Grundlagen der UML

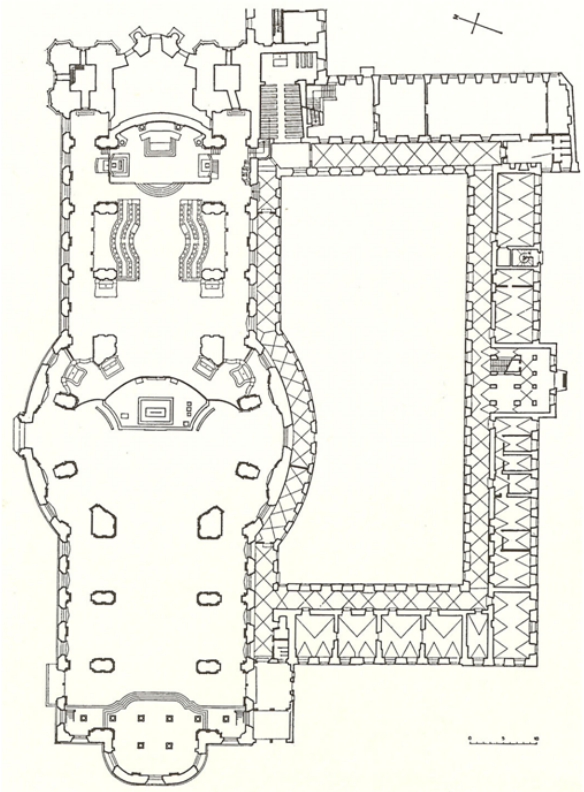
Ein praktischer Einstieg

Dipl.Ing.(FH) Karsten Hain

9. November 2011

Inhaltsverzeichnis

Softwareentwicklung



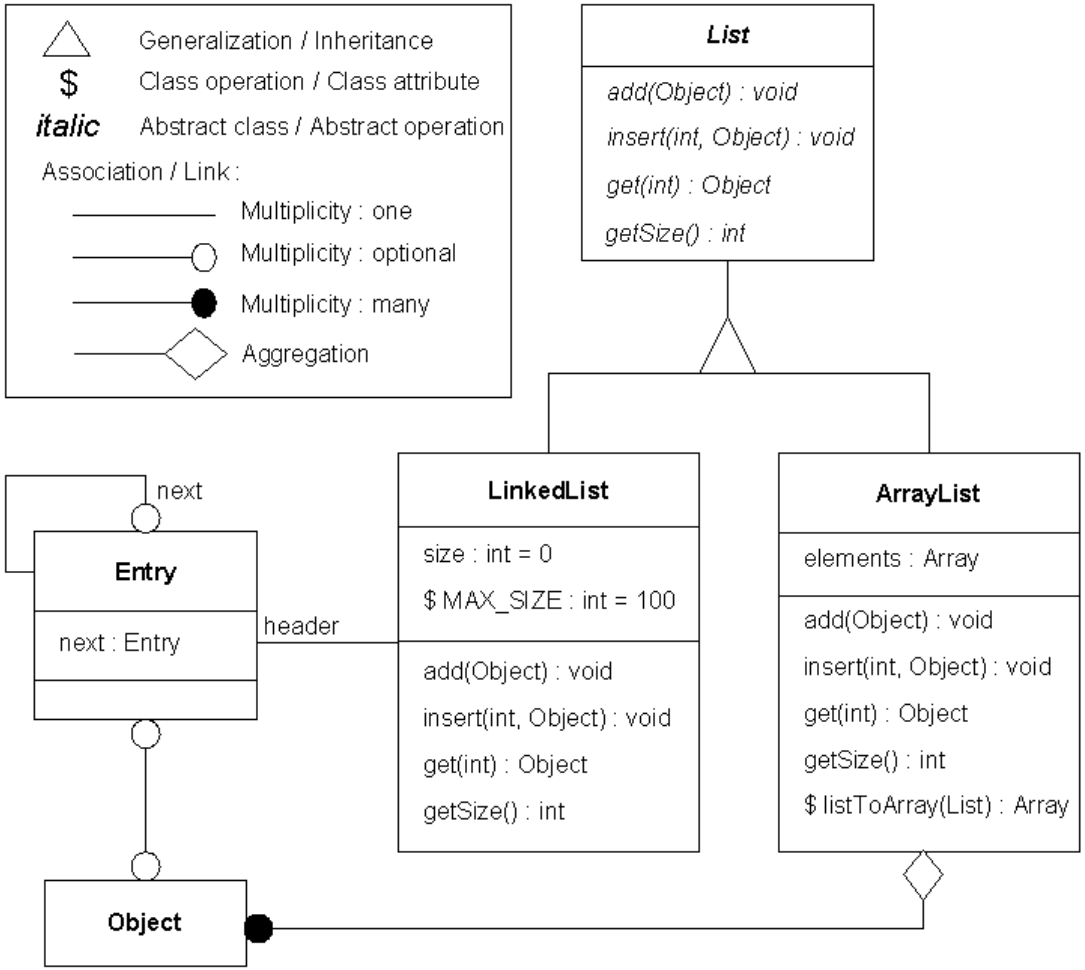
Stiftskirche St. Gallen Grundriss um 1760 - Stiftskirche St. Gallen (Foto: Petar Marjanovic)

Geschichte der Objektorientierung

- 1967 - Objektorientiertes Programmierparadigma (OOP)
- 1979 - C++ von Bjarne Stroustrup bei AT&T als Erweiterung von C
- 1980er - Entwicklung von Methoden zur OO-Modellierung
- Diagramme dienten zur Darstellung der Modelle

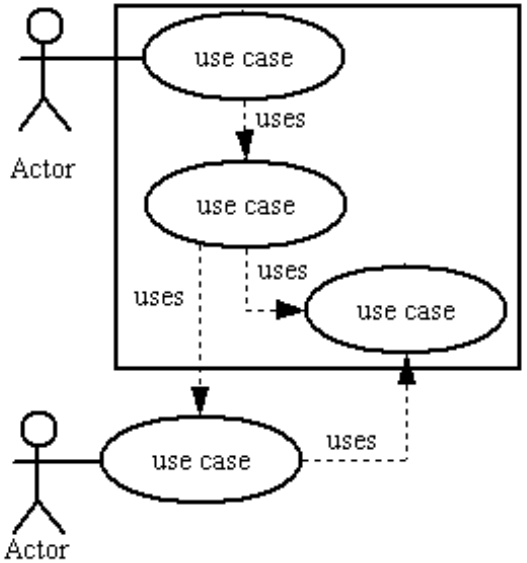
Der kommerzielle Erfolg der OOP bzw. OO-Modellierung führte zu dem Wunsch der Industrie nach Standardisierung!

1991 - OMT - Object Modeling Technique (James Rumbaugh)



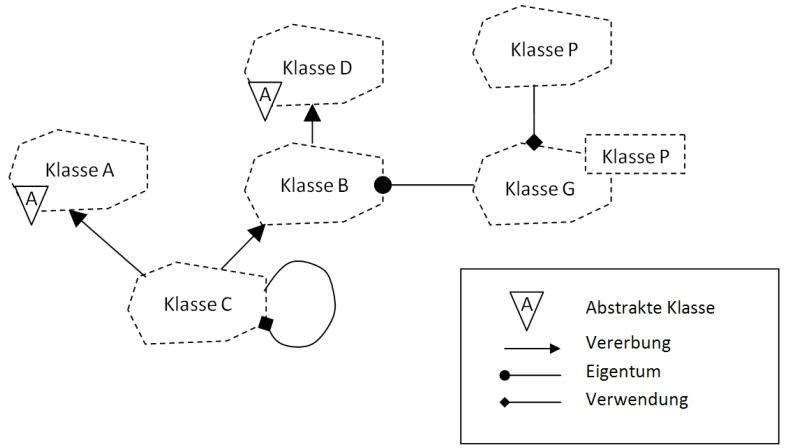
1992 - OOSE - Object Oriented Software Engineering (Ivar Jacobson)

- erste Methode, welche Anwendungsfälle verwendet



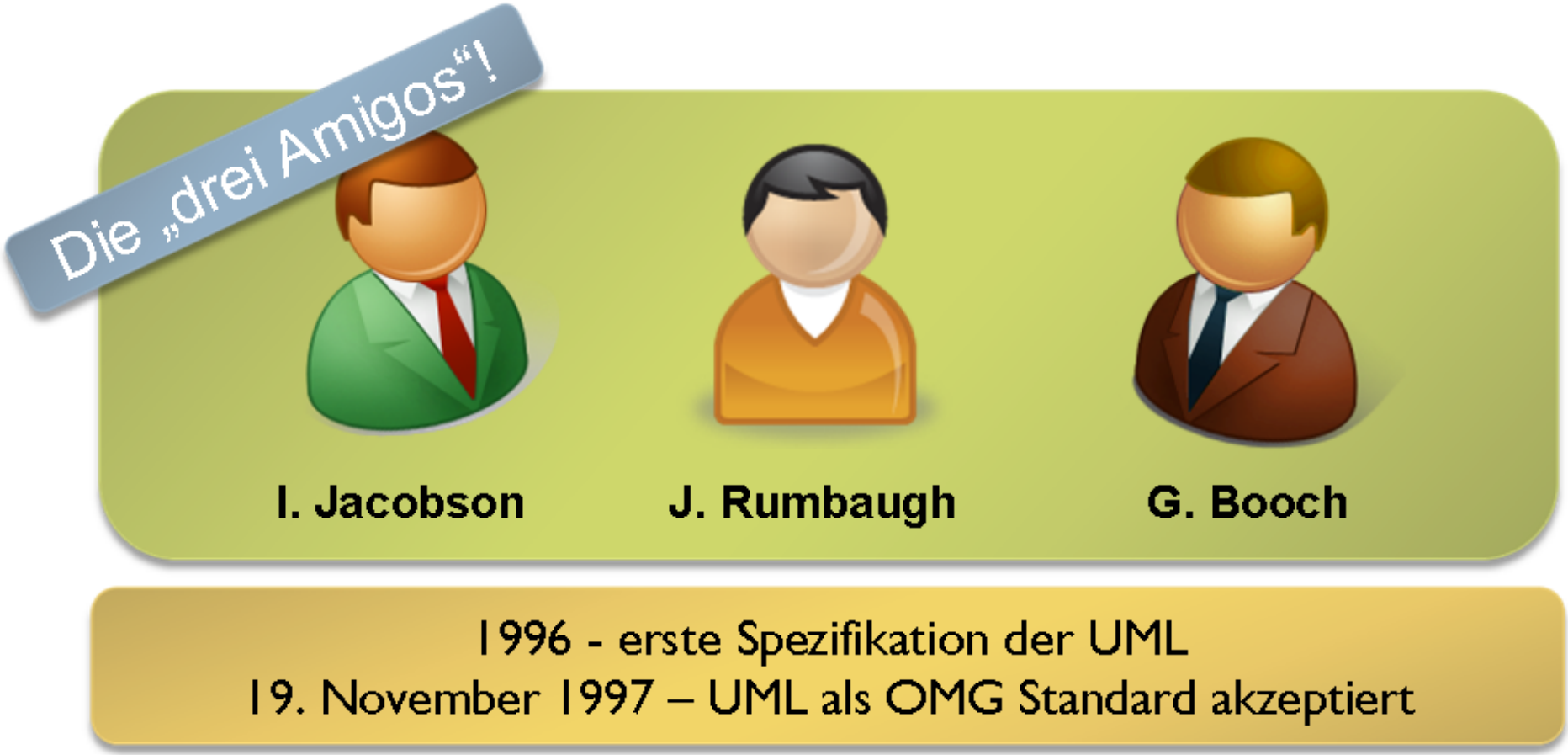
Booch-Notation - grafische Notation für Klassen (Grady Booch)

- seit 1980 Chief Scientist bei der Firma Rational Software Inc.



Drei Amigos

- James Rumbaugh & Ivar Jacobson wechselten zu Rational Software Inc.
- IBM kaufte Rational Software Inc. im Jahr 2002



UML - Der Standard

- UML - Unified Modeling Language
- vereinheitlichte Modellierungssprache
- durch die Object Management Group (OMG) entwickelt
- standardisiert nach ISO/IEC 19501

Rolle

- Analytiker
- Designer
- Entwickler
- Anwender

Zweck

- Spezifizieren
- Konstruieren
- Visualisieren
- Dokumentieren

UML - Der Standard

Was ist die UML?

- Beschreibungssprache zur Erstellung aussagekräftiger Modelle
- Erstellung austauschbarer Modelle
- allgemein verwendbar
- sprachunabhängig
- prozessunabhängig
- klare Semantik auf standardisierter Basis

Was ist die UML nicht?

- kein Entwicklungsprozess
- kein semantisches Metamodell
- kein Werkzeug
- keine visuelle Programmiersprache

UML - Der Standard

Standardisierungsgremium:	OMG - Object Management Group
Gründung:	1989
Form:	Industriekonsortium
Mitglieder:	IBM, Apple, SUN, später Microsoft
Ziele:	Verabschiedung von herstellerunabhängigen systemübergreifenden Standards für OO-Programmierung
Beispiele:	BPMN, CORBA, MOF, MDA, XMI ...
URL:	http://www.omg.org

UML - Versionen

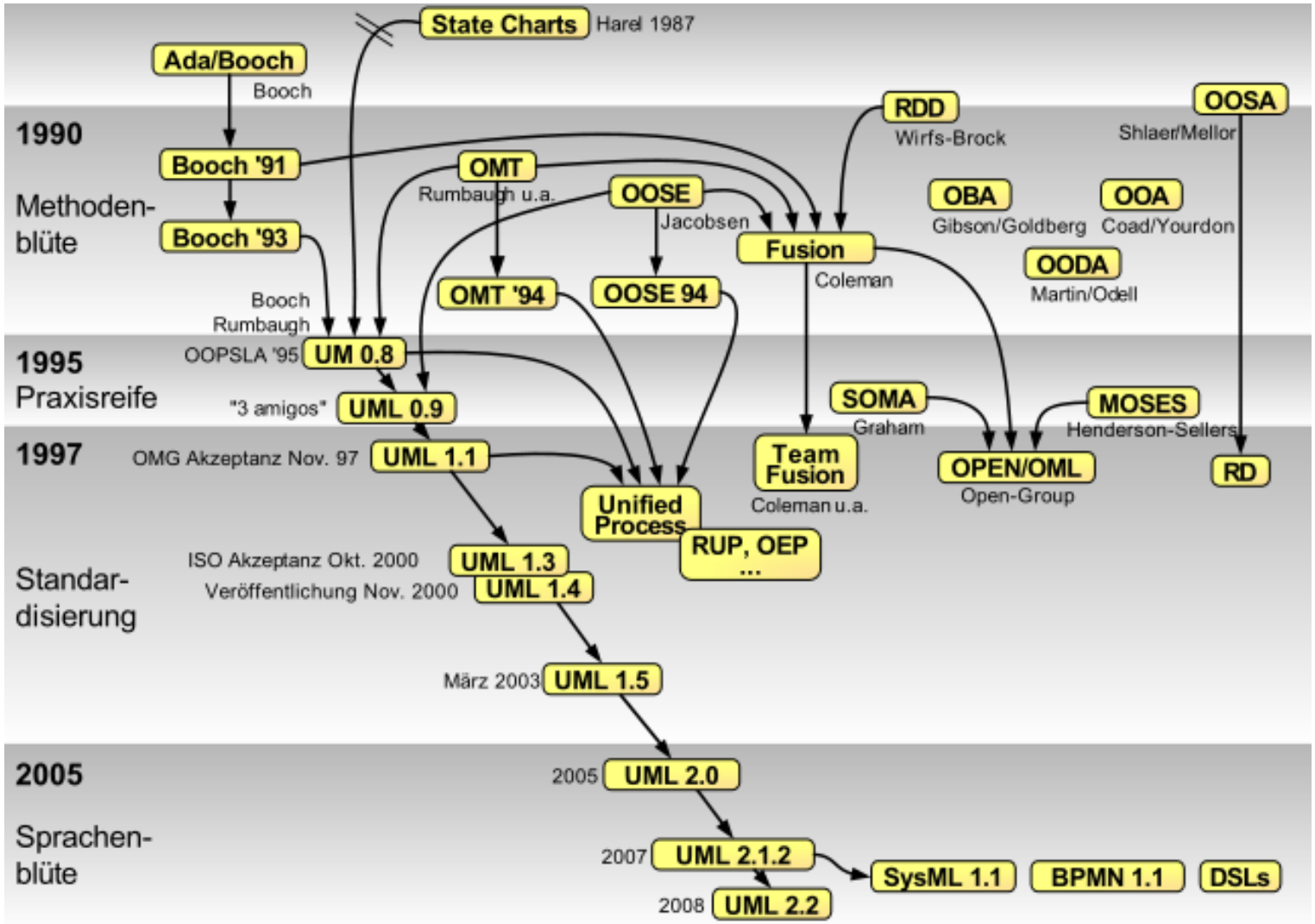


Abbildung: Historie der objektorientierten Methoden und Notationen [Axe08]

Aufbau der UML

Bezugsquelle: http://www.omg.org/spec/UML/2.3/ bzw. http://www.omg.org/

3 + 1 Teilspezifikation

- OMG UML, Infrastructure Version 2.3
- OMG UML, Superstructure Version 2.3
- Object Constraint Language Version 2.0
- UML 2.0 Diagram Interchange Specification

UML 2.3 Infrastructure

URL: http://www.omg.org/spec/UML/2.3/Infrastructure/ - (PDF)

Dokument: Umfang 226 Seiten

Version: 2.3

Inhalt:

- Beschreibung der UML - Kernarchitektur
- Beschreibung der Modellelemente
- Klassenkonzept
- Assoziation
- Multiplizitäten
- Stereotypen

UML 2.3 Superstructure

URL: http://www.omg.org/spec/UML/2.3/Superstructure/ - (PDF)

Dokument: Umfang 758 Seiten

Version: 2.3

Inhalt:

- Definition von statischen Modellelementen
- Definition von dynamischen Modellelementen
- Beispiel: Anwendungsfall, Aktivitäten uvm.

Diagrammtypen der UML 2

Strukturdiagramme:

- Klassendiagramm
- Kompositionsstrukturdiagramm
- Komponentendiagramm
- Verteilungsdiagramm
- Objektdiagramm
- Paketdiagramm

Verhaltensdiagramme:

- Aktivitätsdiagramm
- Anwendungsfalldiagramm
- Interaktionsübersichtsdiagramm
- Kommunikationsdiagramm
- Sequenzdiagramm
- Zeitverlaufsdiagramm
- Zustandsdiagramm

Object Constraint Language Version

URL: http://www.omg.org/spec/OCL/2.2/ - (PDF)

Dokument: Umfang 238 Seiten

Version: 2.2

Verwendung:

- Spezifikation von Invarianten
- allgemeine Formulierung von Bedingungen
- Formulierung von Vor- und Nachbedingungen

UML 2.0 Diagram Interchange Specification

URL: http://www.omg.org/spec/UMLDI/1.0/ - (PDF)

Dokument: Umfang 86 Seiten

Version: 1.0

Inhalt:

- Austausch von UML Diagrammen zwischen Werkzeugen verschiedener Hersteller

UML - Werkzeuge

Kriterien zur Auswahl von Programmen:

- Unterstützung von UML 2.X
- Diagrammunterstützung
- Roundtrip - Engineering
- Modelltransformation
- Refactoring
- Codeerzeugung
- Reverse Engineering
- Diagrammaustausch

UML - Werkzeuge

kommerzielle Produkte:

Visual Paradigm	www.visual-paradigm.com
ObjectIF	www.microTOOL.de
Sparx Systems Enterprise Architect	www.sparxsystems.com.au
Rational Rose	www.ibm.com/software/rational/
Poseidon	http://www.gentleware.com/
eUML2	http://www.soyatec.com/main.php

freie UML Werkzeuge:

ArgoUML UML 1.4!	argouml.tigris.org/
Eclipse Uml2Tools	www.eclipse.org
Netbeans	www.netbeans.org/
Dia	www.gnome.org/projects/dia
Umbrello	uml.sourceforge.net/

Sonstiges

Nutzen Sie das Internet!

- Wikipedia
- http://www.omg.org/uml/
- http://www.oose.de/glossar
- http://www.jeckle.de/

Bücher:

- http://www.amazon.de/
- empfehlenswerter Autor/-in: Bernd Oestereich, Chris Rup

Der Akteur

Definition

Als Akteur werden außerhalb des betrachteten Systems liegende Personen bzw. andere technische Systeme bezeichnet, welche mit dem System zum Erreichen eines konkreten Ziels interagieren.

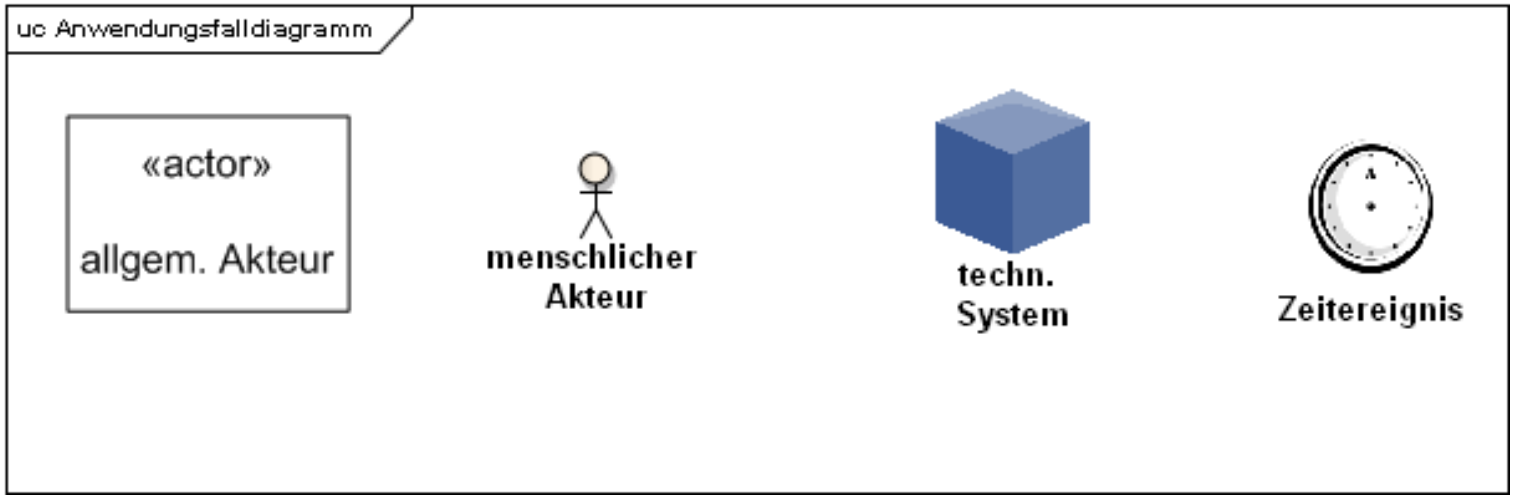
siehe UML Superstructure Specification v 2.2, S. 604 (10-05-05.pdf, S. 620)

- Person: Mensch als Anwender
- externe Ereignisse sind als Akteure umsetzbar

Der Akteur

- es werden keine konkreten Personen betrachtet, sondern Rollen, die diese Personen spielen
- Rollen sind generalisier- und spezialisierbar
- eine Person kann mehrere Rollen spielen
- Akteure können untereinander, ohne Einfluss auf das System, interagieren
- Akteure stehen mit Anwendungsfällen in Beziehung, wenn sie an diesen beteiligt sind

Der Akteur - Notation



Der Akteur - Tipps

- Ein Akteur ist keine konkrete Person!
- Ist der Akteur eineindeutig bezeichnet?
- Steht der Akteur zu dem System in einer Beziehung?
- Ist der Akteur eine Mensch oder ein technisches System?

Der Anwendungsfall

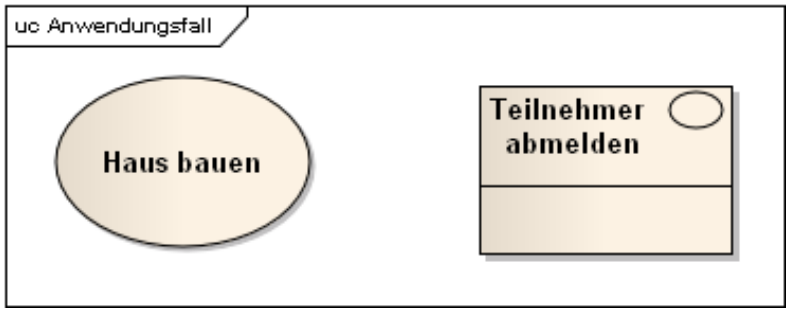
Definition

Durch einen Anwendungsfall wird eine abgeschlossene Teilfunktionalität des Anwendungssystems beschrieben, welche für alle beteiligten Akteure ein erkennbares Ergebnis liefert.

siehe UML Superstructure Specification v2.3, S. 603 (10-05-05.pdf, S. 619)

- Beschreibung einer typischen Interaktion zwischen Anwender und dem System
- Darstellung des extern wahrnehmbaren Verhaltens einer Arbeitssituation
- Beschreibung des „Was“ nicht des „Wie“
- liefert ein positives bzw. negatives Ergebnis für den Geschäftsprozess
- eine Ausprägung eines Anwendungsfalls ist ein Szenario
- ein Anwendungsfall kann mehrere Szenarien beinhalten

Der Anwendungsfall - Notation



- Ellipse mit den Namen des Anwendungsfalls
- Namensgebung: **Substantiv + aktives Verb**
- **Namensgebung erfolgt aus der Perspektive des Systems**

  Beispiel: Autohaus - Verkauf von Neuwagen
  - Perspektive des Akteurs: PKW kaufen (falsch)
  - Perspektive des Systems: PKW verkaufen

Der Anwendungsfall - Beschreibung

Bei einem Anwendungsfall steht die Beschreibung in Textform im Vordergrund!

- UML definiert keine Form für die Beschreibung
- Formulierung in der Sprache der Anwender
- Beschreibung mit begrenzten Umfang (max. 1/2 A4 Seite)
- Problem:

Darstellung der Sachverhalte erfolgt durch verschiedene Autoren auf sprachlich unterschiedliche Art und Weise

Der Anwendungsfall - Dokumentationsschablone

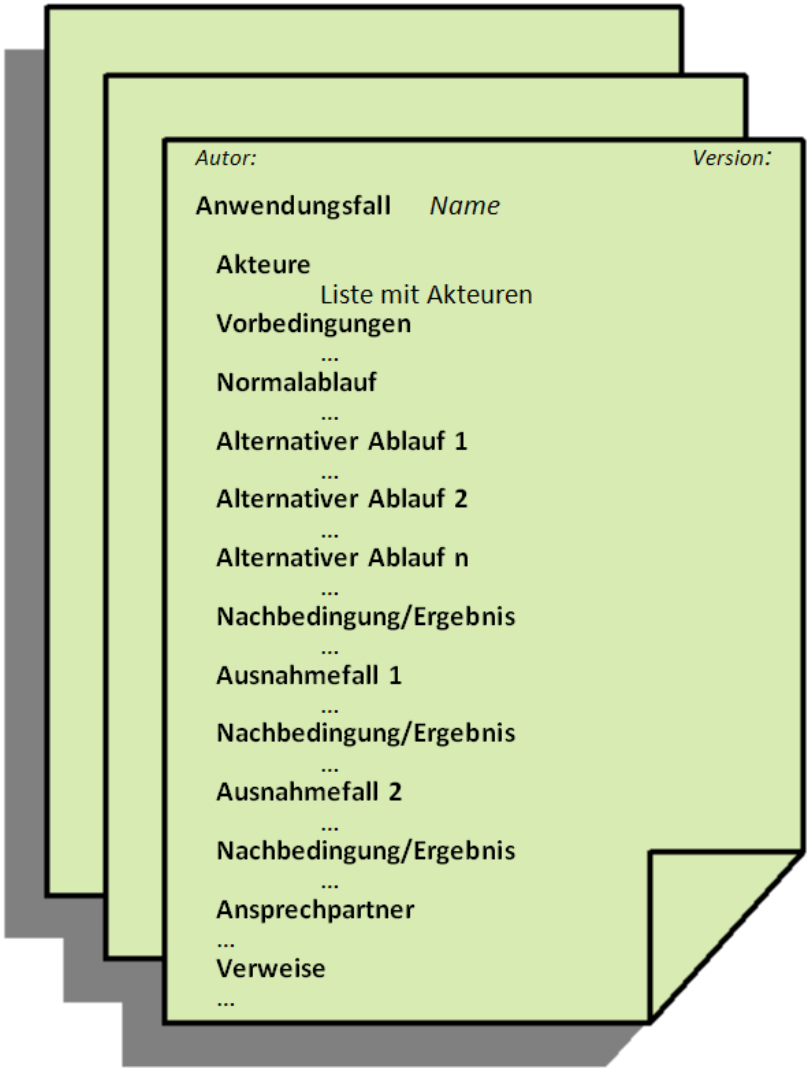


Abbildung: Beispiel einer Dokumentationsschablone

Der Anwendungsfall - Tipps

Grundsätze:

1. Anwendungsfalldiagramme sind ein Hilfsmittel zur Anforderungsermittlung
2. Für jeden menschlichen Akteur muss mindestens eine Person existieren, welche die Rolle des Akteurs spielt.
3. Jeder Anwendungsfall behandelt eine klar abgegrenzte Aufgabe und liefert ein relevantes Ergebnis.
4. Anwendungsfälle ohne Akteure sind fragwürdig.
5. Anwendungsfälle beschreiben die Systembenutzung, nicht das System.
6. Anwendungsfälle werden von Analytikern, nicht von Anwendern geschrieben.

Der Anwendungsfall - Tipps

Grundsätze:

7. Einfachheit und Problemadäquatheit der Anwendungsfälle gehen vor der Eleganz des Anwendungsfallmodells.
8. Die textuelle Spezifikation eines Anwendungsfalls sollte eine Seite nicht überschreiten.
9. Die Beschreibung eines Anwendungsfalls erfolgt immer aus Sicht des Systems.
10. Beziehungen zwischen Anwendungsfällen zeigen statische funktionale Zerlegungen aber keine zeitlichen Abläufe.
11. Eine Paketbildung von Anwendungsfällen erfolgt nach fachlicher Zusammengehörigkeit.

Das Anwendungsfalldiagramm

Definition

Das Anwendungsfalldiagramm stellt Akteure, Anwendungsfälle und deren Beziehungen dar.

siehe UML Superstructure Specification v2.3, S. 617 (10-05-05.pdf, S. 633)

- Hilfsmittel zur Anforderungsermittlung
- Unterstützung der Kommunikation zwischen dem Entwickler und dem Anwender
- Darstellung der Zusammenhänge zwischen Anwendungsfällen und Akteuren
- Darstellung der Interaktion mit dem System
- keine Beschreibung eines Verhaltens oder von Abläufen
- keine Aussage über das Systemdesign

Das Anwendungsfalldiagramm - Beispiel

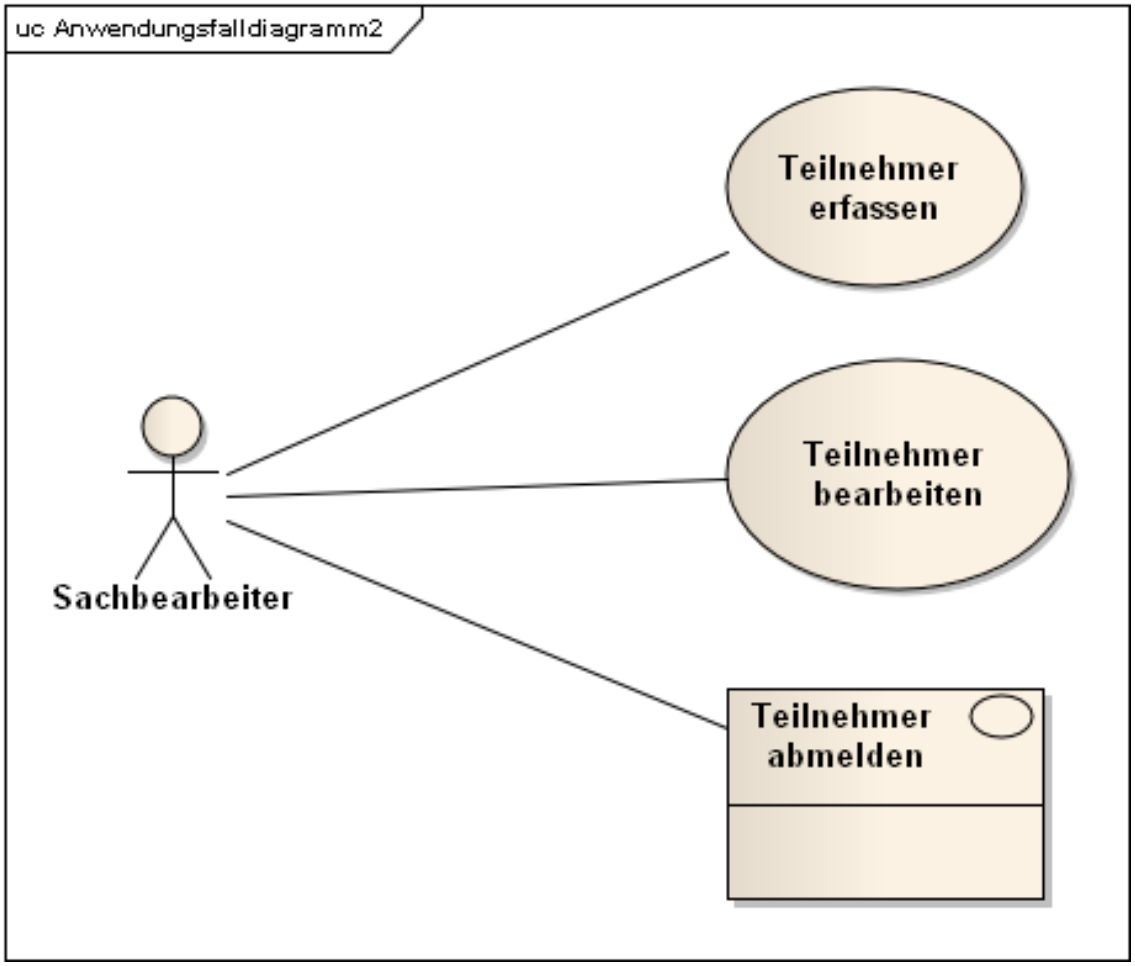


Abbildung: einfaches Anwendungsfalldiagramm

Die Assoziation

Definition

Die Assoziation beschreibt eine Beziehung zwischen einem Anwendungsfall und einem Akteur. Durch eine gerichtete Assoziation wird angegeben, von welchem Element eine Interaktion ausgeht.

- durch eine durchgezogene Linie zwischen den Elementen dargestellt
- eine Angabe der Mehrwertigkeit an den Enden der Linie ist optional
- der Pfeil einer gerichteten Assoziation zeigt vom Initiator weg

Die Assoziation - Beispiel

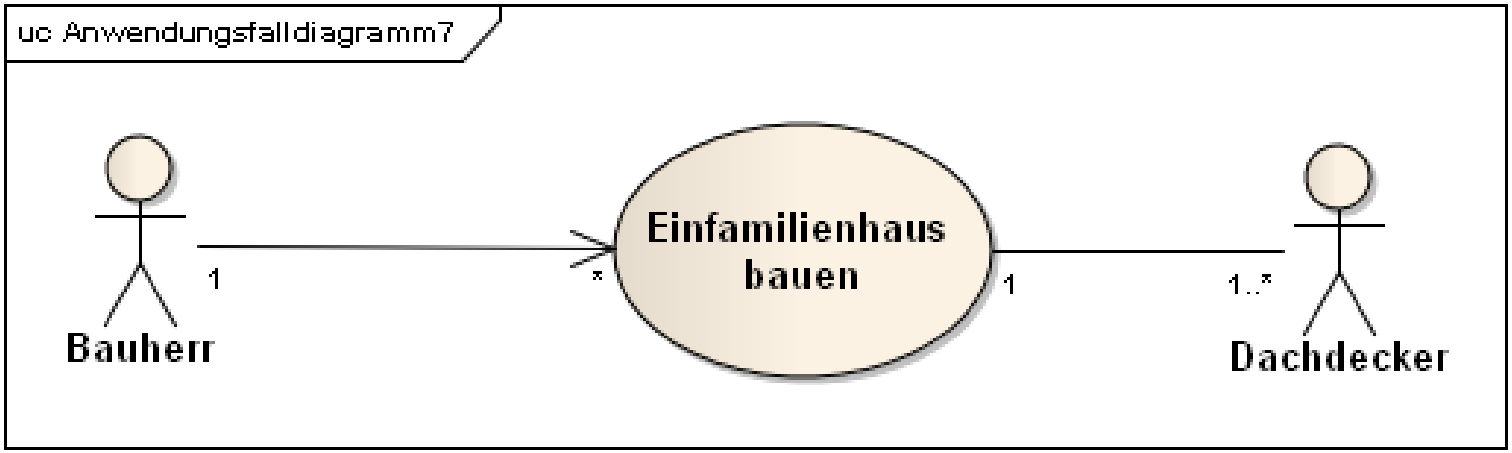


Abbildung: Die Assoziation - Beispiel

Die Enthält- und Erweitertbeziehung

Definition

Mit der Enthältbeziehung wird dargestellt, dass ein Anwendungsfall Teil eines anderen Anwendungsfalls ist.

Definition

Mit der Erweitertbeziehung wird dargestellt, dass ein Anwendungsfall unter bestimmten Bedingungen durch einen anderen Anwendungsfalls ergänzt wird.

- Ausgliederung von identischen Teilen mehrerer Anwendungsfälle
- Vermeidung von Redundanz durch Einbindung über die Enthältbeziehung
- keine Vererbung
- Erweiterung eines Anwendungsfalls ist von Bedingungen abhängig
- Bedingungen sind separat zu notieren

Die Enthält- und Erweitertbeziehung

Enthältbeziehung:

- gestrichelte Linie mit offenem Pfeil
- Richtung auf den enthaltenen Anwendungsfall
- mit `include` gekennzeichnet

Erweitertbeziehung:

- gestrichelte Linie mit offenem Pfeil
- Richtung auf den zu erweiternden Anwendungsfall
- mit `extend` gekennzeichnet

Die Enthält- und Erweitertbeziehung - Beispiel

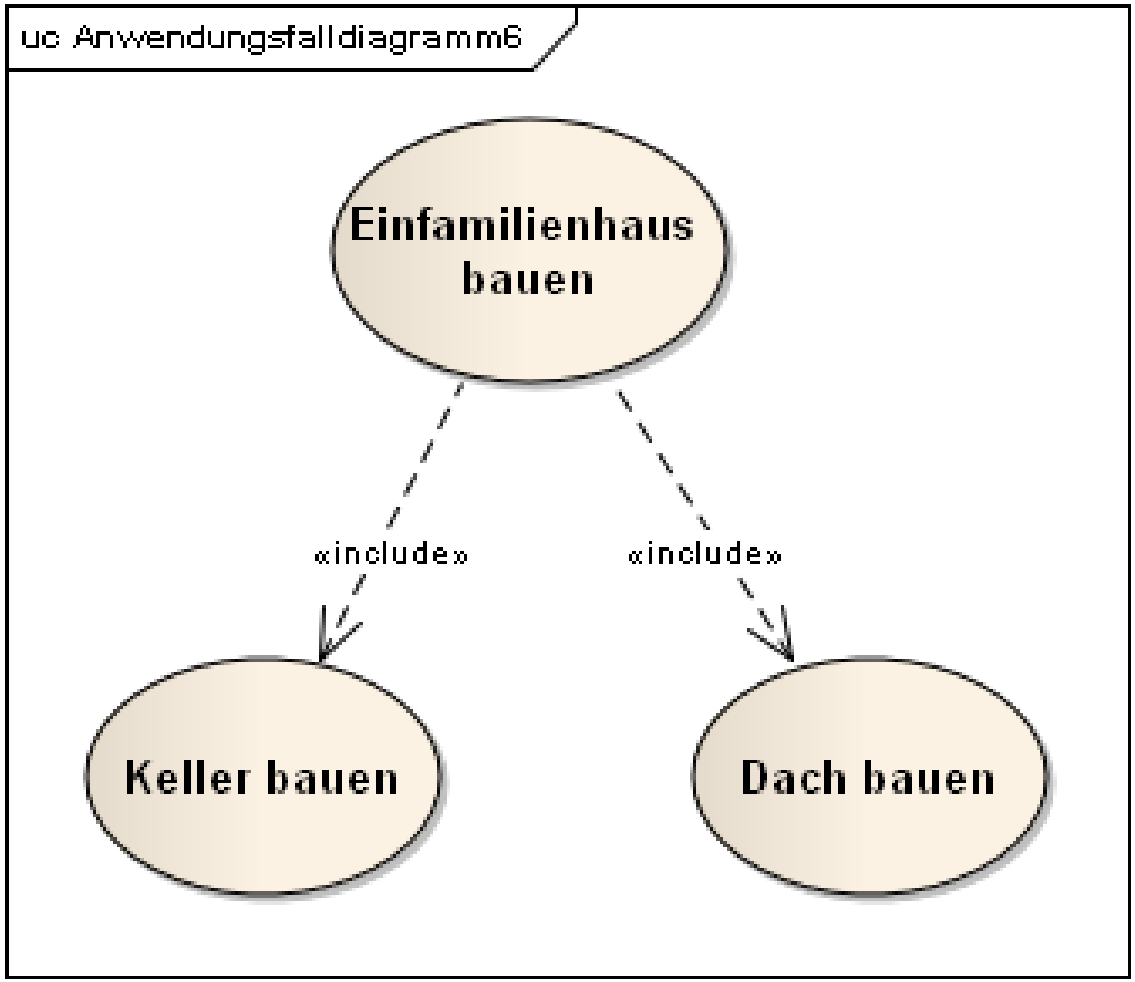


Abbildung: Die Enthält- und Erweitertbeziehung

Die Enthält- und Erweitertbeziehung - Tipps

- mind. zwei Anwendungsteile enthalten gleiche Teile -> Enthältbeziehung
- Enthältbeziehung bindet meist sekundäre Anwendungsfälle ein
- Erweiterungsbeziehung umstritten (Pfeilrichtung)
- Enthätbeziehung ist ausreichend

Die Aktivität

Definition

Die Aktivität modelliert das Verhalten eines Systems, indem mit Hilfe von Kontroll- und Objektflüssen elementare Verhaltensbausteine, den Aktionen, zu komplexerem Verhalten kombiniert werden.

siehe UML Superstructure Specification v2.3, S. 303 (10-05-05.pdf, S. 319)

- Aktivitäten beschreiben Abläufe und Prozesse
- UML 1.x Aktivitätsdiagramme $\longrightarrow$ UML 2.x Aktivitäten
- Semantik der Petrinetze zur Modellierung der Aktivitäten

Beispiel

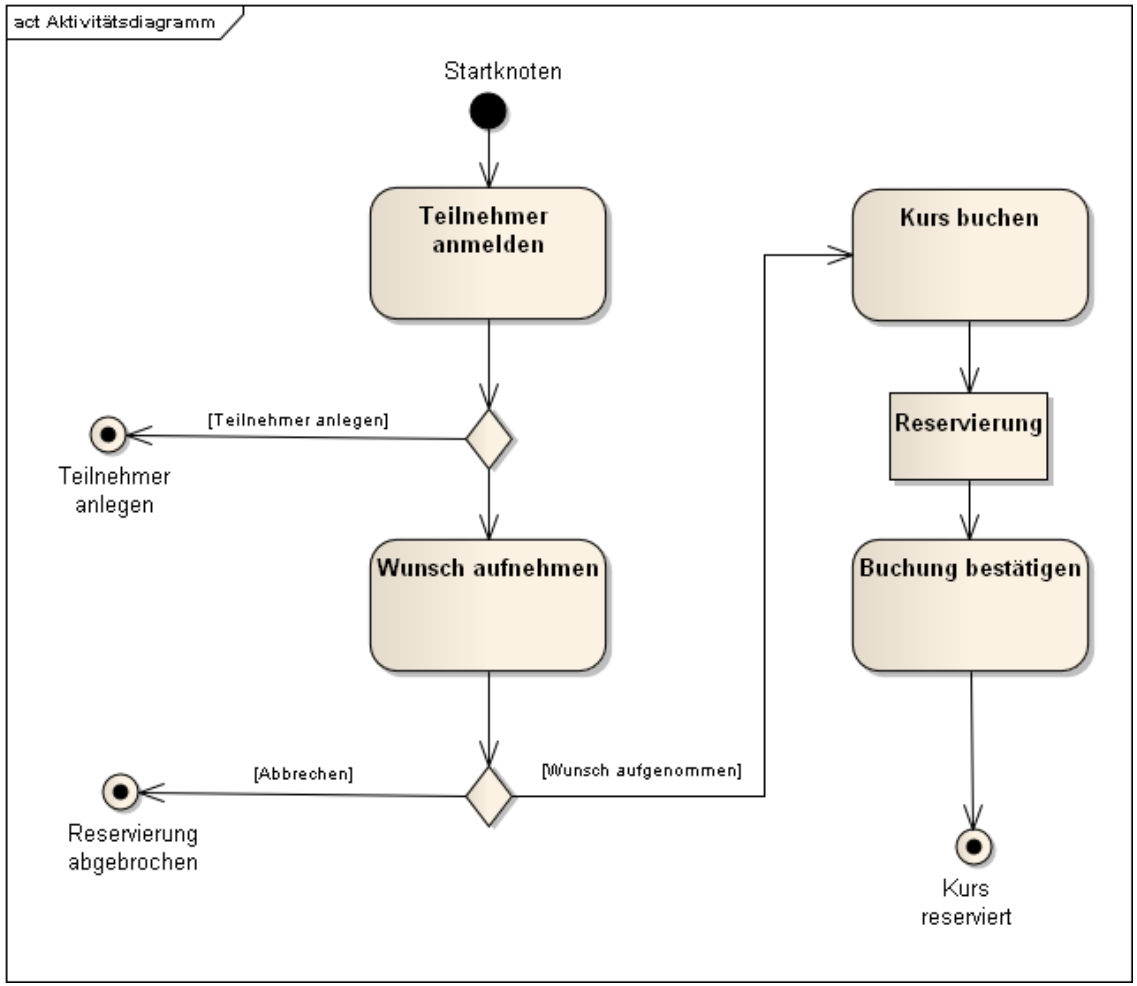


Abbildung: Die Aktivität - Kursbuchung durchführen

Aktivitäten - Praxis

- Beschreibung von Abläufen, z. B. Anwendungsfälle
- natürliche Sprache eignet sich nur für sehr einfache Abläufe
- Aktivitäten eignen sich für die grafische Darstellung komplexer Abläufe
- Komplexität: Ausnahmen, Varianten, Sprünge, Wiederholungen

Aktivitäten - Semantik

eingehende Kontrollflüsse - Start der Aktion

- eine Aktion hat mindestens einen ein- und ausgehenden Kontrollfluss
- ein eingehender Kontrollfluss löst die Aktion aus
- mehrere eingehende Kontrollflüsse:
  alle Kontrollflüsse müssen vorliegen, damit die Aktion ausgelöst wird
  (implizite Synchronisation)

Aktivitäten - Semantik

ausgehende Kontrollflüsse - Ende der Aktion

- Bereitstellung eines Tokens an den ausgehenden Kontrollfluss
- mehrere ausgehende Kontrollflüsse sind möglich
- alle ausgehenden Kontrollflüsse feuern gleichzeitig
- sind Bedingungen definiert, warten alle ausgehenden Kontrollflüsse bis diese erfüllt sind

Start- und Endknoten

- Startknoten: hat nur ausgehende Kontrollflüsse
- Ablaufende, Endknoten haben keine ausgehenden Kontrollflüsse
- eine Aktivität hat mindestens einen Start- und einen Endknoten
- mehrere Startknoten lösen nebenläufige Prozesse aus
- mehrere Endknoten: jeder Endknoten beendet sofort alle Aktionen in der gesamten Aktivität
- Ablaufende beenden nur den zugeordneten Kontrollfluss, der Rest macht weiter

Aktivitäten - Semantik

Kontrollknoten

- Startknoten
- Endknoten
- Ablaufende
- Entscheidung
- Synchronisation
- Teilung (Splitting)
- Zusammenführung

Aktivitäten - Semantik

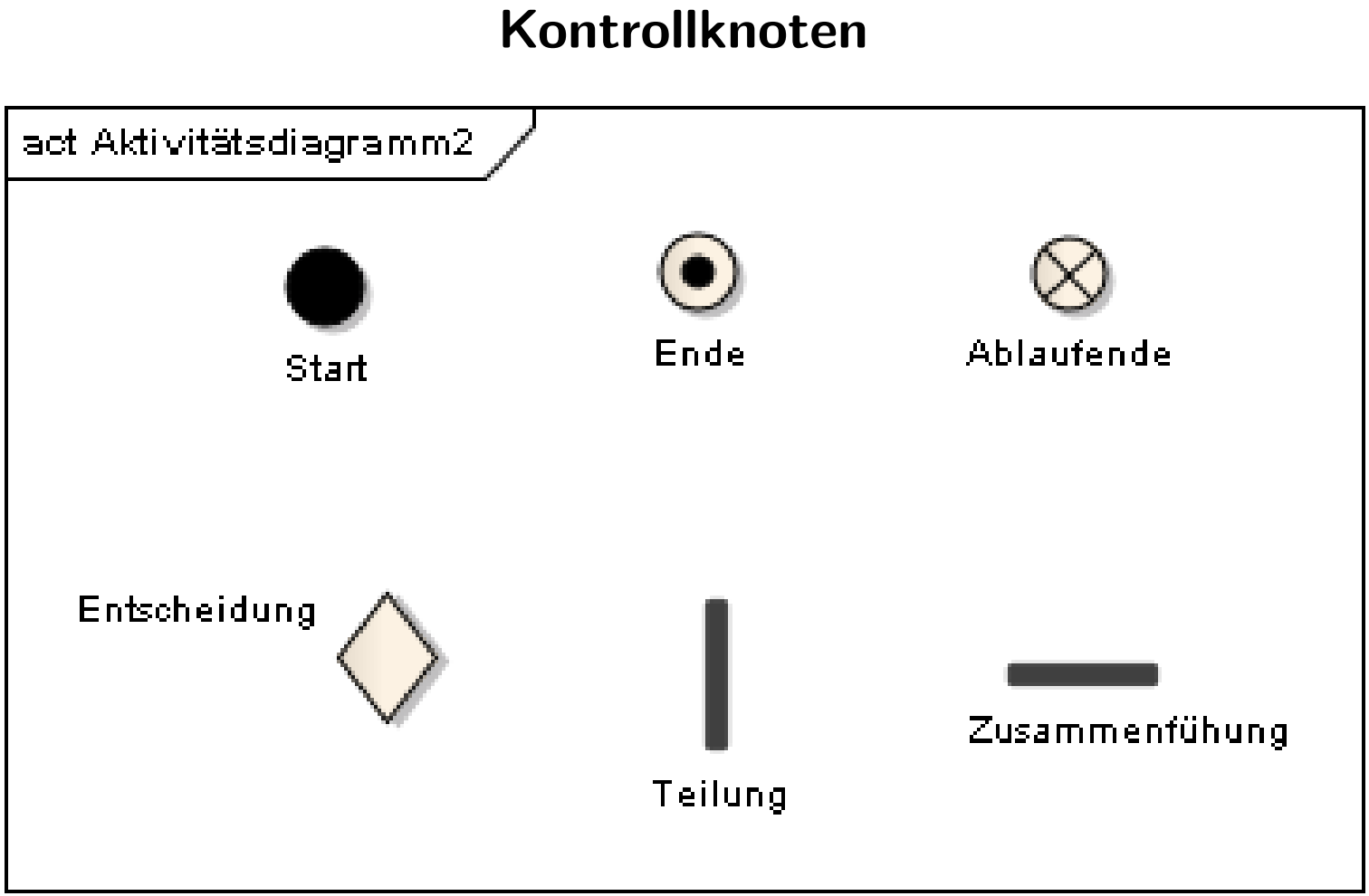
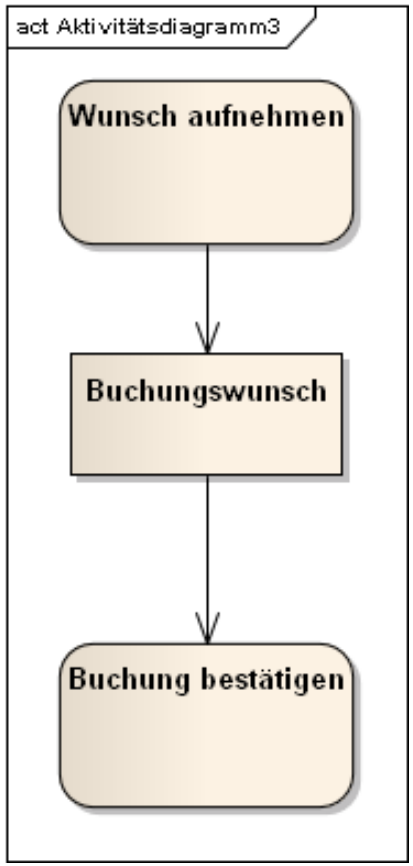


Abbildung: Kontrollknoten

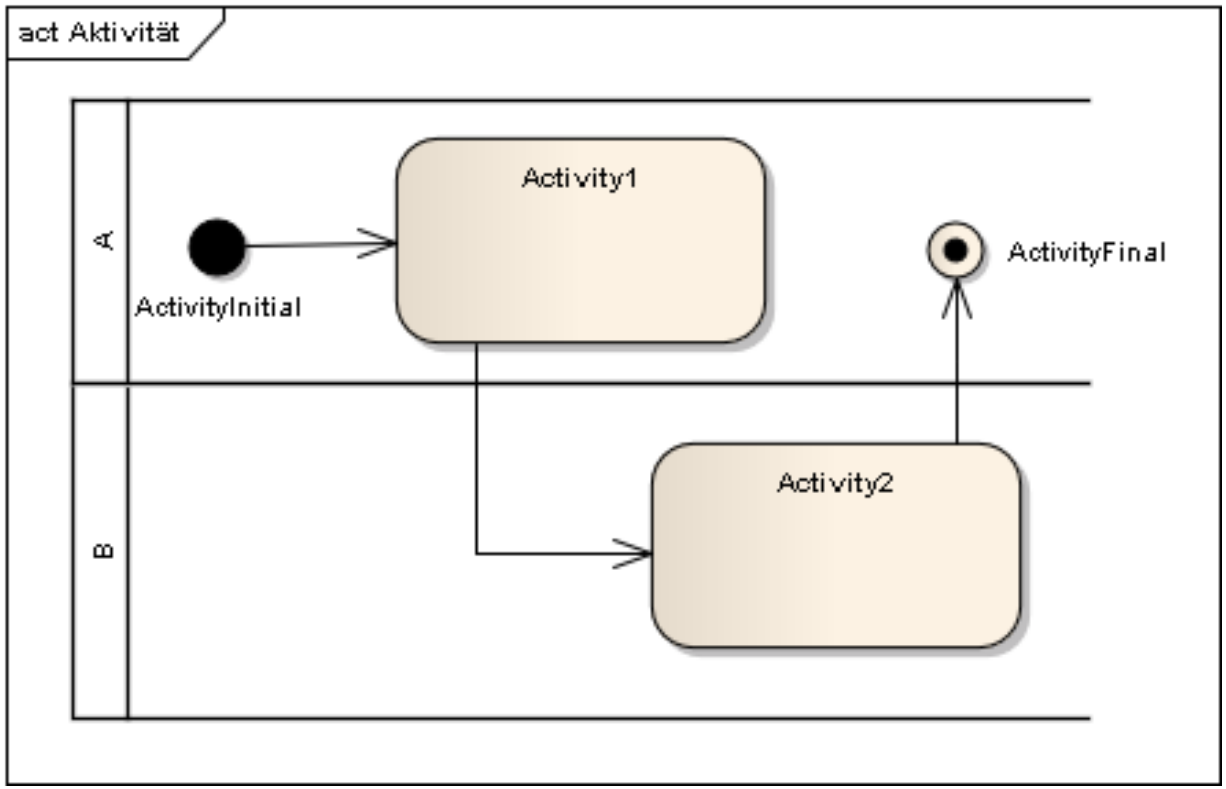
Objektknoten

Ein Objektknoten zeigt an, dass eines oder mehrere Objekte existieren. Diese können sowohl als Eingangs- als auch Ausgangsparameter in den Aktivitäten sein.
Ein Objektfluss bezeichnet den Transport von Objekten.



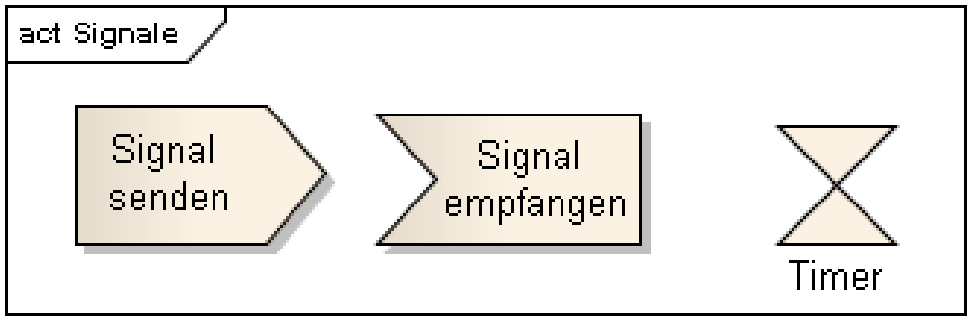
Partitionen

Partitionen kennzeichnen Eigenschafts- oder Verantwortungsbereichen für eine Menge von Knoten.



Signale

Ein empfangenes Signal kennzeichnet ein zu beachtendes Ereignis, welches ein Objektfluss auslöst. Ein gesendetes Signal benachrichtigt über ein Ereignis während eines Kontrollflusses.



Die Klasse

Definition

Eine Klasse definiert Attribute, Operationen und die Semantik für eine Menge von Objekten. Alle abgeleiteten Objekte einer Klasse entsprechen dieser Definition.

Die Klasse - Eigenschaften

- Beschreibung des Verhaltens und Struktur von Objekten
- auch als Typ bezeichnet
- Objekte werden von Klassen produziert
- Objekte sind die agierenden Einheiten
- Verhalten eines Objektes leitet sich aus den Nachrichten ab, die es verstehen kann
- zur Nachrichtenverarbeitung sind Operationen nötig
- Definition von Zusicherungen und Stereotypen
- UML erweitert Klassendefinition um Ports und Signalempfänger
- eine Klasse kann andere Klassen spezialisieren
- eine Klasse kann anderen Klassen in Beziehungen stehen

Die Klasse

- durch Rechtecke mit fettgedruckten Namen
- ergänzt durch Attribute und Operationen
- Rubriken durch horizontale Linien getrennt
- Klassennamen beginnen immer mit Großbuchstaben (i.d.R. Substantive im Singular)
- Paketname kann dem Klassenname durch zwei :: getrennt vorangestellt werden
- oberhalb des Namens kann in spitzen Klammern ein Stereotyp angegeben werden «Fachklasse»
- unterhalb des Namens in geschweiften Klammern können Eigenschaftswerte angegeben werden
- Operationen durch Namen, optional mit Parametern, Initialwerten, Eigenschaften und Zusicherungen notiert
- für einige Stereotypen definiert die UML eigene Symbole
- Attribute UML 2.x Property

Die Klasse - Beispiel

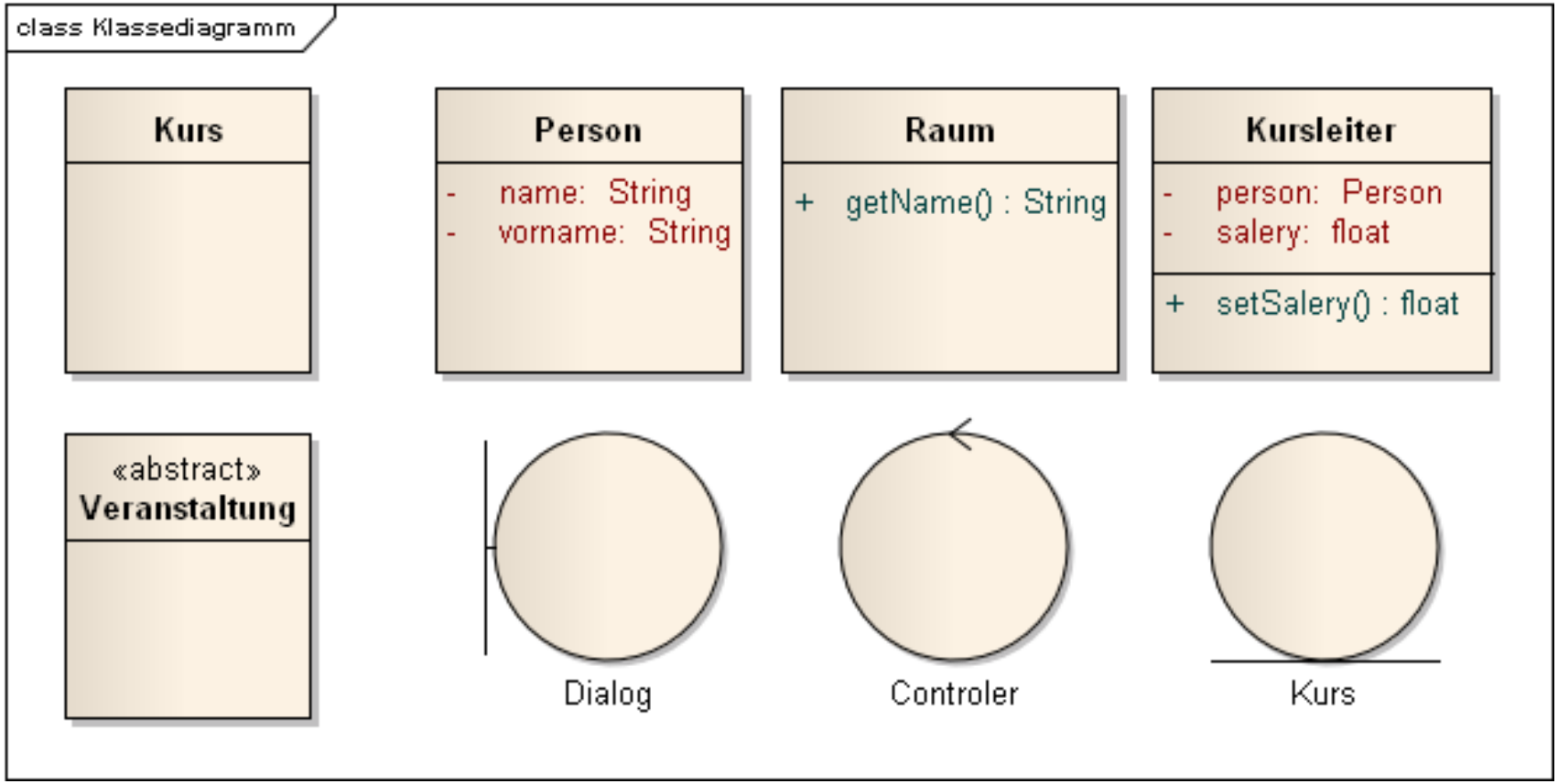


Abbildung: Die Klasse

Das Attribut

Definition

Das Attribut ist ein Element, welches in jedem Objekt einer Klasse einen individuellen Wert für das Objekt repräsentiert. Außerhalb des Objektes haben Attribute keine Identität, d. h. Attribute werden ausschließlich durch das umhüllende Objekt kontrolliert.

- durch seinen Namen beschrieben
- Ergänzung des Datentyps, Inititalwert, Zusicherungen sind möglich
- starke Programmiersprachenabhängigkeit
- Zusicherununngen können den Wertebereich einschränken
- Zusicherungen können separat notiert werden
  (Beispiel: OCL: `context Kreis inv: radius > 0`)
- Eigenschaftswerte beschreiben Besonderheiten `{read only}`, `{frozen}`
- optimale / obligatorische Attribute können durch Angabe der Multiplizität differenziert werden

Das Attribut

- Dynamische Arrays:
  - nur angeben, wenn nicht [1], z. B. bei Arrays mit [$*$] (Komposition)
  - Angabe von Sortierreihenfolgen **unorderd**, orderd

  - abgeleitete Attribute:
  - nicht physisch durch einen Wert repräsentiert
  - automatisch berechnet
  - nicht direkt änderbar
  - nur von objektinternen Elementen abhängend
  - abgeleitete Attribute für caching Kennzeichnung sinnvoll, um Performanz zu optimieren

  - Klassenattribut:
  - gehören nicht zum Objekt sondern zur Klasse, alle Objekte können auf das gemeinsame Klassenattribut zugreifen (Zählung von Objekten)

Das Attribut

- Sichtbarkeit: Programmiersprachenabhängig
- public: für alle sicht- und benutzbar
- protected: nur die Klassen, Unterklassen und die als friend deklarierte Klassen haben Zugriff
- private: nur die Klassen und die als friend deklarierte Klassen haben Zugriff
- package: nur Klassen im selben Paket haben Zugriff
- Verwendung nur über die Klassen in der Attribute verwendet werden
- andere Klassen stets über Operationen

Das Attribut - Notation

- beginnen mit einem Kleinbuchstaben
- Typnamen mit Großbuchstaben
- Eigenschaften und Zusicherungen in geschweiften Klammern
- abgeleitet Attribute durch „/“ gekennzeichnet
- Klassenattribute werden unterstrichen
- Sichtbarkeiten mit +, #, -, ~ dargestellt

Das Attribut - Beispiel

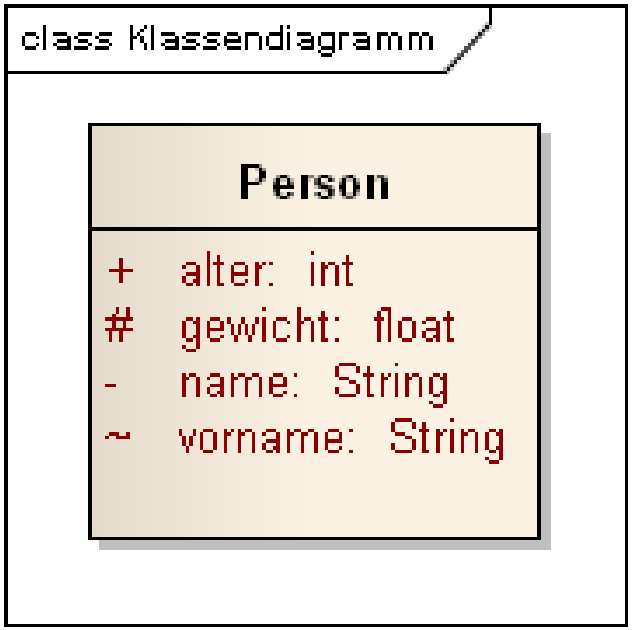


Abbildung: Das Attribut - Beispiel

Die Operation

Definition

Operationen sind „Dienstleistungen“, welche von einem Objekt angefordert werden. Diese werden durch eine Signatur beschrieben, welche aus Operationsname, Parameter und ggf. einem Rückgabetyp besteht. Operationen werden durch Methoden, als Folge von Anweisungen implementiert.

- Begriff „Methode“ und „Operation“ werden meist synonym verwendet, siehe Definition der Programmiersprache
- Nachricht: besteht aus Namen (Selektor), einer Liste von Argumenten, genau ein Empfänger
- Sender erhält genau ein Antwortobjekt
- eindeutige Signatur innerhalb der Klasse
- mit Zusicherungen ausstattbar
- Richtung (in, out, inout)

Die Operation

- Eigenschaftswerte heben Besonderheiten hervor, z.B.: `abstract`, `deprecated`
- abstrakte Operationen sind solche, die nur durch Signatur repräsentiert werden, Impl. erst in Unterklassen (c++ virtuell)
- abstrakte Operationen nur in abstrakten Klassen
- nicht implemen abstrakte Operationen sind sinnlos
- Operationen ohne Seiteneffekte auf den Zustand des Objektes oder Anderer sind mit query zu bezeichnen
- Objekte untereinander kommunizieren durch den Austausch von Botschaften
- jedes Objekt versteht genau die Botschaften zu denen es entsprechende Operationen hat
- jede Klasse kann die Operationen mehrfach definieren

Die Operation - Notation

- Name beginnt in Kleinbuchstaben
- Argumente beginnen mit einem Kleinbuchstaben und werden durch einen Datentyp näher bestimmt
- Code im Rumpf ist programmiersprachenspezifisch
- Eigenschaftswerte stehen in geschweiften Klammern
- Zusicherungen als OCL
- Sichtbarkeit:
  - public: für alle sicht- und benutzbar
  - protected: nur die Klassen, Unterklassen und die als friend deklarierten Klassen haben Zugriff
  - private: nur die Klassen und die als friend deklarierten Klassen haben Zugriff
  - package: nur Klassen im selben Paket haben Zugriff
  - Verwendung nur über die Klassen in der Attribute verwendet werden

Die Operation - Beispiele

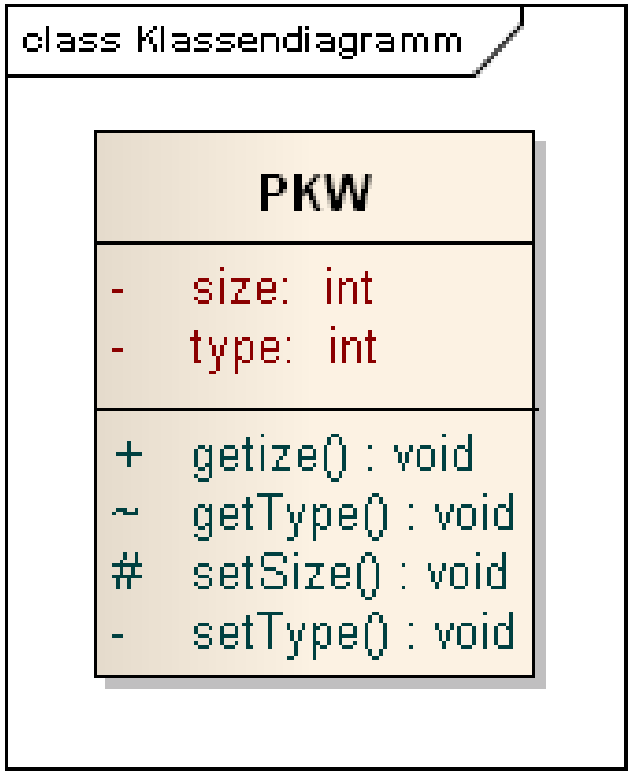


Abbildung: Die Operation - Beispiel

Das Klassendiagramm

Definition

Ein Klassendiagramm ist ein Strukturdiagramm der UML, welches zur grafischen Darstellung der existerenden Klassen und deren Beziehungen untereinander dient.

- Aktivitäten beschreiben Abläufe und Prozesse
- UML 1.x Aktivitätsdiagramme $\rightarrow$ UML 2.x Aktivitäten
- Semantik der Petrinetze zur Modellierung der Aktivitäten

Das Klassendiagramm - Praxis

Verwendung:

- Domänenklassendiagramm als Teil der Anforderungsspezifikation
- Analyseklassendiagramm als Teil der Analysespezifikation
- Geschäfts- oder Fachklassen für fachliche Begriffsmodelle
- Designklassen für Strukturen
- aus Quellcode / Reverse-Engineering für Visualisierung der Code-Struktur

Beziehungselemente

Darstellung statischer Modellsachverhalte

- Assoziation
- gerichtete Assoziation
- qualifizierte Assoziation
- Attributierte Assoziation
- Mehrgliedrige Assoziation
- Generalisierung und Spezialisierung
- Realisierung
- Aggregation
- Komposition
- Abhängigkeit

Die Assoziation

Definition

Die Assoziation ist eine Relation zwischen UML-Elementen, die eine Menge von Objektverbindungen mit gleicher Semantik und Struktur beschreibt.

- Assoziationen visualisieren Kommuikationsbeziehungen zwischen Klassen
- zwischen verschiedenen Klassen
- rekursive Verbinungen zur gleichen Klasse (verschiedene Objekte der Klasse) erlaubt
- spezialisiert durch Aggregation und Komposition
- mit Namen benennbar
- Rollenbezeichnung und Beschränkungen für die Verbindung an den Enden der Assoziation
- Gültigkeit über den Existenzzeitraum oder nur zeitweilig «temporary»
- Multiplizität ist die Angabe der Anzahl der assoziierten Objekte der benachbarten Klasse

Die Assoziation - Beispiel

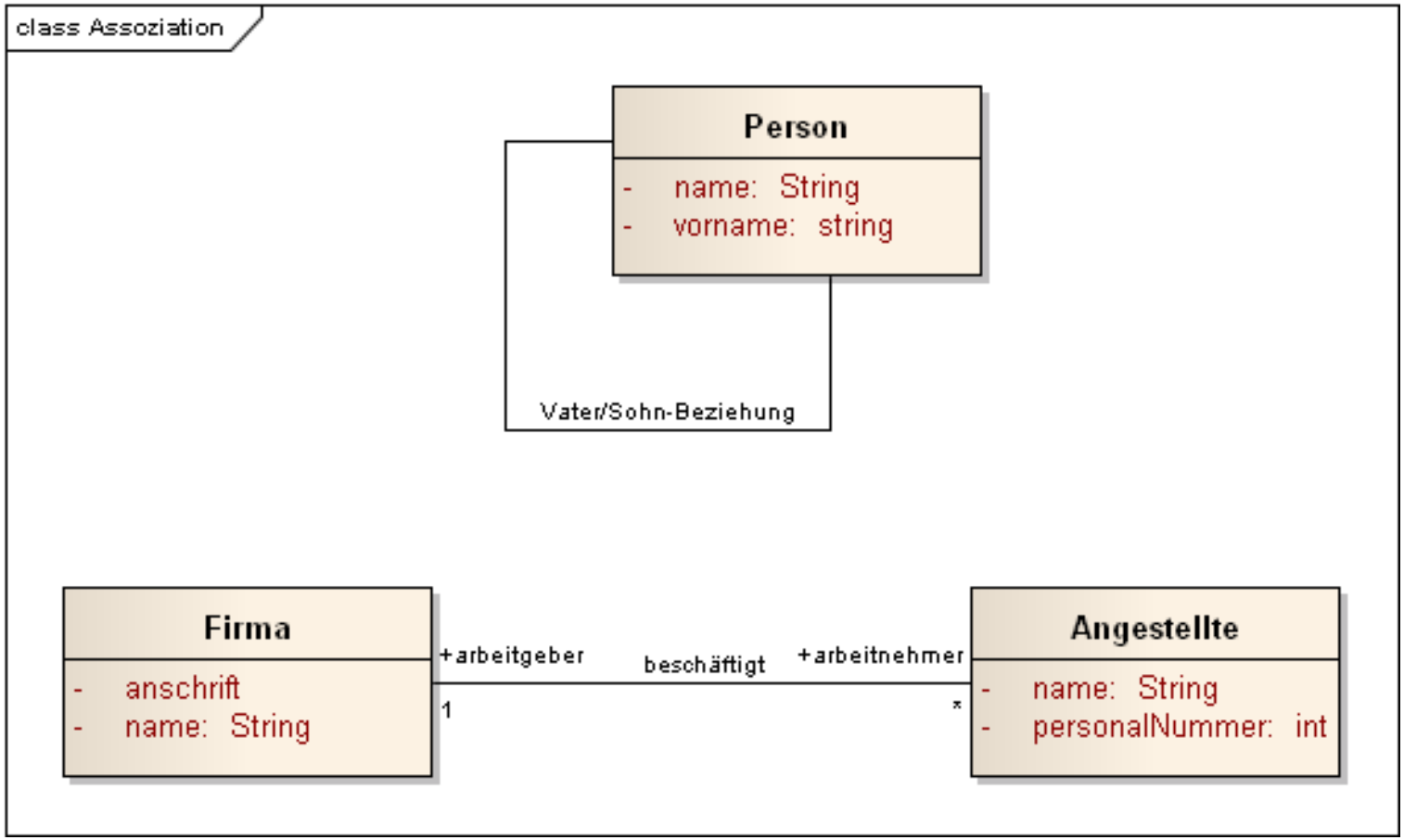


Abbildung: Die Assoziation

Die gerichtete Assoziation

Definition

Die gerichtete Assoziation ist eine unidirektionale Beziehung zwischen zwei Elementen.

- Notation wie Assoziation mit offener Pfeilspitze
- expliziter Ausschluss einer Richtung durch Kreuz

Die gerichtete Assoziation

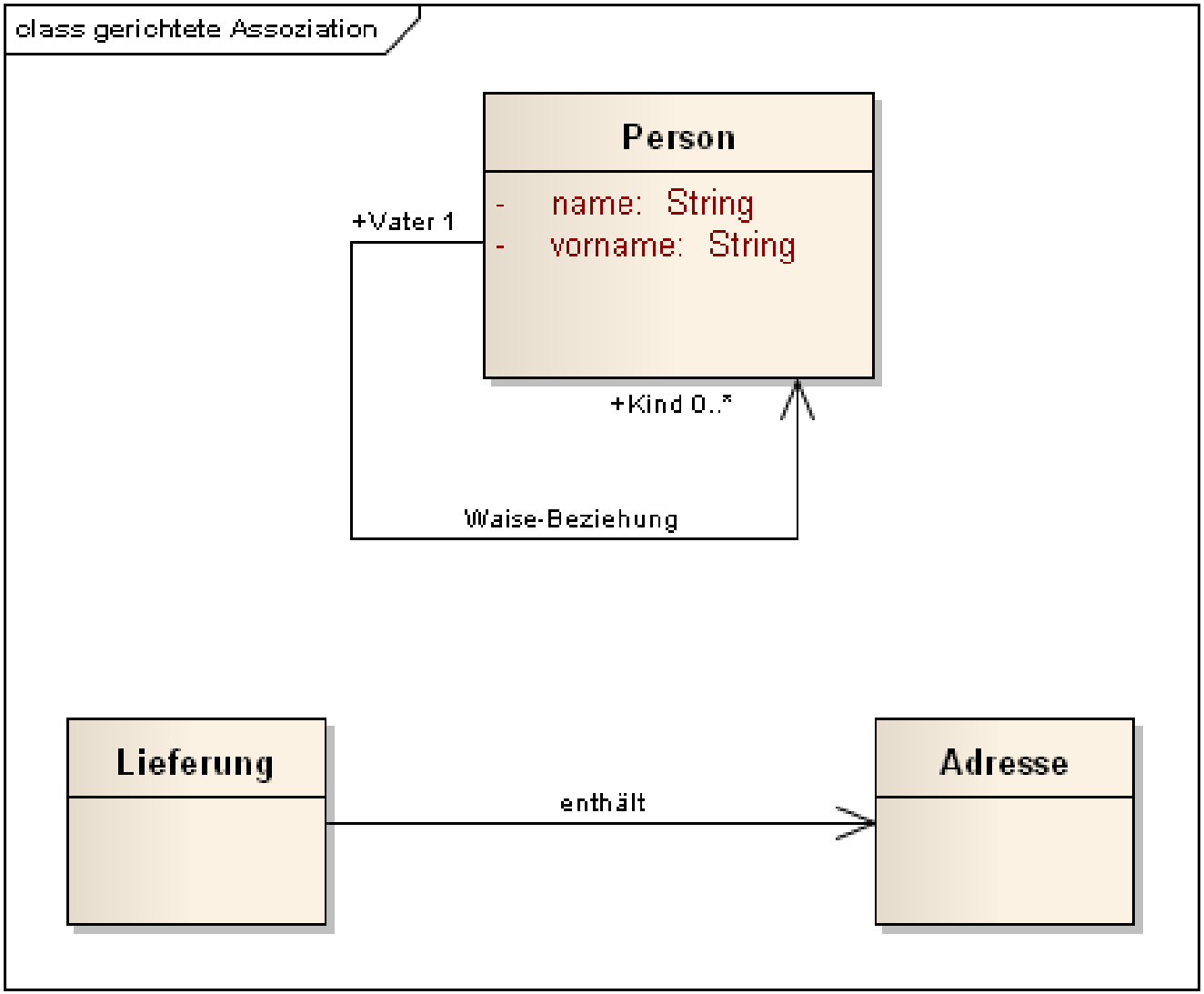


Abbildung: Die gerichtete Assoziation

Die Aggregation

Definition

Die Aggregation ist eine besondere Art der Assoziation zwischen Öbjekten". Ein Objekt ist dabei Teil eines anderen und beide können für sich existieren.

Die Aggregation

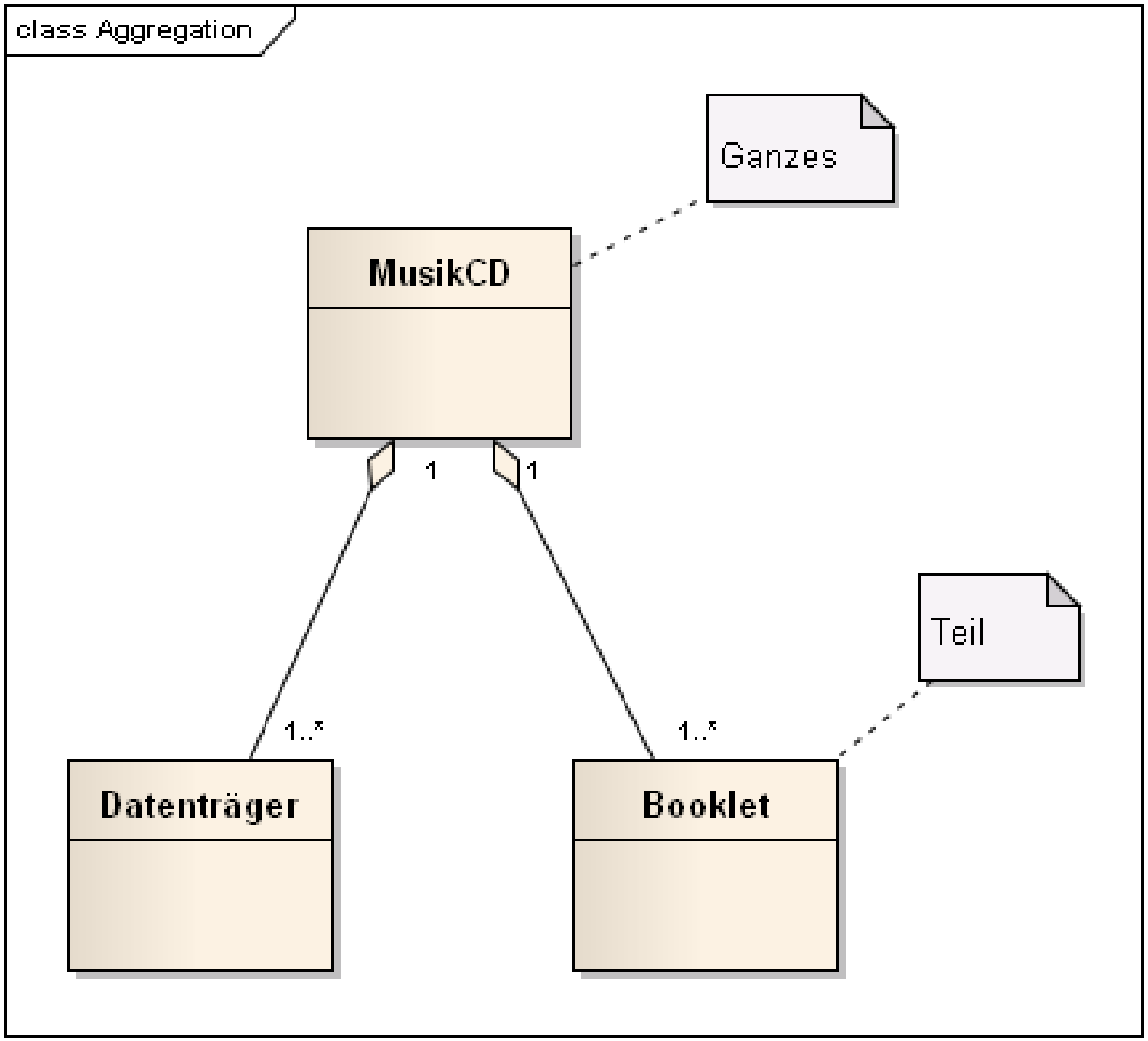


Abbildung: Die Aggregation

Die Komposition

Definition

Die Komposition ist eine besondere Art der Assoziation zwischen Öbjekten". Ein Objekt ist dabei Teil eines anderen, wobei das Teilobjekt nicht für sich existieren kann.

Die Komposition

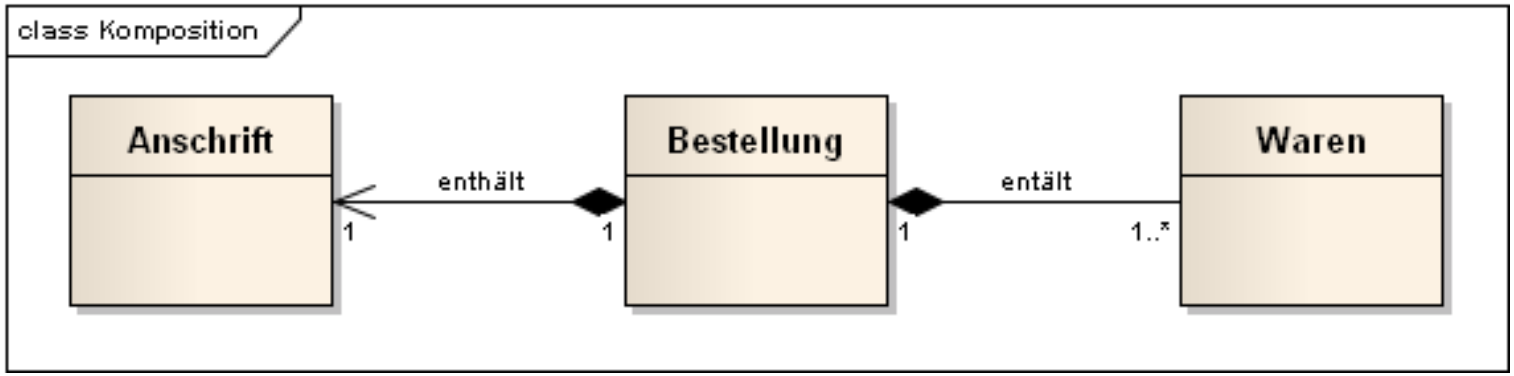


Abbildung: Die Komposition

Generalisierung und Spezialisierung

Definition

Die Generalisierung und die Spezialisierung stellt eine Ordnungsbeziehung zwischen einem allgemeinen und einem speziellen Element dar. Das speziellere Element fügt weitere Eigenschaften zu einem allgemeineren Element hinzu, verhält sich aber weiterhin kompatibel zu diesen.

- hierarchische Gliederung von Eigenschaften
- Oberklassen beinhalten allgemeine Eigenschaften, welche an die Unterklassen weitergegeben werden
- Unterklassen beinhalten spezifischere Eigenschaften sowie die Eigenschaften der Oberklasse
- Unterklassen können geerbte Eigenschaften der Oberklasse überschreiben, aber nicht entfernen
- Unterscheidung erfolgt aufgrund eines Charakteristikums (Diskriminator)
- die Menge aller Unterklassen, welche auf einem Diskriminator beruhen heißen Partition oder Generalisierungsmenge

Generalisierung und Spezialisierung- Notation

- großer nicht ausgefüllter Pfeil
- Richtung von der Unterklasse zur Oberklasse
- Zusammenfassung mehrerer Pfeile möglich
- Angabe des Diskriminators über den Pfeil
- als gestrichelte Linie zwischen den Pfeilen

Generalisierung und Spezialisierung - Beispiel

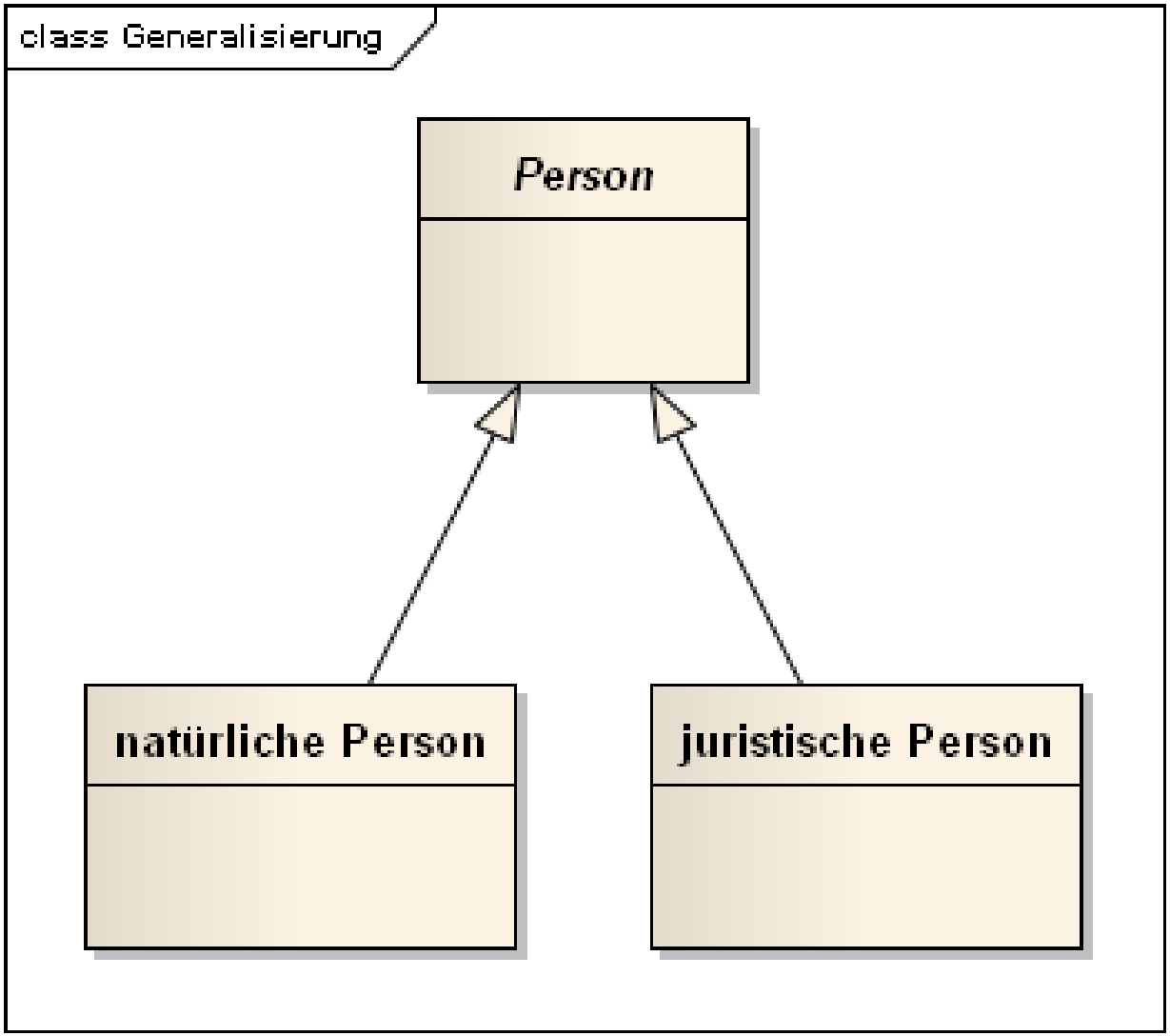


Abbildung: Generalisierung und Spezialisierung

Mehrfachvererbung

- bisher: Jede Unterklasse hat genau eine Oberklasse
- neu: Mehrfachvererbung - jede Unterklasse kann mehrere Oberklassen besitzen
- Programmiersprachenabhängig (nicht in Smalltalk, Java)
- Probleme: verschiedene Oberklassen beinhalten gleichnamige Eigenschaften
- Alternative: Delegation

Zusammenfassung

Im Vordergrund der Anforderungsermittlung stehen Problemadäquadheit und die Fragen:

- nach dem „Was“ - Funktionsumfang
- nach dem „Warum“ - Ziele
- nach dem „Womit“ - z. B. Objekte und Klassen der Problemwelt

Die Frage nach dem „Wie“ (konkrete Realisierung) spielt erst später Rolle!

- Sichtweisen auf Funktion, Struktur und Verhalten
- Validierung mit dem Anwender

Der Zustand

Definition

Ein Zustand ist eine Abstraktion einer Menge von Eigenschaften, welche ein Objekt durch die Belegung der Attribute einnehmen kann.

siehe UML Superstructure Specification v 2.2, S. 541 (10-05-05.pdf, S. 557)

- Zustand ergibt sich aus der Wertbelegung der Attribute, bei denen sich das Verhalten des Objektes stark ändert
- Besonderheiten sind Start- und Endzustände
- Modellierung nicht für jede Klasse notwendig
- Zustandsübergänge werden durch ein Ereignis ausgelöst
- das Ereignis hat einen Namen und mögliche Argumente
- die UML definierte Aktionen `entry`, `do`, `exit` Aktionen die automatisch ausgelöst werden

Der Zustand - Notation

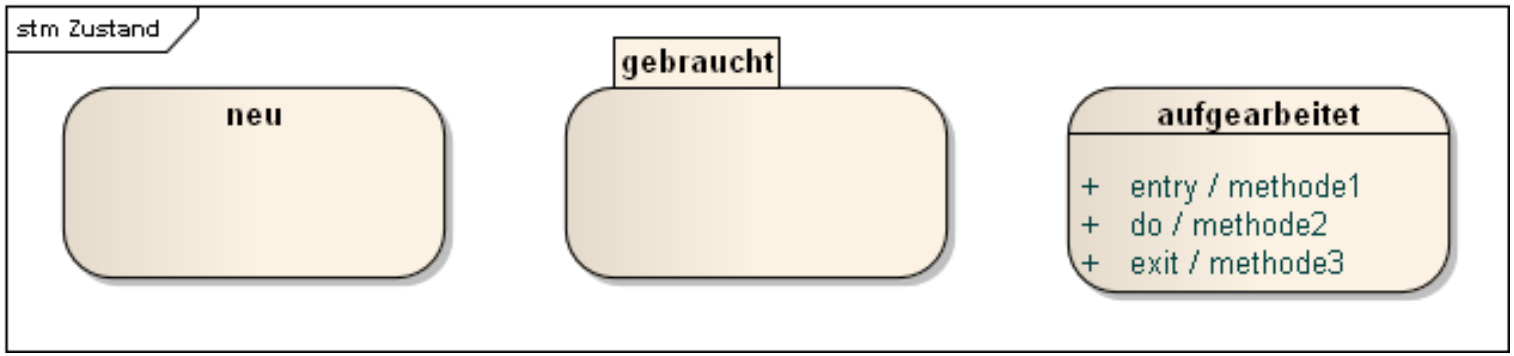


Abbildung: Der Zustand

Der Unterzustand

Definition

Zustände lassen sich schachteln. Der umschlossene Zustand ist der Unterzustand.

Varianten:

- sequentielle Verschachtelung
- konkurrierende Verschachtelung

Der Unterzustand

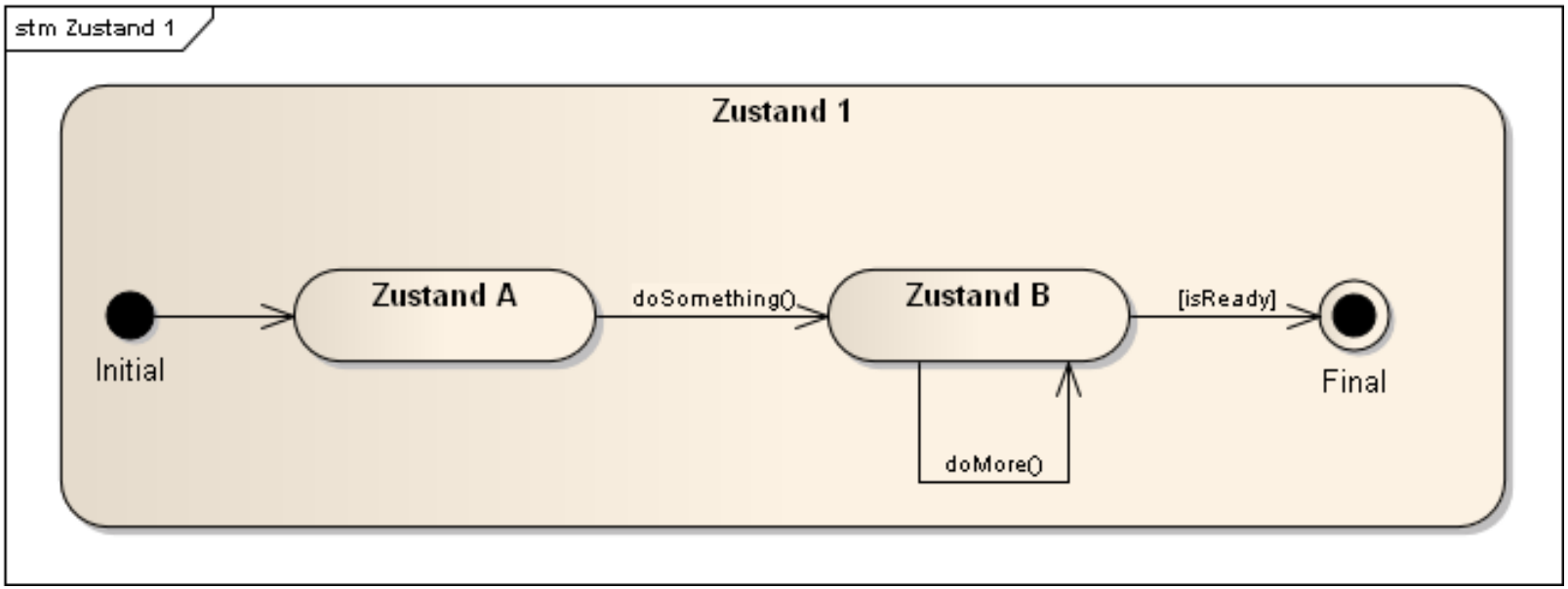


Abbildung: Verschachtelter Unterzustand

Der Unterzustand

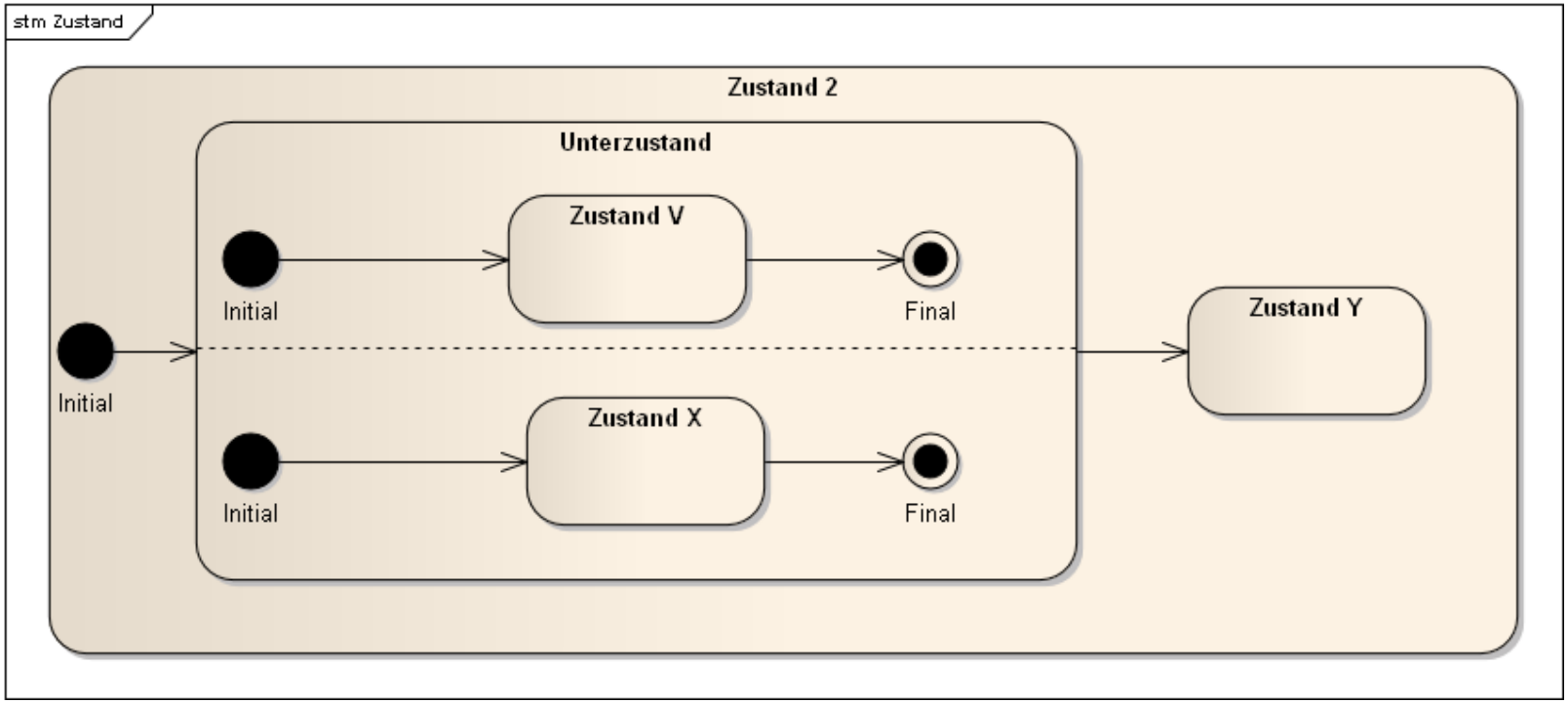


Abbildung: Paraleller Unterzustand

Das Ereignis

Definition

Ein räumlich und zeitlich einordenbares mit einer Bedeutung behaftetes Geschehen wird als Ereignis bezeichnet. Dieses Ereignis löst gewöhnlich einen Zustandsübergang (Transition) aus.

- Transitionsbeschreibung:
  `ereignis(args) [bedingung] /operation(args`
- Ursachen für ein Ereignis:
  - eine definierte Bedingung wird erfüllt
  - das betrachtete Objekt erhält eine Nachricht
- die Verarbeitung der Ereignisse ist abhängig vom Objektzustand
- abhängig vom Zustand kann ein Ereignis kann zu verschiedenen Aktionen führen

Das Ereignis

- Transitionen werden durch ein Ereignis ausgelöst und mit diesen beschriftet
- Transitionen ohne Beschriftung werden automatisch ausgelöst
- Ereignisse mit Bedingungen als Voraussetzung für einen Zustandswechsel
- `when (Ausdruck)`: beschreibt einen absoluten Zeitpunkt (day=today), dann feuert diese Transition
- `after(Ausdruck)`: relativer Zeitpunkt zum vorherigen Zustandswechsel

Das Ereignis - Notation

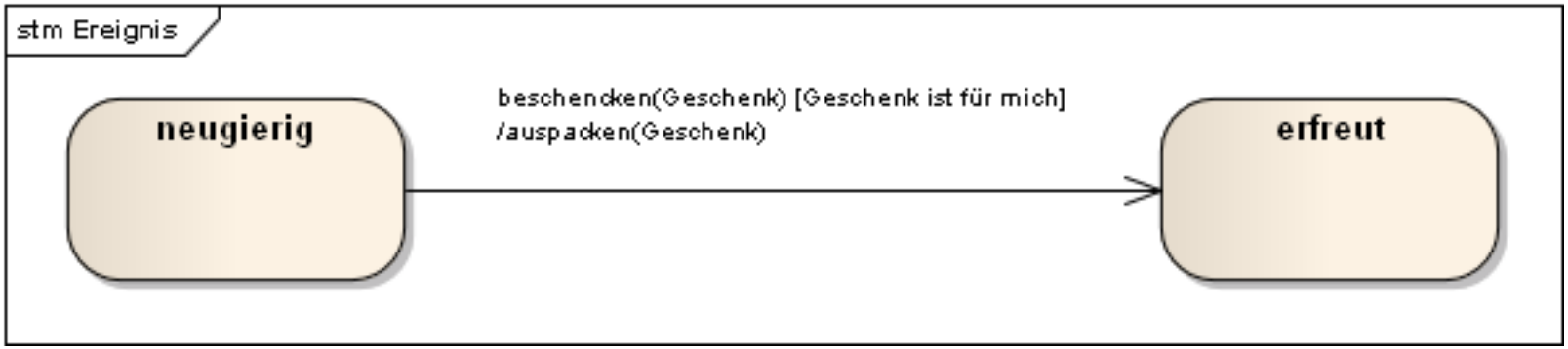


Abbildung: Das Ereignis

Das Zustandsdiagramm

Definition

In einem Zustandsdiagramm werden die möglichen Zustände eines Objektes und Ereignisse die zu einem Zustandswechsel führen über die Lebenszeit des Objektes dargestellt.

- Modell des endlichen Zustandsautomaten (finite state machine - FSM)
- Darstellung einer begrenzten und nicht leere Menge von Zuständen
- Darstellung der Start- und Endzustände
- Darstellung einer begrenzten Menge, nicht leere Menge von Ereignissen

Das Zustandsdiagramm

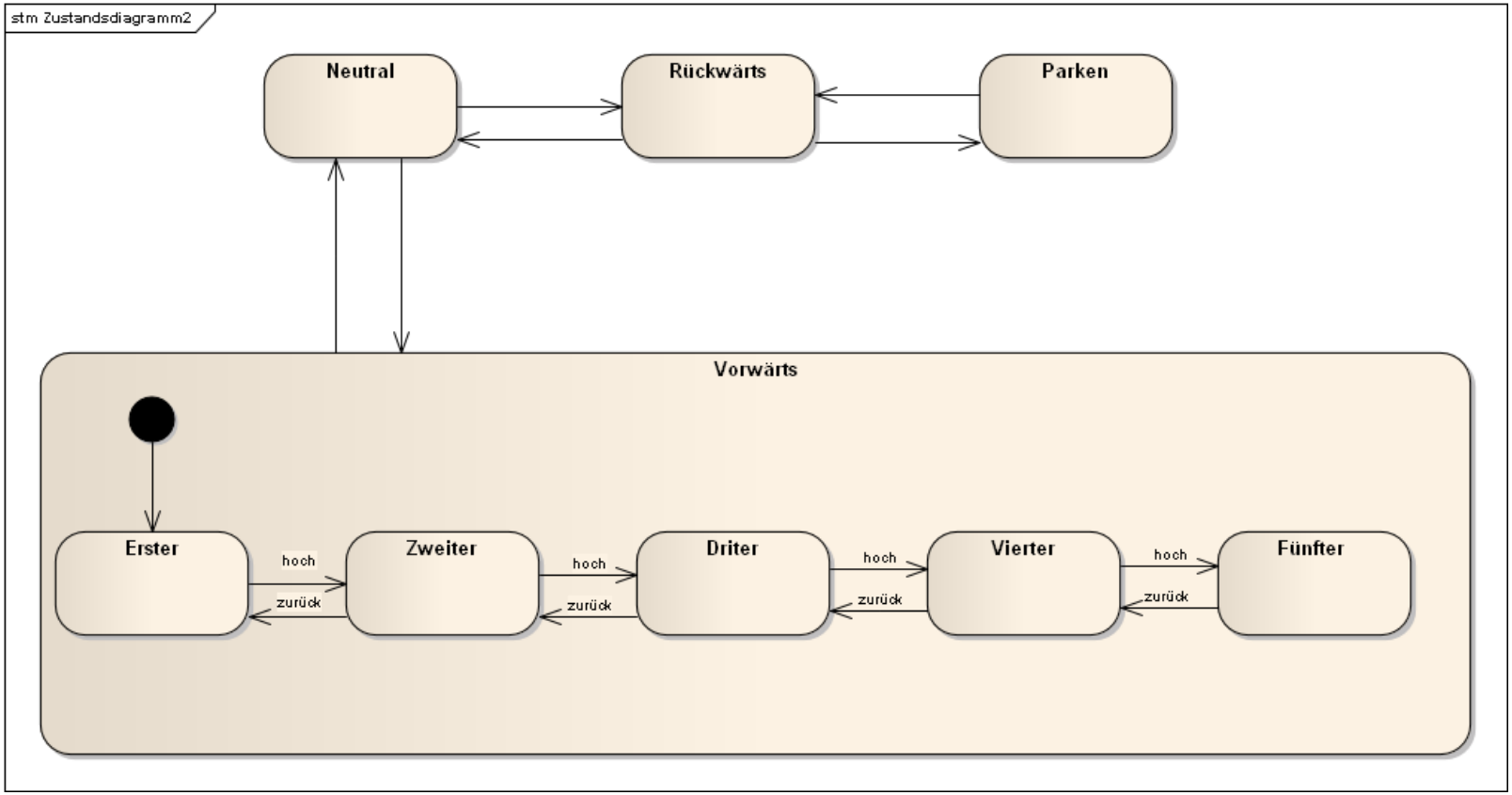


Abbildung: Das Zustandsdiagramm

Das Sequenzdiagramm

Definition

Das Sequenzdiagramm visualisiert Nachrichten, welche eine begrenzte Menge von beteiligten Akteuren oder Objekten in einer zeitlich begrenzten Situation austauscht.

siehe UML Superstructure Specification v 2.2, S. 473 (10-05-05.pdf, S. 489)

Im Vordergrund steht der zeitliche Verlauf der Nachrichten!

Das Sequenzdiagramm - Notation

Allgemeines zur Notation:

- Leserichtung von oben nach unten
- Sequenzdiagramme sind verschachtelbar
- eine Menge von verschiedenen Abläufen (Aktivitäten) wird in verschiedene Sequenzen zerlegt
- Darstellung ausgewählter besonderer Abläufe

Das Sequenzdiagramm - Notation

Objekte:

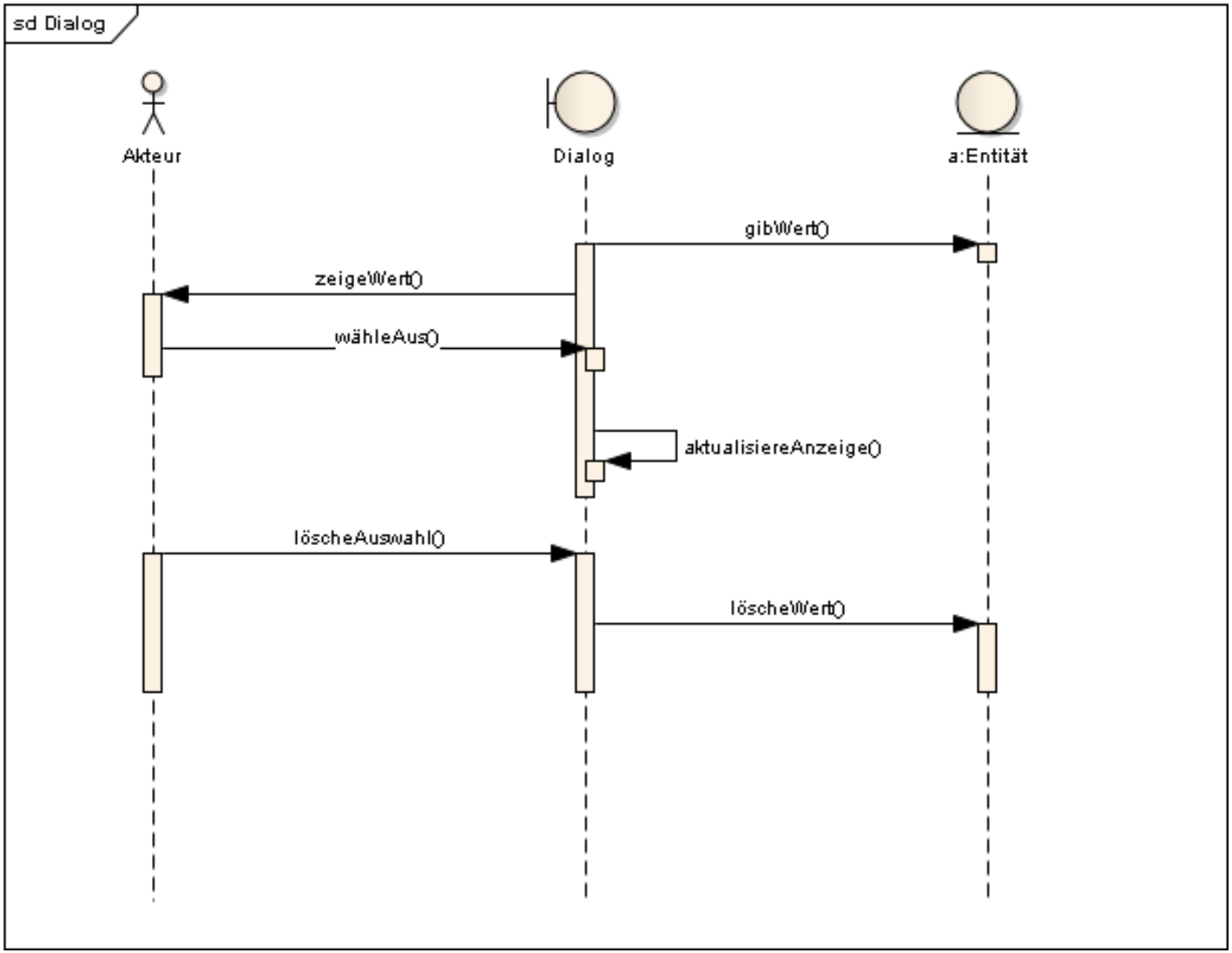
- Objekte werden oben durch das entsprechende Objektsymbol bzw. als Rechteck und eine senkrechte gestrichelte Linien dargestellt.
- Objektkonstruktion kann durch eine Nachricht dargestellt werden
- Objektdesktuktion kann durch ein Kreuz am Ende der Lebenslinie dargestellt werden
- Objektzustände werden auf der Lebenslinie dargestellt

Das Sequenzdiagramm - Notation

Antworten:

- Antworten auf Nachrichten sind optional
- Darstellung erfolgt als gestrichelte Linie mit offener Pfeilspitze

Das Sequenzdiagramm - Beispiel



Das Sequenzdiagramm - Notation

Nachrichten:

- Nachrichten werden als waagerechte Pfeile zwischen den Lebenslinien der Objekte dargestellt. Die dazugehörige Nachricht wird in der Form nachricht(argumente) über den Pfeil notiert.
- synchrone Nachrichten haben gefüllte Pfeilspitzen
- asynchrone Nachrichten haben offene Pfeilspitzen

Das Sequenzdiagramm - Notation

Steuerungsfokus:

- Angabe Steuerungsfokus ist optional
- Kennzeichnet aktive Objekte bzw. solche die die Programmkontrolle haben
- Darstellung erfolgt als nicht ausgefüllter senkrechter Balken an der Stelle der Lebenslinie

lokale Attribute:

- die Angabe erfolgt oben links im Diagramm
- Schleifenzähler

Fragen

Zeit für Ihre Fragen!

Quellenverzeichnis

[Axe08] AxelScheithauer:
Historie der objektorientierten Methoden und Notationen.
http://de.wikipedia.org/wiki/Datei:OO-historie.svg.
http://de.wikipedia.org/wiki/Datei:OO-historie.svg.
Version: August 2008. –
Zugriff: 08.09.2009