Willkommen zur Vorlesung

Modellbasierte Softwaretechniken für sichere Systeme

im Sommersemester 2012

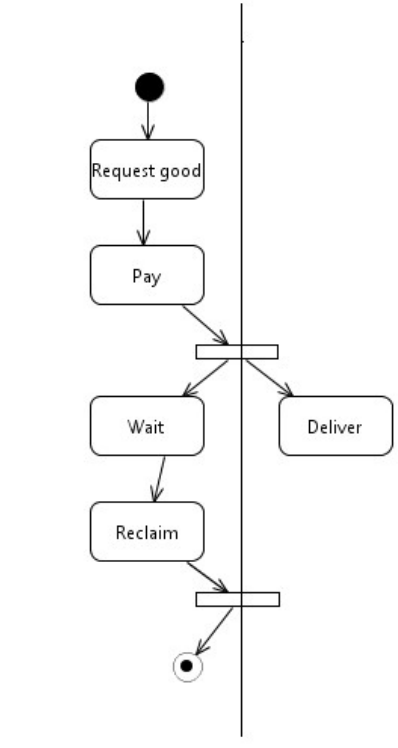
Prof. Dr. Jan Jürjens

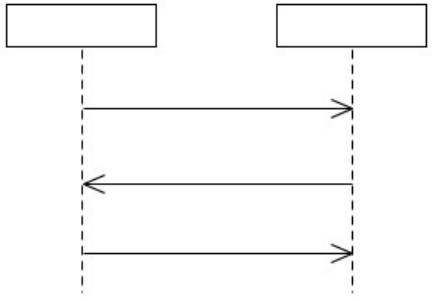
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

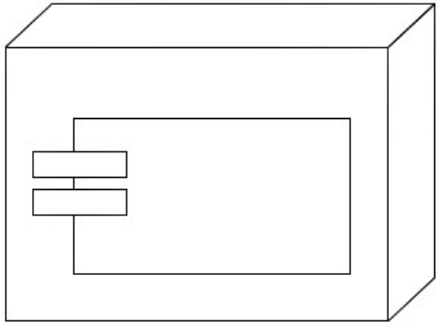
3. Modell-basierte Sicherheit mit UML

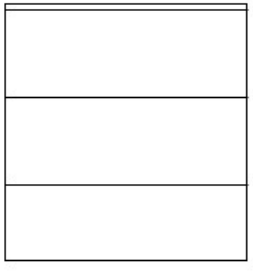
[inkl. Beiträge von
Prof. Dr. Volker Gruhn, Universität Duisburg-Essen]

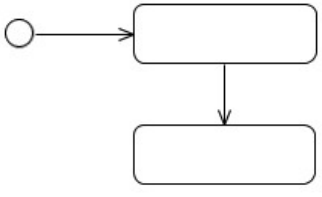
UML: Einige Diagrammarten

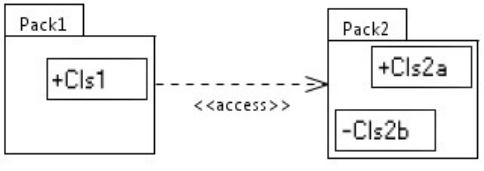




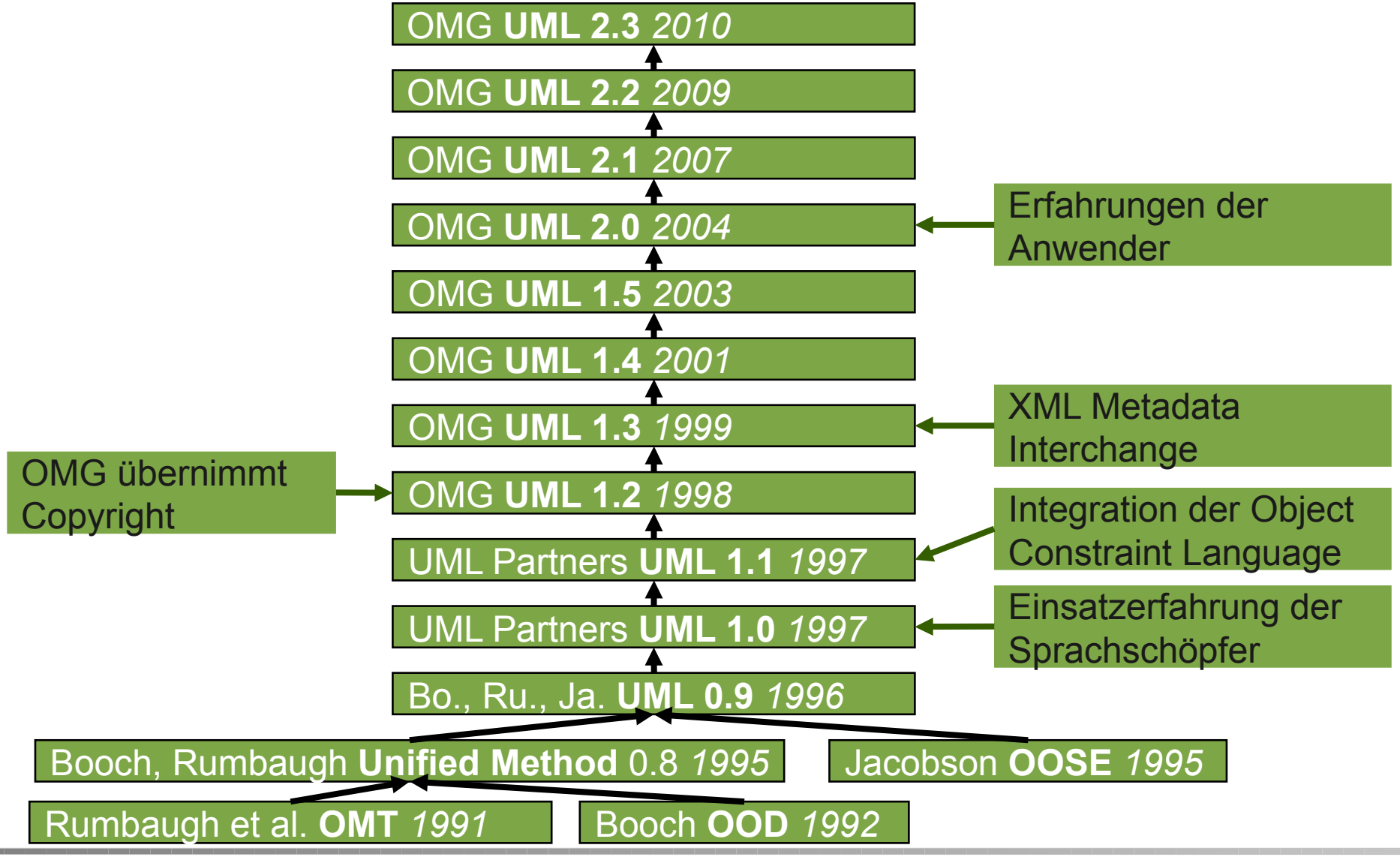




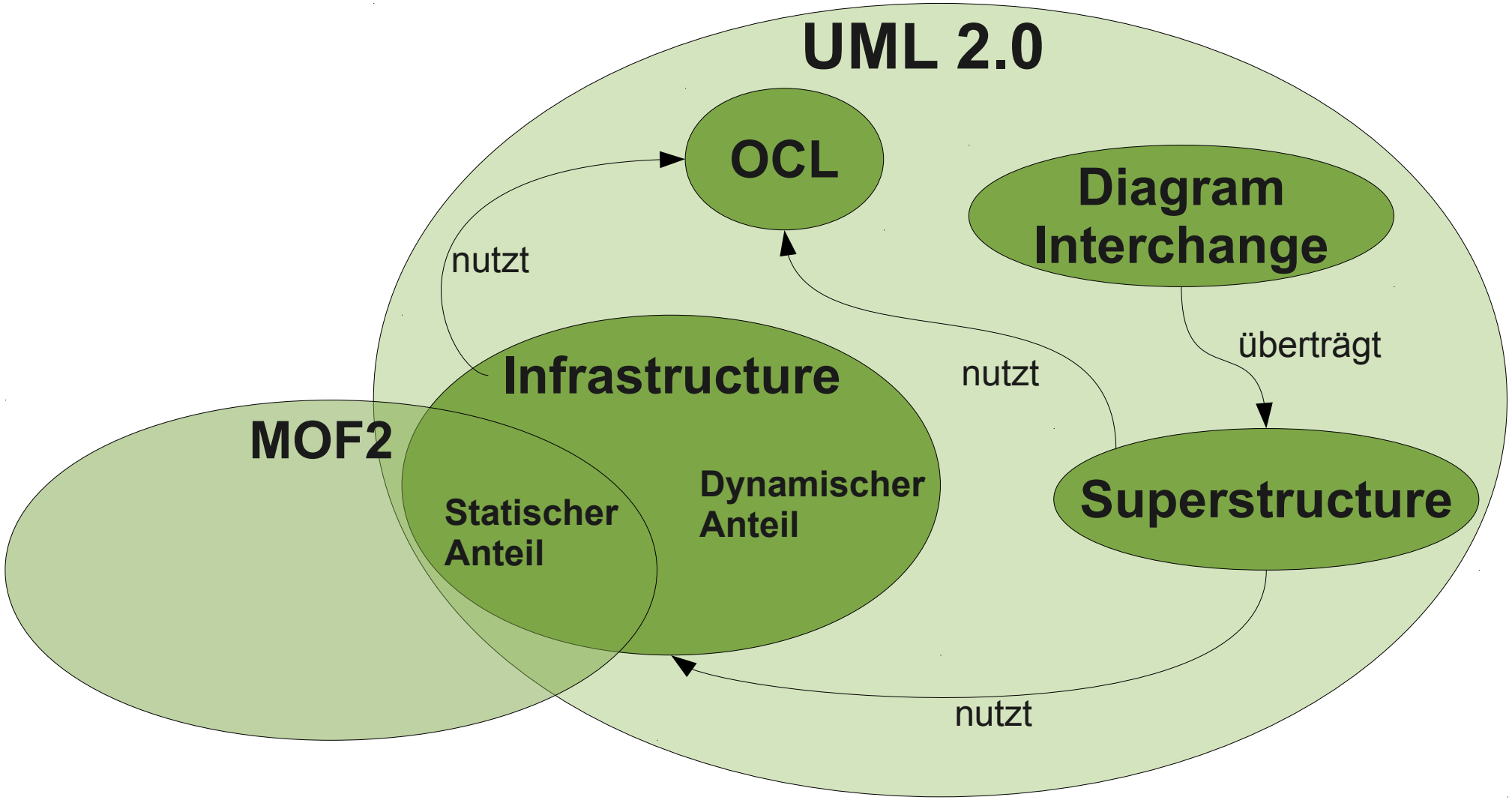




Geschichte der UML



UML Spezifikation

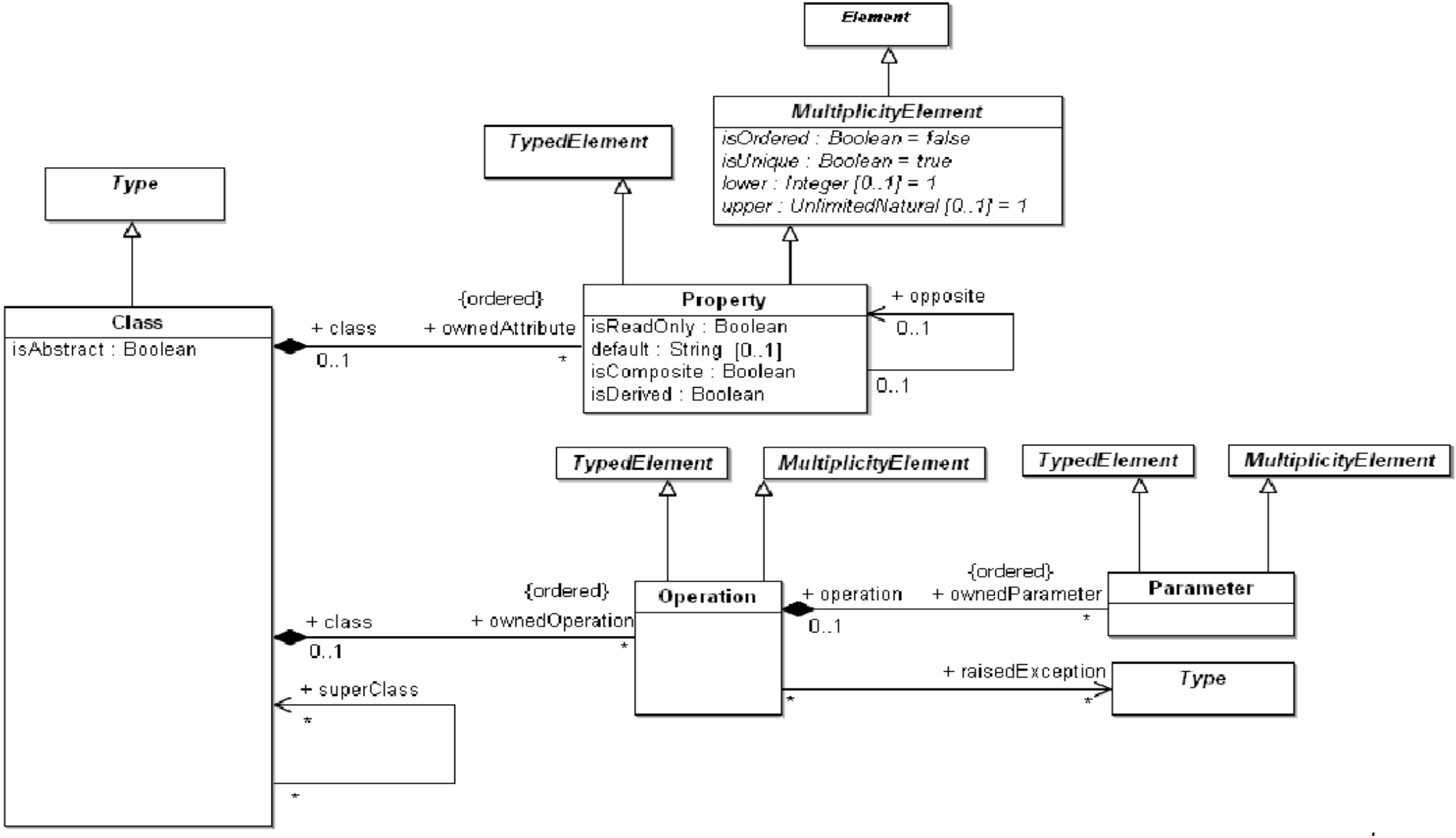


UML Spezifikation

- Definition der Modellierungssprache
- Bestandteile
  - Abstrakte Syntax
    - Modellelemente
  - Statische Semantik
    - Wohlgeformtheit
  - Konkrete Syntax
    - Notation, ggf. graphische Symbole
  - Dynamische Semantik
    - Verwendung, Bedeutung

Metamodell
+ Constraints (OCL)

UML Metamodell Beispiel
Core::Basic:ClassDiagram



Metamodellierung

- Modelle sind Instanzen ihrer Metamodelle
- Beispiel Diagramm vs. Metamodell-Instanzen
- Metamodelle definieren Modellelemente
  - „Sprachdefinition“
- Metamodelle für:
  - Maschinenlesbarkeit
  - Validierung
  - Speicherung von Modellen (Repositories)
  - Datenaustausch/Interoperabilität
  - Definition von Transformationen

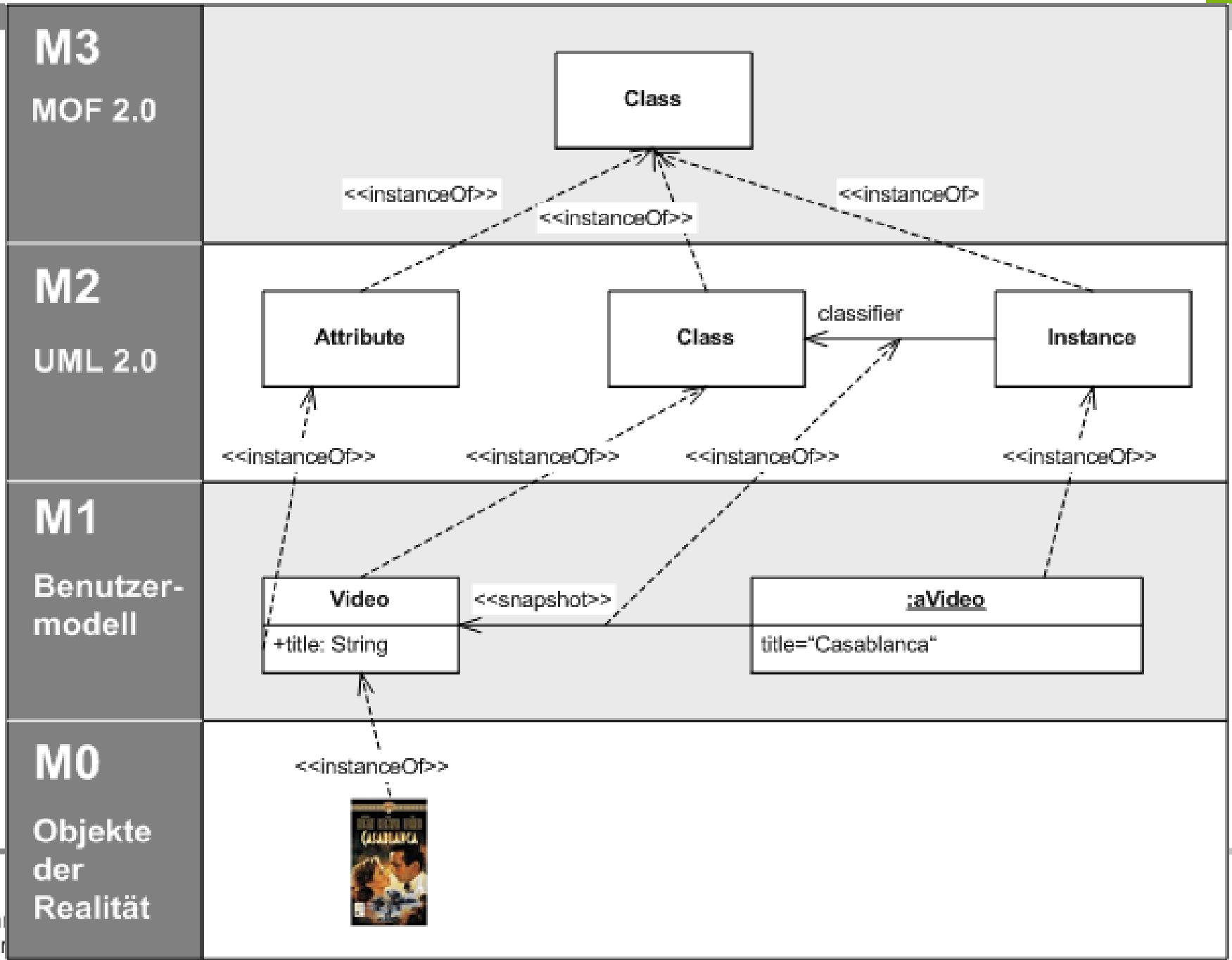
Meta Ebenen

- M0-Ebene
  - Konkret. Ausgeprägte Daten.
- M1-Ebene
  - Modelle. Zum Beispiel physikalische oder logische Daten- oder Prozessmodelle oder konkrete Ausprägungen von UML- bzw. Objekt-Modellen, welche die Daten der M0-Ebene definieren.
- M2-Ebene
  - Meta-Modelle. Definieren, wie die Modelle aufgebaut und strukturiert sind. Zum Beispiel definieren Sprachelemente wie Klassen, Assoziationen und Attribute der UML 2.0, wie konkrete UML-Modelle aufgebaut sein können.

Meta Ebenen

- M3-Ebene
  - Meta-Meta-Modelle (bzw. MOF-Ebene). Abstrakte Ebene, die zur Definition der M2-Ebene herangezogen wird. Die Definition der M3-Ebene erfolgt mit den Mitteln der M3-Ebene selbst, dies stellt den Abschluss einer sonst unendlichen Metaisierung dar.

Meta Ebenen


M3
MOF 2.0
Class
<<instanceOf>>
<<instanceOf>>
<<instanceOf>>
M2
UML 2.0
Attribute
Class
classifier
Instance
<<instanceOf>>
<<instanceOf>>
<<instanceOf>>
<<instanceOf>>
M1
Benutzer-modell
Video
+title: String
<<snapshot>>
:aVideo
title="Casablanca"
M0
Objekte der Realität
<<instanceOf>>

Metamodellierung

- Wege zum „eigenen“ Metamodell
  - Eigenes Metamodell „from scratch“
    - aufwändig
  - Direkte, beliebige Erweiterung/Anpassung eines Metamodells
    - Setzt uneingeschränkten Zugriff auf das Metamodell voraus und verletzt ggf. exisitierende Semantik
  - Erweiterung des UML-Metamodells durch
    - UML Profile
    - Vorgesehene Schnittstelle zum UML-Metamodell
    - Lediglich Verfeinerung

UML Profil
Erweiterungsmöglichkeiten

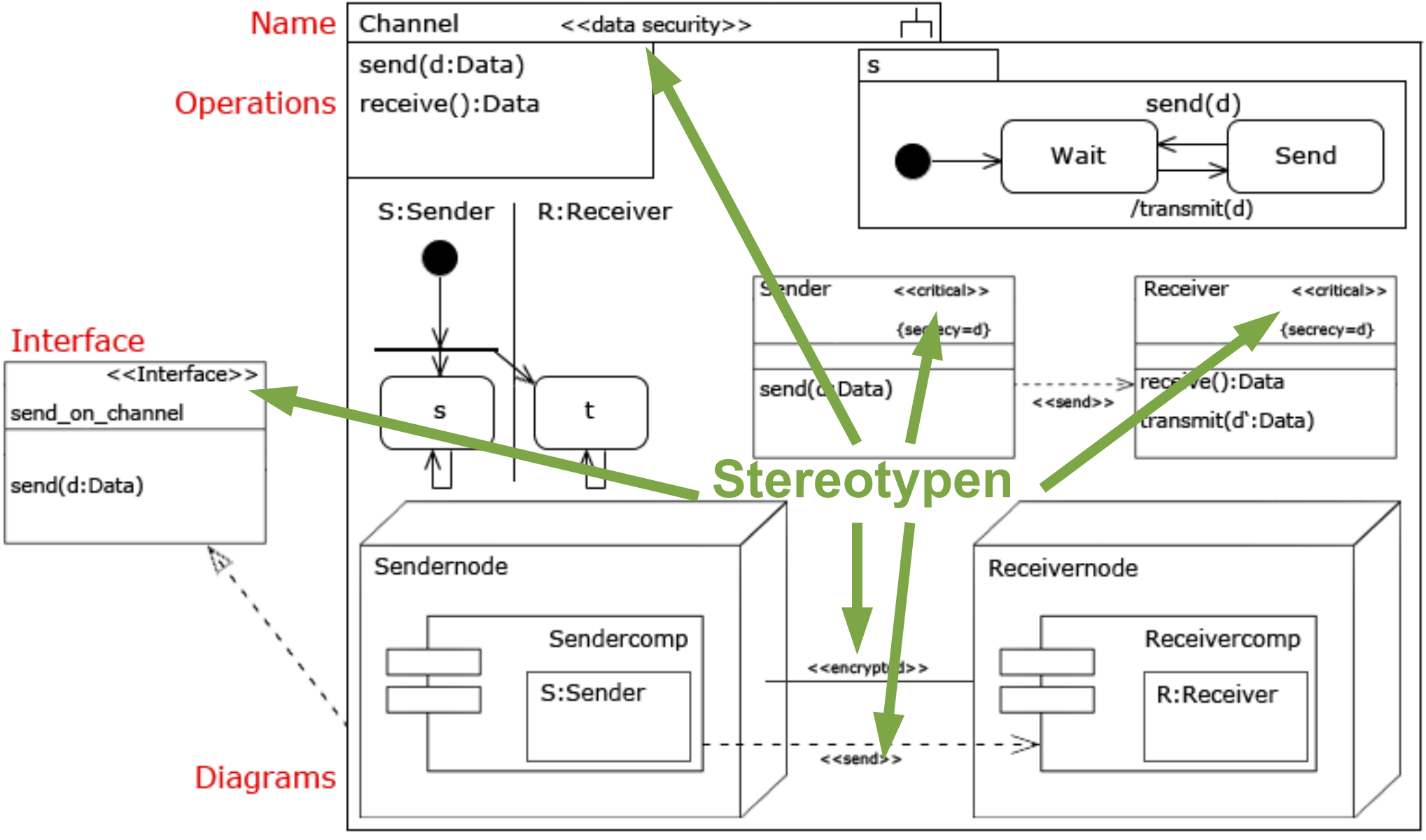
Stereotypen: spezialisiert die Benutzung von Modellelementen <<label>>.

Tagged value: fügt {tag=value} Paare zu stereotypisierten Elementen hinzu.

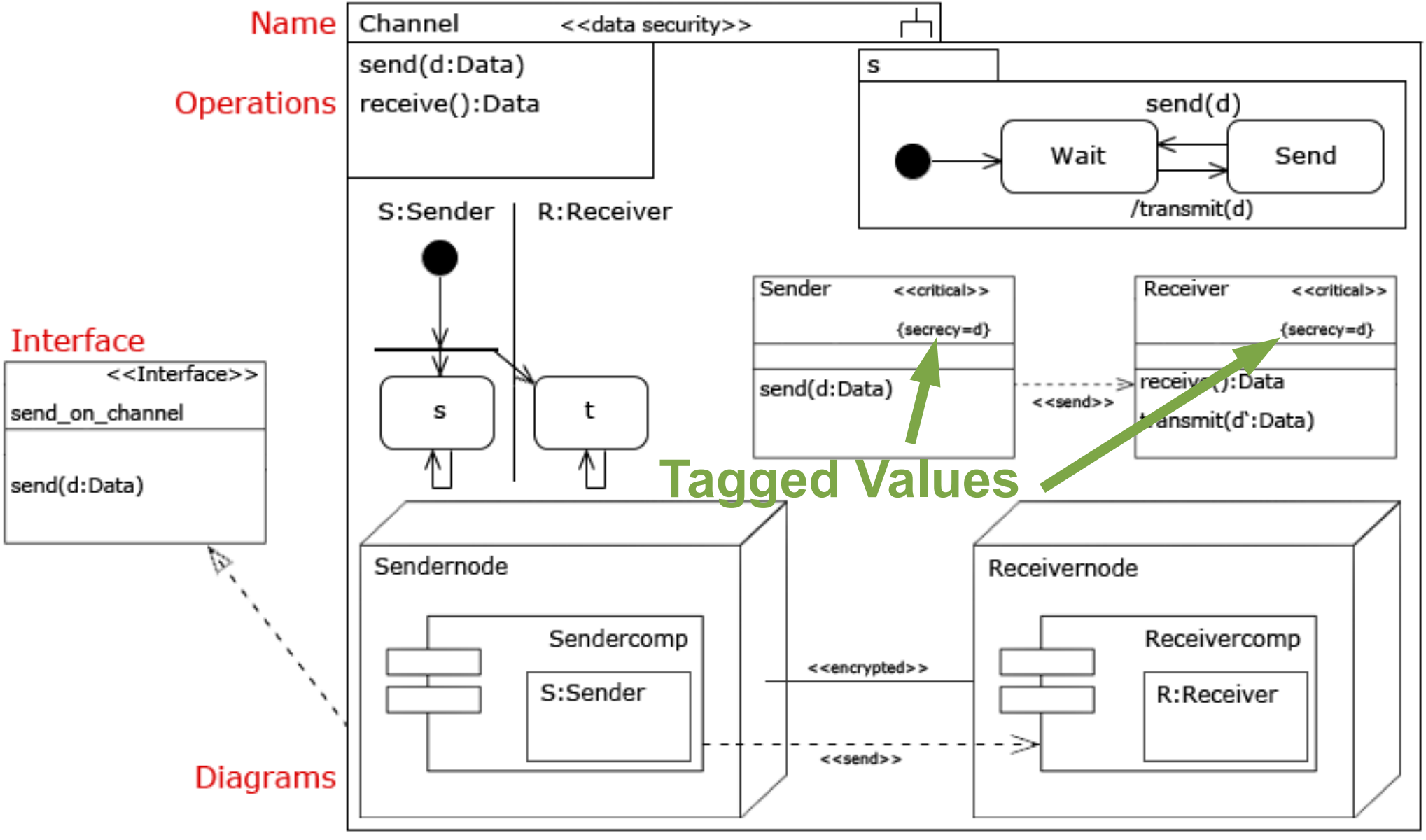
Constraint: verfeinert die Semantik eines stereotypisierten Elements.

Profil: sammelt obige Informationen.

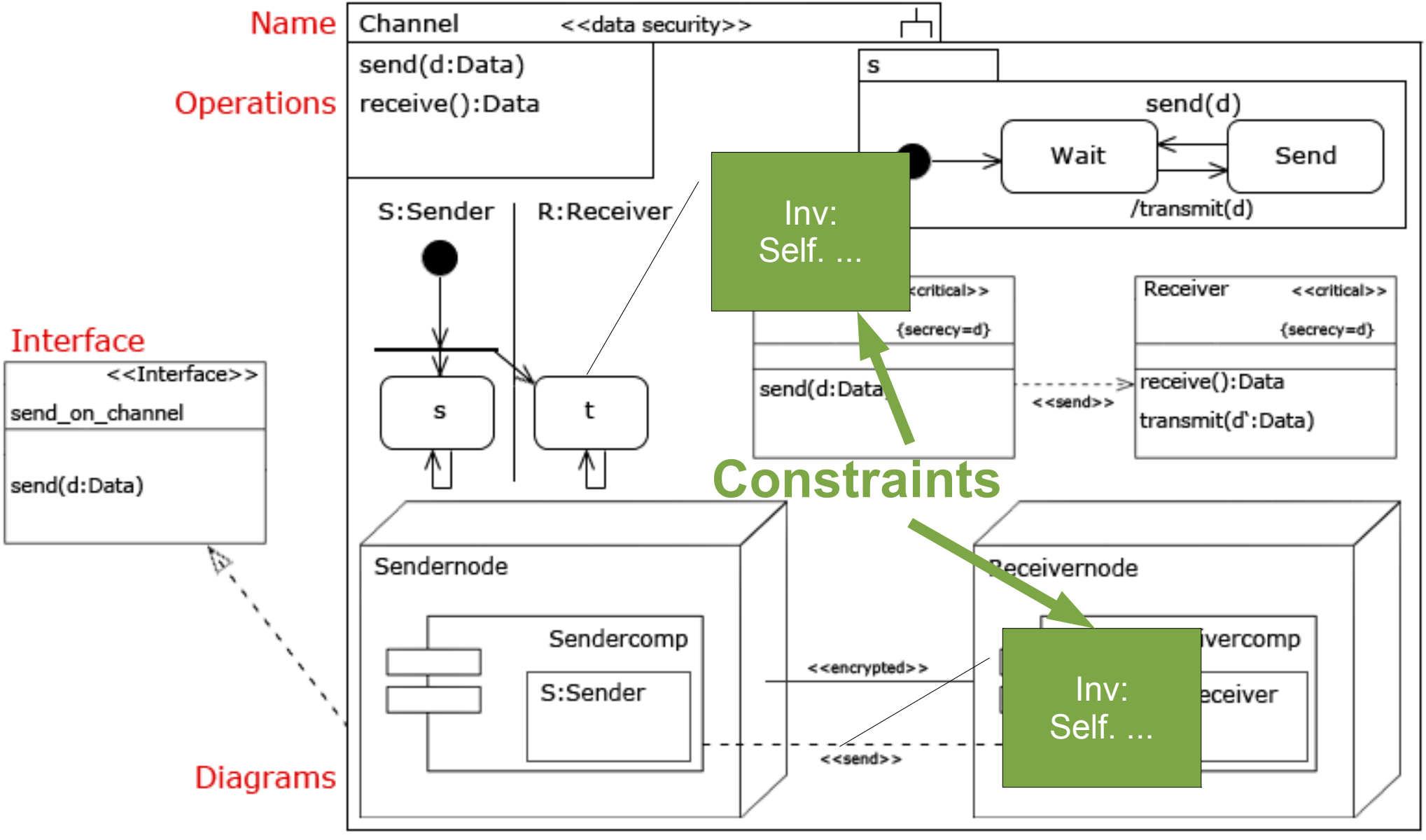
UML Profil
Erweiterungsmöglichkeiten



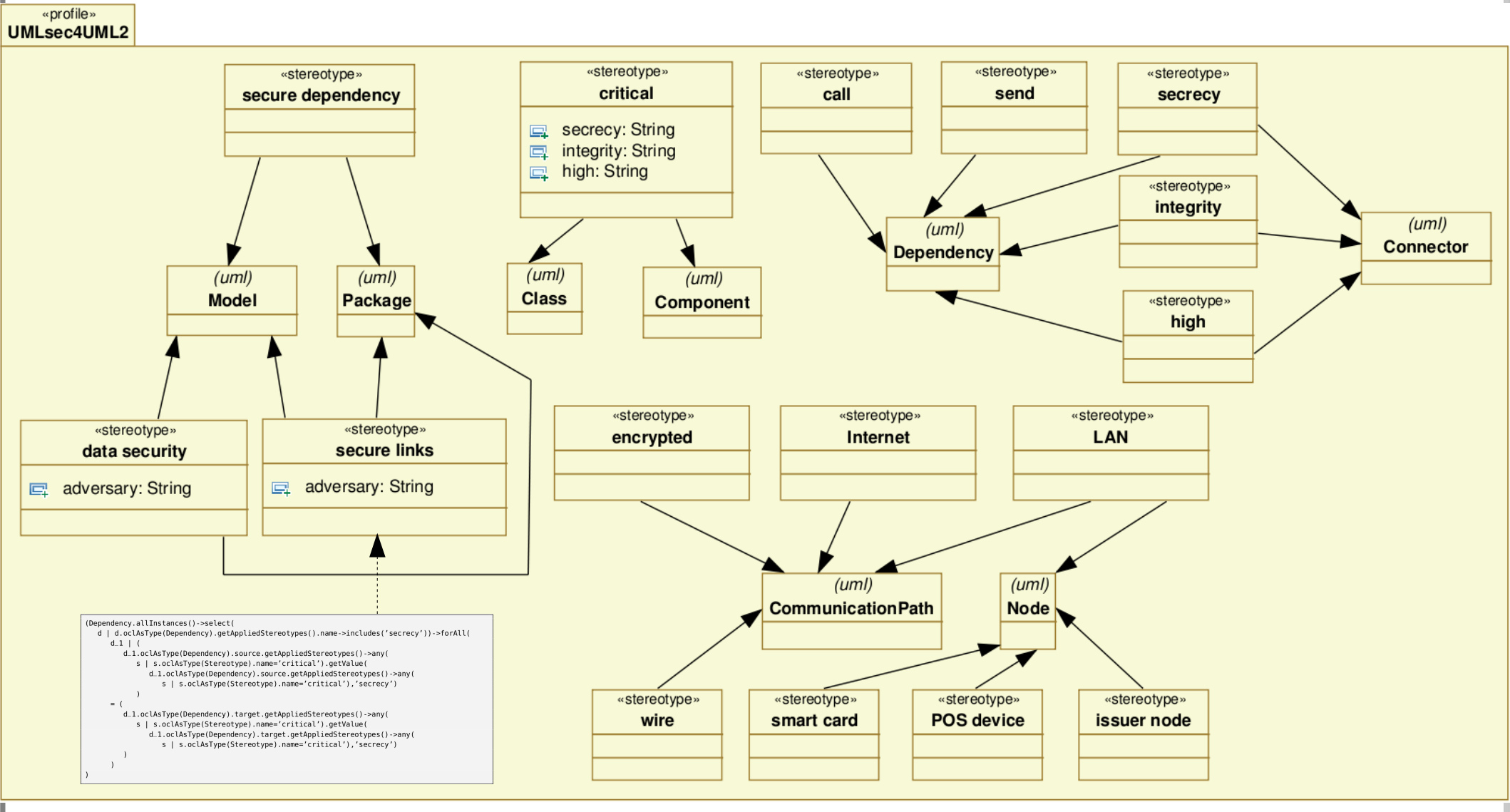
UML Profil
Erweiterungsmöglichkeiten



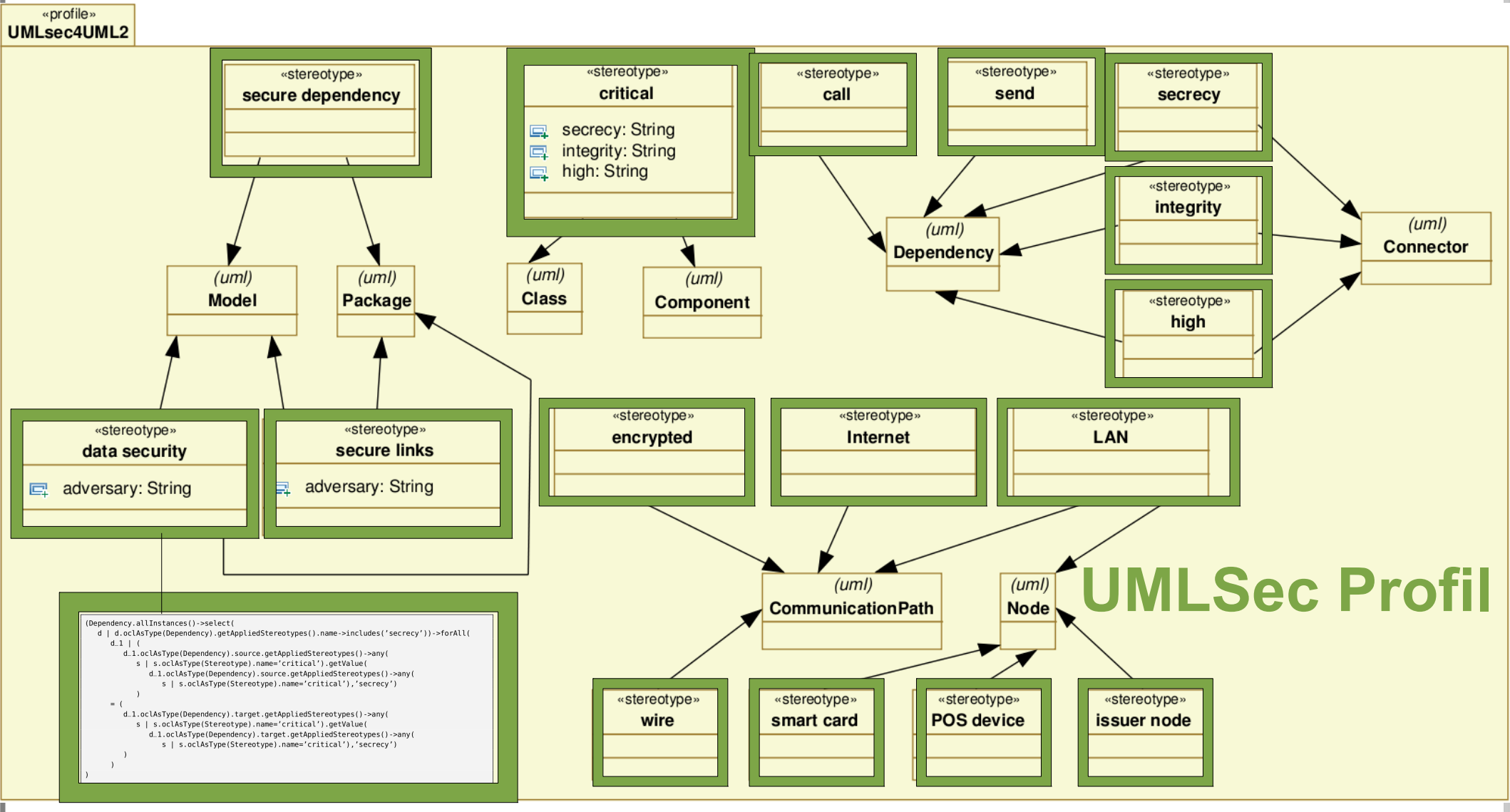
UML Profil
Erweiterungsmöglichkeiten



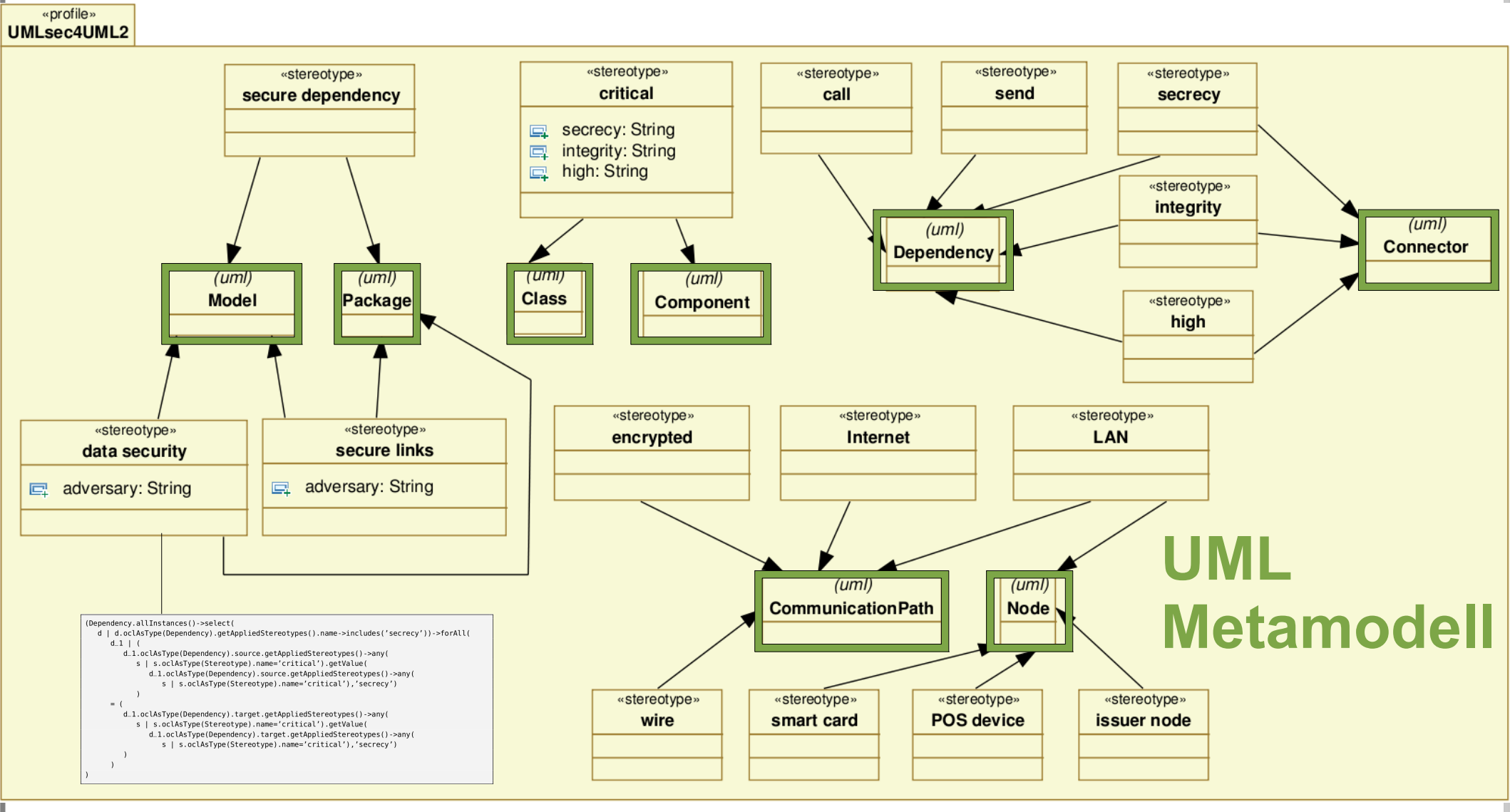
UML Profil
Beispiel



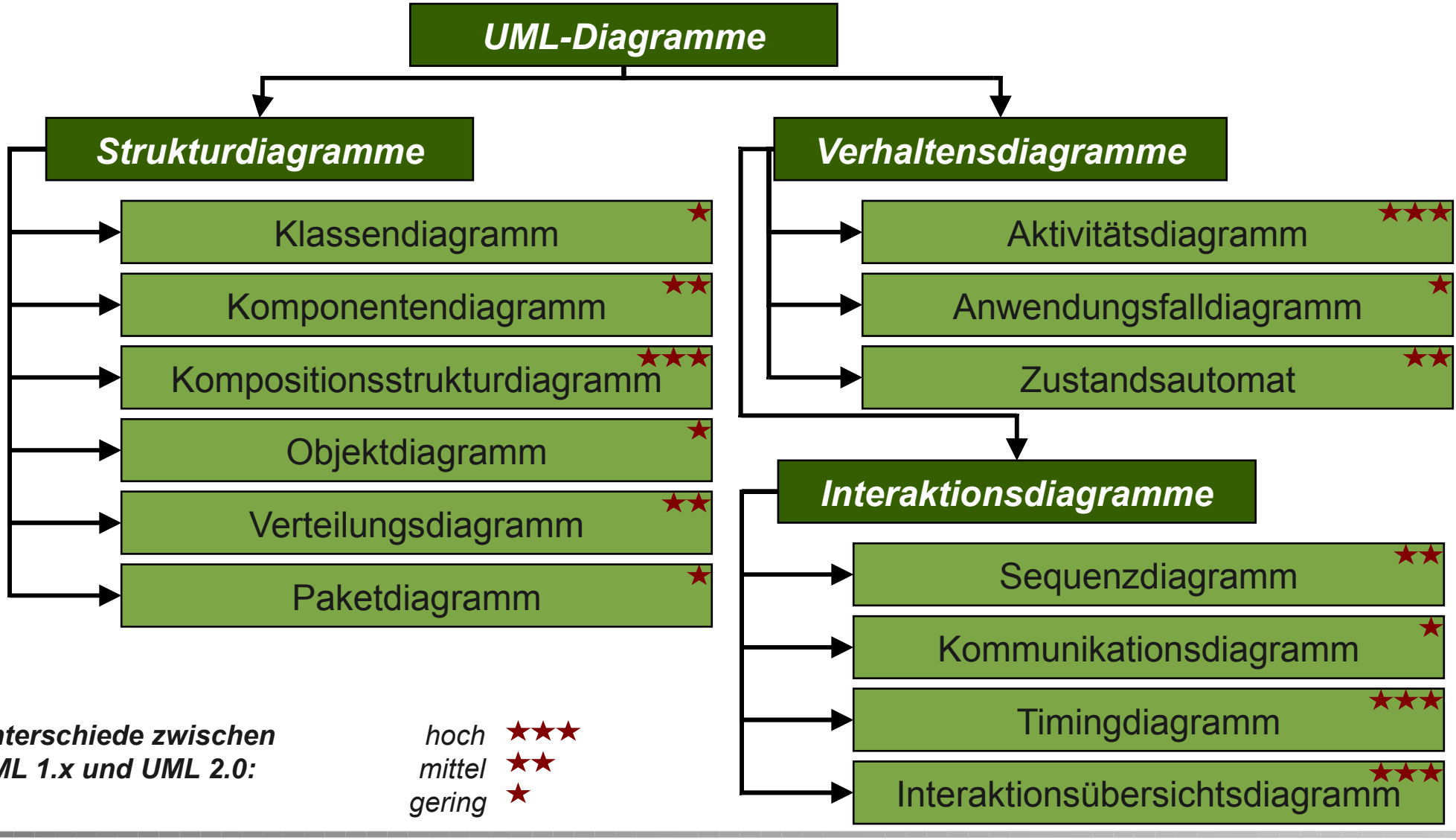
UML Profil
Beispiel



UML Profil
Beispiel



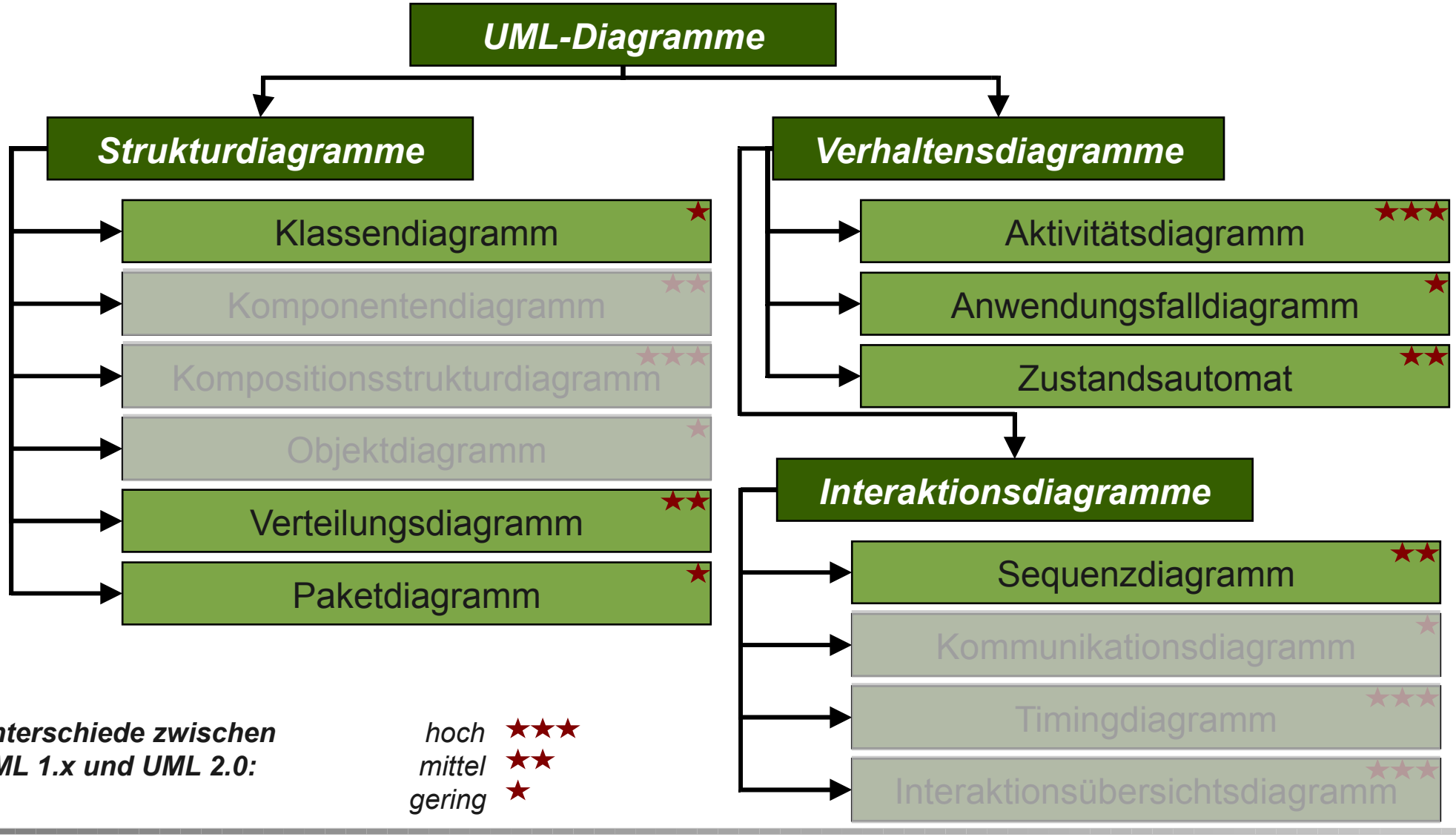
UML Diagrammtypen



Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

UMLsec-relevante Diagrammtypen



Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

UMLsec-relevante Diagrammtypen

Anwendungsfalldiagramm: Enthält die Anforderungen an das System

Klassendiagramm: Datenstruktur des Systems

Zustandsdiagramm: dynamisches Verhalten der Komponenten

Aktivitätsdiagramm: Steuerung des Ablaufs zwischen den Komponenten

Sequenzdiagramm: Interaktion durch Nachrichtenaustausch

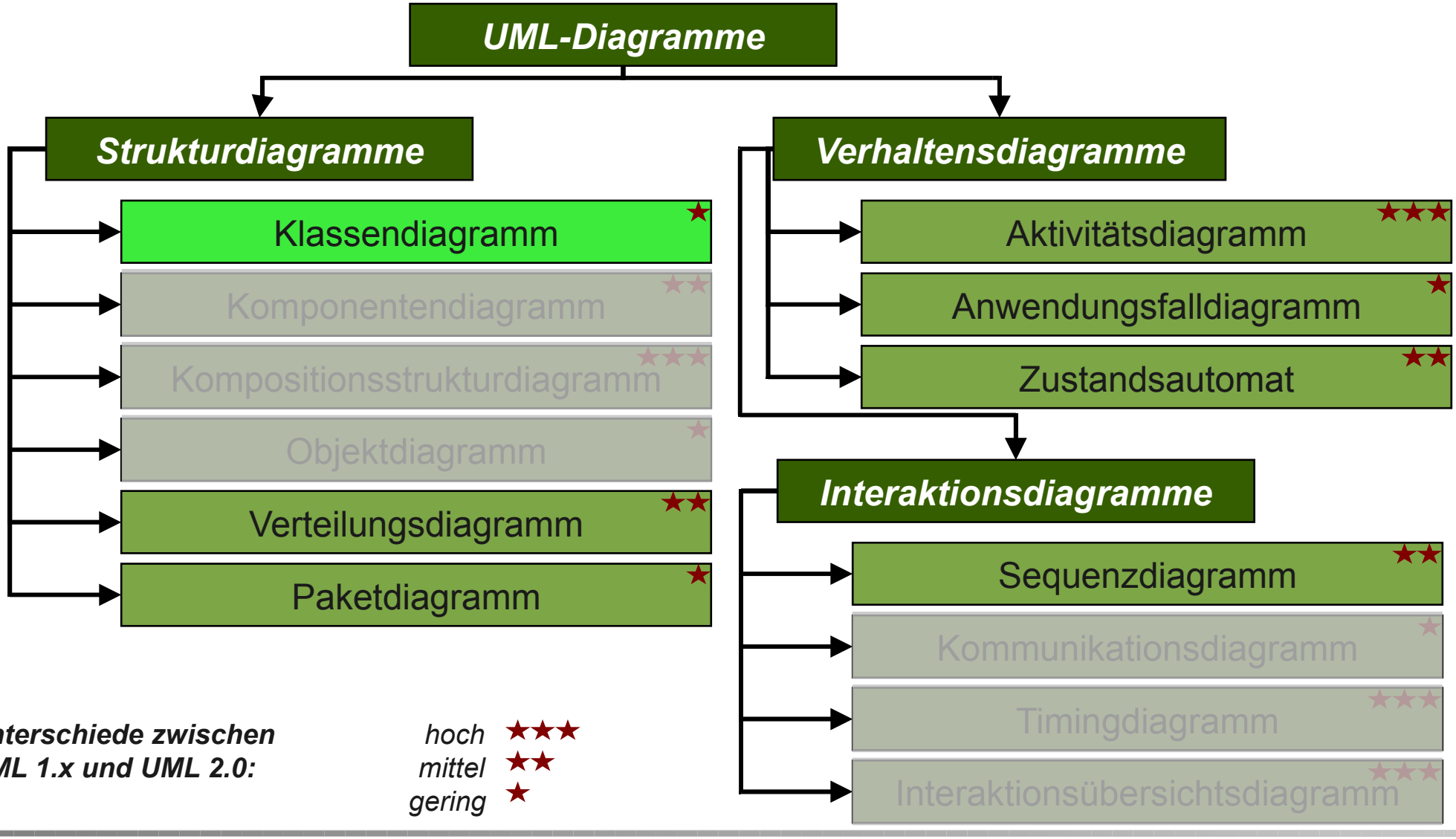
Verteilungsdiagramm: Physikalische Umgebung

Paket/Subsystem: Fasst Diagramme eines Systemteils zusammen

UMLsec Anmerkungen

- UML2 Vorstellung auf UMLsec relevante Diagramme beschränkt
- UMLsec auf UML 1.5 definiert
  - Beispiele können von UML 2 Abweichen
- UMLsec wurde zum Großteil schon auf UML 2 angehoben (Siehe UMLSec4UML2 Profil), aber um Verwirrung vorzubeugen alle Beispiele in UML 1.5

UML im Schnelldurchlauf
Klassendiagramm



Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

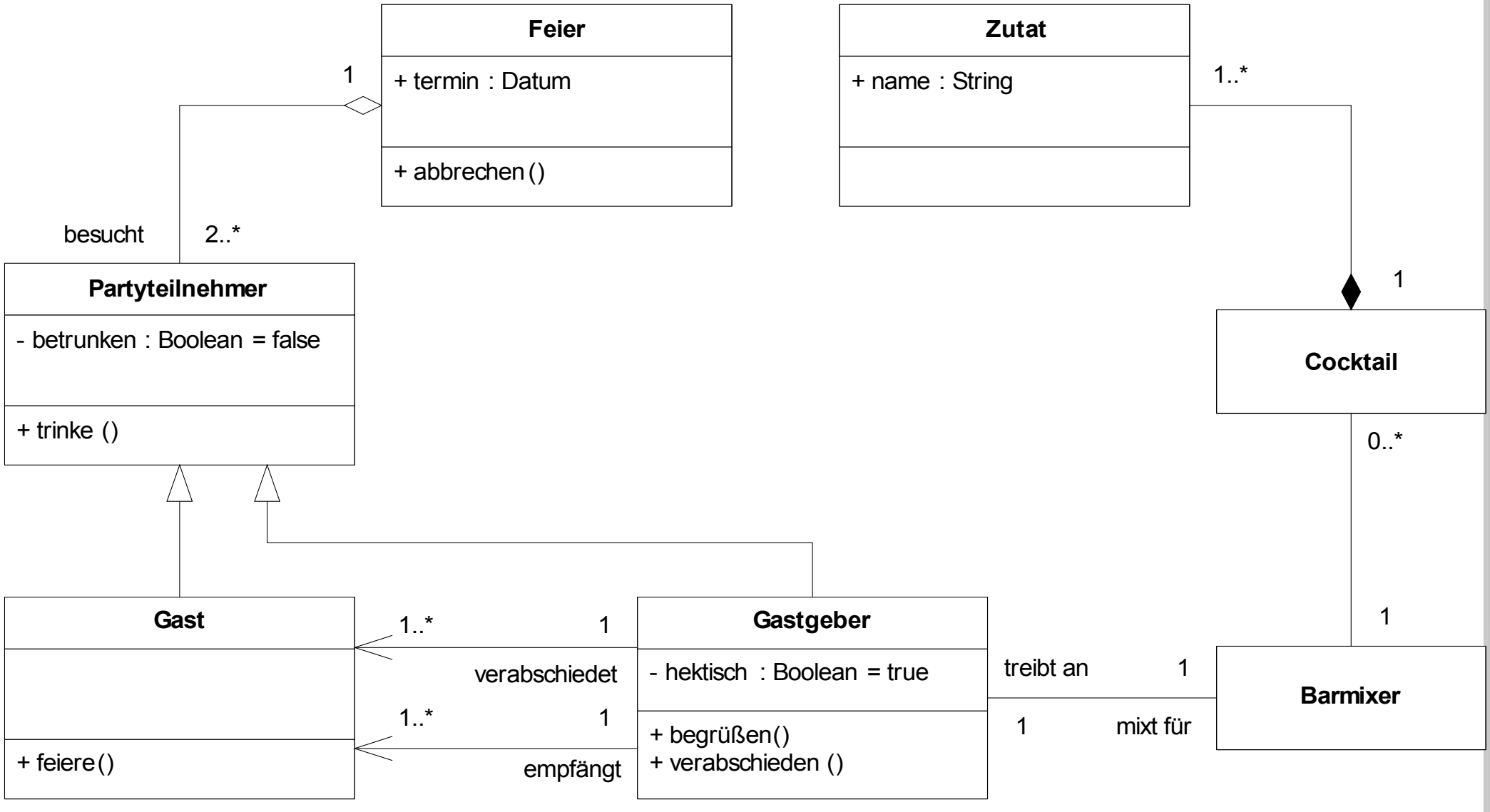
- Diese zentrale Frage beantwortet das Diagramm:
  - Aus welchen Klassen besteht mein System und wie stehen diese untereinander in Beziehung?

- Diese Stärken hat das Diagramm:
  - beschreibt die statische Struktur des Systems
  - enthält alle Strukturzusammenhänge und Datentypen
  - ist Brücke zu dynamischen Diagrammen

- Änderungen gegenüber früheren UML-Versionen
  - Unsauberkeiten wurden beseitigt, einige Elemente wurden überarbeitet (z. B. copy- oder become-Stereotypen).
  - Sonst keine wesentlichen Änderungen.

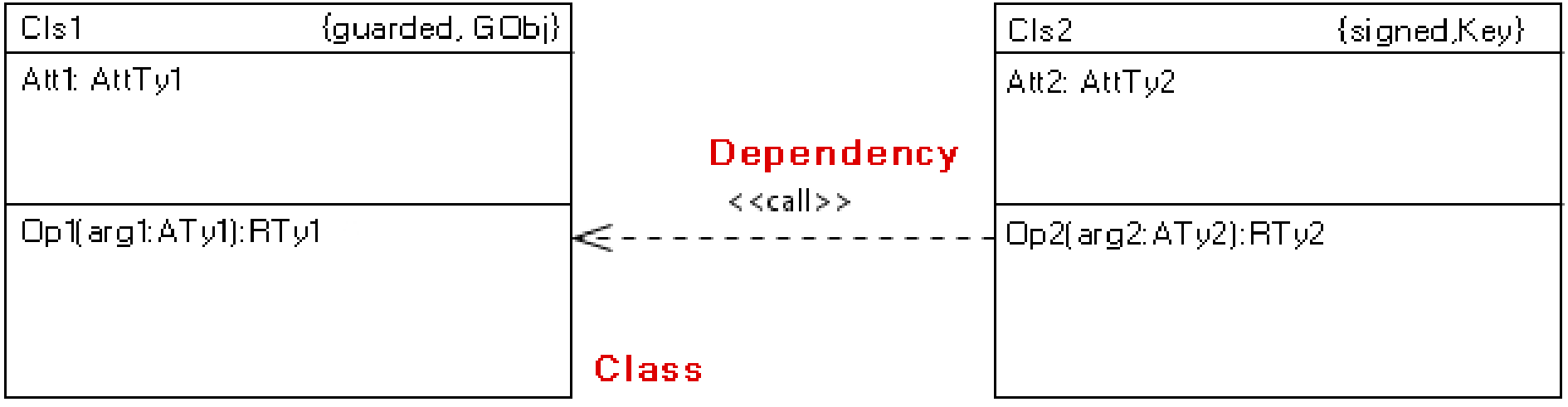
- Besondere Hinweise
  - Klassen sollten mit einem Hauptwort im Singular benannt werden.
  - In den Namen sollten Sonderzeichen und Umlaute vermieden werden.

UML im Schnelldurchlauf
Klassendiagramm

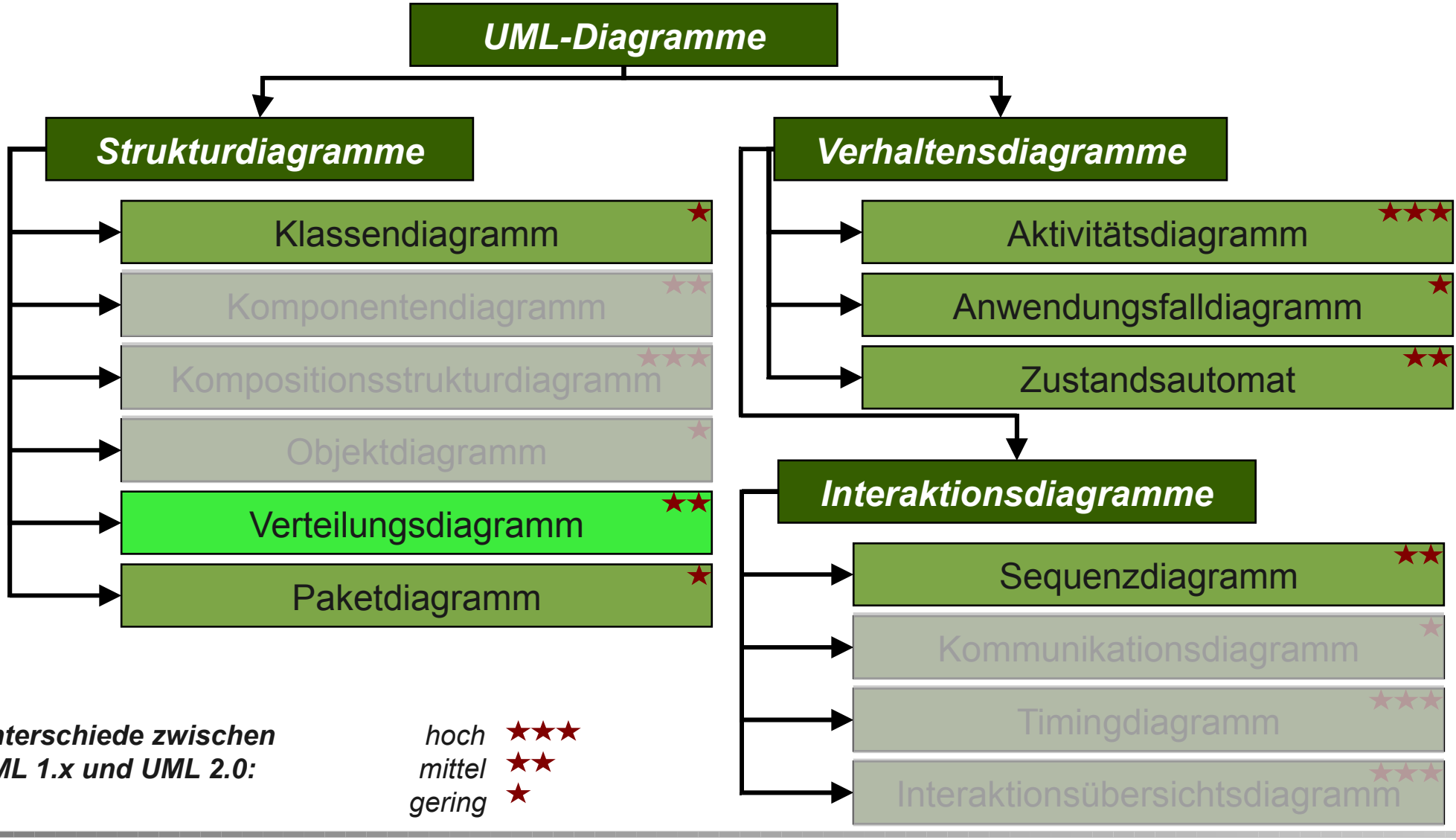


Klassenstruktur des Systems.

Klasse mit Attributen und Operation / Signalen,
Beziehungen zwischen Klassen.



UML im Schnelldurchlauf
Verteilungsdiagramm



Unterschiede zwischen UML 1.x und UML 2.0:

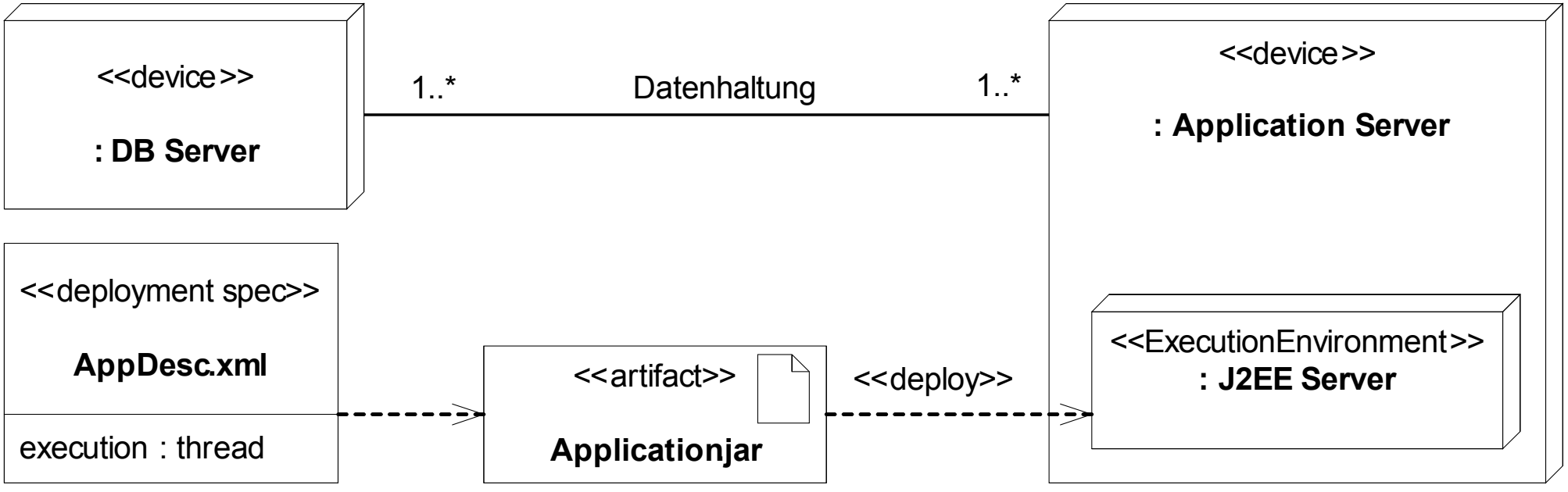
hoch ★★★
mittel ★★
gering ★

- Problem:
  - Es soll die Verteilung der Software des Systems auf die Hardware beschrieben werden.
- Diese zentrale Frage beantwortet das Diagramm:
  - Wie sieht das Einsatzumfeld (Hardware, Server, Datenbanken etc.) des Systems aus?
  - Wie werden die Komponenten zur Laufzeit wohin verteilt?

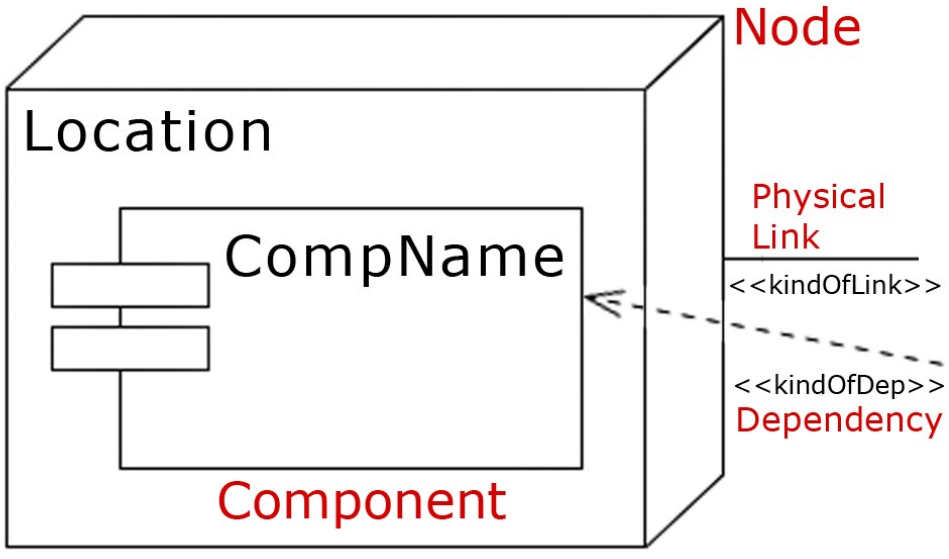
- Diese Stärken hat das Diagramm:
  - zeigt das Laufzeitumfeld des Systems mit den greifbaren Systemteilen
  - Darstellung von „Softwareservern“ möglich
  - hohes Abstraktionsniveau, kaum Notationselemente

- Änderungen gegenüber früheren UML-Versionen
  - Es wurde eine bessere Unterscheidung zwischen Soft- und Hardware-Ausführungsumgebungen vorgenommen.
  - Verwendung von Artefakten.
- Besondere Hinweise
  - Beschränkt euch durchgängig auf eine Darstellungsvariante, wenn mehrere Notationsvarianten möglich sind (z.B. mit oder ohne Deploy-Beziehungen).

UML im Schnelldurchlauf
Verteilungsdiagramm

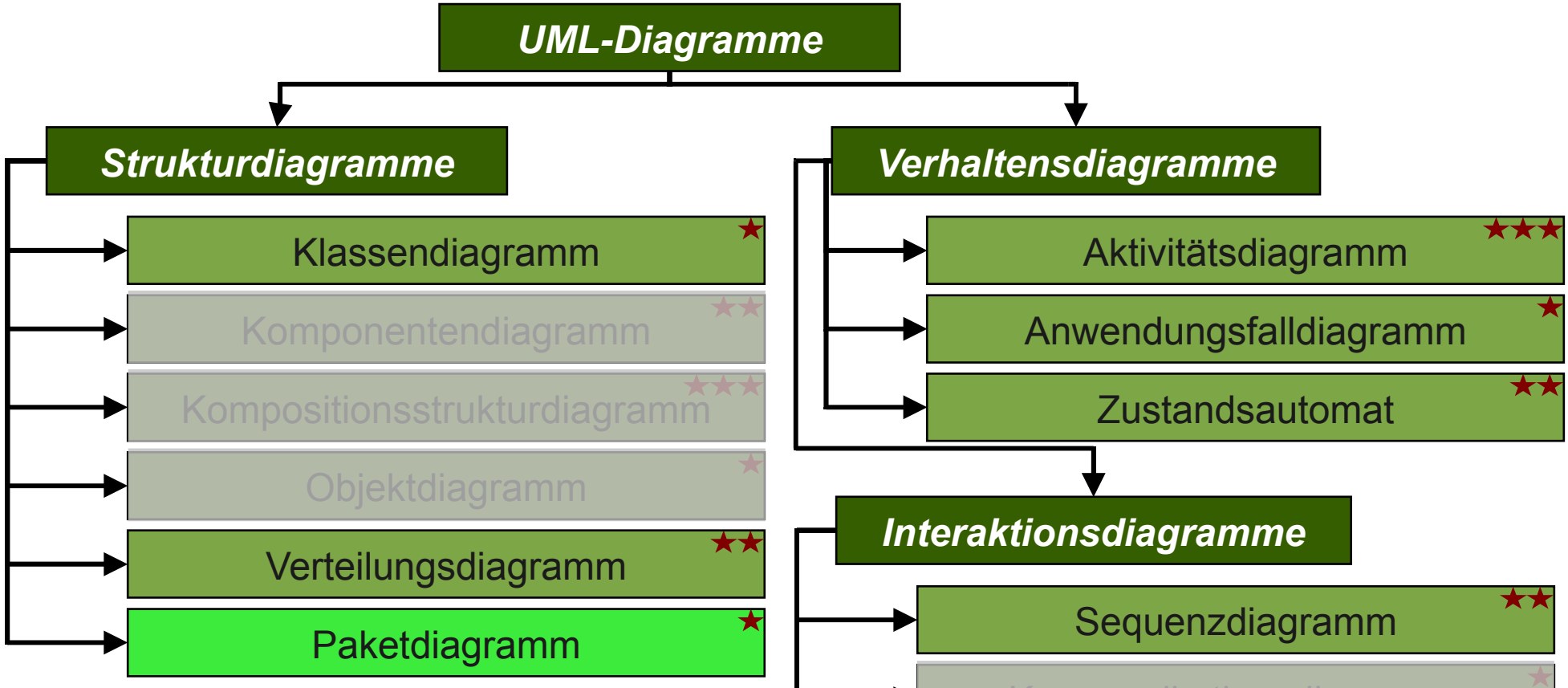


UMLsec Beispiel Verteilungsdiagramm



Erklärt die physikalische Ebene, auf der das System implementiert wird.

UML im Schnelldurchlauf
Paketdiagramm



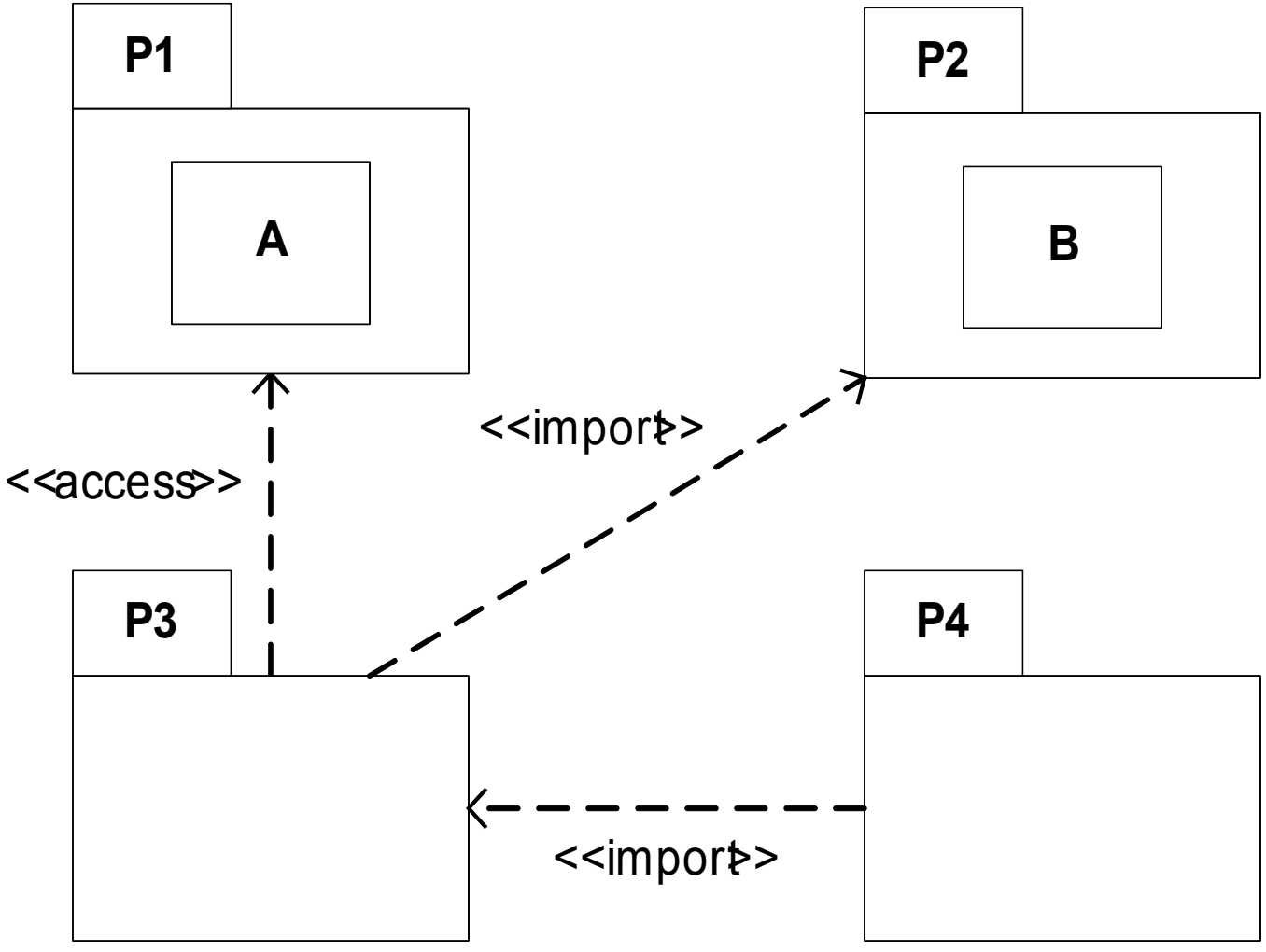
Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

- Problem:
  - Große Softwaresysteme mit mehreren 100 Klassen lassen sich nicht mehr von einer Person überblicken.
- Diese zentrale Frage beantwortet das Diagramm:
  - Wie kann ich mein Modell so schneiden, dass ich den Überblick bewahre?
- Diese Stärken hat das Diagramm:
  - organisiert das Systemmodell in größeren Einheiten durch logische Zusammenfassung von Modellelementen
  - Modellierung von Abhängigkeiten und Inklusionen ist möglich

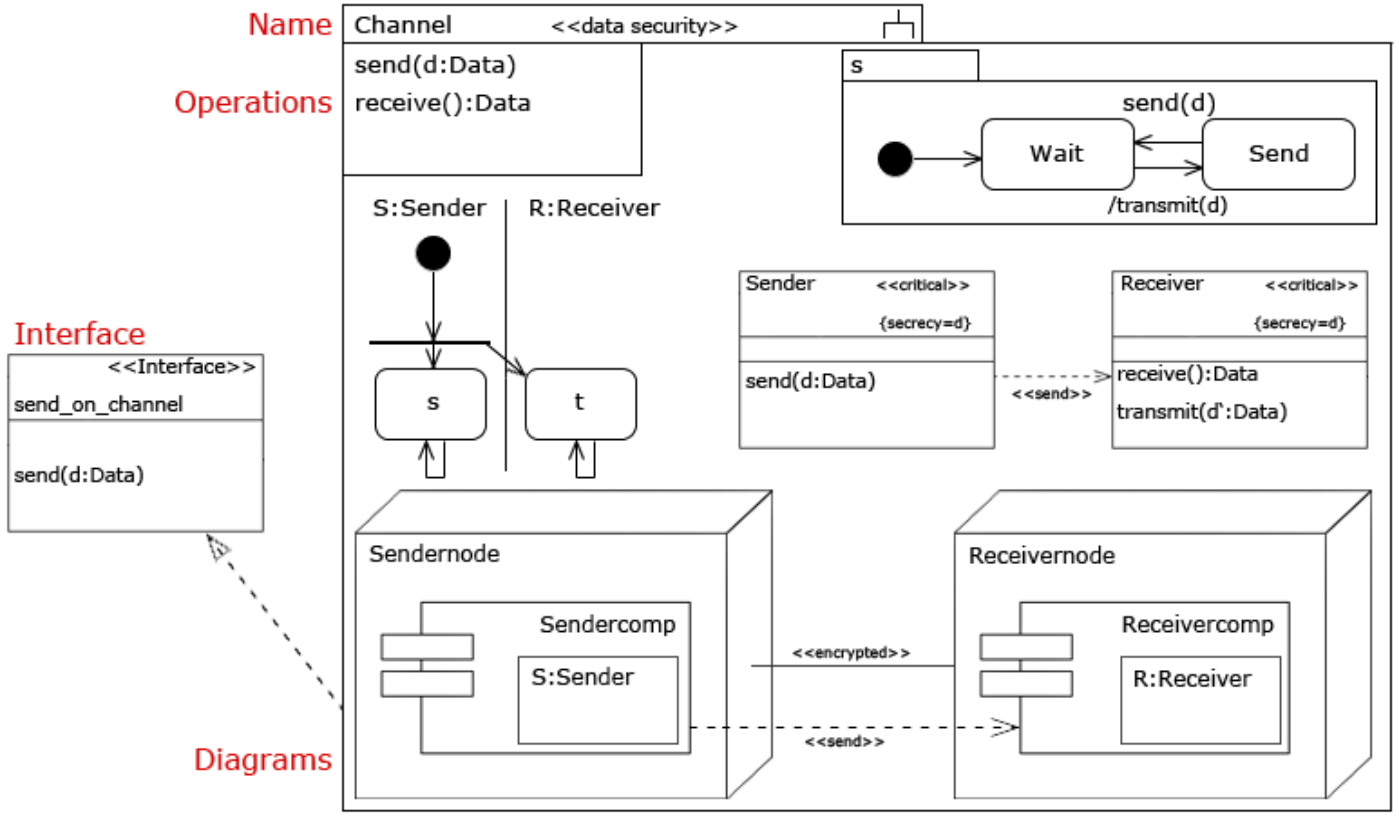
- Änderungen durch UML 2:
  - Insgesamt: keine Änderungen!
  - Jedoch: Der Importmechanismus (insbesondere Behandlung und Verschmelzung von Namensräumen) prominenter dokumentiert

UML im Schnelldurchlauf
Paketdiagramm



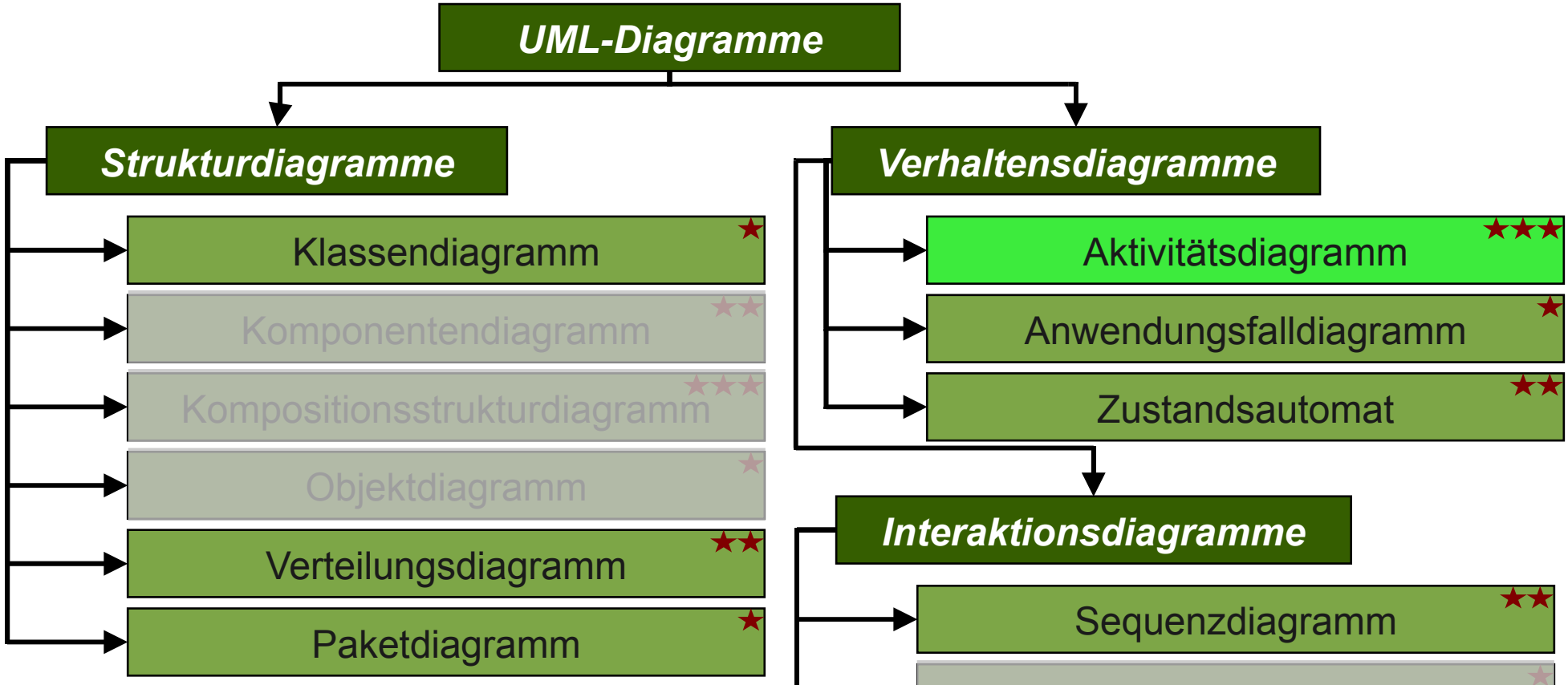
- Inhalt:
  - P3 importiert A (P1) als private
  - P3 importiert B (P2) als public
  - P4 importiert P3
- Folge:
  - P4 darf nicht auf A zugreifen
  - P4 darf auf B zugreifen

UMLsec Beispiel
Paketdiagramm



Kann benutzt werden, um UML Element zu einer Gruppe zusammen zu fügen.

UML im Schnelldurchlauf
Aktivitätsdiagramm



Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

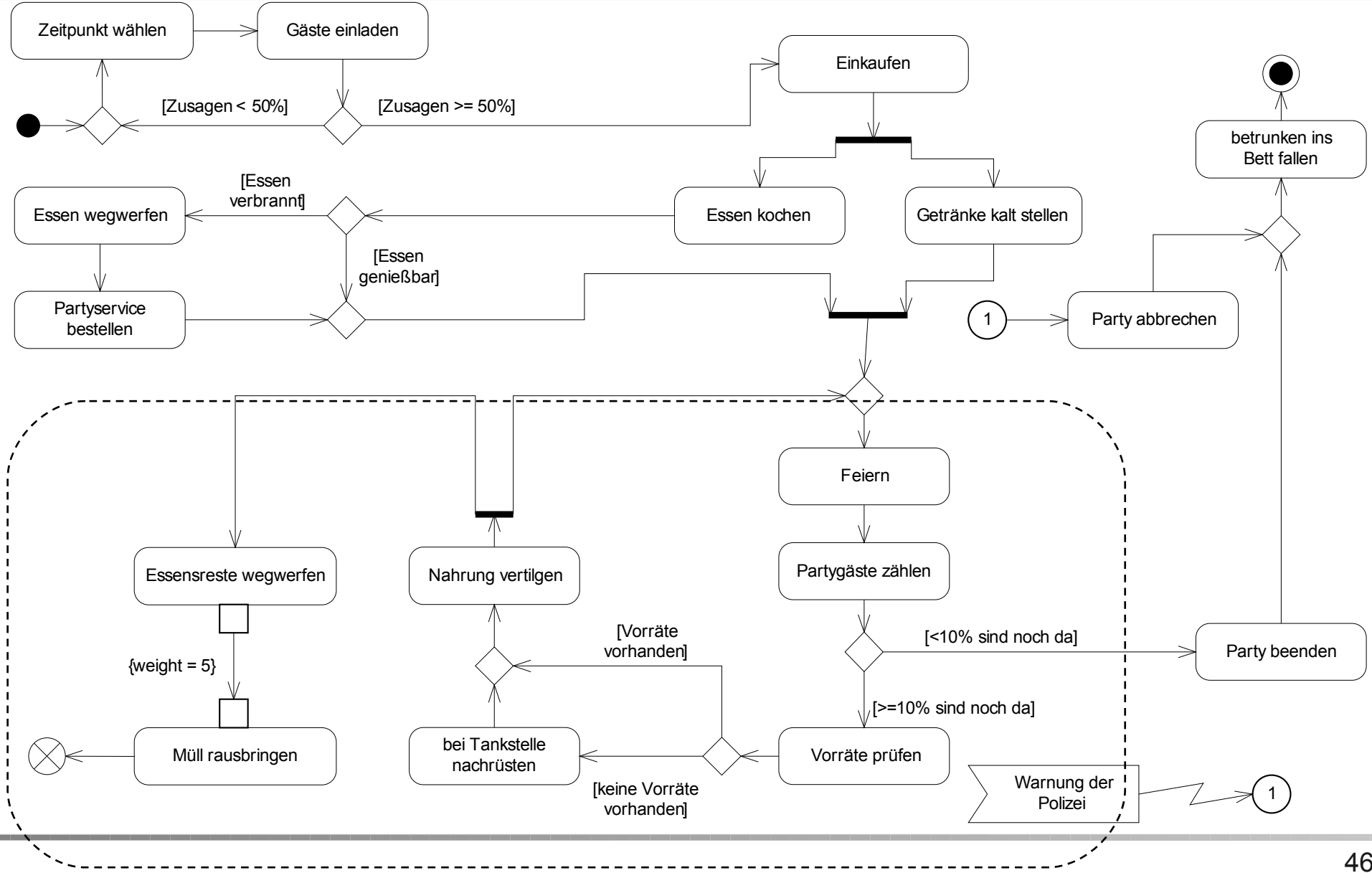
- Problem:
  - Es sollen Abläufe, z.B. Geschäftsprozesse, modelliert werden. Im Vordergrund steht dabei eine Aufgabe, die in Einzelschritte zerlegt werden soll.
  - Es sollen Details eines Anwendungsfalles festgelegt werden.
- Diese zentrale Frage beantwortet das Diagramm:
  - Wie realisiert mein System ein bestimmtes Verhalten?

- Diese Stärken hat das Diagramm:
  - Detaillierte Visualisierung von Abläufen mit Bedingungen, Schleifen und Verzweigungen.
  - Parallelisierung und Synchronisation ist möglich.
  - Darstellung von Daten- und Kontrollflüssen.

- Änderungen gegenüber früheren UML-Versionen
  - sind jetzt keine Sonderform der Zustandsdiagramme mehr, sondern basieren auf erweiterten Petri-Netzen
  - damit sind viele Einschränkungen beseitigt:
    - verbesserte Testbarkeit
    - fast automatisch erkennbare Verklemmungsfreiheit
    - bessere Unterstützung paralleler Flüsse
    - Ausführbarkeit fast vollständig möglich
    - mehr Flexibilität in der Modellierung durch das Tokenkonzept

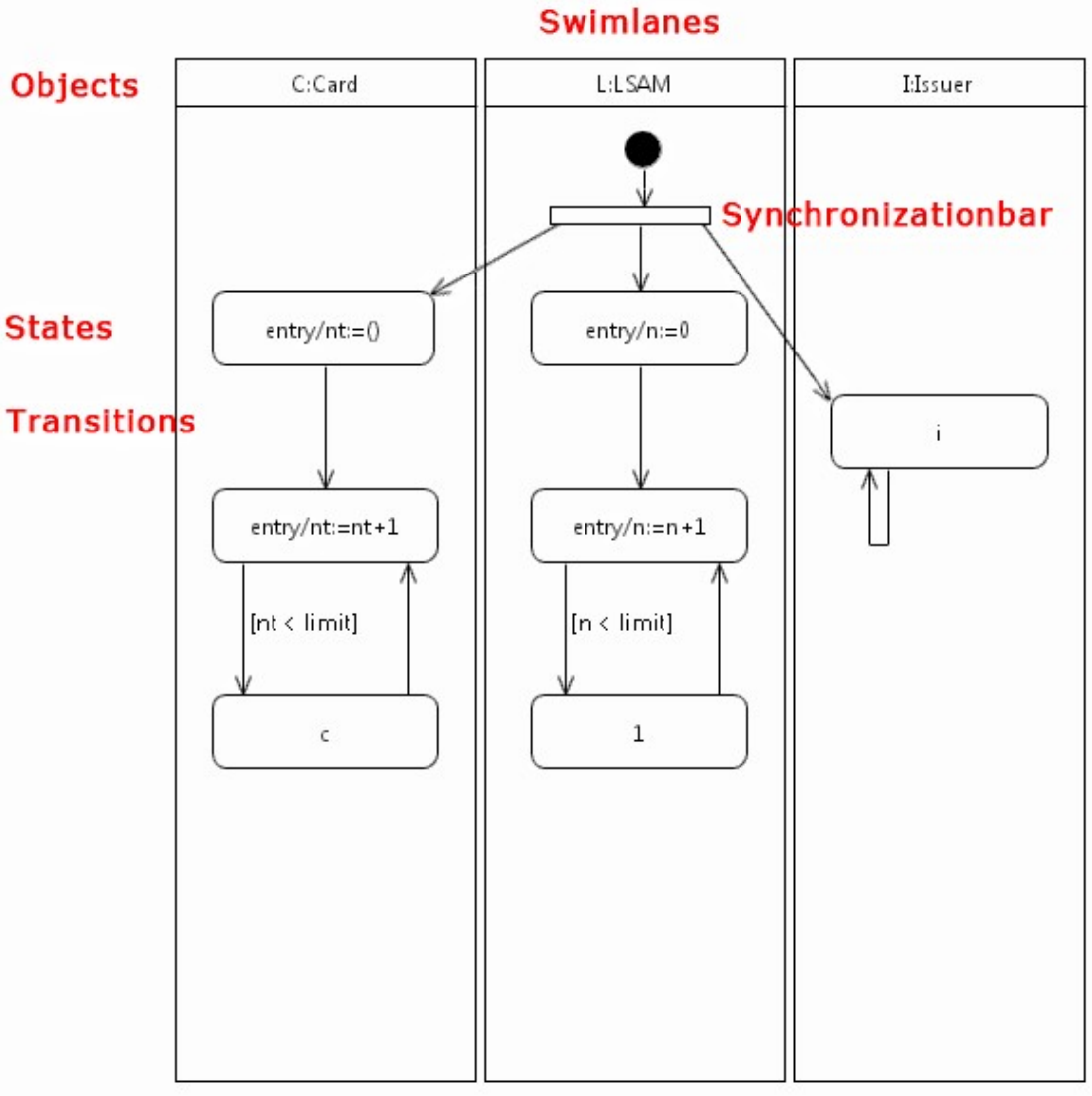
- Tokenkonzept
  - Ein Token (auch: Marke) zeigt an, an welchem Punkt sich der Ablauf gerade befindet.
  - Es können beliebig viele Token unterwegs sein (parallele Abläufe).
  - Token werden graphisch nicht dargestellt, sondern dienen nur der Erklärung der Abläufe.

- Besondere Hinweise
  - Für die Modellierung reaktiver Systeme sollten lieber Zustandsautomaten verwendet werden.
  - Verzweigungen und deren Bedingungen sollten immer exakt modelliert werden.

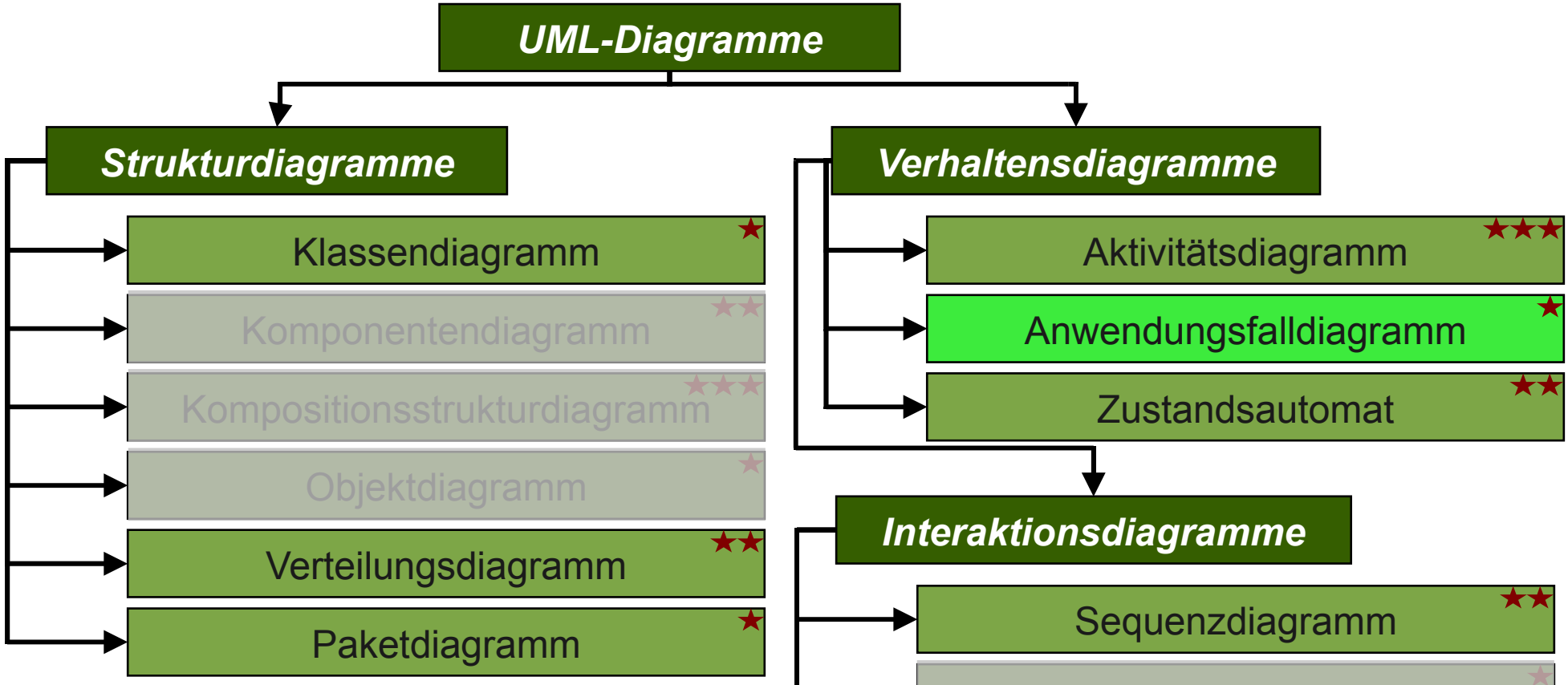

Zeitpunkt wählen
Gäste einladen
[Zusagen < 50%]
[Zusagen >= 50%]
Einkaufen
Essen wegwerfen
[Essen verbrannt]
Essen kochen
Getränke kalt stellen
[Essen genießbar]
Partyservice bestellen
1
Party abbrechen
betrunken ins Bett fallen
Feiern
Essensreste wegwerfen
Nahrung vertilgen
Partygäste zählen
{weight = 5}
[Vorräte vorhanden]
[<10% sind noch da]
Party beenden
[>=10% sind noch da]
Müll rausbringen
bei Tankstelle nachrüsten
Vorräte prüfen
[keine Vorräte vorhanden]
Warnung der Polizei
1

UMLsec Beispiel Aktivitätsdiagramm

Spezifiziert den Kontrollfluss zwischen Komponenten desselben Systems. Ist auf einer höheren Abstraktionsebene als Zustands- und Sequenzdiagramme.



UML im Schnelldurchlauf
Anwendungsfalldiagramm

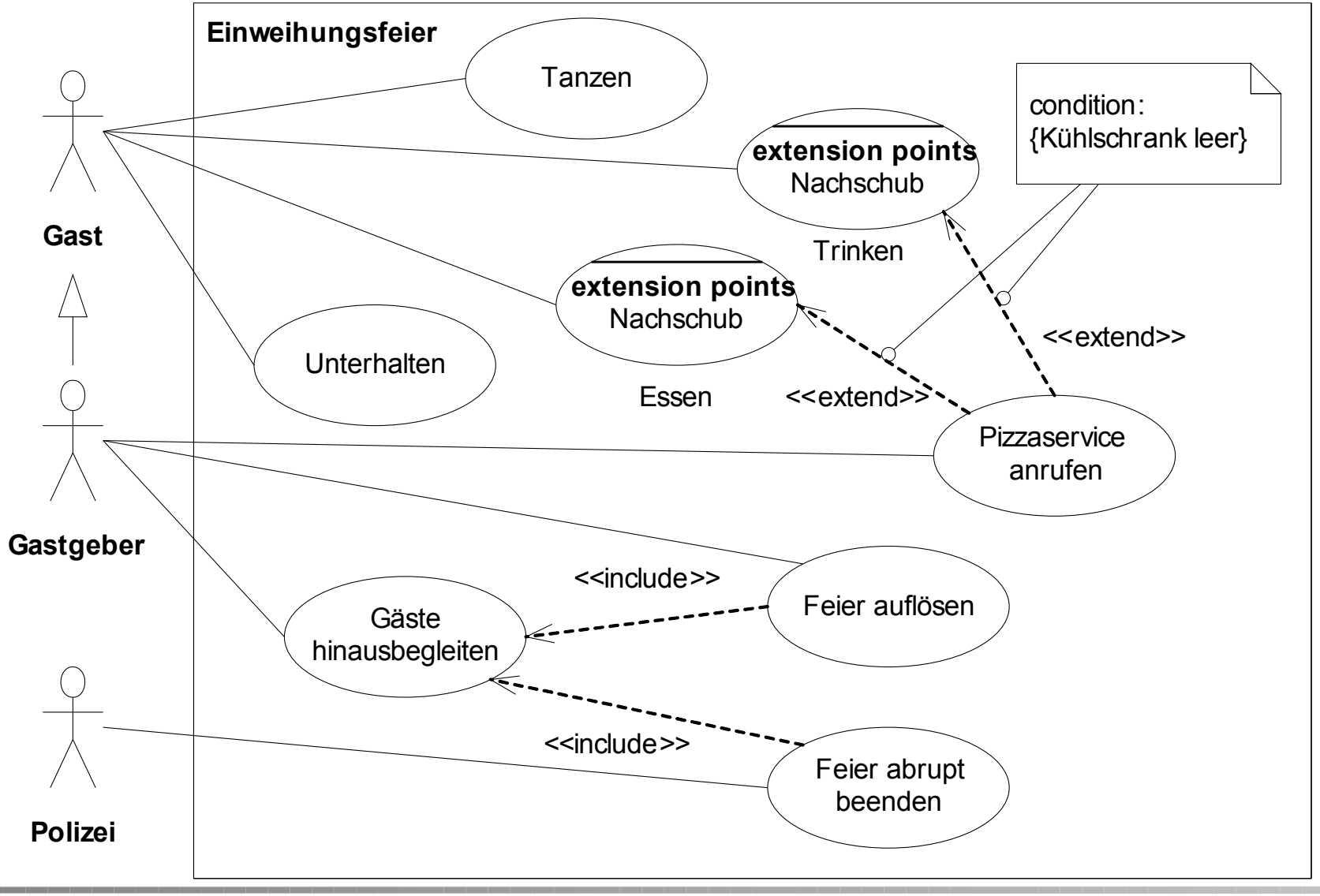


Unterschiede zwischen UML 1.x und UML 2.0:

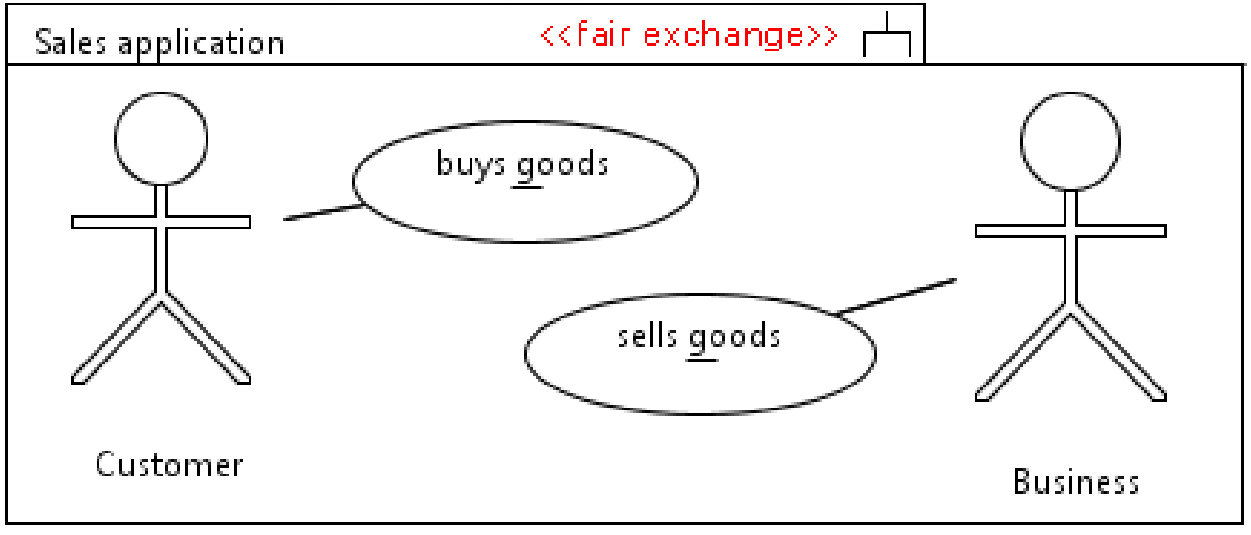
hoch ★★★
mittel ★★
gering ★

- Problem:
  - Das externe Verhalten des Systems soll aus Nutzersicht dargestellt werden.
- Diese zentrale Frage beantwortet das Diagramm:
  - Was leistet mein System für seine Umwelt (Nachbarsysteme, Stakeholder)?
- Diese Stärken hat das Diagramm:
  - präsentiert die Außensicht auf das System
  - geeignet zur Kontextabgrenzung
  - hohes Abstraktionsniveau, einfache Notationsmittel

- Änderungen gegenüber früheren UML-Versionen
  - Akteure müssen benannt werden
  - Die <<extend>>-Beziehung kann Vorbedingungen und extension points besitzen.
- Besondere Hinweise
  - Anwendungsfälle sollten nicht zur detaillierten Beschreibung von Operationen oder Funktionen verwendet werden.
  - Es sollten keine nicht-funktionalen Anforderungen beschrieben werden.
  - Anwendungsfälle sollten immer die Anwendersicht, nie die Entwicklersicht widerspiegeln.

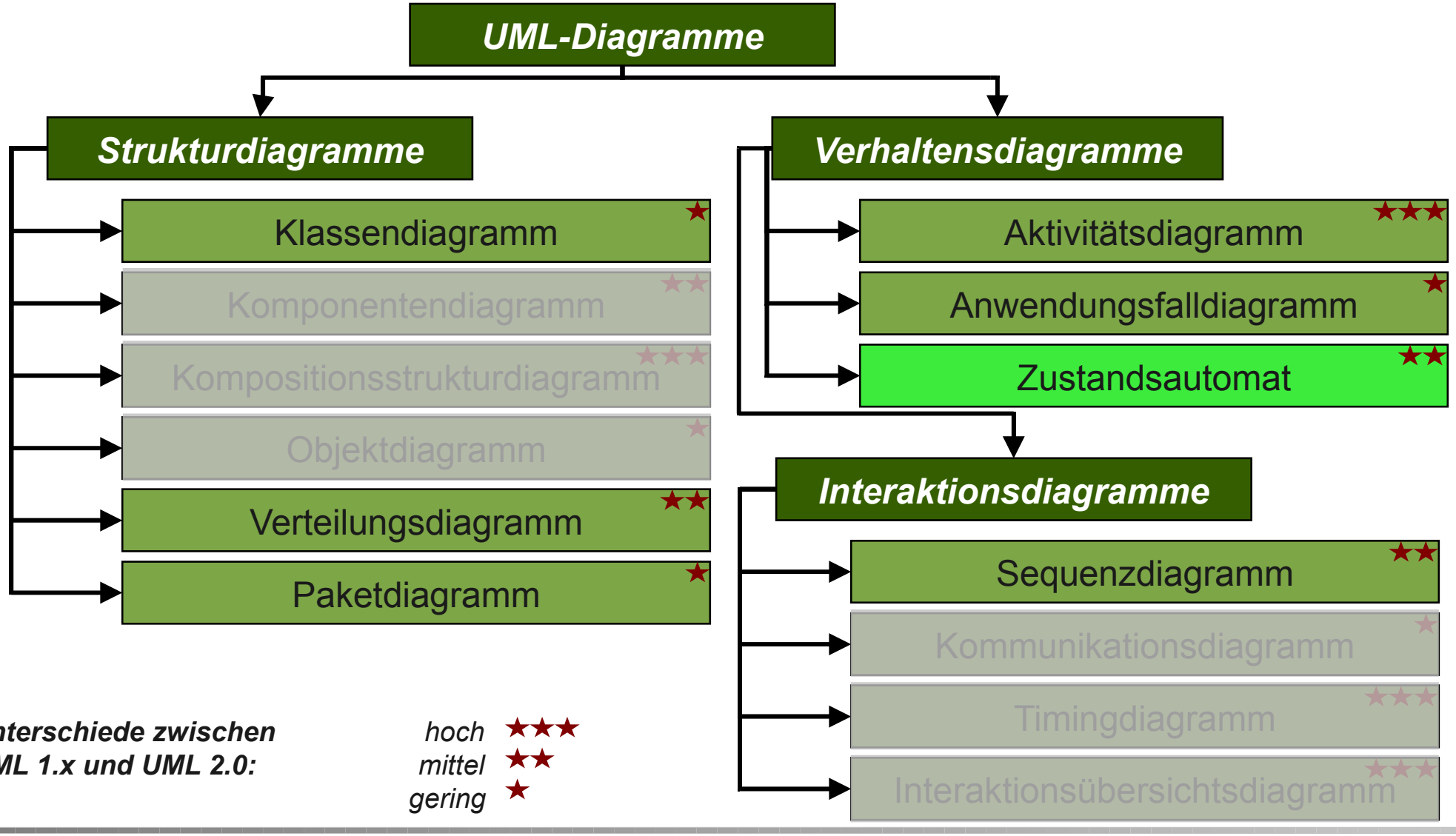

Einweihungsfeier
Tanzen
extension points
Nachschub
Trinken
condition:
{Kühlschrank leer}
Gast
extension points
Nachschub
Unterhalten
Essen
<<extend>>
<<extend>>
Pizzaservice
anrufen
Gastgeber
<<include>>
Gäste
hinausbegleiten
Feier auflösen
<<include>>
Feier abrupt
beenden
Polizei

UMLsec Beipspiel
Anwendungsfalldiagramm



Spezifiziert einen Anwendungsfall des Systems:
Szenario einer Funktionalität, die einem Benutzer oder einem anderen System angeboten wird.

Akteure die mit einer Aktivität verknüpft sind.

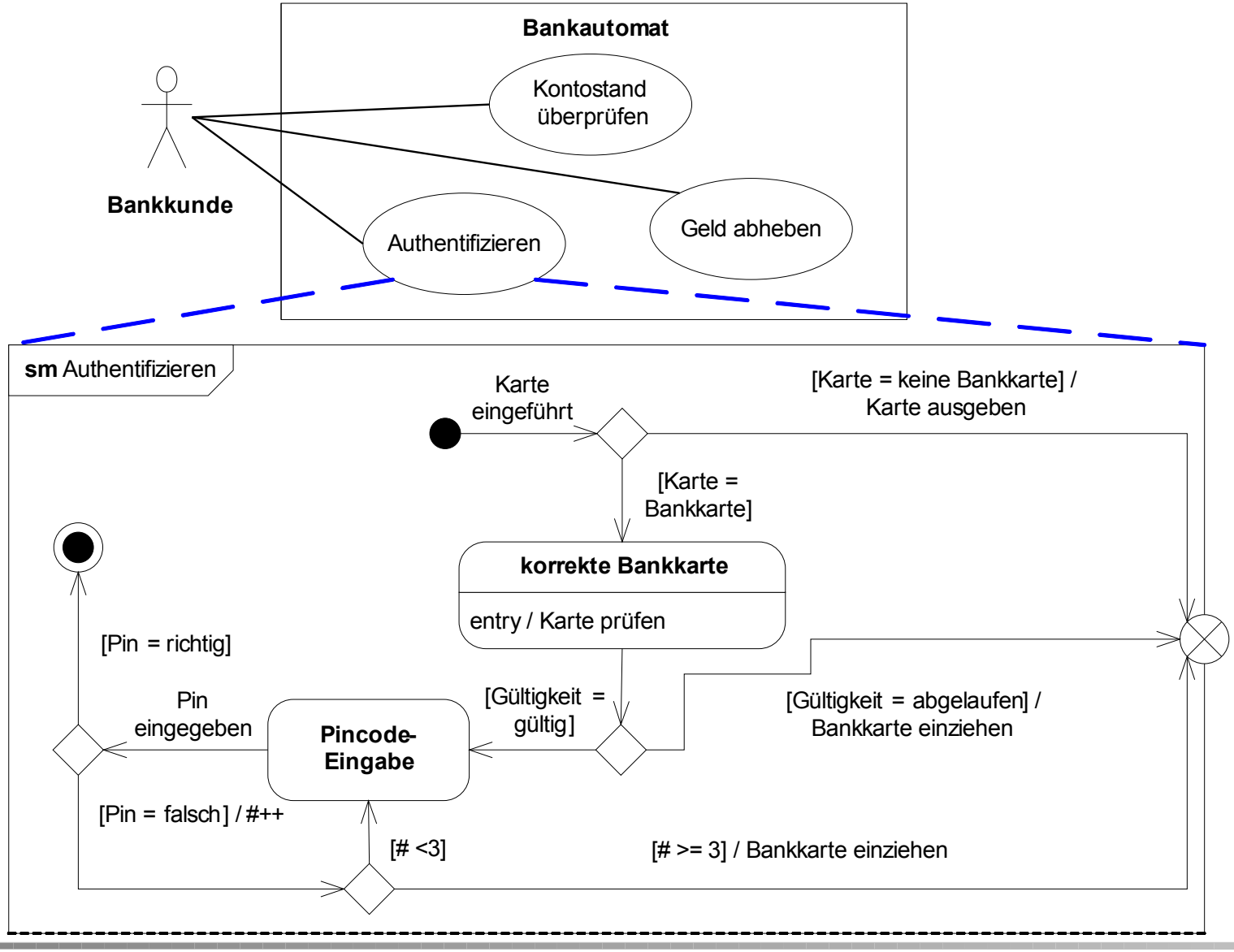

UML-Diagramme
Strukturdiagramme
Klassendiagramm ★
Komponentendiagramm ★★
Kompositionsstrukturdiagramm ★★★
Objektdiagramm ★
Verteilungsdiagramm ★★
Paketdiagramm ★
Verhaltensdiagramme
Aktivitätsdiagramm ★★★
Anwendungsfalldiagramm ★
Zustandsautomat ★★
Interaktionsdiagramme
Sequenzdiagramm ★★
Kommunikationsdiagramm ★
Timingdiagramm ★★★
Interaktionsübersichtsdiagramm ★★★
Unterschiede zwischen UML 1.x und UML 2.0:
hoch ★★★
mittel ★★
gering ★

- Diese zentrale Frage beantwortet das Diagramm:
  - Welche Zustände kann ein Objekt, eine Schnittstelle, ein Anwendungsfall etc. bei welchen Ereignissen annehmen?
- Diese Stärken hat das Diagramm:
  - Präzise Abbildung eines Zustandmodells mit Zuständen, Ereignissen, Nebenläufigkeiten, Bedingungen, Ein- und Austrittsaktionen
  - Schachtelung möglich

- Beschreibung des Verhaltens von Anwendungsfällen
  - Die formale Modellierung von Anwendungsfällen hat folgende Vorteile:
    - Sie sind eindeutig und weniger interpretierbar.
    - Es können Testfälle abgeleitet werden.
- Beschreibung des Verhaltens von Klassen
  - Dem Attribut einer Klasse wird im Allgemeinen ein Datentyp zugeordnet.
  - Wenn der Datentyp eine endliche Menge von gültigen Werten besitzt, dann ergeben sich die verschiedenen Zustände der Klasse aus allen möglichen Kombinationen dieser Zustandsvariablen.

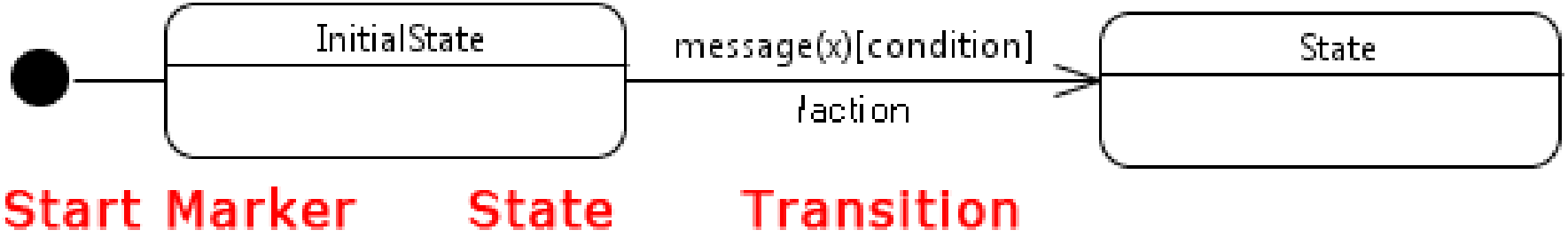
- Protokollzustandsautomaten
  - Unter einem Protokoll versteht man hier die erlaubte Abfolge von Aufrufen der Operationen, die von einem Classifier angeboten werden.

- Änderungen gegenüber früheren UML-Versionen
  - verbesserte Verknüpfung von statischen Elementen und dahinter liegenden Zustandsmodellen
  - Protokollzustandsautomaten wurden zur präzisen Definition von Schnittstellen und Ports eingeführt.
- Besondere Hinweise
  - Es sollten niemals Zustände existieren, die nur eingehende, aber keine ausgehenden Transitionen besitzen (ansonsten ist das Modell unvollständig).
  - Es sollten nach Möglichkeit häufig Kreuzungspunkte verwendet werden, um die Lesbarkeit zu erhöhen.

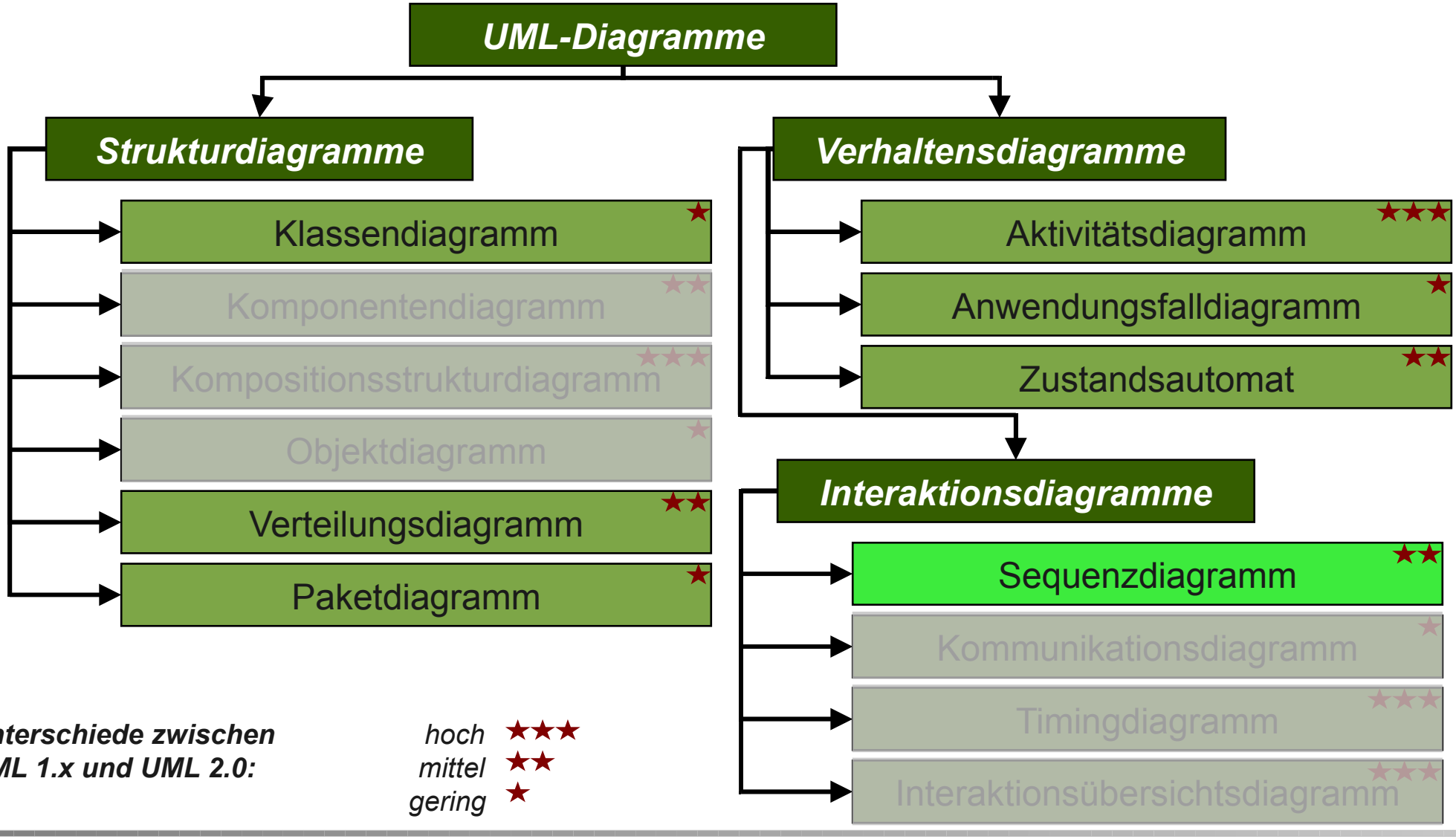

Bankautomat
Kontostand überprüfen
Bankkunde
Authentifizieren
Geld abheben
sm Authentifizieren
Karte eingeführt
[Karte = keine Bankkarte] / Karte ausgeben
[Karte = Bankkarte]
korrekte Bankkarte
entry / Karte prüfen
[Pin = richtig]
Pin eingegeben
Pincode-Eingabe
[Gültigkeit = gültig]
[Gültigkeit = abgelaufen] / Bankkarte einziehen
[Pin = falsch] / #++
[# <3]
[# >= 3] / Bankkarte einziehen

Dynamisches Verhalten der einzelnen Komponenten.

Die Eingabe verursacht einen Zustandsübergang und möglicherweise eine Ausgabe.



UML im Schnelldurchlauf
Sequenzdiagramm



Unterschiede zwischen UML 1.x und UML 2.0:

hoch ★★★
mittel ★★
gering ★

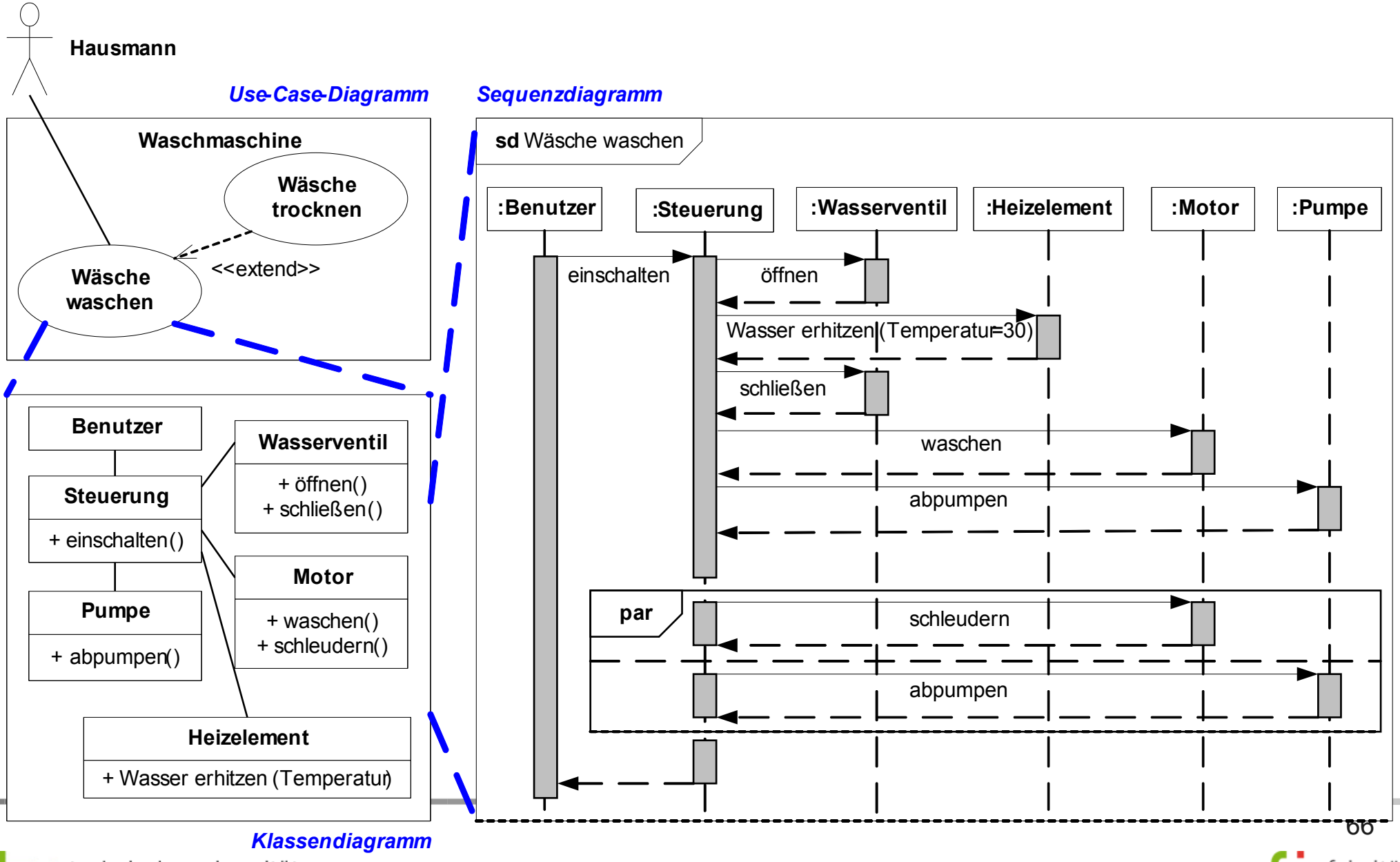
- Diese zentrale Frage beantwortet das Diagramm:
  - Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?
- Diese Stärken hat das Diagramm:
  - stellt detailliert den Informationsaustausch zwischen Kommunikationspartnern dar
  - sehr präzise Darstellung der zeitlichen Abfolge auch mit Nebenläufigkeiten
  - Schachtelung und Flusssteuerung (Bedingungen, Schleifen, Verzweigungen) möglich

- Bedeutung des Diagramms
  - Das Sequenzdiagramm ist das meistverwendeste unter den Interaktionsdiagrammen.
- Gegenstand des Diagramms
  - Es zeigt den zeitlichen Verlauf der Interaktion zwischen mehreren Kommunikationspartnern (2 Dimensionen).

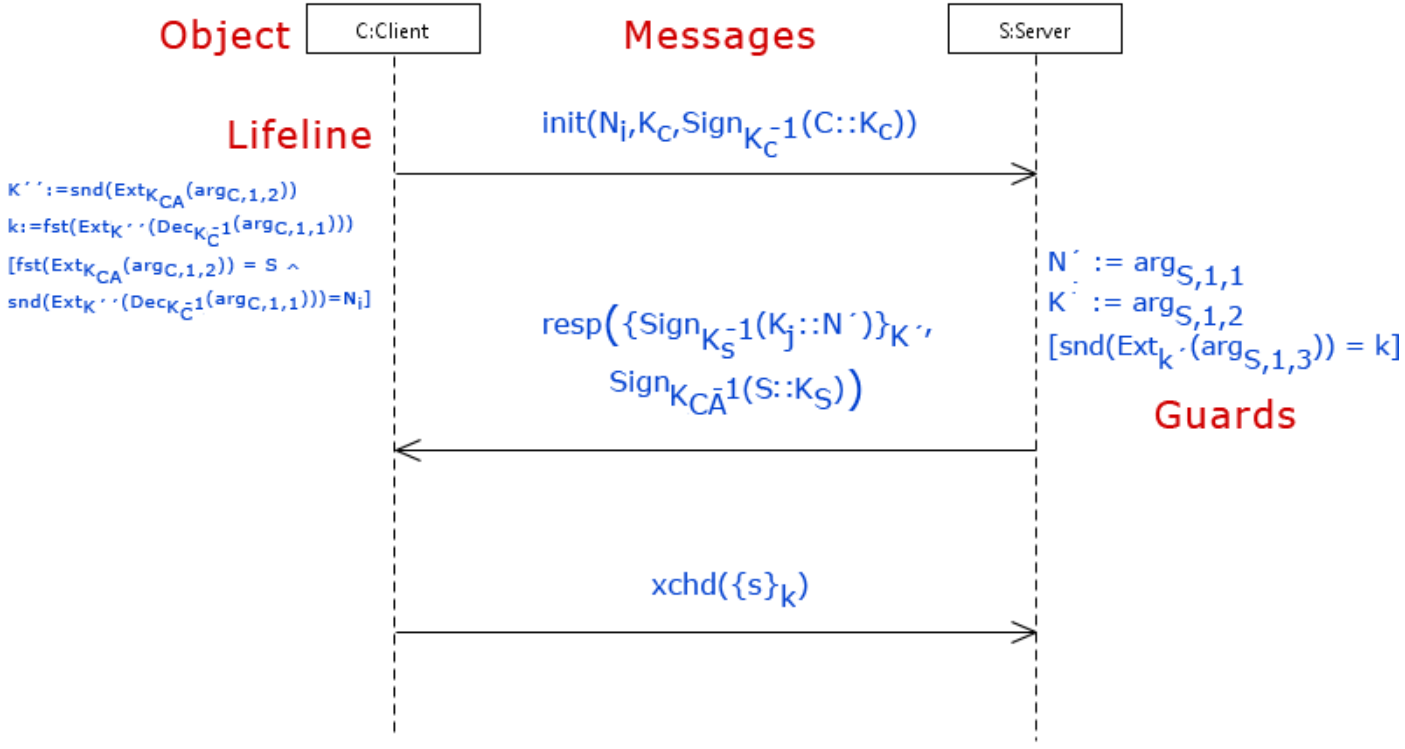
- Häufige Anwendungsfälle
  - Die Abfolge der Nachrichten ist wichtig.
  - Die durch Nachrichten verursachten Zustandsübergänge sind kaum relevant.
  - Die Interaktionen sind kompliziert.
  - Die strukturelle Verbindung zwischen den Kommunikationspartner ist nicht relevant.
  - Es stehen Ablaufdetails im Vordergrund.

- Änderungen gegenüber früheren UML-Versionen
  - ist strukturier- und zerlegbar
  - Sequenzen können damit beliebig ineinander verschachtelt werden.
  - viele Möglichkeiten zur Steuerung von Kontrollflüssen und Nebenläufigkeiten
  - Alle wesentlichen Konstrukte aus den beliebten Message Sequence Charts (MSC) wurden übernommen.

- Besondere Hinweise
  - Es sollten häufig Interaktionsreferenzen genutzt werden, um das Diagramm übersichtlich zu gestalten.
  - Es sollten an kritischen Stellen auch die falschen Abläufe mit den entsprechenden kombinierten Fragmenten modelliert werden.


Hausmann
Use-Case-Diagramm
Sequenzdiagramm
Waschmaschine
Wäsche trocknen
<<extend>>
Wäsche waschen
Benutzer
Steuerung
+ einschalten()
Pumpe
+ abpumpen()
Wasserventil
+ öffnen()
+ schließen()
Motor
+ waschen()
+ schleudern()
Heizelement
+ Wasser erhitzen (Temperatur)
Klassendiagramm
sd Wäsche waschen
:Benutzer
:Steuerung
:Wasserventil
:Heizelement
:Motor
:Pumpe
einschalten
öffnen
Wasser erhitzen (Temperatur=30)
schließen
waschen
abpumpen
par
schleudern
abpumpen

UMLsec Beispiele
Sequenzdiagramm



Verdeutlicht die Interaktion zwischen Objekten und Komponenten per Nachrichtenaustausch.

Zusammenfassung

- Diese Folien beinhalteten:
  - UML & Metaebenen
  - UML Profil
  - UML Diagramme
    - Klassendiagramm
    - Verteilungsdiagramm
    - Paketdiagramm
    - Aktivitätsdiagramm
    - Anwendungsfalldiagramm
    - Zustandsdiagramm
    - Sequenzdiagramm