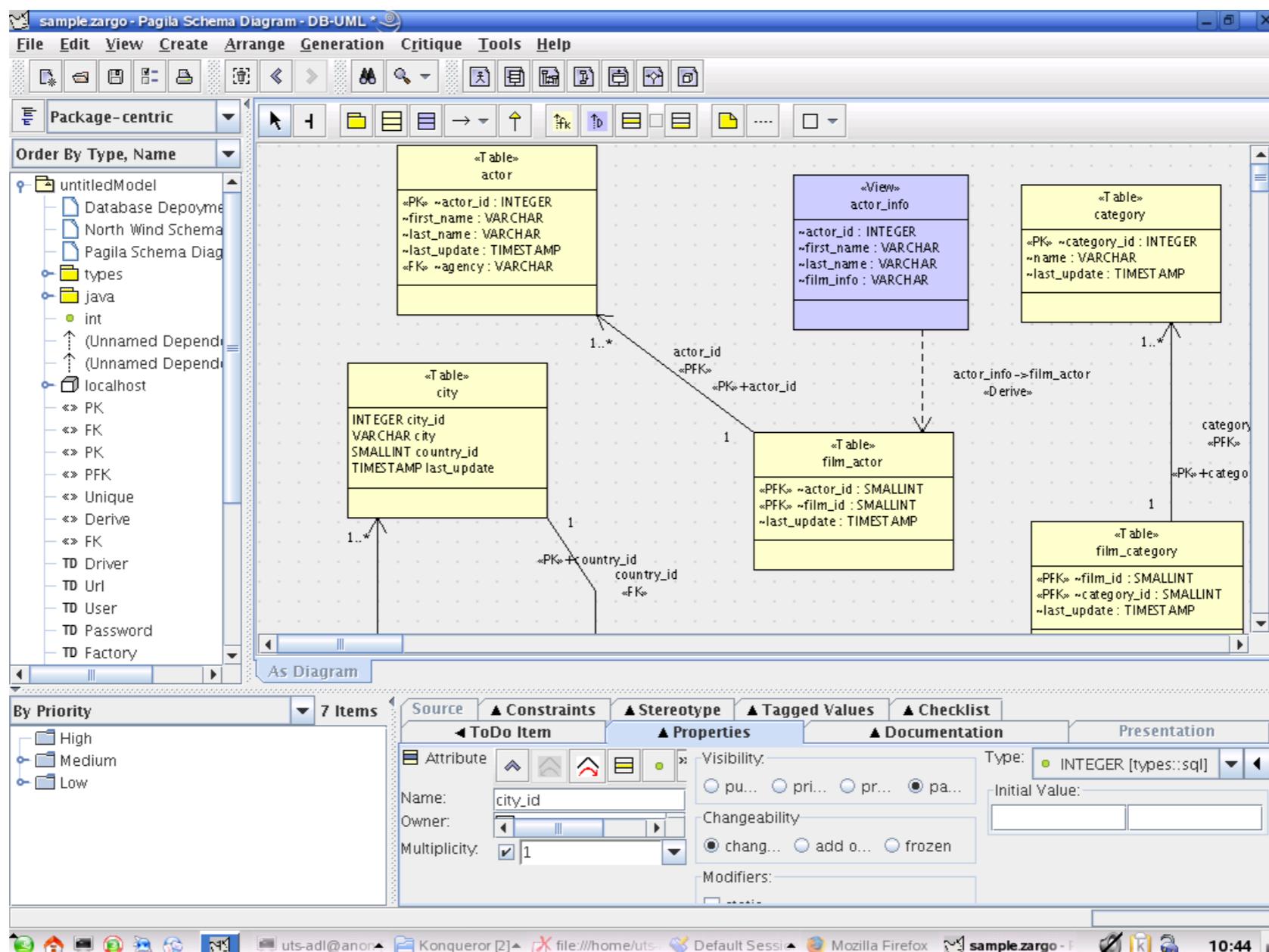
Werkzeuge



Enterprise Architect (SparxSystems)

UML

Werkzeuge

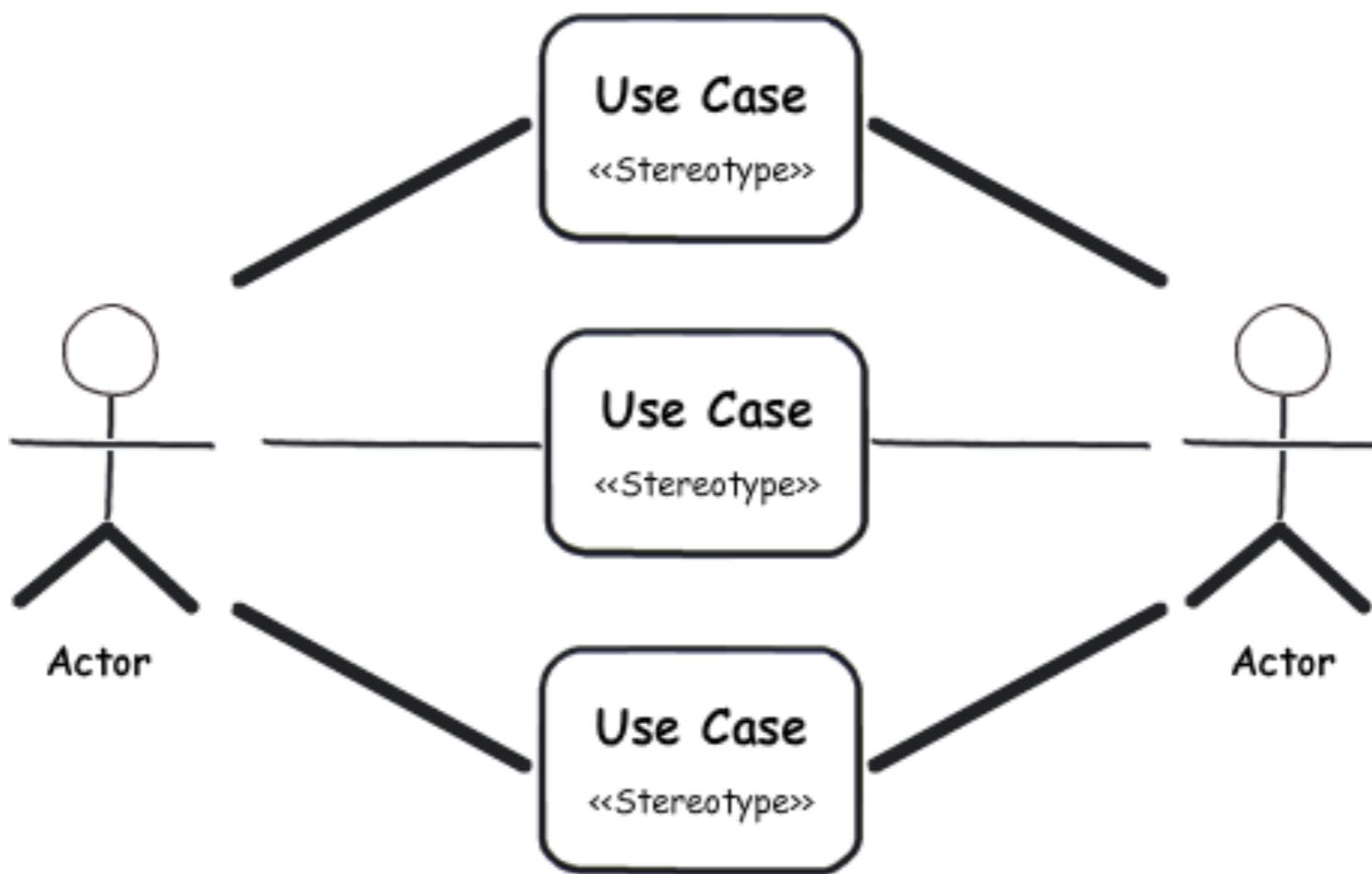


ArgoUML (Tigris)

UML

Werkzeuge

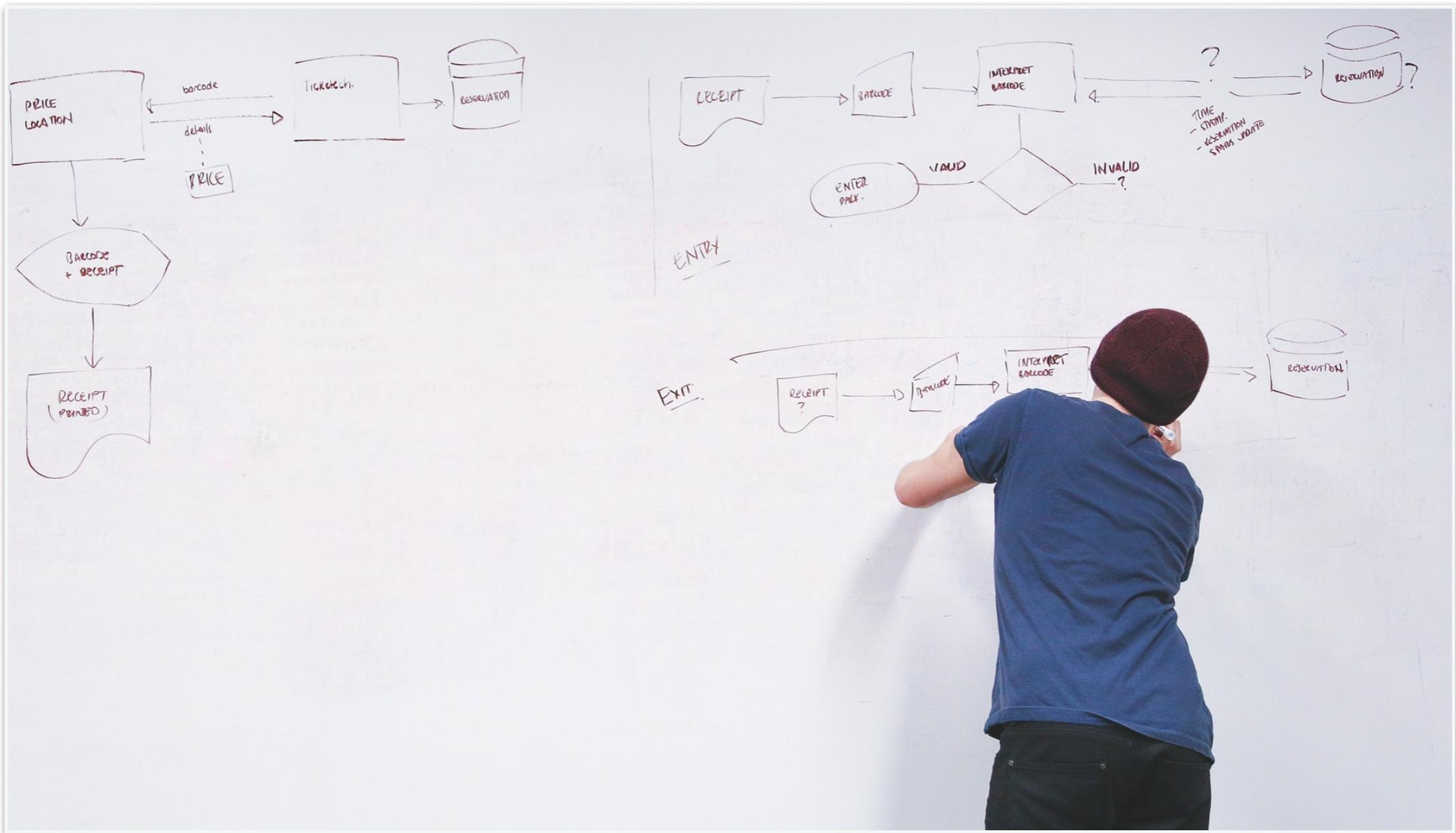
Three Use Cases



Mock4U (Balsamiq)

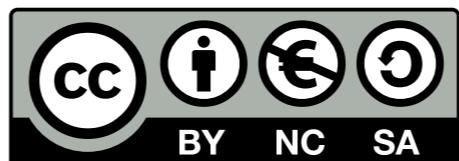
UML

Werkzeuge



Whiteboards, Kaffee und eine lebendige Diskussion

Vielen Dank.



Dieses Werk ist lizenziert unter einer Creative Commons 4.0 International Lizenz mit folgenden Eigenschaften:

- Namensnennung
- Nicht-kommerzielle Nutzung
- Weitergabe unter gleichen Bedingungen.

Quellen

- Objektorientierung
<http://www.itwissen.info/definition/lexikon/Objektorientierte-Programmierung-OOP-object-oriented-programming.html>
- https://www.unibw.de/inf4/professuren/geoinformatik/lehre/skripten/skripte/skripten_ft_09/umlvorlesung2009.pdf
- http://www.tutorialspoint.com/uml/uml_2_overview.htm
- <http://agilemodeling.com>