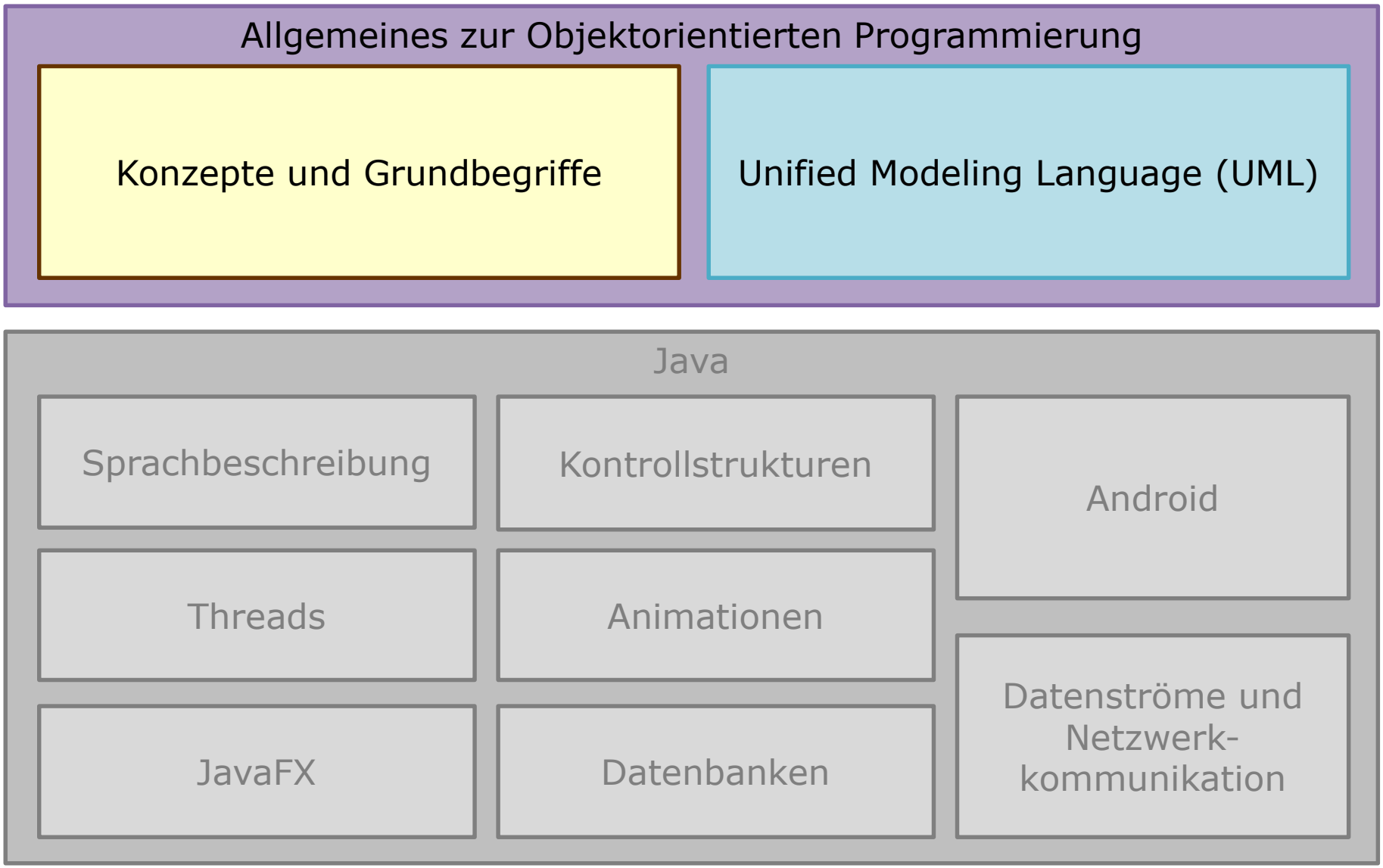
BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Objektorientierte Programmierung

Unified Modeling Language (UML)

Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-sachsen.de

Aufbau der Lehrveranstaltung



Das Design-Problem

UML – Steckbrief (1/2)

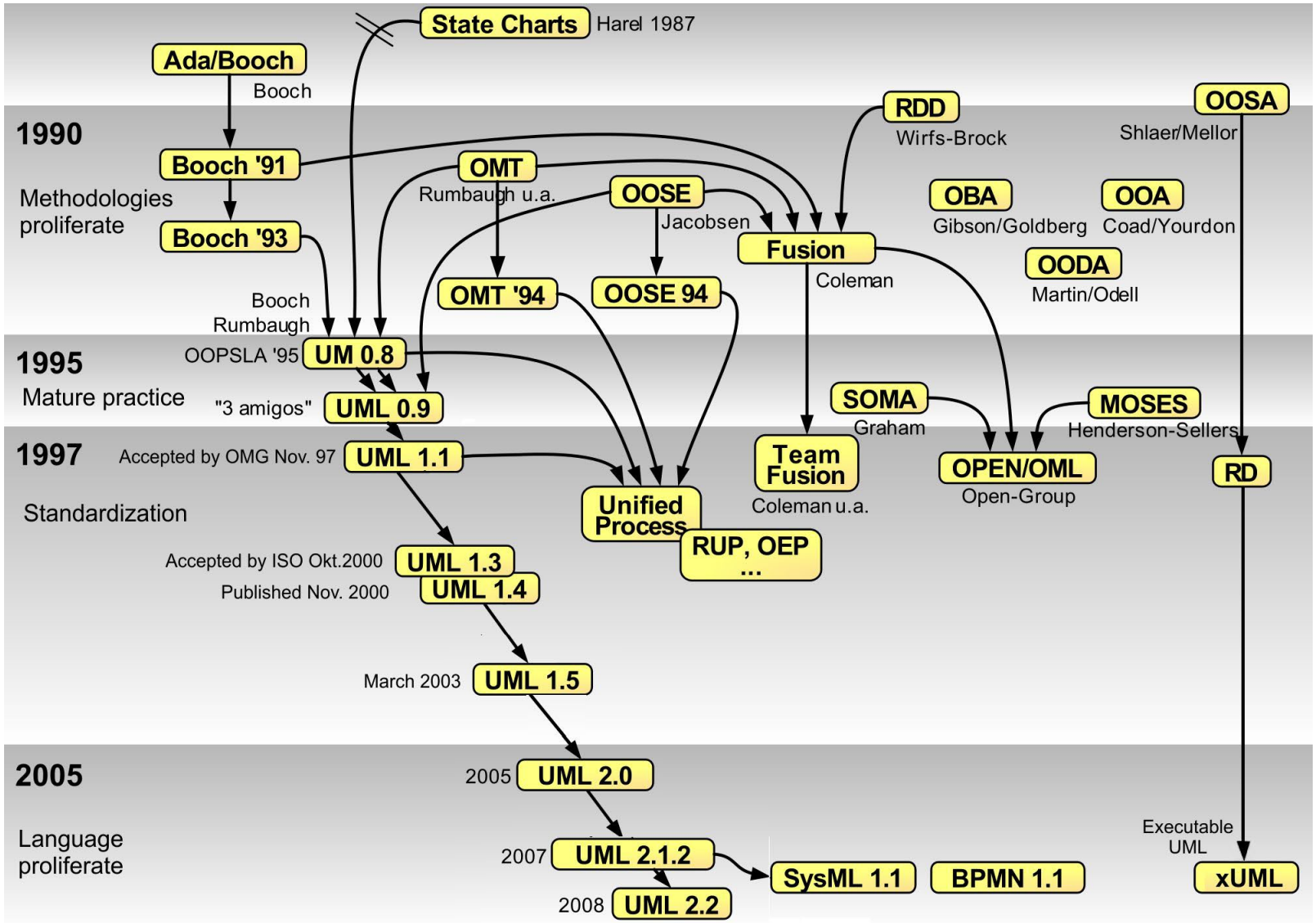
- Visuelle Modellierungssprache
- Dient dazu, komplexe Softwaresysteme zu
  - modellieren
  - dokumentieren
  - spezifizieren
  - visualisieren
- Unterstützung von Objektorientierter Analyse und Design (OOA/OOD) sowie Architekturspezifikation von Softwaresystemen
- Industriestandard der Object Management Group (OMG; seit 1989)

UML – Steckbrief (2/2)

UML ist…

- …weder vollständig noch konsistent
- …keine Programmiersprache
- …keine rein formale Sprache
- …nicht domänenspezifisch
- …kein Ersatz einer textuellen Beschreibung
- …keine Methode oder Vorgehensmodell

UML – HISTORIE



Geburt der UML 2

- vier konkurrierende Vorschläge zur UML 2 Spezifikation
- akzeptierter Vorschlag vom U2 Partners-Konsortium
  - u.a. Alcatel, HP, Motorola, Oracle, Rational, Ericsson, DaimlerChrysler, France Telecom, …
- „Ziel: Vermeidung des Second System Syndroms bei der UML 2
  - Entfernung von bislang kaum verwendeten Sprachelementen
  - Entfernung von zu methoden- oder sprachspezifischen Elementen
  - Entfernung von Elementen ohne präzise Semantik

UML 2 – wesentliche Neuerungen

- Neuformulierung des Metamodells (der „Grammatik“) der UML
- stärkere Verknüpfung der Diagrammtypen
- bessere Unterstützung von Echtzeit- und Komponentensystemen
- Schärfung der Semantik, wo nötig
- …

Struktur der UML 2

- UML-2-Standard umfasst zwei sich ergänzende Dokumente
  - Infrastruktur
    - grundlegende Sprachkonstrukte
    - Basisarchitektur
  - Superstruktur
    - Diagrammnotation
    - Semantik
- aktuelle Version: UML 2.5.1 (Dezember 2017)
- aktuelle Infrastruktur- und Superstrukturstandards: UML 2.4.1

Nutzen der UML-Modellierung

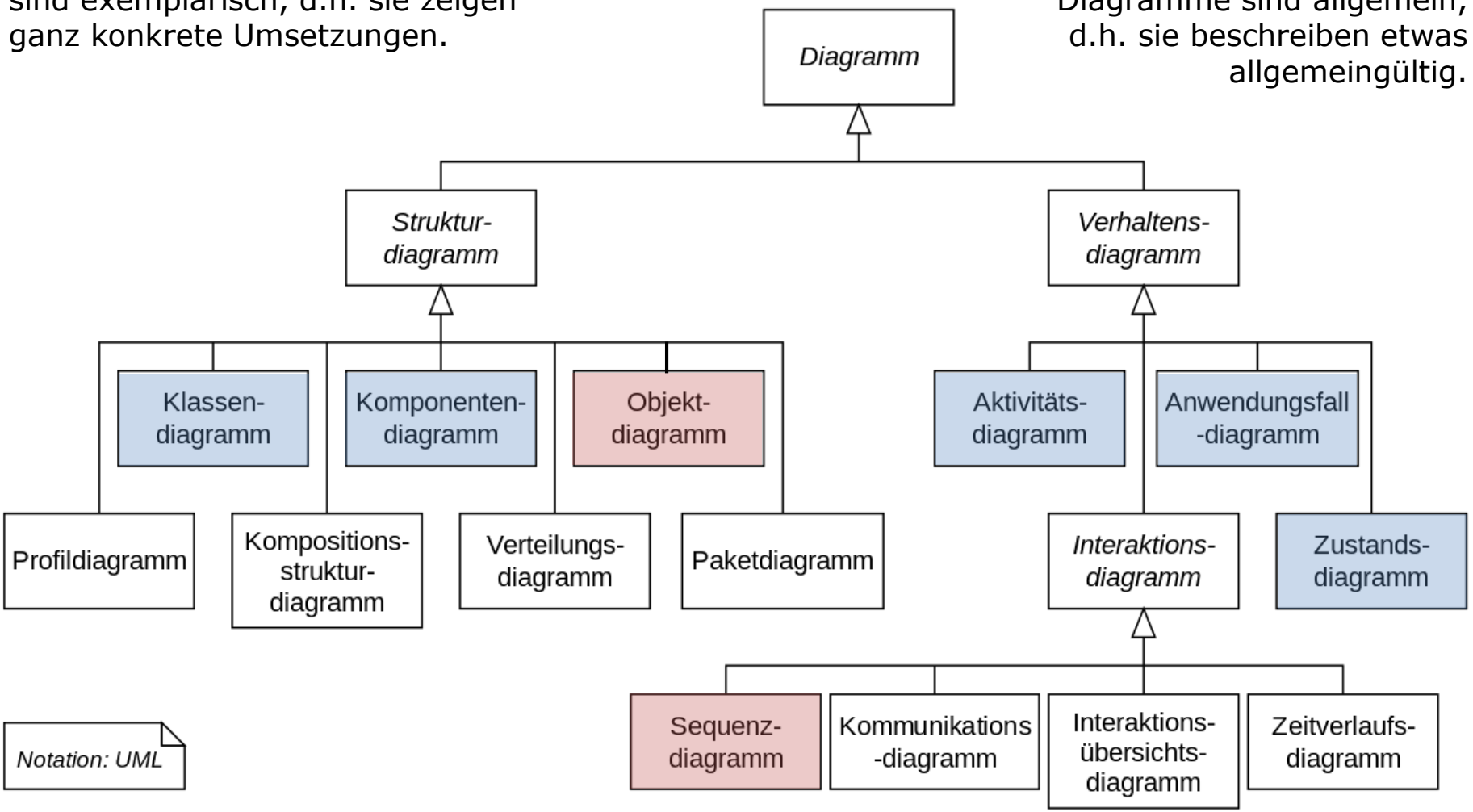
- Einsatz der UML muss an die Entwicklungsmethode angepasst werden
  - Wasserfallmodell, V-Modell, Spiralmodell, agile Methoden, …
- Wissen explizieren (d.h. zu Papier bringen) ist wichtig
  - Kommunikationshilfe
  - Reflektion und Klärung bisher unklarer Sachverhalte
  - Diskussionsgrundlage
- grafische Modelle helfen bei der Analyse der Auswirkungen von Anforderungsänderungen
  - Systemanforderungen stehen selten zu Projektbeginn fest
    - Hinzulernen der Projektbeteiligten
    - Änderungen der Projektumgebung

ausgewählte Typen von UML-Diagrammen

UML 2 – Übersicht

Die rot markierten Diagramme sind exemplarisch, d.h. sie zeigen ganz konkrete Umsetzungen.

Die blau markierten Diagramme sind allgemein, d.h. sie beschreiben etwas allgemeingültig.



https://commons.wikimedia.org/wiki/File:UML-Diagrammhierarchie.svg

UML 2 – Struktur- und Verhaltensdiagramme

- Strukturdiagramme
  - Strukturdiagramme zeigen statische Modellelemente und ihre Beziehungen → Organisation und Abhängigkeiten
    - *Klassendiagramm*
    - *Paketdiagramm*
    - Objektdiagramm
    - Kompositionsstrukturdiagramm
    - *Komponentendiagramm*
    - Verteilungsdiagramm
    - Sonderstellung: Profildiagramm (seit UML 2.2)
- Verhaltensdiagramme
  - Verhaltensdiagramme zeigen Zustände, Abläufe oder Kommunikation → Änderungen und Aktionen
    - Use-Case-Diagramm
    - Aktivitätsdiagramm
    - Zustandsautomat
  - Interaktionsdiagramme
    - Sequenzdiagramm
    - Kommunikationsdiagramm
    - Timing-Diagramm
    - Interaktionsübersichts-diagramm

Vollständige Spezifikation

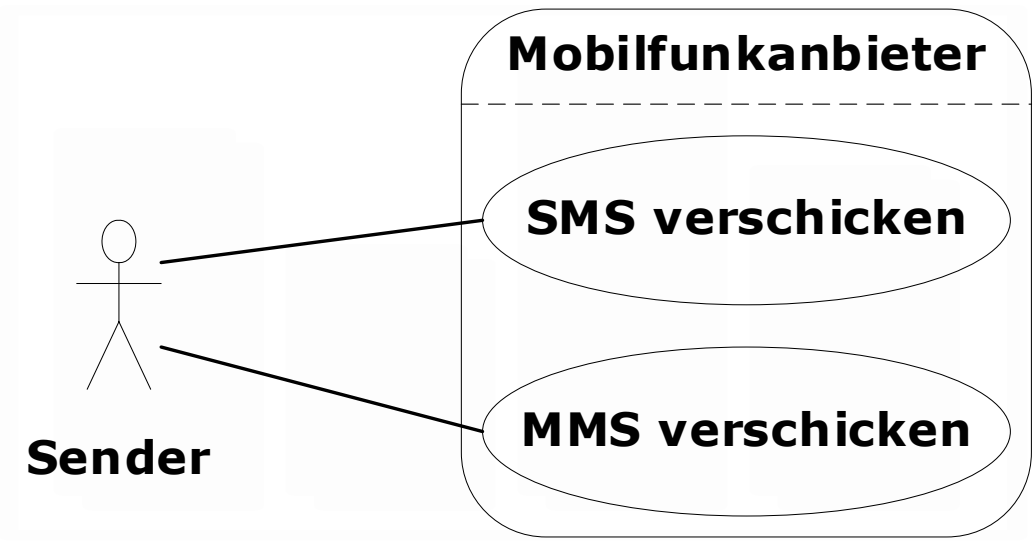
- UML ist zu komplex, um in dieser Lehrveranstaltungsreihe vollständig erklärt zu werden
- Fokus auf die wichtigsten Aspekte
- ausführliche Erklärungen zum selbständigen Lernen:
  - https://www.visual-paradigm.com/guide/
  - https://www.omg.org/spec/UML/

Use-Case-Diagramm

USE-CASE-DIAGRAMM (ANWENDUNGSFALL-DIAGRAMM)

Ziel: Darstellung der Funktionen des Systems im Kontext seiner Umwelt

- Visualisierung der Außensicht des Systems
- verwendbar zur Kontextabgrenzung
- sehr hohes Abstraktionsniveau, relativ einfache Notationsmittel



Use-Case-Diagramm (Anwendungsfall-Diagramm)

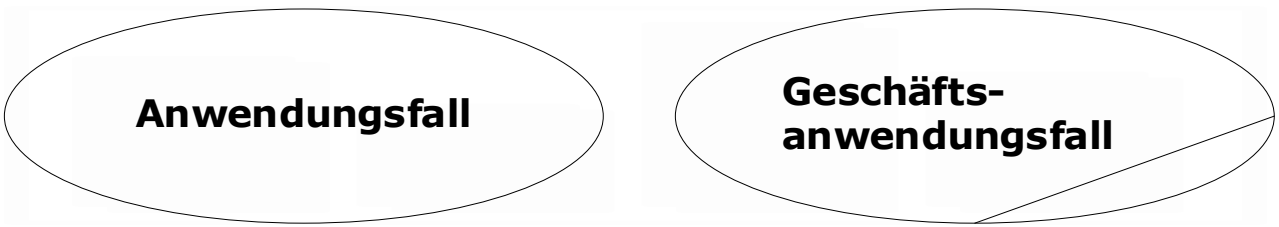
- Was soll das System können? Wie soll es nach außen reagieren?
  - beschreiben das (erwartete) Verhalten des Systems
  - repräsentieren Sicht der zukünftigen Anwender
  - sind wichtiges Kommunikationsmittel
  - dienen der Anforderungsermittlung und der Vermeidung von Missverständnissen
  - beschreiben keine chronologischen Abläufe
- Use-Case vs. User-Story
  - User-Storys sind kleiner → einzelne Szenarien eines Use-Cases
  - User-Storys sind Planungsinstrument für die „Iteration“
  - Use-Cases beschreiben (langlebig) umzusetzenden Zusammenhang

USE-CASE-DIAGRAMM (ANWENDUNGSFALL-DIAGRAMM)

Vorgang, zeitlich ununterbrochene Interaktion

- Geschäftsanwendungsfall: abstrakter Vorgang aus Firmensicht
- (System-)Anwendungsfall: konkreter Vorgang des Softwaresystems

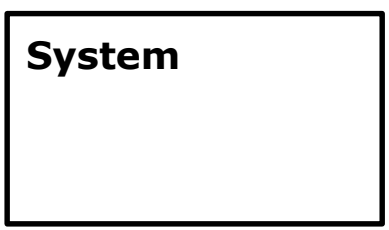
Notation:



Akteure

- sind Nutzer des Systems (auch Fremdsysteme)
- Abstrakt: nicht „Frau Müller", sondern „Controller"

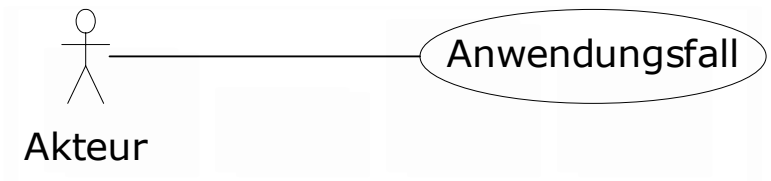
Notation:



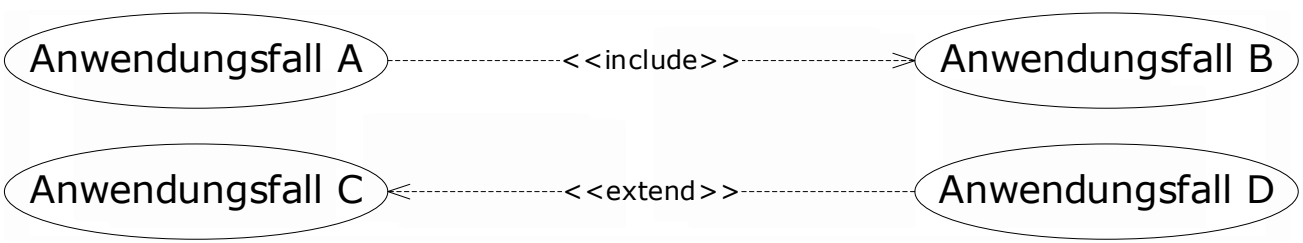
USE-CASE-DIAGRAMM (ANWENDUNGSFALL-DIAGRAMM)

Assoziationen

- Beziehung zwischen Akteur und Anwendungsfall



- Beziehung zwischen Anwendungsfällen
  - Anwendungsfall A schließt Anwendungsfall B mit ein
  - Anwendungsfall D erweitert Anwendungsfall C



Aktivitätsdiagramm

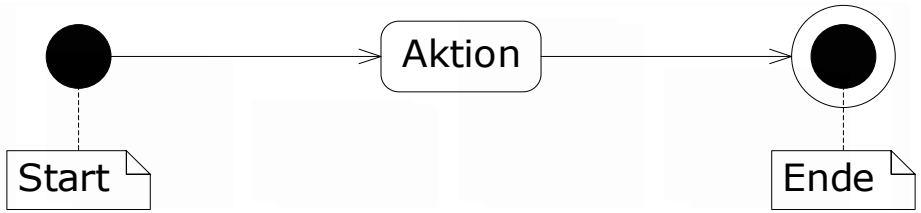
Aktivitätsdiagramm

Ziel: Darstellung eines flussorientierten Prozesses oder Algorithmus'

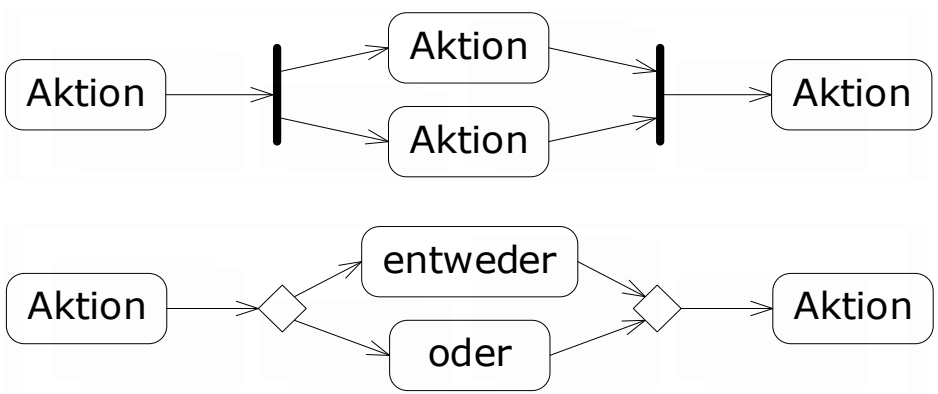
- detaillierte Darstellung von Abläufen (Bedingungen, Verzweigungen, Schleifen, Parallelität, Synchronisation)

- Was passiert beim Ablauf?
  - stellen Verläufe dar, bspw. Ablauf eines Anwendungsfalls
  - verbindliches Festlegen der Aktionsabfolge im Kontrollfluss/Datenfluss
  - verschiedene Darstellungsebenen möglich
  - sehr universell einsetzbar
  - bestehen aus Knoten und Kanten

Elemente Aktivitätsdiagramm – Knoten



- Aktion: elementarer Verhaltensbaustein
- Kontrollknoten
  - Start- und Endknoten (siehe oben)
  - Parallelisierung- und Synchronisationsknoten („Trennlinie“)
  - Verzweigungs- und Verbindungsknoten („Entscheidungsbox“)



Elemente Aktivitätsdiagramm – Knoten

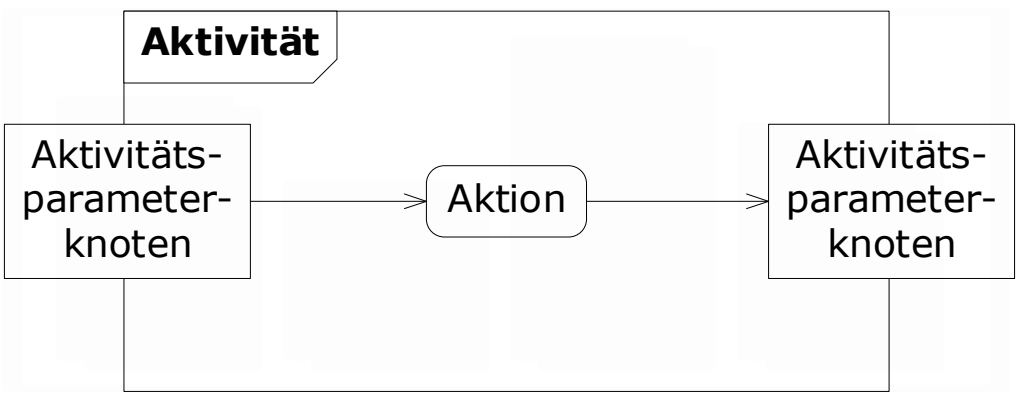
Objektknoten

- dienen dem Zwischenspeichern von Objekten im Objektfluss (Datenfluss)
- Kästchen ohne Rundung: Daten (Objekte)

Darstellung

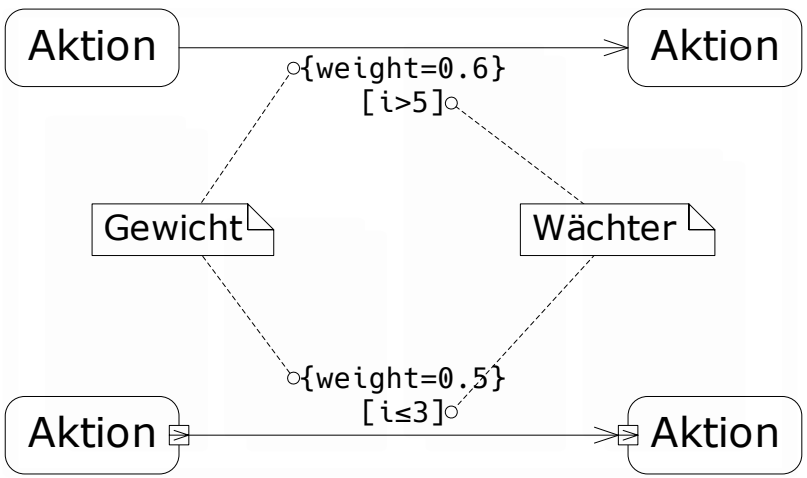
- Pins: gehören zu einer bestimmten Aktion

- Aktivitätsparameterknoten: Objekte an Aktivität übergeben

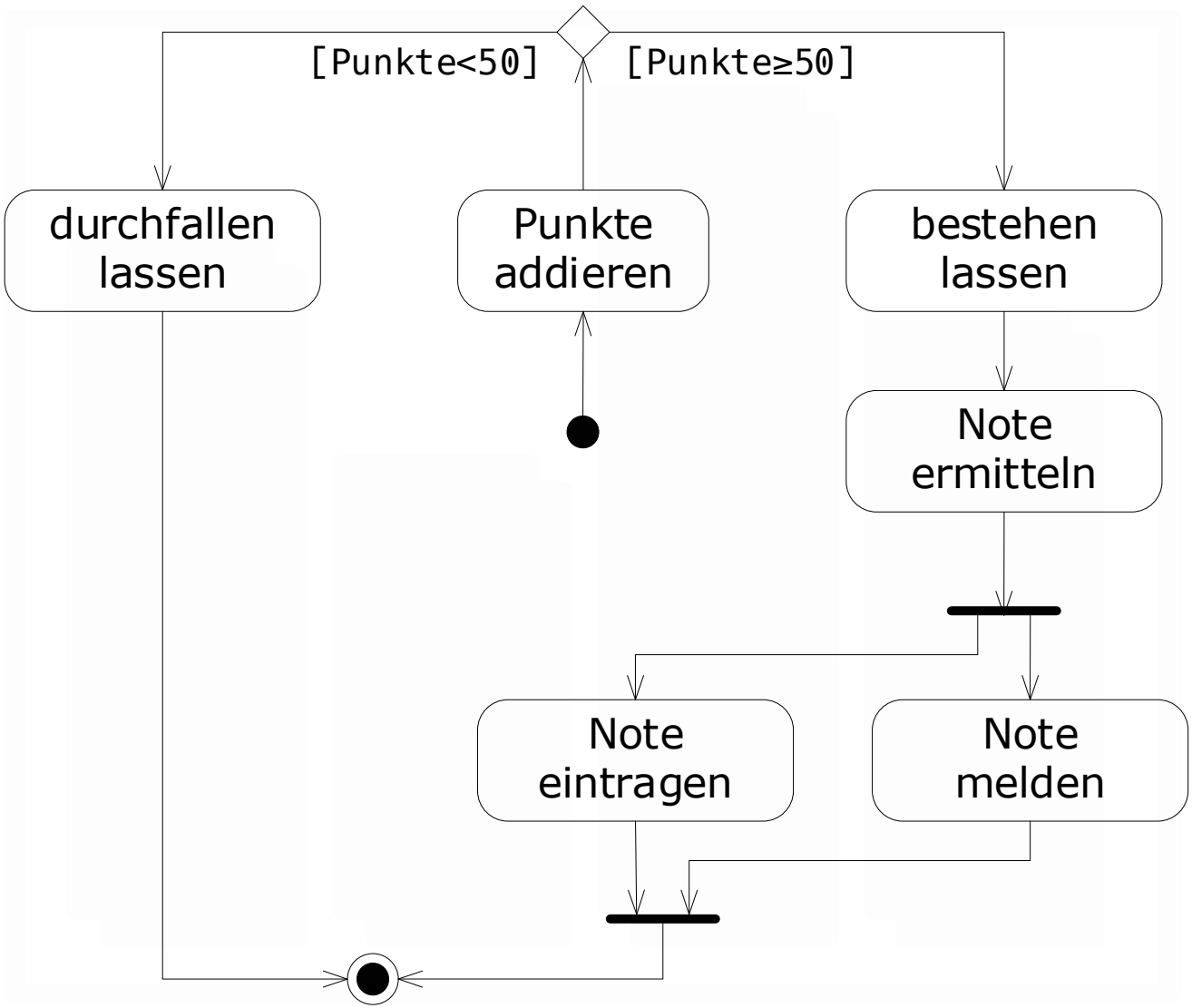


Elemente Aktivitätsdiagramm – Kanten (Pfeile)

- Kontrollflusskante
  - Übergang zwischen Aktionen oder Aktion und Kontrollknoten
  - Gerichtet (Pfeile)
  - können Gewicht haben
  - können einen Wächter haben
- Objektflusskante
  - Übergang zwischen Objektknoten oder Objektknoten und Kontrollknoten



Aktivitätsdiagramm – Beispiel

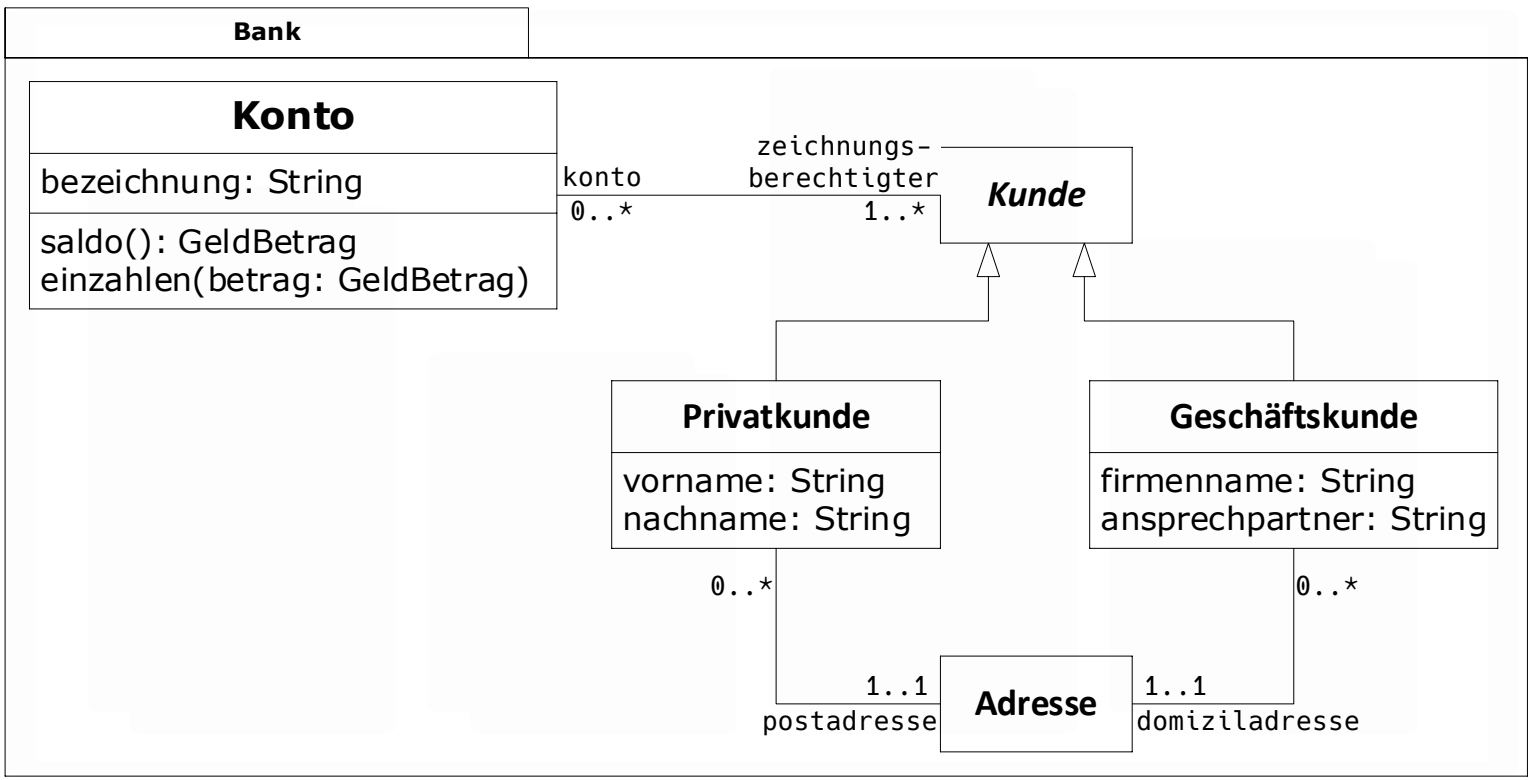


Klassendiagramm

KLASSENDIAGRAMM

Ziel: Modellierung der Beziehungen zwischen Systemklassen

- statische Struktur des abzubildenden Systems
- Brücke zu den dynamischen Diagrammen



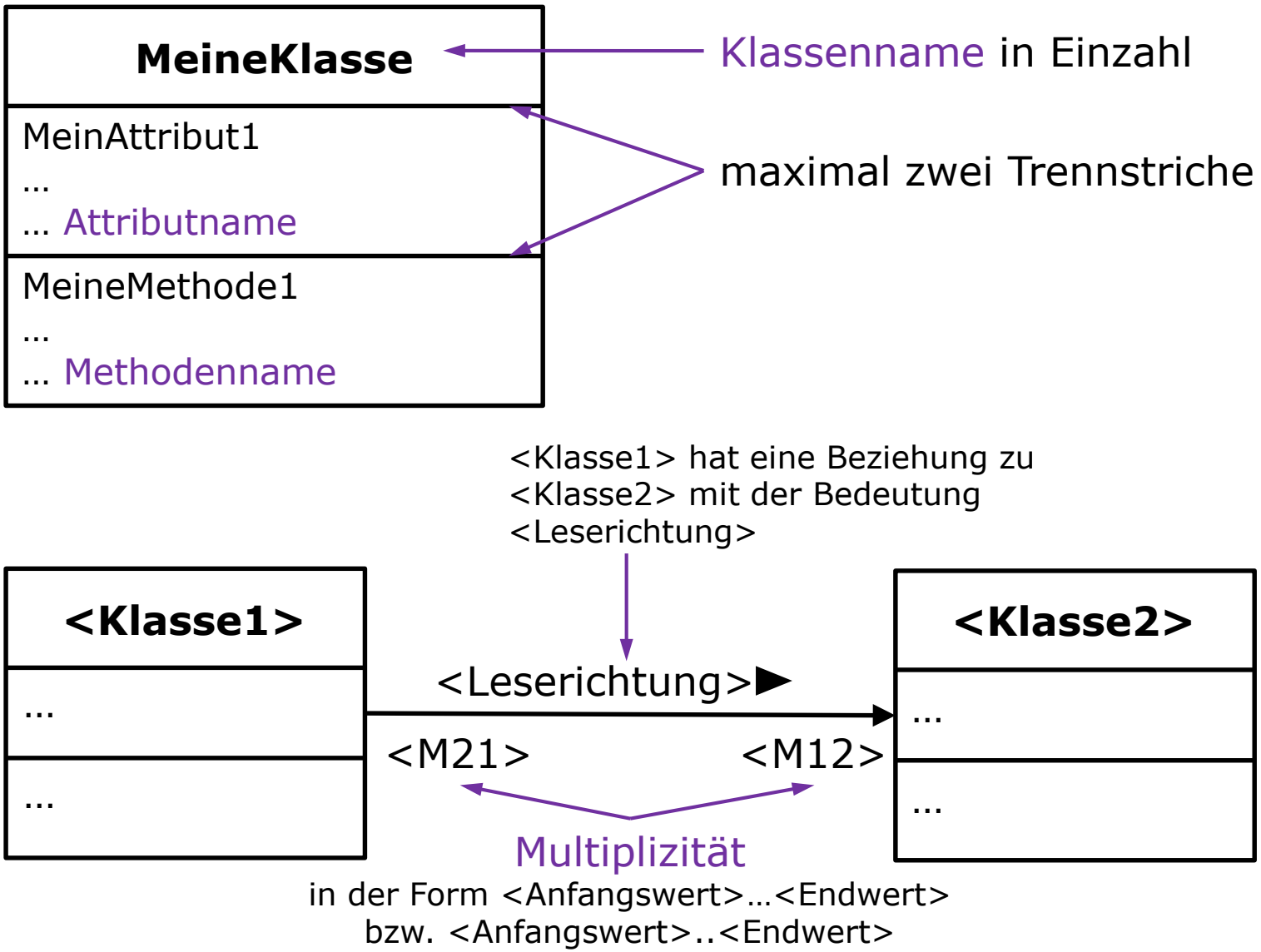
KLASSENDIAGRAMM

- Wie relevante Objekte unabhängig von der konkreten Instanz darstellen?
- Wie deren Beziehungen strukturieren?
- Wie deren Funktionalität abbilden?
  - Klassendiagramm ist das wichtigste Strukturdiagramm in der OOP
  - angepasst an die Phasen der Entwicklung
    - Analyseklassendiagramm
    - Entwurfsklassendiagramm

Klassendiagramm – Analyseklassendiagramm

- beschreibt den Problembereich
- legt abzubildende Eigenschaften fest
- visualisiert Zusammenhänge und Mengenverhältnisse zwischen Objekten
- legt Aufgaben (Funktionen) von Klassen fest
- in der Analysephase keine Implementierungsdetails wie
  - Sichtbarkeiten
  - Datentypen
  - Funktionsparameter

Klassendiagramm – Notation Analyseklassendiagramm



Klassendiagramm – Beziehungen

Assoziation

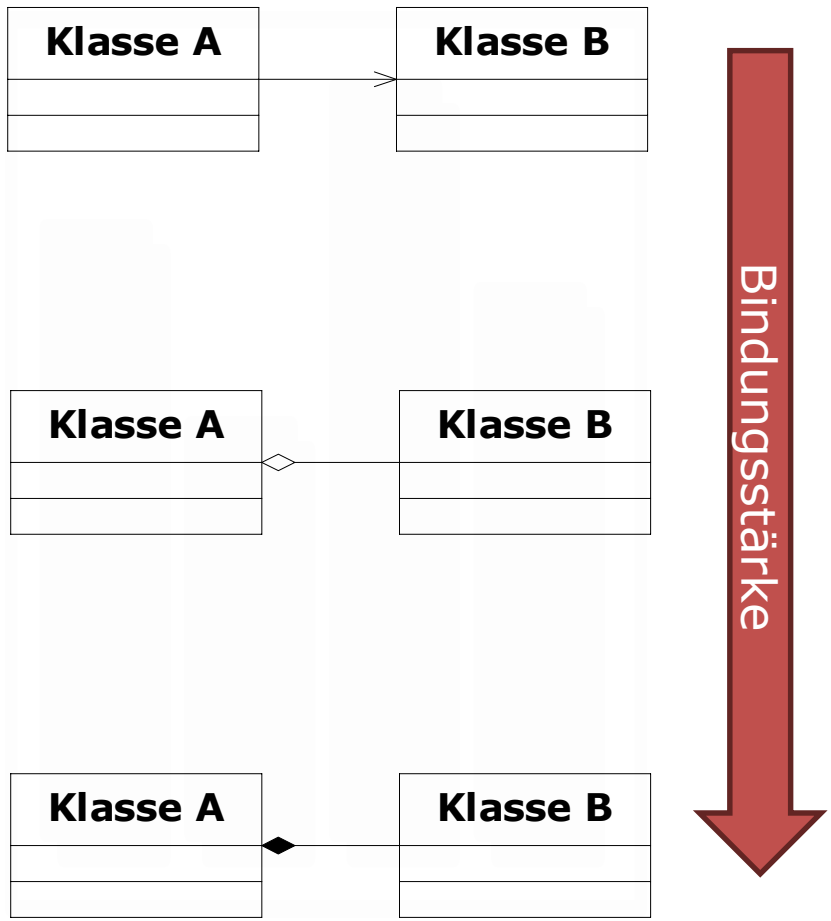
- Klasse A kennt Klasse B
- Klasse A hat Beziehung zu Klasse B

Aggregation

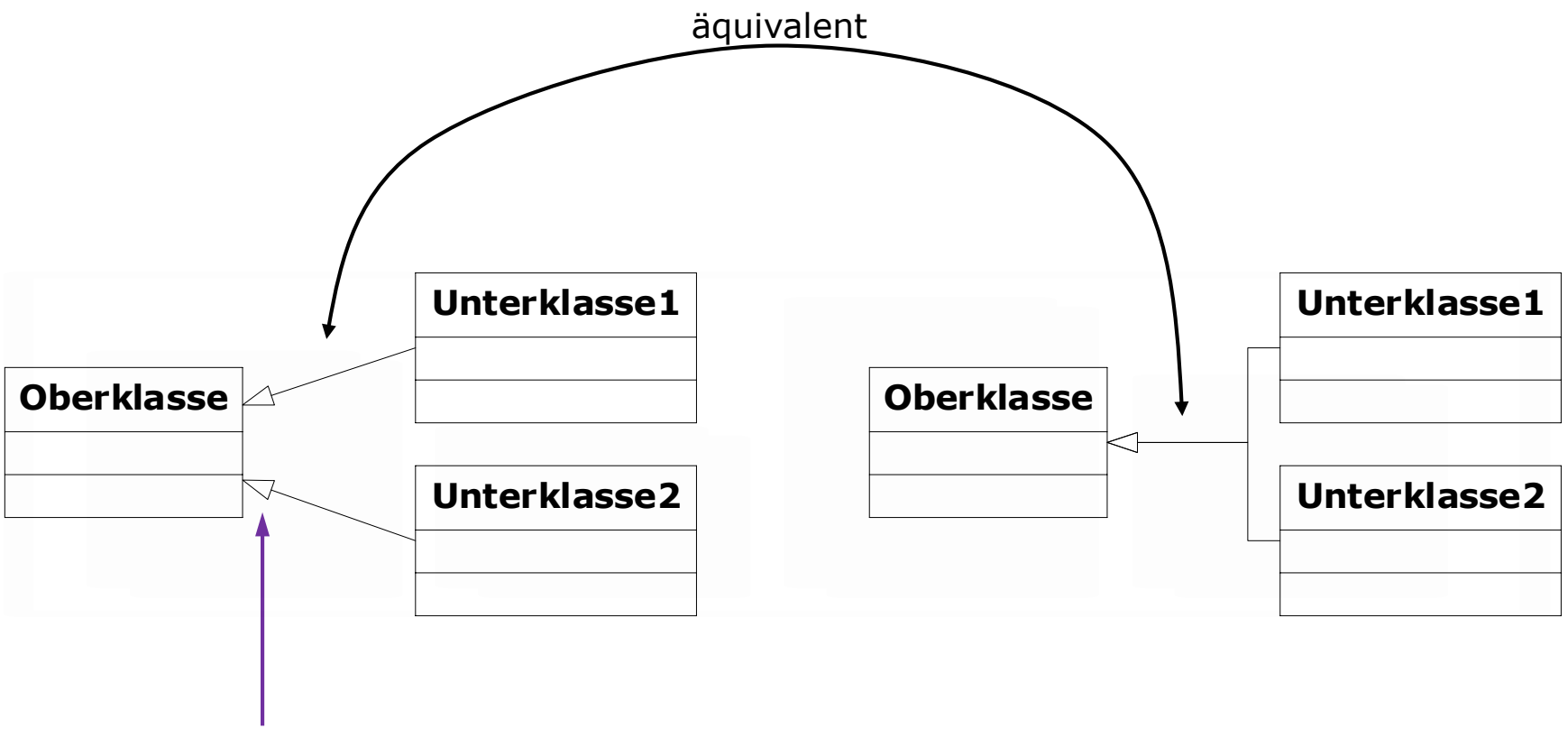
- Klasse B ist Bestandteil von Klasse A
- Klasse A besitzt/enthält Klasse B

Komposition

- Klasse B ist existenz-relevanter Bestandteil von Klasse A
- Klasse B ist Teil von Klasse A



KLASSENDIAGRAMM – VERERBUNG



Pfeil mit leerer geschlossener Spitze:
<Unterklasse> erbt von <Oberklasse>

Klassendiagramm – Entwurfsklassendiagramm (1/3)

Wie kann ich die in der Analyse gefundene Struktur so erweitern, dass sie direkt in Quellkode überführt werden kann?

Welche Modellelemente sind zusätzlich notwendig, damit für die Implementierung alles vollständig beschrieben ist?

- Festlegung der Zielsprache und der verwendeten Bibliotheken
- alle implementierungsspezifischen Details
- Vervollständigung von bisher offen gelassenen Modellaspekten
- Leserichtungen werden zu Rollennamen
- Sichtbarkeiten
- Funktionen
- Attributen
- Rollen

Klassendiagramm – Entwurfsklassendiagramm (2/3)

Welche Modellelemente sind zusätzlich notwendig, damit für die Implementierung alles vollständig beschrieben ist?
(fortgesetzt…)

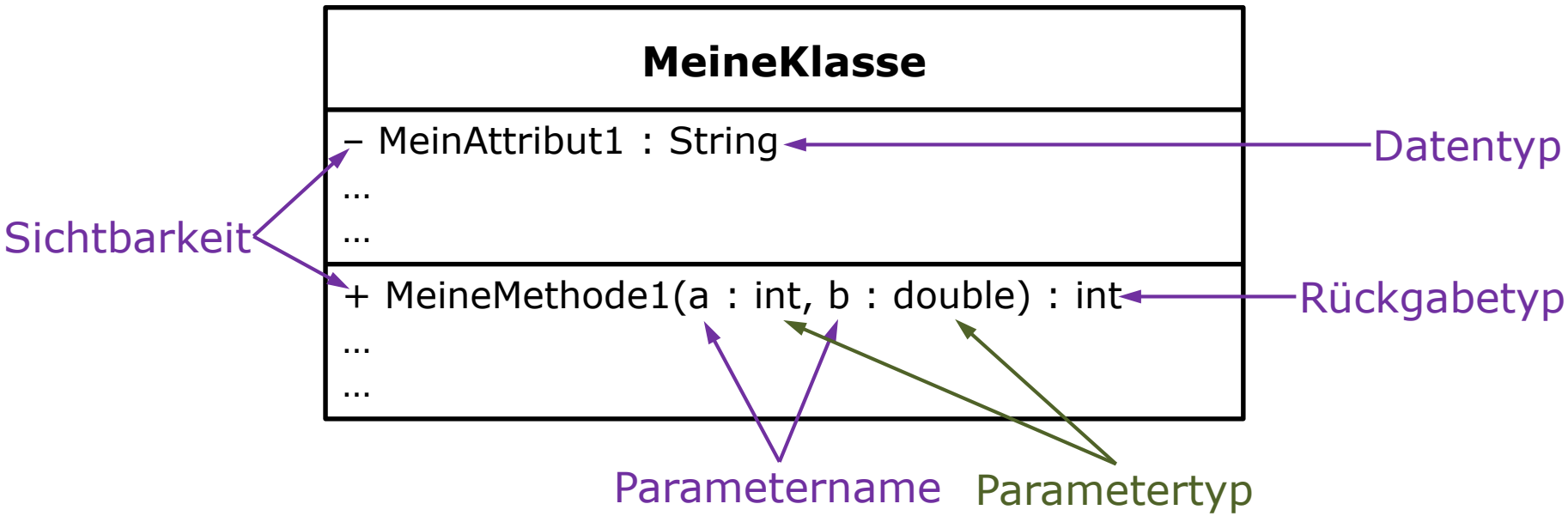
- Parameter der Funktionen
- Datentypen von
  - Parametern
  - Funktionen (Rückgabetyp)
  - Attributen
- abstrakte Klassen und Funktionen
- Schnittstellen
- Enumerationen
- Abhängigkeiten zu Bibliotheken usw.

KLASSENDIAGRAMM – ENTWURFSKLASSENDIAGRAMM (3/3)

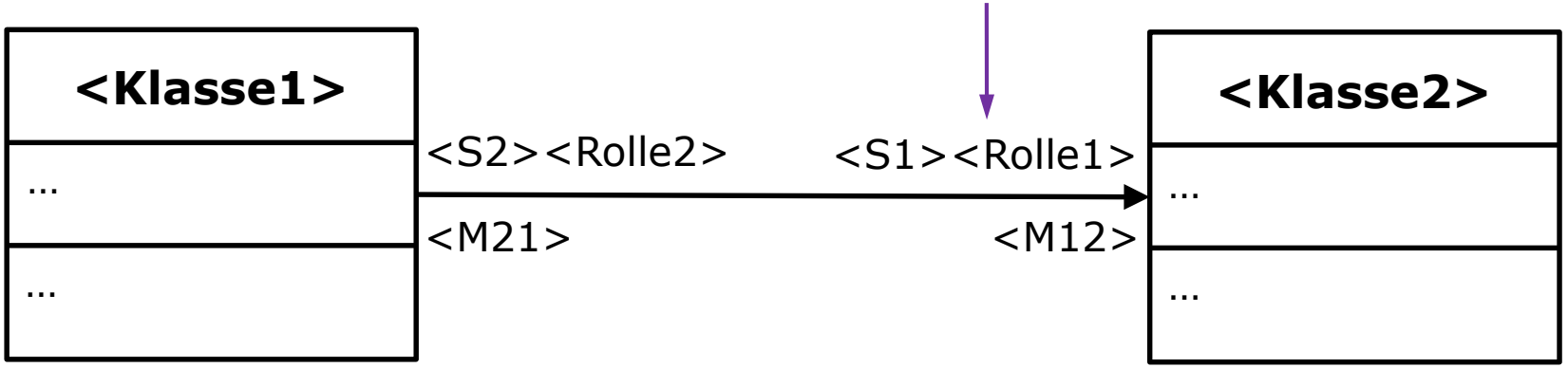
Welche Modellelemente sind zusätzlich notwendig, damit für die Implementierung alles vollständig beschrieben ist?
(fortgesetzt…)

- Stereotypen
  - spezialisieren Bedeutung des Modellelementes
  - vorgegeben (z.B. `«interface»`, `«abstract»`)
  - selbst definiert (z.B. `«warnfaehig»`)
  - oft verwendet für Wahl der Datenstrukturen bei Umsetzung von Multiplizitäten (z.B. `«list»`)
- Zusicherungen (Constraints)
  - OCL – ObjectConstraintLanguage
  - Beispiel: `{ordered}` – wenn geordnete Reihenfolge nötig
  - oft eingesetzt zur Auswahl der korrekten Datenverwaltungsstruktur

Klassendiagramm – Notation Entwurfsklassendiagramm



Objekt von <Klasse1> kennt <M12> Objekte von <Klasse2>
unter dem Namen <Rolle1> mit der Sichtbarkeit <S1>



Objektdiagramm, Paketdiagramm und Komponenten-diagramm

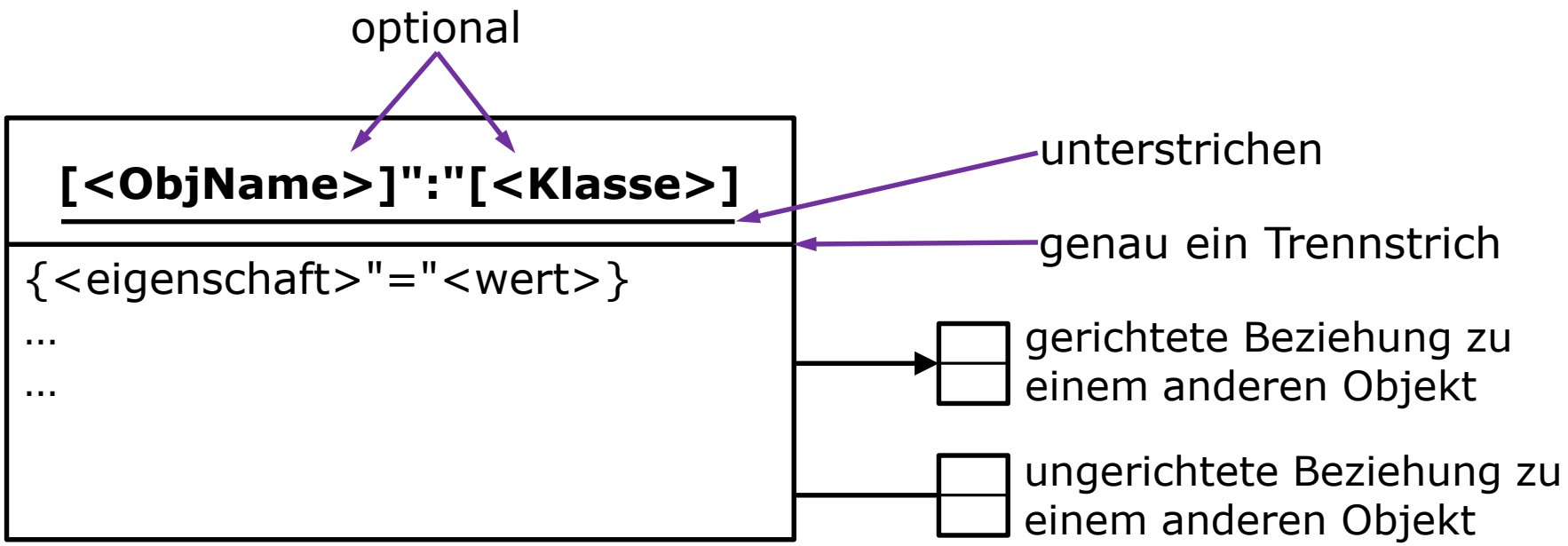
Objektdiagramm

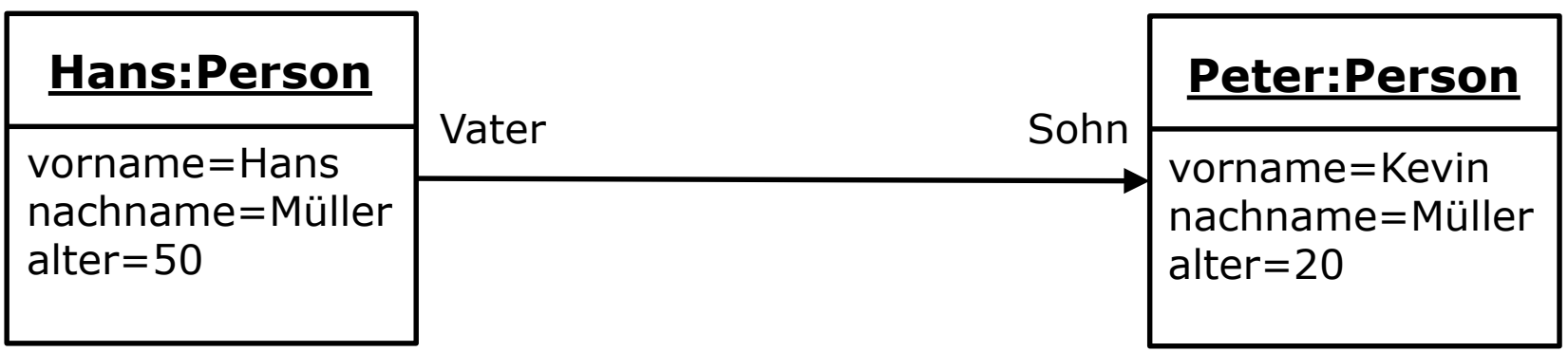
Welche Instanzen existieren gerade und welche Eigenschaften haben sie?

Ziel: Abbildung der inneren Struktur des Systems zur Laufzeit

- „Schnappschuss“ von Objekten und Attributbelegungen
- beispielhafte illustrative Verwendung
- Darstellung von Mengenverhältnissen
- Detaildarstellung vergleichbar mit der des Klassendiagramms
- sind Schnappschüsse aus der Laufzeit der Software
- zeigen nur relevante Objekte
- oft nur kleiner Ausschnitt möglich
- zeigen Beziehungen der Instanzen zueinander
- definieren konkrete Attributbelegungen der Instanzen
- stellen keine Funktionen/Methoden dar

Objektdiagramm – Notation

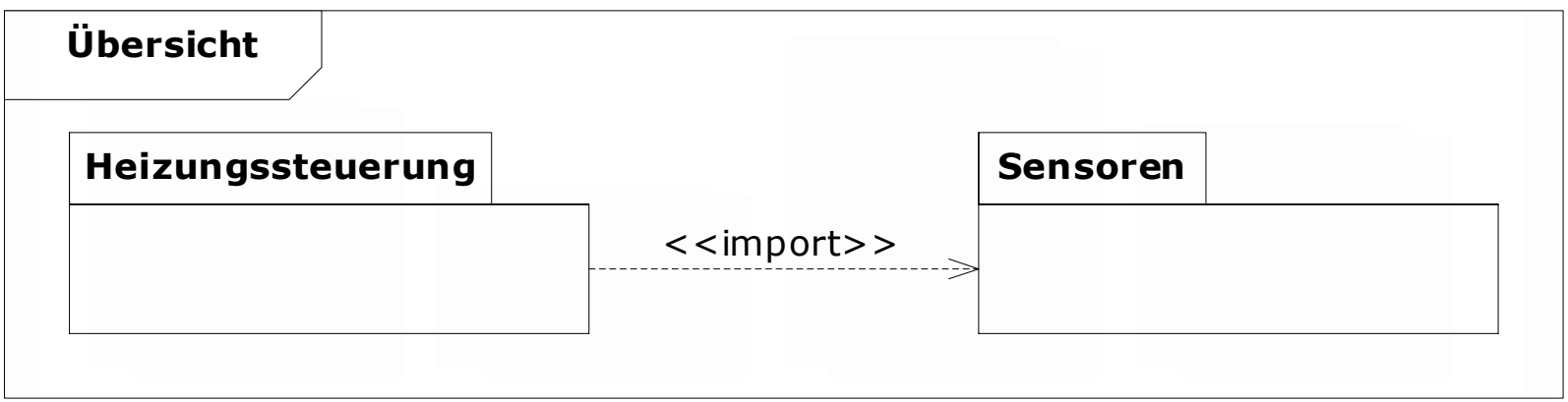




PAKETDIAGRAMM

Ziel: Unterteilung des Modells in übersichtliche Teile (Pakete)

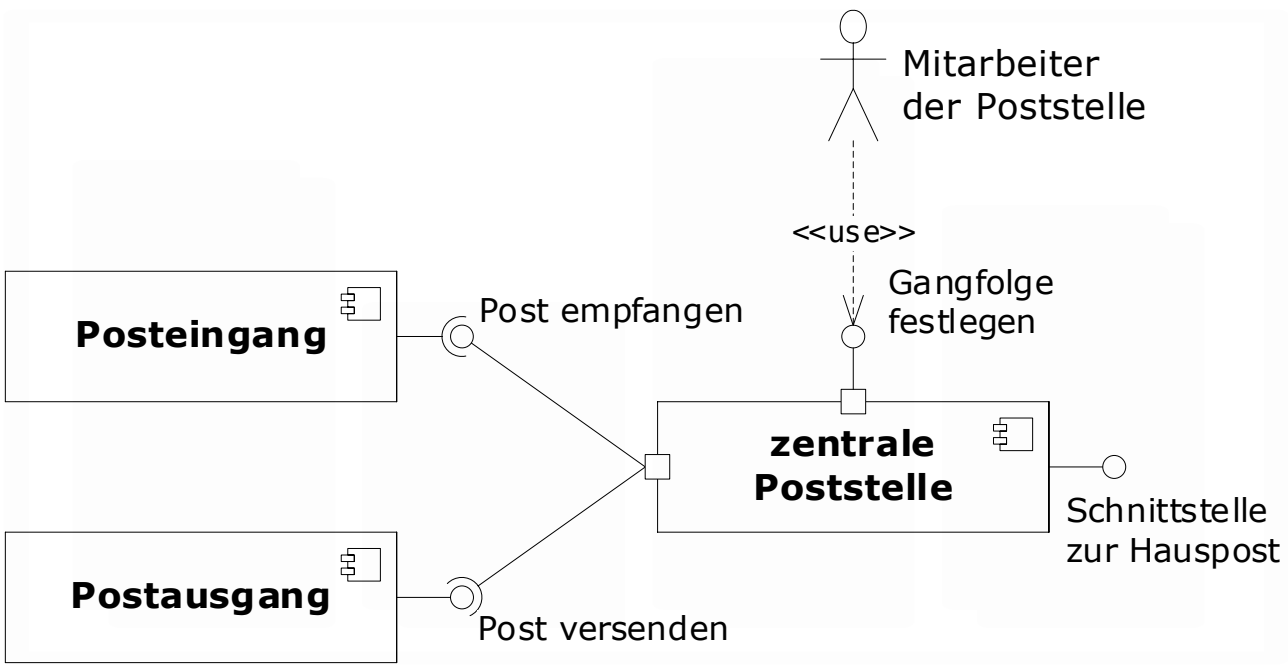
- Organisation des Systemmodells in übersichtlichere Einheiten
- Logische Zusammenfassung von Modellelementen
- Modellierung von Abhängigkeiten zwischen Einheiten



KOMPONENTENDIAGRAMM (1/2)

Ziel: Zusammenfassung der Klassen zu wiederverwendbaren, einfach(er) zu verwaltenden Komponenten und Modellierung der Beziehungen zwischen diesen Komponenten

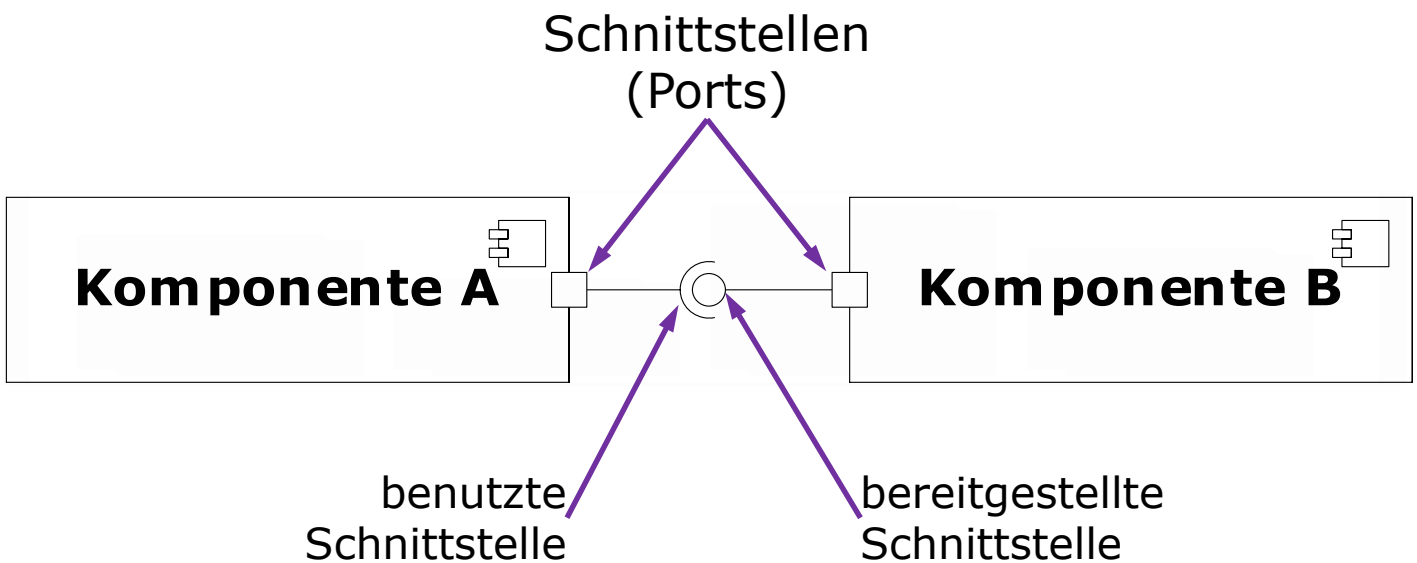
- Visualisierung der Organisation und von Abhängigkeiten einzelner Systemkomponenten
- Modellierung von angebotenen und benötigten Schnittstellen



KOMPONENTENDIAGRAMM (2/2)

Aus welchen Teilen besteht das System? Wie interagieren die Teile?

- beschreibt die Komponenten
- angebotene und erforderliche Schnittstellen der Komponenten
- Darstellung von Abhängigkeitsbeziehung der Komponenten über Verknüpfung der Schnittstellen
- interner Aufbau der Komponenten ist verborgen

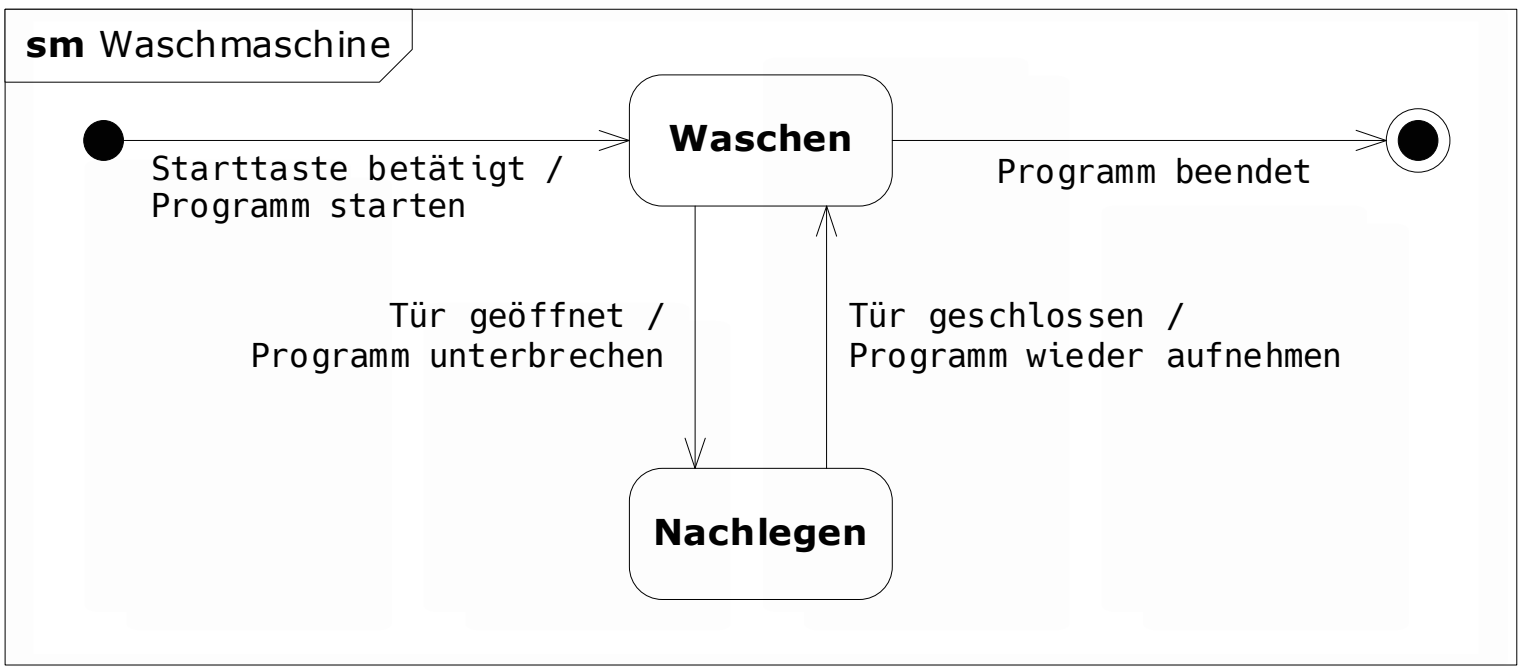


Zustandsdiagramm

ZUSTANDSDIAGRAMM

Ziel: Darstellung von ereignisabhängigen Zuständen eines Elements

- sehr präzise und mächtige Zustandsmodellierung
- Modellierung von Zuständen, Ereignissen, Nebenläufigkeit, Bedingungen, Ein- und Austrittsaktionen
- hierarchische Zustandsmodellierung möglich



Zustandsdiagramm – Typen (1/2)

Welche Zustände können vorkommen?

Welches Ereignis führt von einem Zustand in einen anderen Zustand?

Zustandsdiagramme modellieren

- Zustände der Objekte
- mögliche Transitionen (Zustandsübergänge)
- Trigger (Ereignisse), die Transitionen auslösen
- Aktivitäten während einer Transition
- Aktivitäten während eines Zustands

Ausprägungen in UML 2

- Verhaltenszustandsdiagramm
- Protokollzustandsdiagramm

Zustandsdiagramm – Typen (2/2)

Verhaltenszustandsdiagramm

- Wie reagiert ein Objekt je nach Zustand?
- definiert Zustände sowie Ereignisse, die Übergänge auslösen
- definiert Aktionen, die ausgeführt werden

Protokollzustandsdiagramm

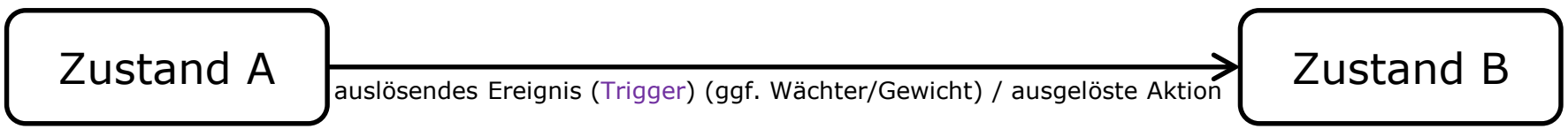
- Welche Übergänge sind zulässig?
- „Protokoll“ ist die Menge der erlaubten Übergänge und Aktionen
- zulässige Abfolge der Zustände

Zustandsdiagramm – Kanten

Kante (Edge, Transition, Übergang)

- verbindet Zustände gerichtet miteinander
- Trigger (Ereignis) führt zur Transition
- kann Wächter und Gewicht besitzen

Notation:



Sonderfall Selbsttransition:

Zustandsdiagramm - Knoten

Knoten (Vertex, einfacher Zustand)

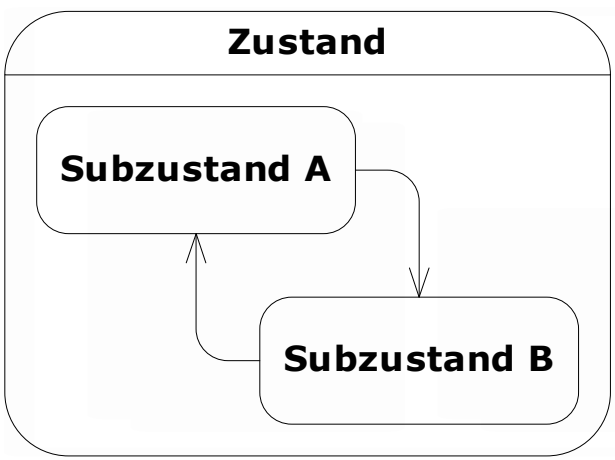
- Zustand bzw. Verhalten eines Objektes oder eines Systems
- kann interne Aktivitäten (Effekte) definieren
- kann interne Transitionen definieren

Notation:

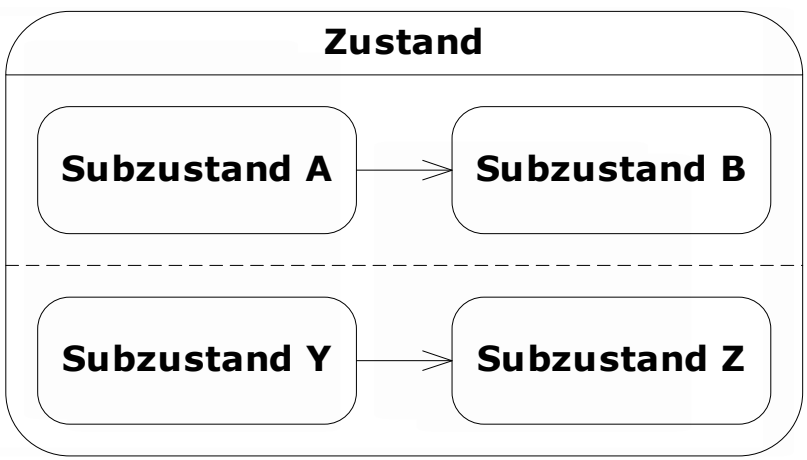
Zustandsdiagramm – zusammengesetzter Zustand

- Zustand enthält Subzustände
- mehrere (orthogonale) Regionen möglich
- orthogonal: nebenläufig, parallel

Notation:



eine Region



zwei orthogonale Regionen

Zustandsdiagramm – Pseudozustände (1/2)

- können, müssen aber nicht auftreten
- dienen der Ablaufsteuerung
- keine Effekte, internen Transitionen oder Subzustände

Anfangszustand und Endzustand

- sollte möglichst in jeder Region einmal auftreten
- Wächterbedingung und Aktion möglich
- kein Trigger

Notation:

Zustandsdiagramm – Pseudozustände (2/2)

Eingangsknoten und Ausgangsknoten

- ähnlich Anfangs- und Endknoten
- dienen als Verknüpfungspunkte für Unterdiagramme

Terminierungsknoten

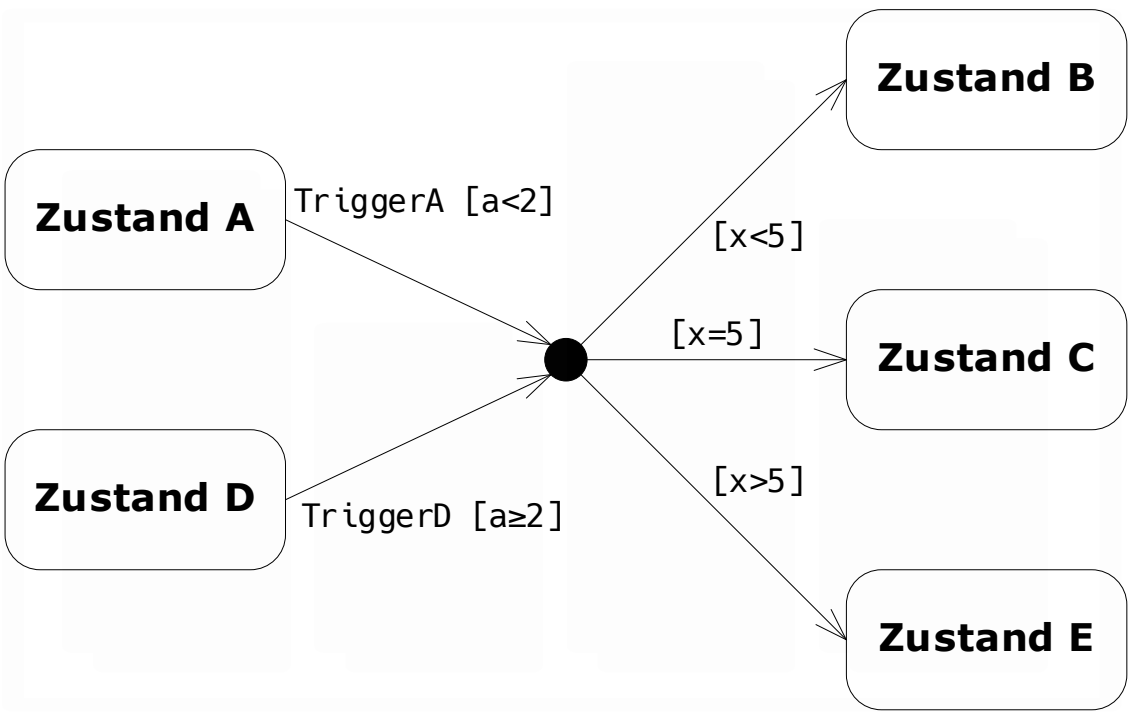
- nicht Endzustand, sondern Abbruch des Zustandsdiagramms

Historie

- für zusammengesetzte Zustände
- merkt sich den letzten Zustand beim Verlassen der Region
- flache Historie: nur oberste Region
- tiefe Historie: gesamte Schachtelungstiefe

Zustandsdiagramm – Verbindungsstelle

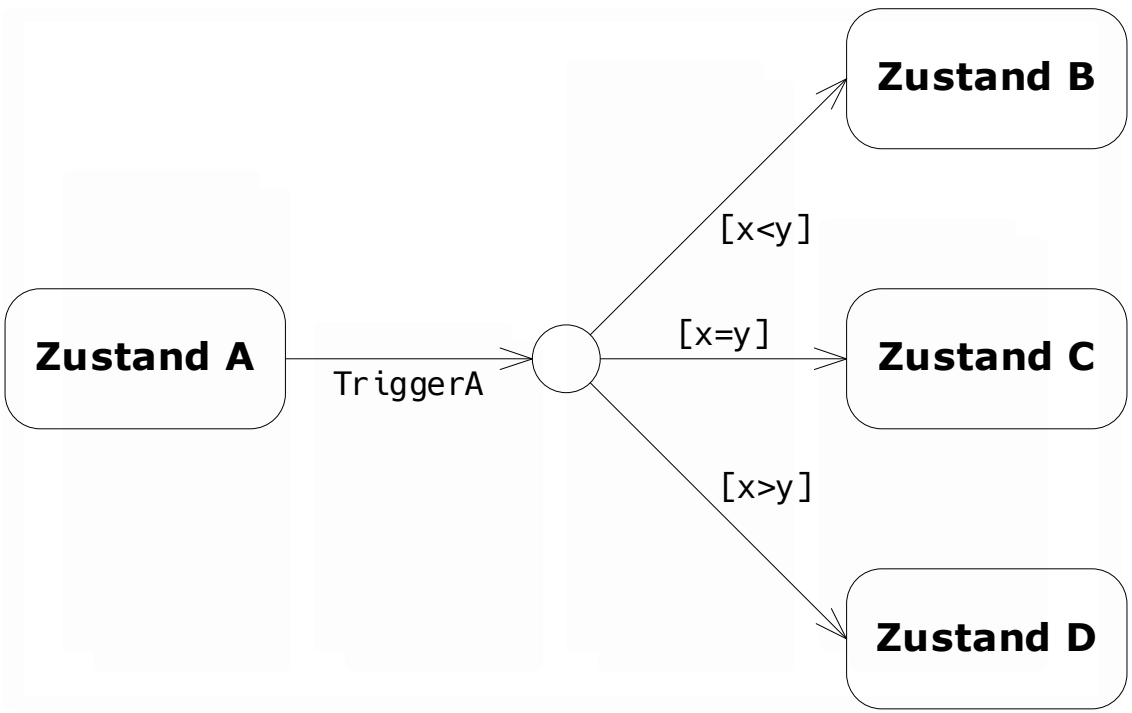
- statisch bedingt
- dient der Zusammenführung oder Unterteilung von Transitionen
- 1 bis beliebig viele Ein- und Ausgänge (1..*)
- Wächter regeln Weg



Zustandsdiagramm – Verzweigungen (1/2)

Sequentielle Verzweigung

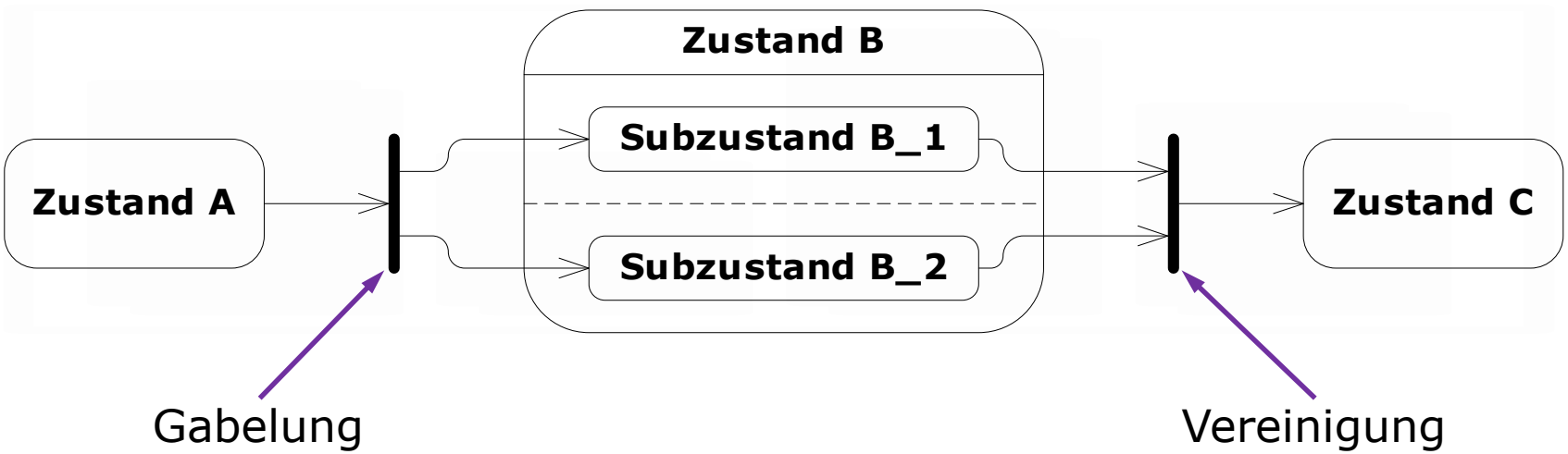
- dynamisch bedingt
- 1 Eingang
- Entscheidungsknoten
- derselbe Trigger, aber unterschiedliche Wächter



Zustandsdiagramm – Verzweigungen (2/2)

Nebenläufigkeitsbalken

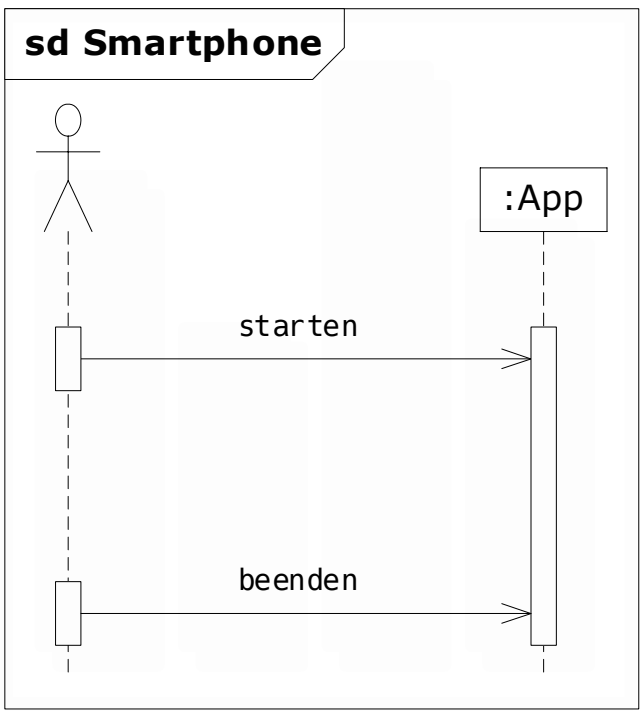
- parallele Ausführung
- für Subzustände
- als Gabelung: führt in orthogonale Regionen hinein
- als Vereinigung: führt aus orthogonalen Regionen hinaus



Sequenzdiagramm und Zeitdiagramm

SEQUENZDIAGRAMM

- Ziel: Darstellung des Informationsaustauschs zwischen Elementen (meist Objekte, Akteure) unter Berücksichtigung der Reihenfolge
- Darstellung des *zeitlichen Ablaufs* des Informationsaustauschs zwischen den Kommunikationspartnern
- Schachtelung und Flusssteuerung möglich



Sequenzdiagramm

Welche Nachrichten senden sich Objekte?

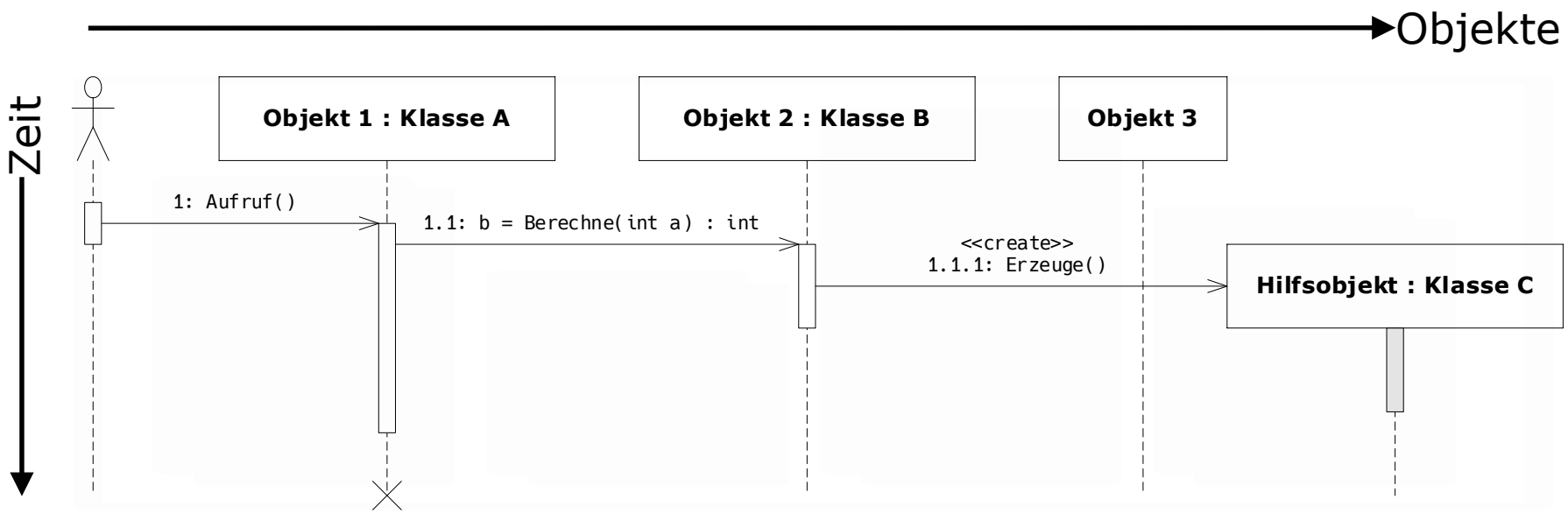
Welche Abfolge haben Nachrichten?

Wie reagieren Objekte auf Nachrichten?

Sequenzdiagramme

- dienen der Strukturierung der Funktionen und Interaktionsfähigkeiten von Objekten
- beschreiben einen konkreten Interaktionsablauf
- zeitliche Ordnung

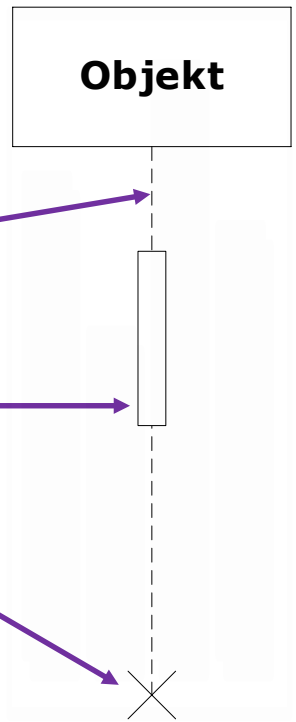
Sequenzdiagramm – Zeitachse und Objektachse



Sequenzdiagramm – Elemente

Interagierende Objekte

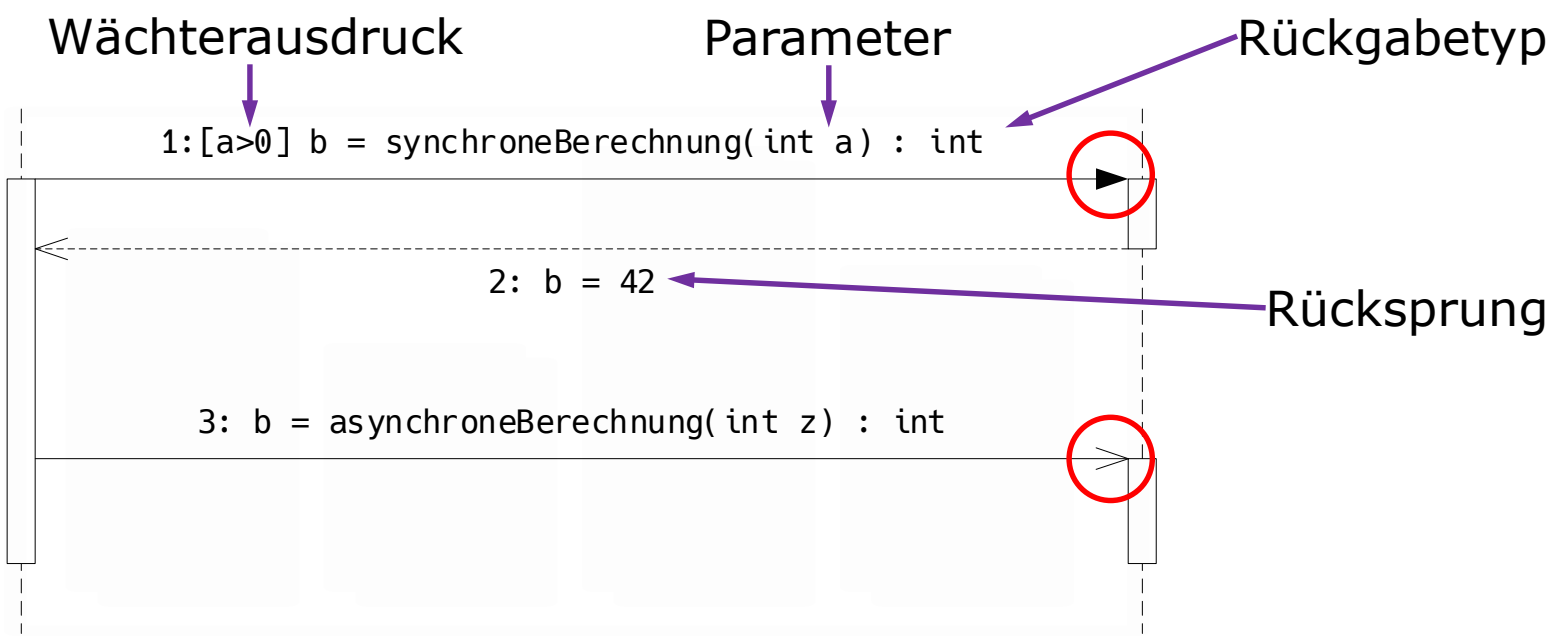
- jedes Objekt hat eigene Lebenslinie
- Aktivierungsbalken
- bei Zerstörung des Objektes: Terminierungssymbol



Sequenzdiagramm – Nachrichten (1/2)

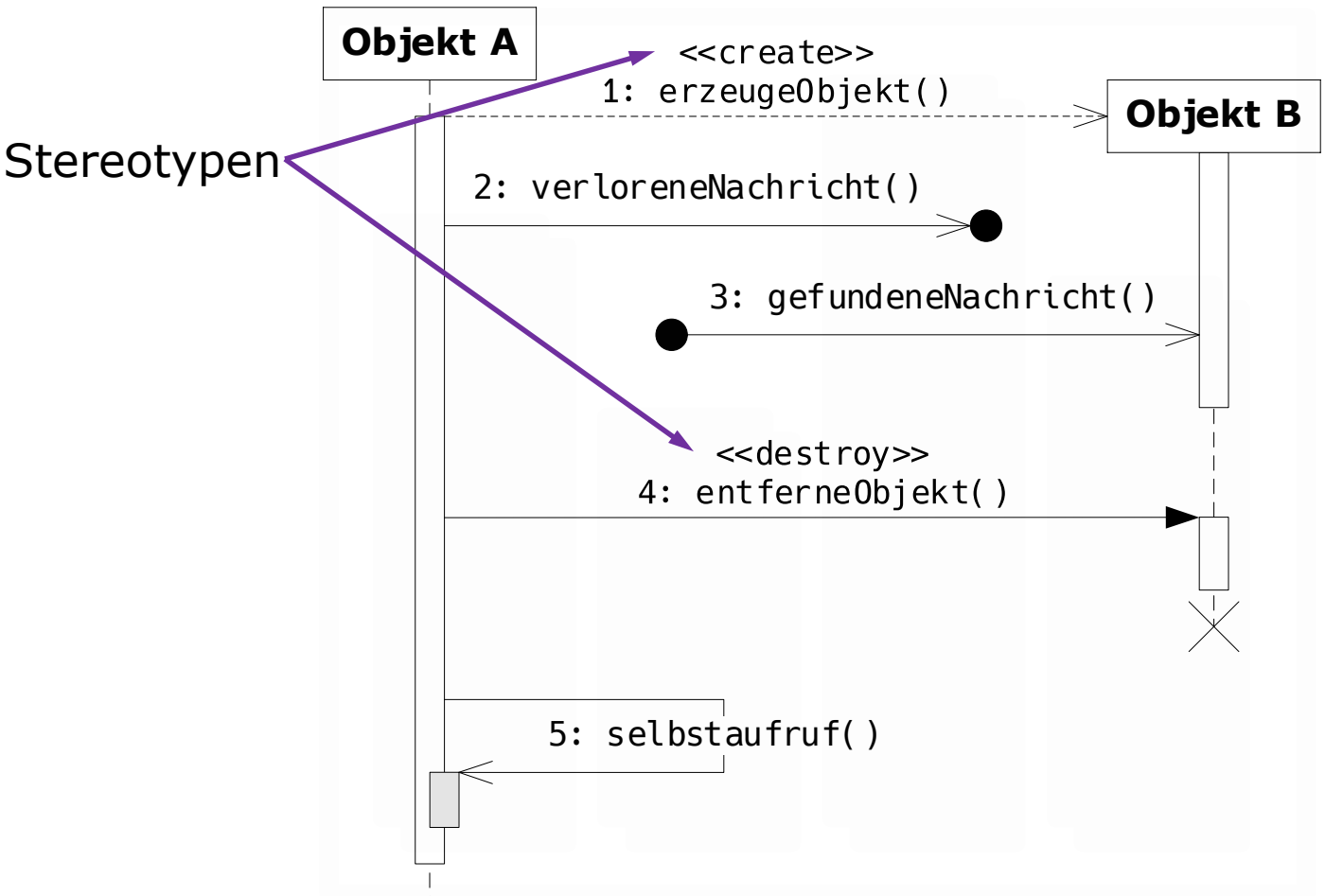
- als verzögerungsfrei angenommen
- Name entspricht Funktionsnamen
- synchrone Nachricht: wartet auf Antwort, erst danach weiterer Aufruf
- asynchrone Nachricht: nur Signal, Abfolge der Antworten nicht festgelegt

Notation:



Sequenzdiagramm – Nachrichten (2/2)

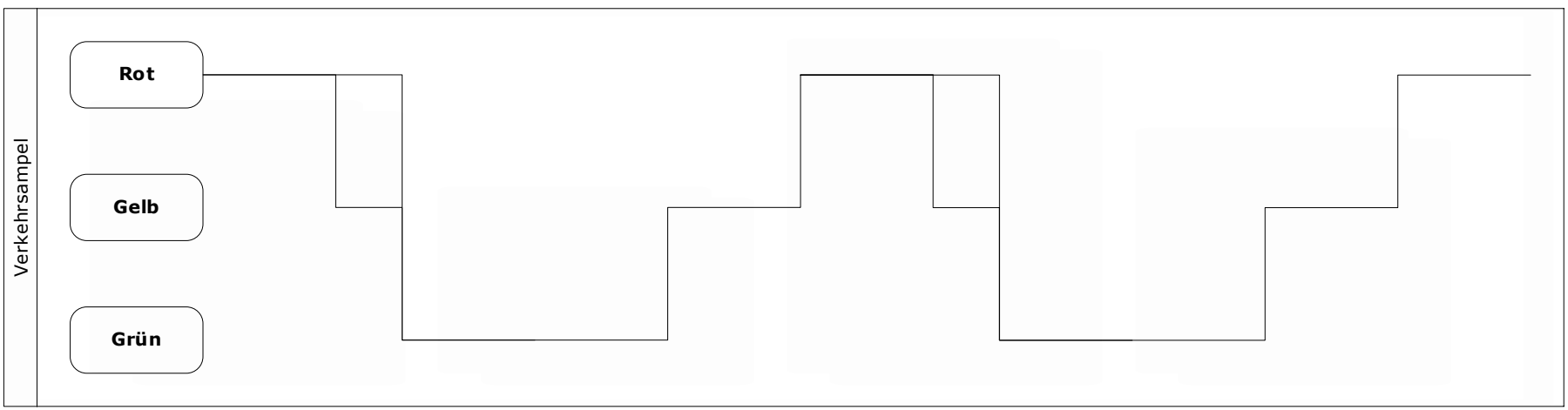
Spezielle Nachrichten



Zeitdiagramm (Timing-Diagramm)

Ziel: Darstellung der Zeitpunkte der Zustände der Interaktionspartner

- Visualisierung des exakten zeitlichen Verhaltens von Klassen, Schnittstellen, usw.
- sinnvoll zu verwenden, wenn der Zeitpunkt des Eintretens eines Ereignisses von Bedeutung ist

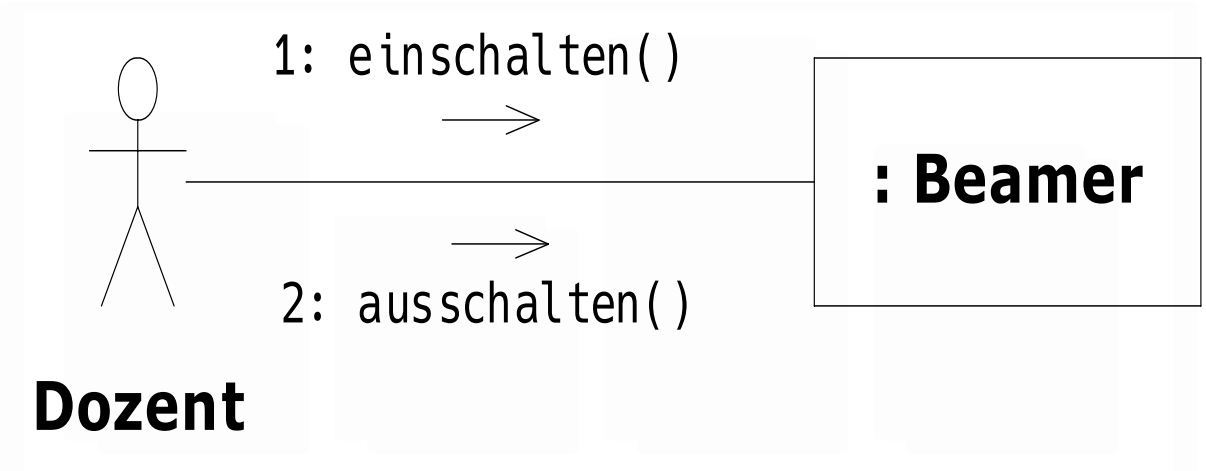


sonstige Diagramme

KOMMUNIKATIONSDIAGRAMM

Ziel: Darstellung der kollaborierenden Kommunikationspartner

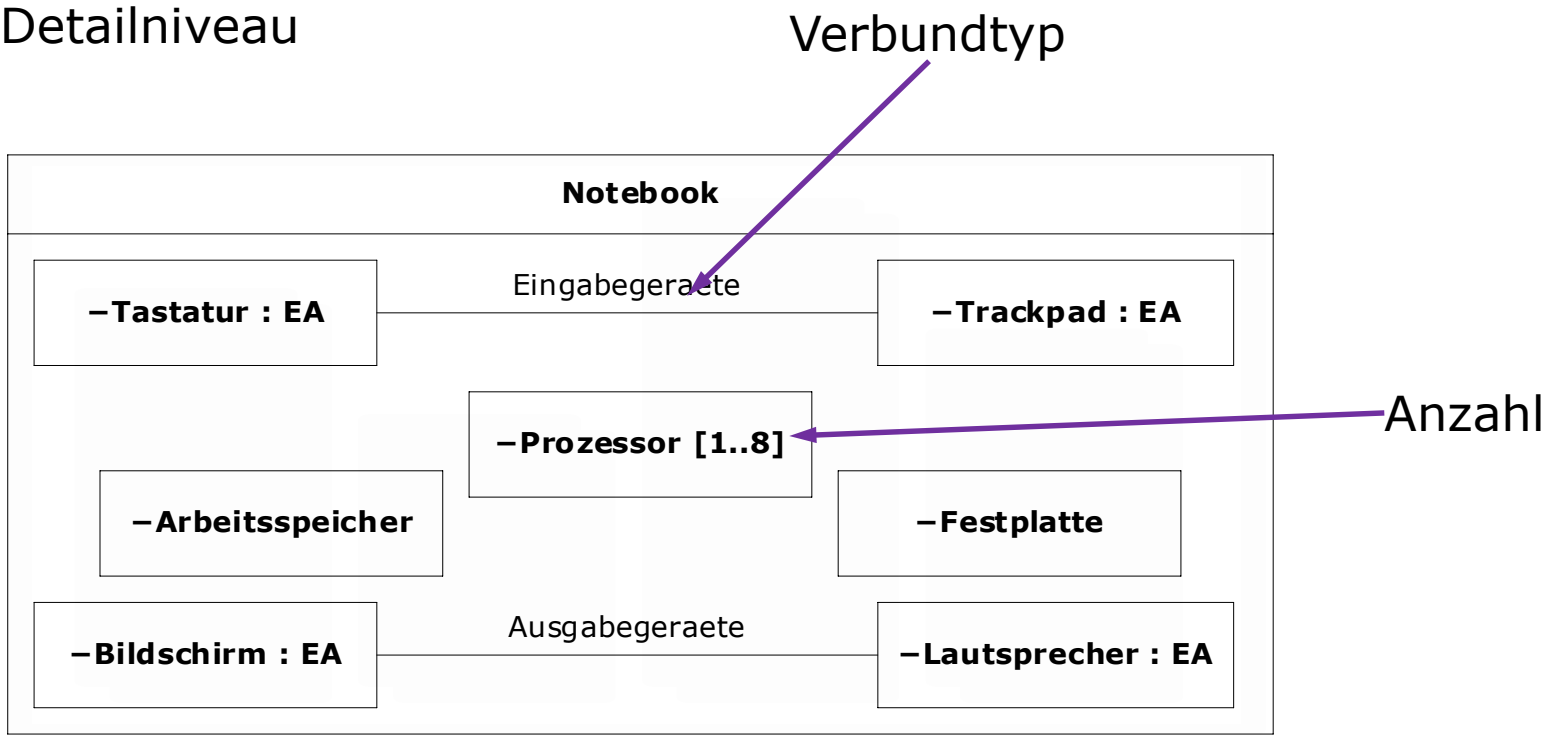
- Darstellung des Informationsaustauschs
- Fokus auf Überblick, weniger auf Abfolge (vgl. Sequenzdiagramm)



KOMPOSITIONSSTRUKTURDIAGRAMM

Ziel: Darstellung des „Innenlebens“ einer Klasse, Komponente oder eines Systemteils

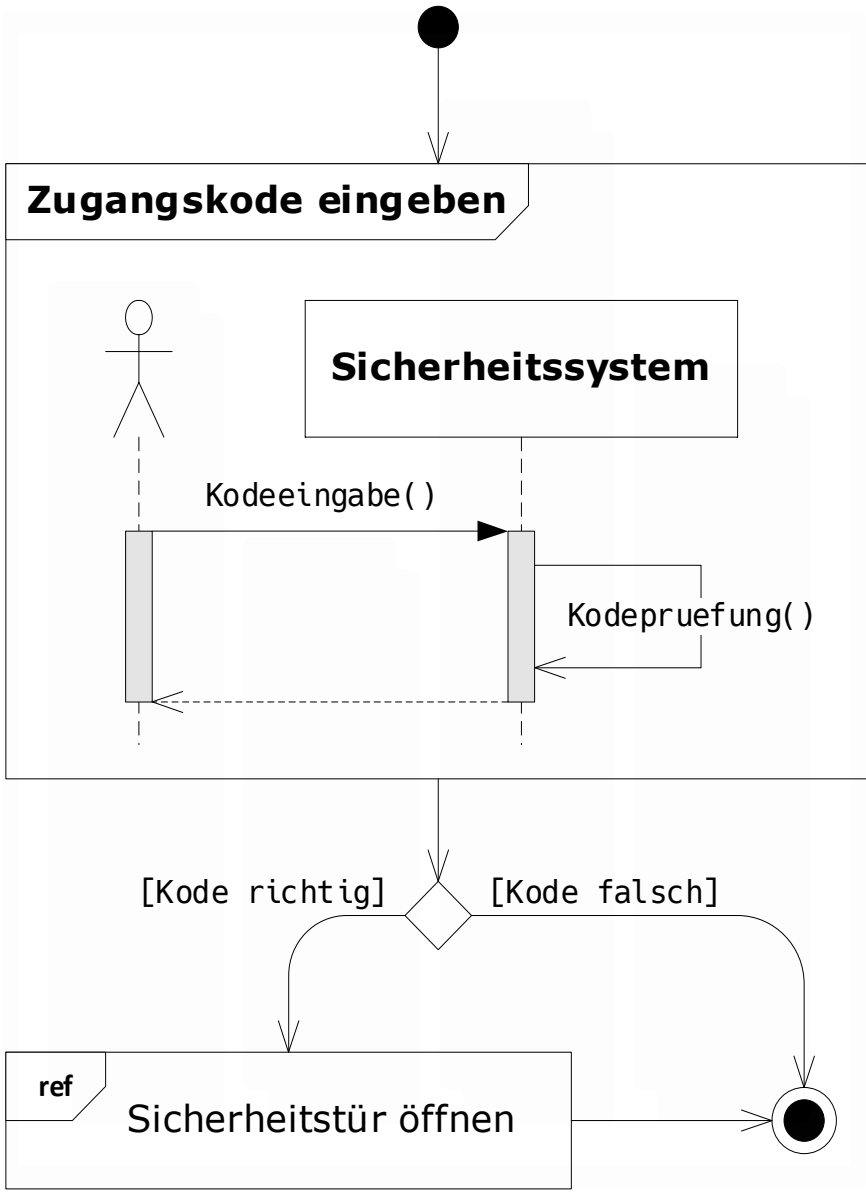
- Top-Down-Modellierung des Systems
- Darstellung von Teilen eines Gesamtelements und ihrer Mengenverhältnisse
- mittleres Detailniveau



INTERAKTIONSÜBERSICHTSDIAGRAMM

Ziel: Visualisierung der Interaktionsablaufzeiten

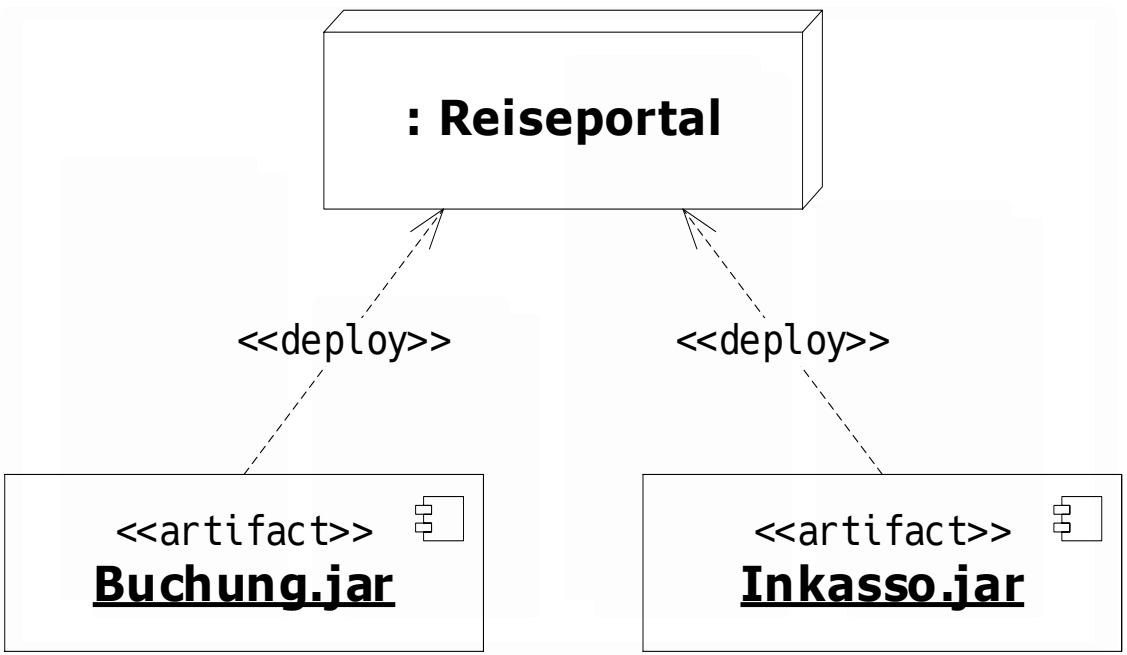
- Interaktionsdiagrammen verbinden
- Interaktionsdiagramme strukturieren



VERTEILUNGSDIAGRAMM

Ziel: Darstellung des Einsatzumfeldes des Systems (auch Hardware) und Verteilung der Komponenten zur Laufzeit

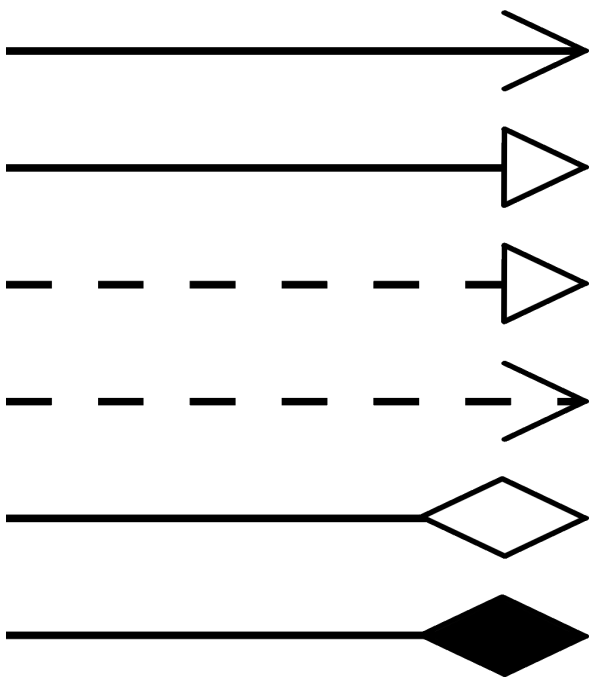
- Visualisierung des Systemlaufzeitumfeldes mit physischen (Hardware-) Systemkomponenten auf hohem Abstraktionsniveau



Beziehungen

BEZIEHUNGEN

Objekte und Klassen können in unterschiedlichen Beziehungen zu einander stehen:



Assoziation	(has a / uses)
Vererbung	(inherits from)
Realisierung	(implements)
Abhängigkeit	(relies on)
Aggregation	(is a)
Komposition	(is part of)

Beispielszenario: Robotorsteuerung

ROBOTERSTEUERUNG

Autonomes Transportsystem aus Produktionsanlage

- Anlage fertigt Maschinenelemente aus Einzelbauteilen
- einzelne Stationen
- Datenbank für Lagerverwaltung
- Selbstständiges Aufladen/Navigieren/ Transportieren/Abladen durch Transportsystem
- Greifer an Transportsystem vorhanden
- Navigation erfolgt über Graustufenlinie am Boden
- Roboter hat Sensoren
- Markierungen neben Graustufenlinie kennzeichnen Stationen

Modellbildung

Welche sind die für meine Aufgabe relevanten Eigenschaften?

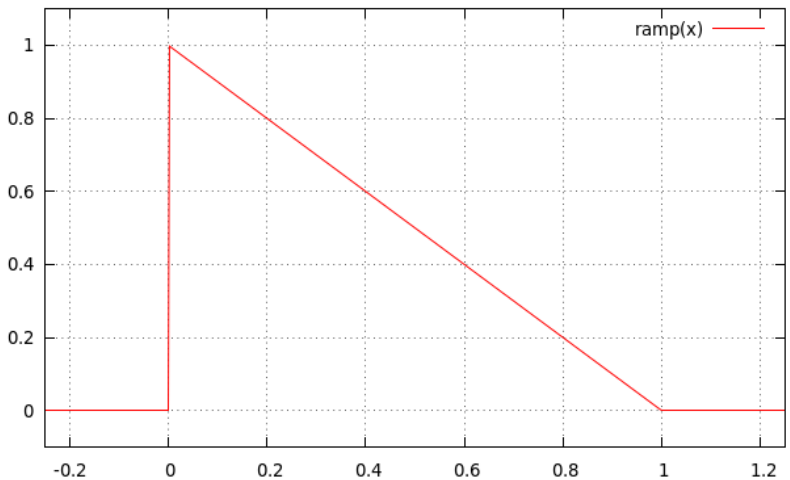
Welche Eigenschaften kann ich ignorieren?

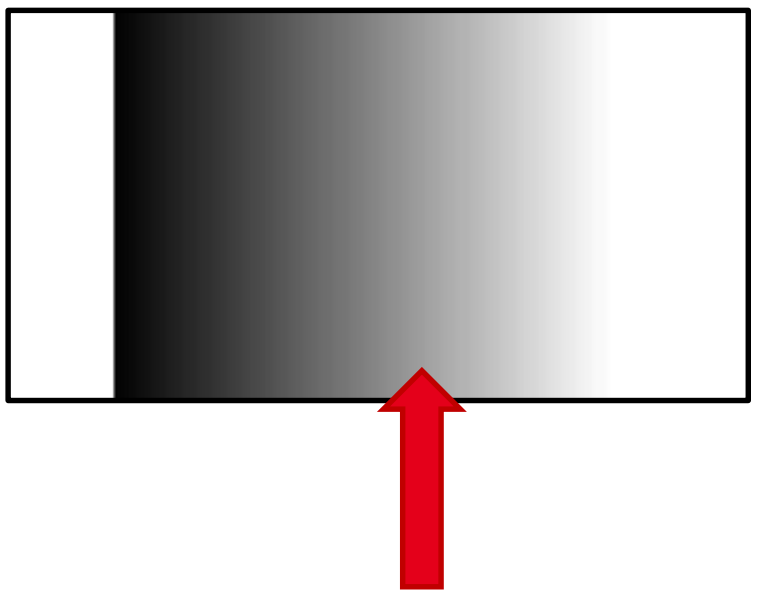
Wie bilde ich die relevanten Eigenschaften ab?

Wie stelle ich das Modell grafisch dar?

STRECKENFINDUNG

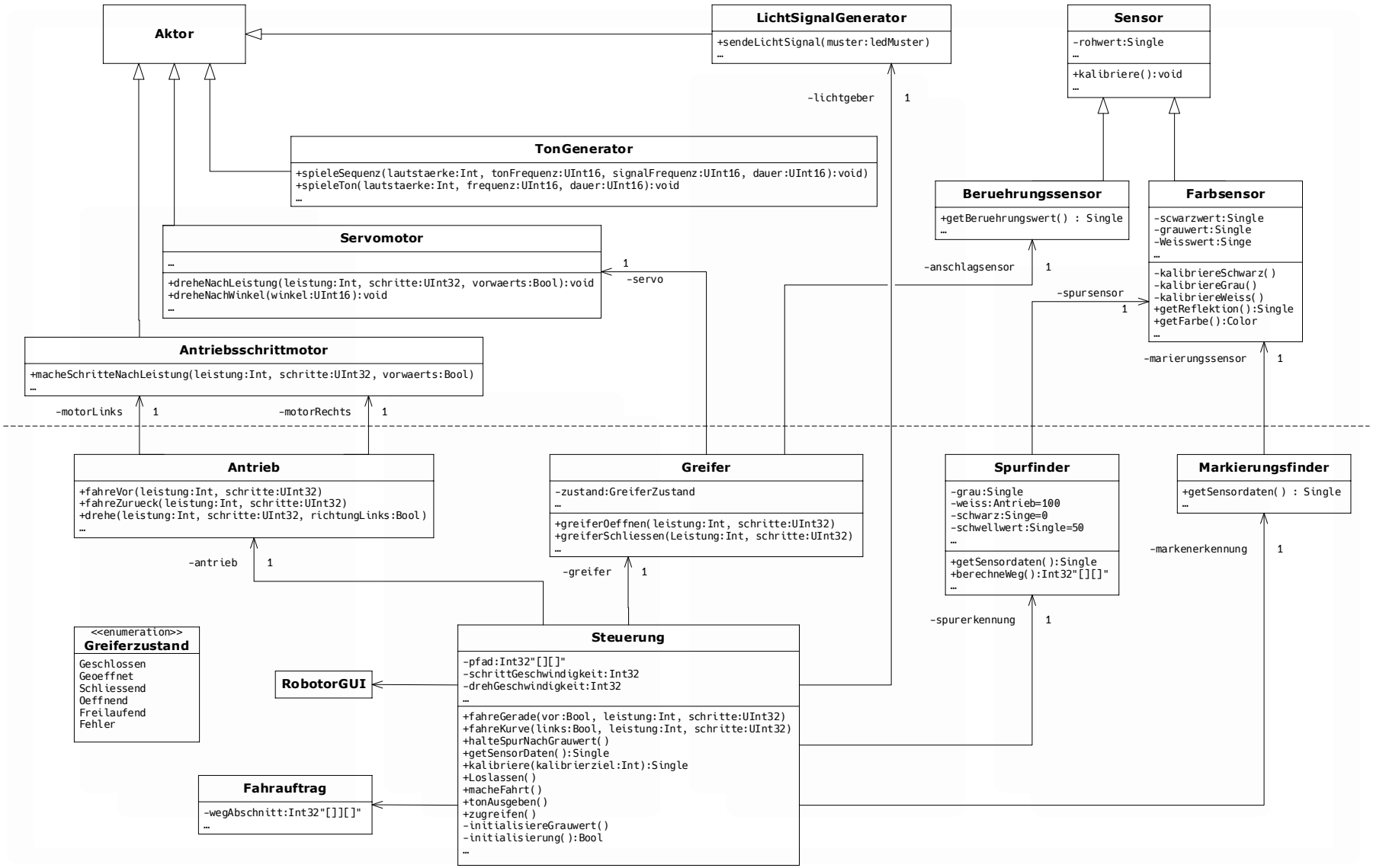
Der Grauwert ist als Rampe definiert:



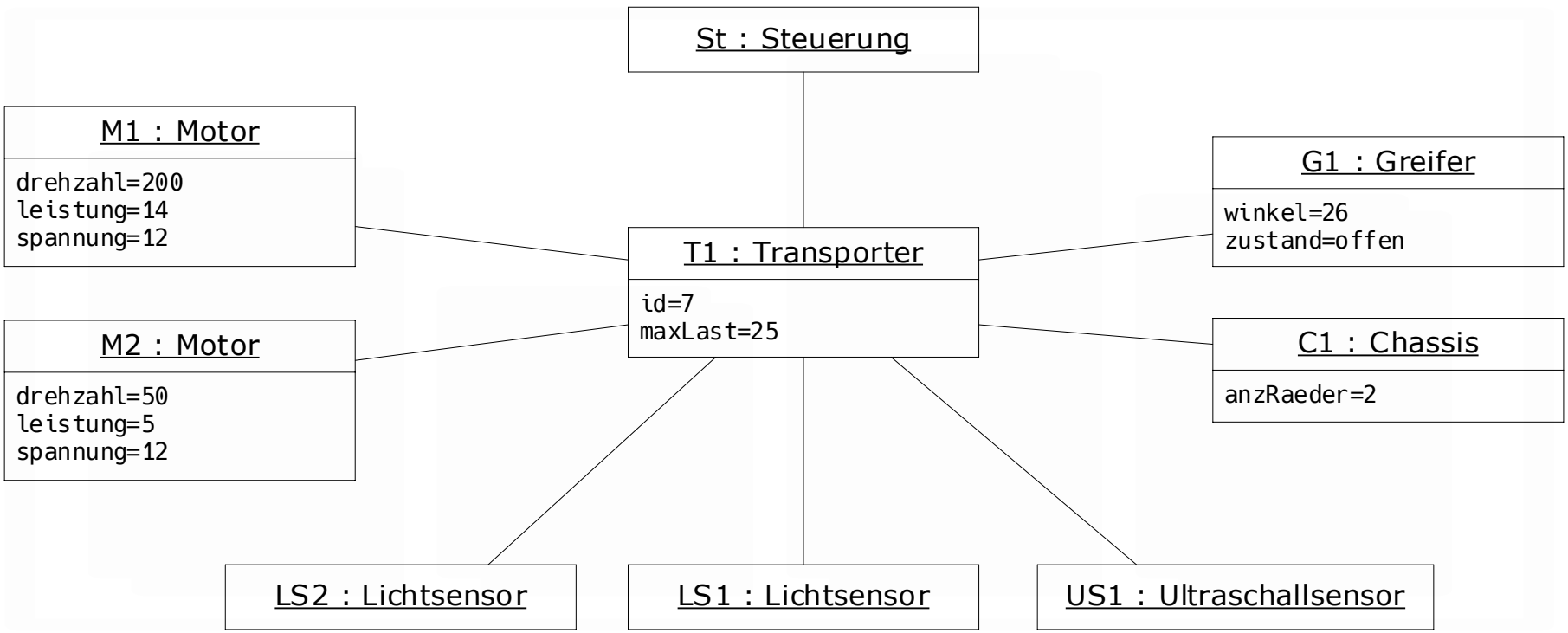


- Roboter bewegt sich entlang der Rampe
- Änderung des Grauwertes dient als Parameter

KLASSENDIAGRAMM



OBJEKTDIAGRAMM



Zeichnen von UML-Diagrammen

Visio ist keine UML-Entwicklungsumgebung!

Sie haben Zugriff auf ein besseres Werkzeug als Visio 2013? Prima!
Nutzen Sie es eigenverantwortlich!

- *Freie Software*: ArgoUML, Eclipse mit Uml2Tools, Nclass, Netbeans, Dia, Fujaba, Gaphor, Modelio Free Edition, OpenAmeos, PingUML, SDMLib, Software Ideas Modeler, StarUML, TOPCASED, Umbrello, UMLet, Violet, WhiteStarUML
- *Proprietäre Software*: Altova Umodel, ARIS, Astah, Borland Together, BOUML, ConceptDraw, EasyCODE, Embedded UML Studio, Innovator, iBS DesignTime for ABAP, MagicDraw UML, Metamill, Microsoft Visual Studio 2010 Ultimate, Modelio Enterprise Edition, objectiF, Poseidon for UML, PowerDesigner von Sybase, PTC Integrity Modeler, radCASE, Rational Rhapsody, Rational Rose, Select Solution Factory von Select Business Solutions, sinelaboreRT, SiSy, Sparx Systems Enterprise Architect, UML Lab, UML 2 Statemachine C/C++ Code Generator Framework, Visual Paradigm, yEd

Visio Stencil and Template for UML 2.5

Home

Visio stencil for UML

Visio stencil for SysML

Tips and Tricks

Update Log

About me

AdChoices

Free Visio

Visio Stencil

Updated December 27, 2013. Check what's new in the Update Log.

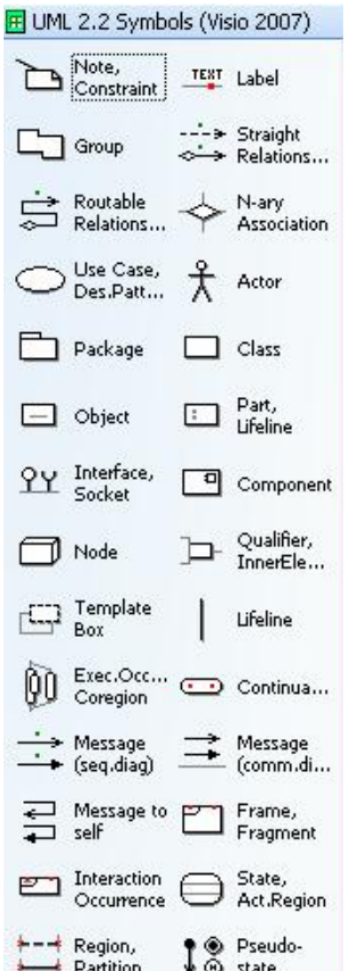
The UML stencil for Microsoft Visio supports complete UML 2.5, i.e. UML use case diagram, class diagram, package diagram, object diagram, composite structure diagram, interaction diagram, sequence diagram, communication diagram, interaction overview diagram, activity diagram, state machine diagram, component diagram, deployment diagram, profile diagram, timing diagram, and all symbols of the UML 2.5, specified in OMG Unified Modeling Language (OMG UML), ptc/2013-09-05, as well as all previous UML versions, UML 2.4, UML 2.3, UML 2.2, UML 2.1, UML 2.0, UML 1.5, UML 1.4, UML 1.3 and UML 1.1.

The Visio stencils can also be used for SoaML and other UML profiles and dialects.

The stencil also contains several non-normative UML symbols, that are not specified in the standard, but used in some UML books and papers. These non-normative symbols are always last items on the right-click menu, below the menu item called "non-normative".

If you discover omissions in the UML stencil or have suggestions for improvements please contact me at ph@ phruby.com.

Content of this page



https://softwarestencils.com/uml/index.html

Literatur

Sommerville, I.: Software Engineering, ADDISON-WESLEY Verlag, ISBN 3-8273-7001-9

Balzert, H.: UML 2 in 5 Tagen, W3L-Verlag | Herdecke | Witten, 2009

Object Management Group (OMG): Unified Modeling Language (UML) 2.5.1 Specification -- https://www.omg.org/spec/UML/2.5.1/PDF