

Projekt Erdbebenfrühwarnung im WiSe 2010/11



Entwicklung verteilter eingebetteter Systeme

Prof. Dr. Joachim Fischer
Dipl.-Inf. Ingmar Eveslage
Dipl.-Inf. Frank Kühnlenz

fischer|eveslage|kuehnlenz@informatik.hu-berlin.de

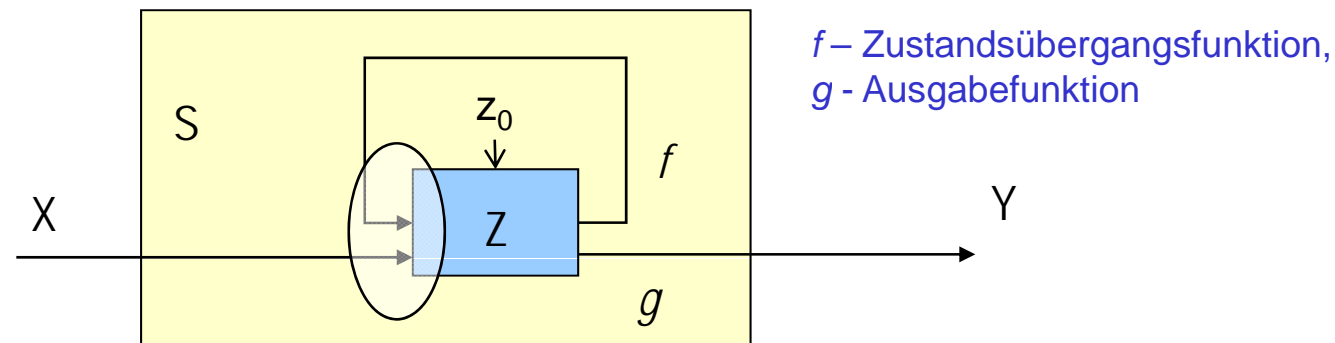
3. Grundlagen der Systemmodellierung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Klassifikation dynamischer Systeme

Zustandsänderungen (Prinzip)

- Sei Z n -dimensionaler Zustandsvektor (Zustand z = Belegung von Z), der Zustandsgrößen eines (Teil-)Systems S zu diesem Zeitpunkt beschreibt
- der (neue) **Zustand** ergibt sich aus dem bisherigen (**aktuellen**) Zustand bei Berücksichtigung von “**Zuwachs**” und “**Reduktion (negativer Zuwachs)**” für die Zustandsgrößen im betrachteten Zeitraum des Zustandswechsels
- ausgehend von einem ausgezeichneten Anfangszustand z_0

Zeit $t_0 \quad t_1 \quad t_2 \quad t_3$
 $z_0 \rightarrow z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \dots$ Zustandsentwicklung in Abhängigkeit der Zeit



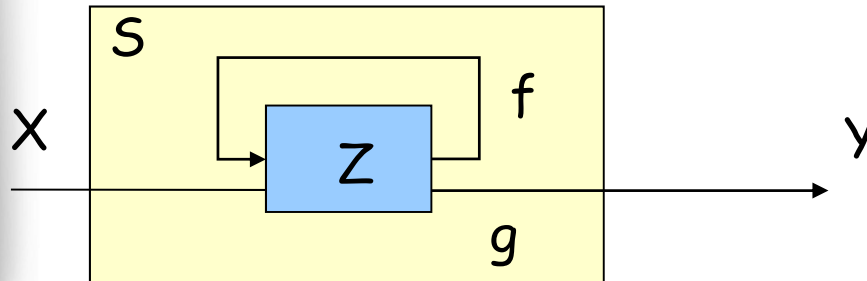
Eingabe X
 Zuwachs
 im Zeitintervall $(t_k, t_{k+1}]$

Änderung des Zustandsvektors Z
 im Zeitintervall $[t_k, t_{k+1}]$
 in Abhängigkeit von $x(t_{k+1}), z(t_k)$

Ausgabe Y
 äußere Reaktion
 des Systems auf die Eingabe
 im Zeitintervall $(t_k, t_{k+1}]$

Allgemeine (Teil-)Systemdefinition

dient mehr der Klassifikation von Verhaltensmodellen

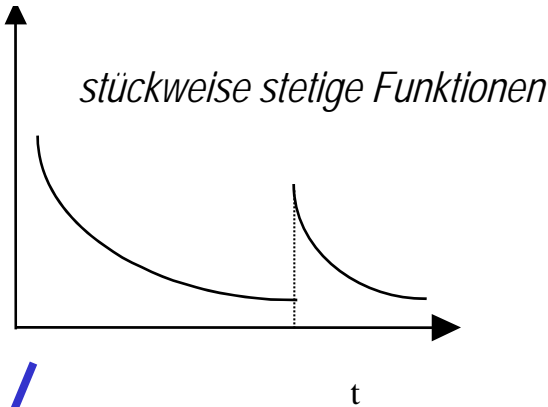


$$S = (Z, z_0, X, Y, f, g, \text{time})$$

- Z Menge der möglichen Zustände
- $z_0 \in Z$ Anfangszustand
- X Menge der möglichen Eingaben
- Y Menge der möglichen Ausgaben
- **time** Zeitbasis als (T, \leq, t_0) mit
 - Menge möglicher Zeitpunkte T ,
 - einer Ordnungsrelation \leq und
 - einem minimalen Element t_0
- f $Z \times X \times T \rightarrow Z$ als Zustandsübergangsfunktion
- g $Z \times X \times T \rightarrow Y$ als Ausgabefunktion

Arten von Zustandsänderungen

zeitkontinuierliche
Zustandsänderung



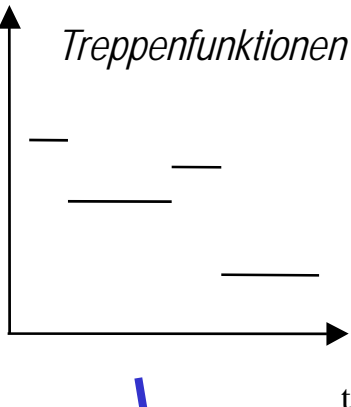
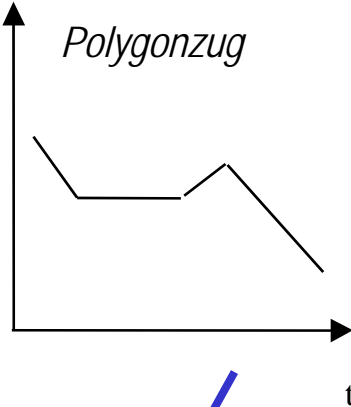
$$z'(t) = f(z(t), x(t), t)$$

mit $z(t) \in Z, x(t) \in X, t \in T$

Differentialgleichungen

numerische
Lösungsverfahren

zeitdiskrete
Zustandsänderung



$$z(t_{n+1}) = f(z(t_n), x(t_{n+1}), t_{n+1})$$

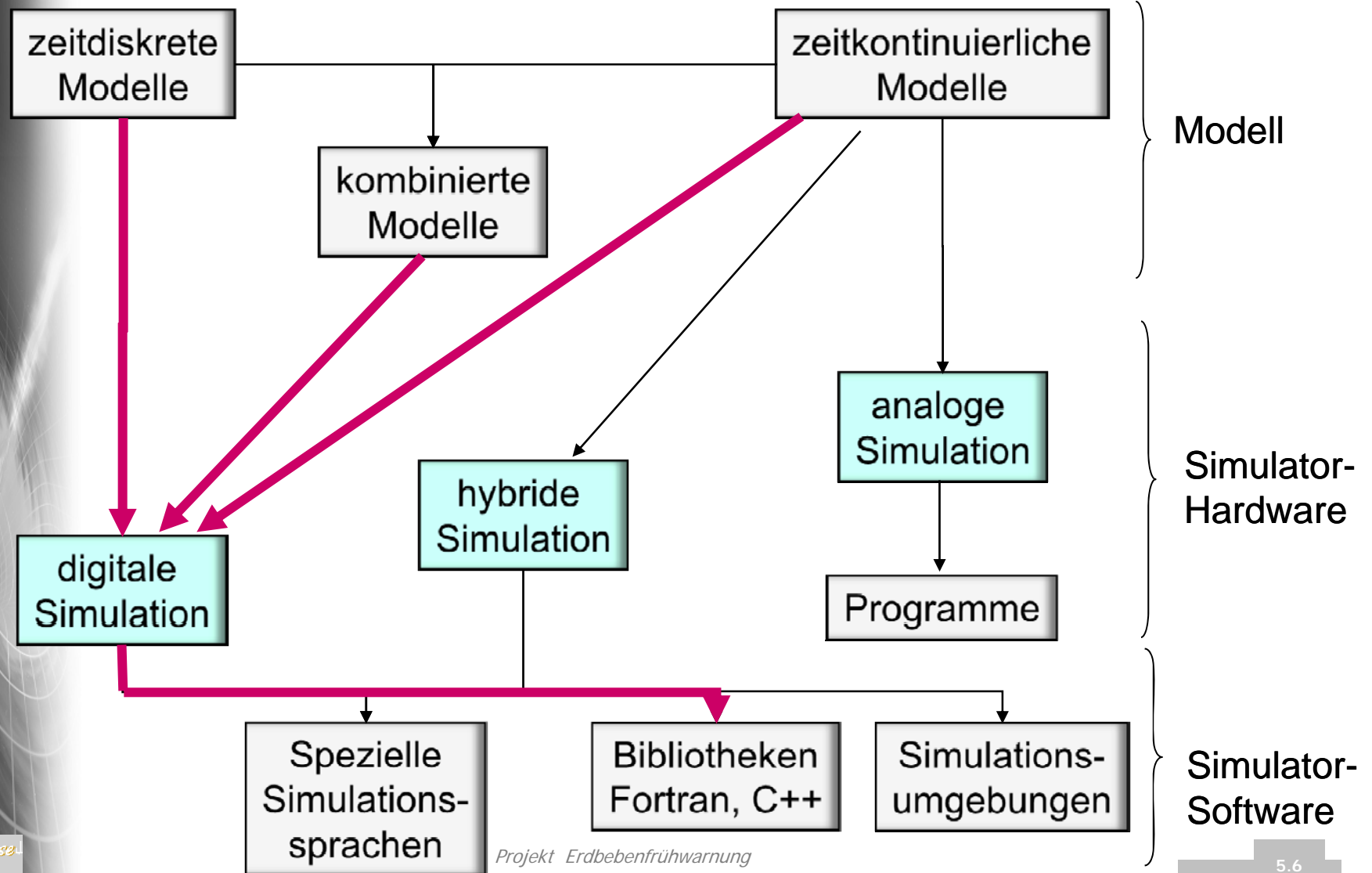
mit $z(t_n) \in Z, x(t_{n+1}) \in X, t_{n+1} \in T$

Differenzgleichungen
zelluläre Automaten

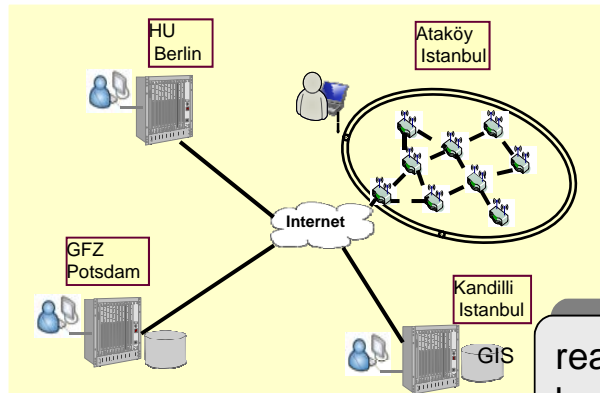
Ereignissimulationen

Prozesssimulation

Klassifizierung von Modellen und Simulationsverfahren



Erdbebenfrühwarnsystem



Sonderfall:
Modell und Original-
Teile
liegen als Software
vor

reales
bzw. gedachtes
Phänomen

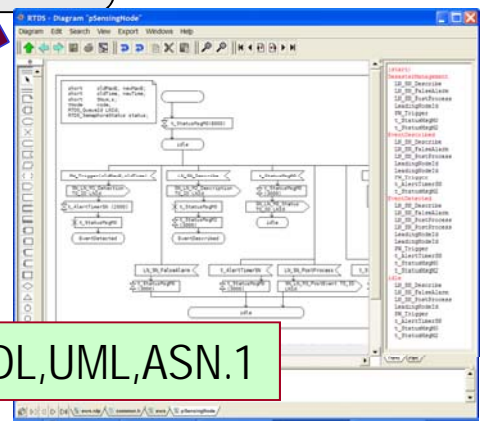
Ausführungs-
Modell im Speicher
Simulator

Ziel

informales
/semiformales
Systemmodell

Experiment-
Management-
System

formales
mathematisches
Simulationsmodell

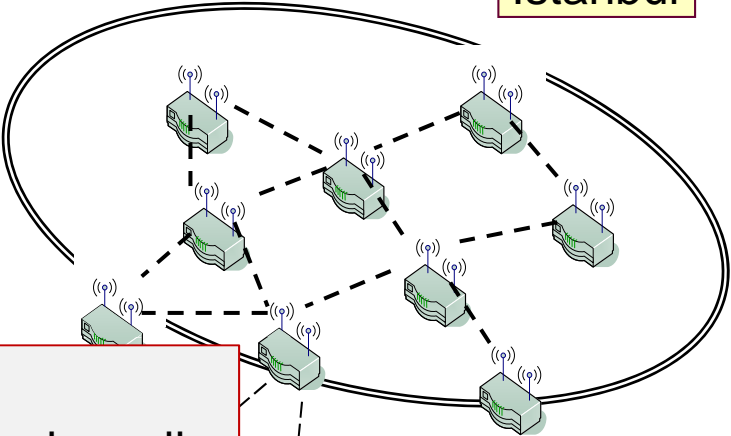


Zielcode: C++, OpenWRT, Boost

Modell: SDL,UML,ASN.1

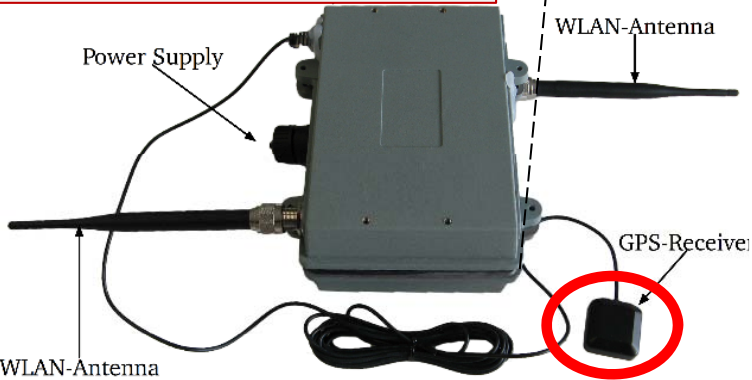
SOSEWIN – Hardware, Software

Ataköy
Istanbul



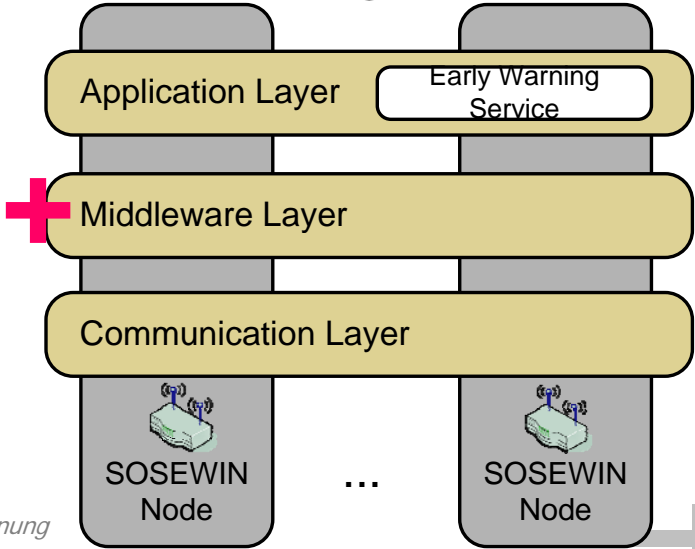
source

- battery with solar cell
- grid

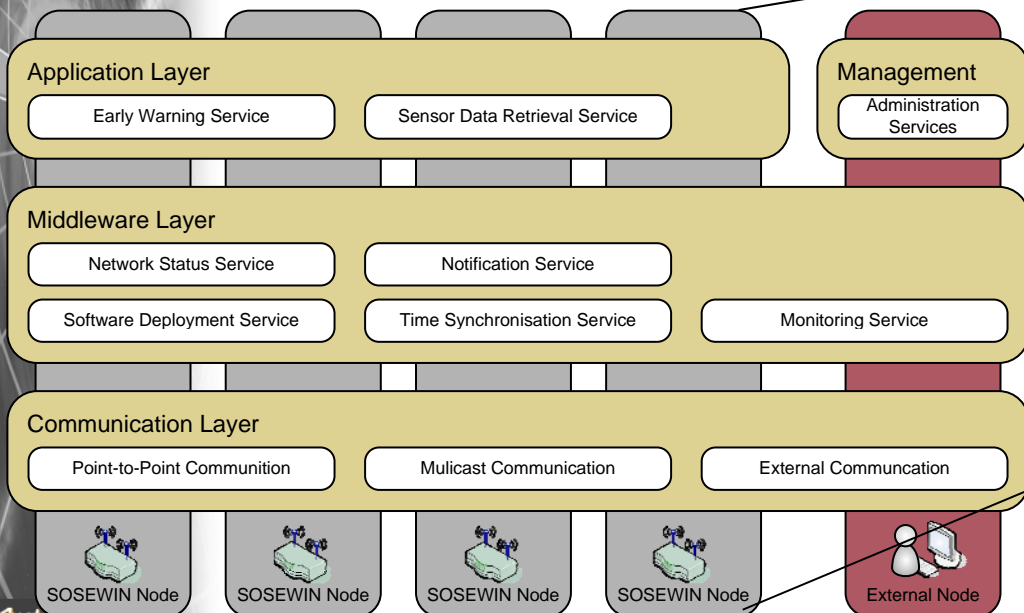
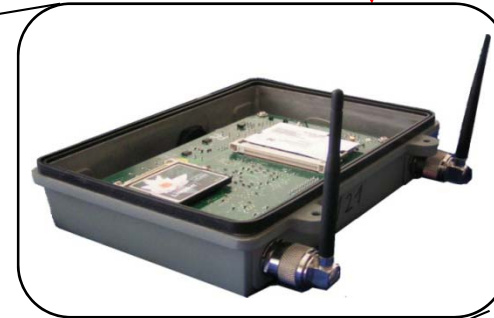
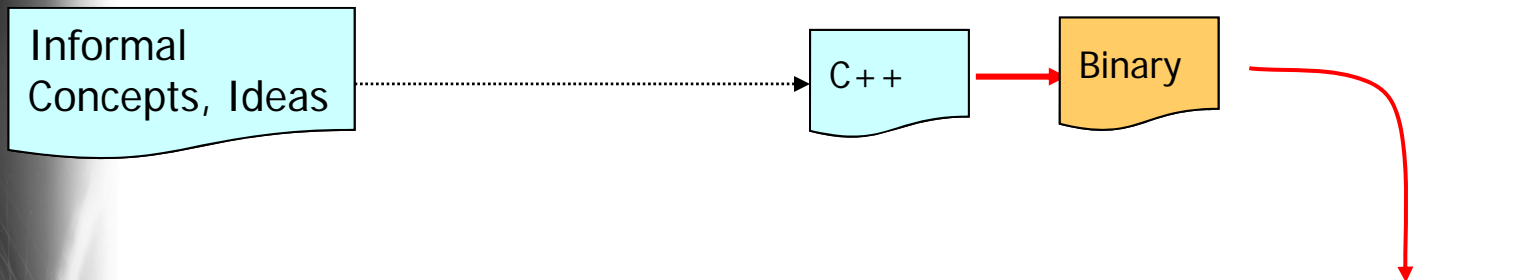


Wrap-Board

- 32-bit Processor (AMD Geode)
- 233 MHz
- 128 MB RAM
- 1024 MB Flash
- 2 x 54 Mb/s Transceiver
- **Linux OS**
(sensor drivers)



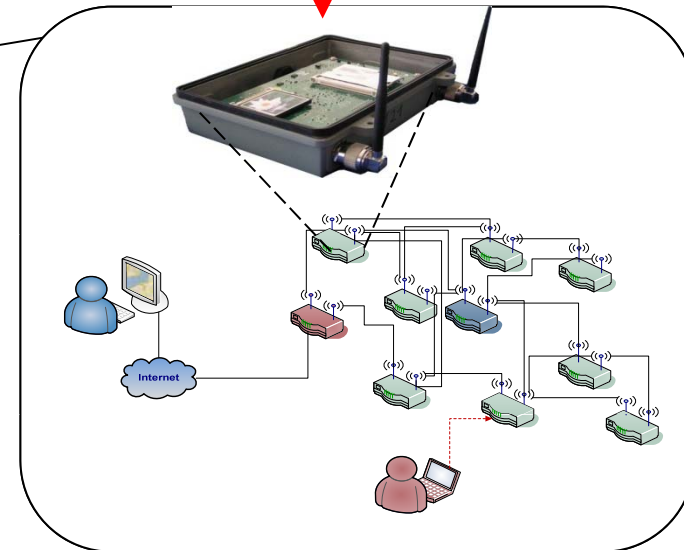
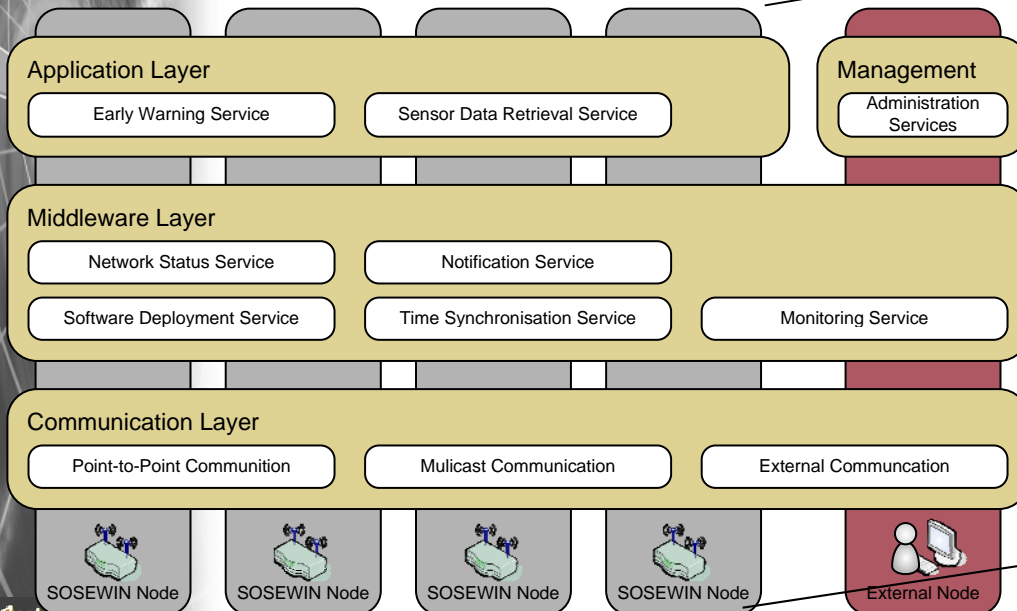
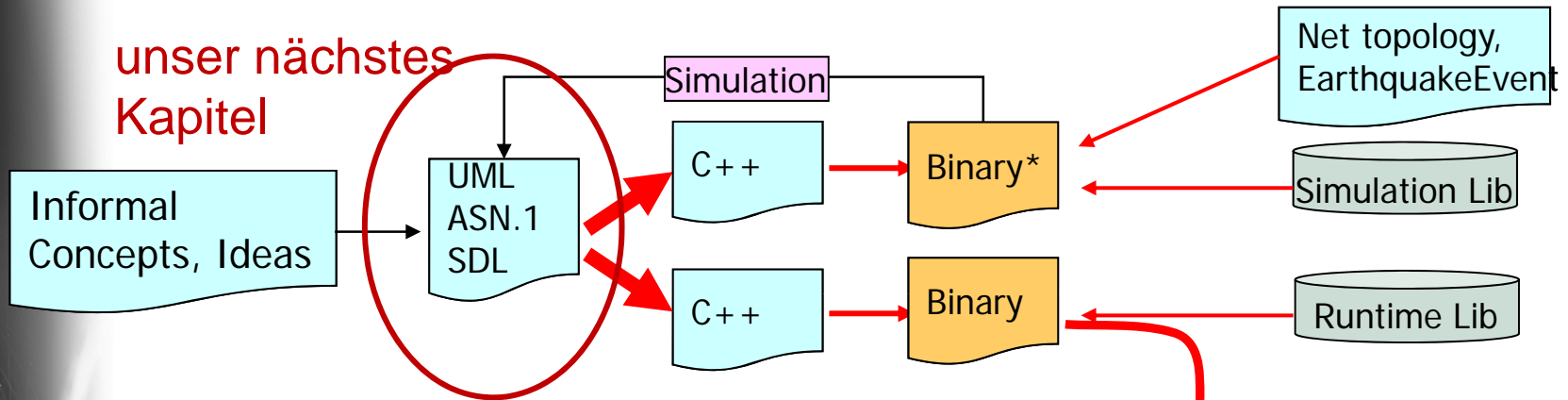
Traditional Software Development



test of a single node is not sufficient and of a complete system problematic

Our Model-Driven Approach

unser nächstes Kapitel

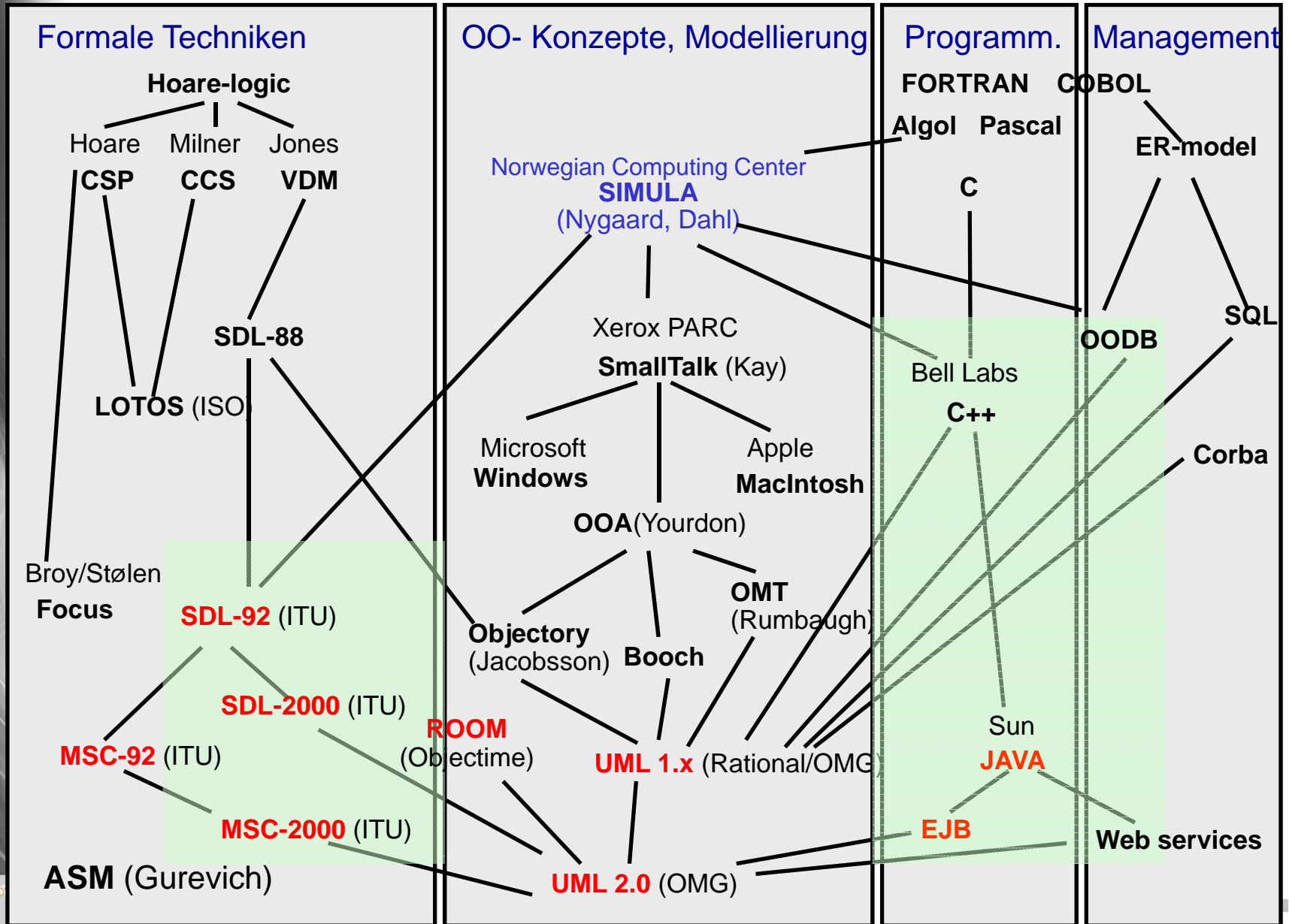


simulation of a complete system simplifies testing

4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards

Die Vorläuferkonzepte von UML



UML-Charakterisierung (Zusammenfassung)

- ist eine Notation/Sprache, keine Methode, Framework für UML-ähnliche Modellierungssprachen
- abstrahiert von
 - architekturellen Vorgaben,
 - Design- und Implementierungs-Styles,
 - Technologien (Software, Hardware, Infrastrukturen, ...),
 - Entwicklungsprozessen
- standardisiert
 - Begriffswelt (Modellierungskonzepte),
 - Semantik (Bedeutung der Modellierungskonzepte),
 - visuelle Darstellung (Notation der Modellierungskonzepte)
- führt Ideen verschiedener Techniken zusammen
 - Booch, OMT, Jacobsson, ROOM, SDL, EDOC, MSC, Component Based Modeling, ...

Nutzen/Ziele des UML-Einsatzes

- Blueprint für die Softwareentwicklung
- hilft Nutzern, Entwicklern und Kunden bei der Kommunikation
 - Anforderungen, Designvorgaben, Deployment-Constraints etc. mittels UML-Modellen festlegbar
- bietet Konzepte zur nachvollziehbaren Modellierung („Traceability“)
 - vom initialen Geschäftsmodell bis zum einsetzbaren System
- zentrale Bibliothek für Wissen und Erfahrungen in Form von Modellen (Wiederverwendbarkeit)
- Verbesserung der Softwarequalität
- langfristige Kostenreduzierung
- Flexibilität für schnelle technische und geschäftliche Änderungen aufgrund der Anwendbarkeit im gesamten Entwicklungsprozess
- große und komplexe Softwaresysteme erfordern sorgfältiges Design

Die UML



„Wenn die Sprache nicht stimmt,
ist das was gesagt wird, nicht das, was gemeint ist.“ (Konfuzius)

- UML = Unified Modeling Language
- ... ist zunächst Standardsprache (der OMG) zur Visualisierung, Spezifikation, Konstruktion und Dokumentation komplexer Softwaresysteme
- ... kombiniert Konzepte der
 - Objektorientierten Modellierung
 - Datenmodellierung (Entity-Relationship-Diagramme)
 - Business-Modellierung (Work Flows)
 - Komponentenmodellierung
 - Verhaltensmodellierung (Erweiterte Zustandsautomaten)
 - ...
- UML-Modelle sind in erster Linie graphische Repräsentationen in Form von Diagrammen

4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards

UML-Modell

- Definition ...aus dem Standard

A **model** captures a view of a physical system.

It is an abstraction of the physical system, with a certain purpose.

This purpose determines what is to be included in the model and what is irrelevant.

Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail.

Modelliere nicht so detailliert wie möglich,
sondern
so detailliert wie nötig !

UML-Grundkonzepte

- UML-Modellbausteine
 - Dinge/Entitäten (im Sinne von Abstraktionen)
 - Beziehungen zwischen den Entitäten
 - Diagramme zur Anordnung von Entitäten
- Regeln zur Komposition der Modellbausteine (später)
- universelle Mechanismen, gültig für gesamte Sprache (später)

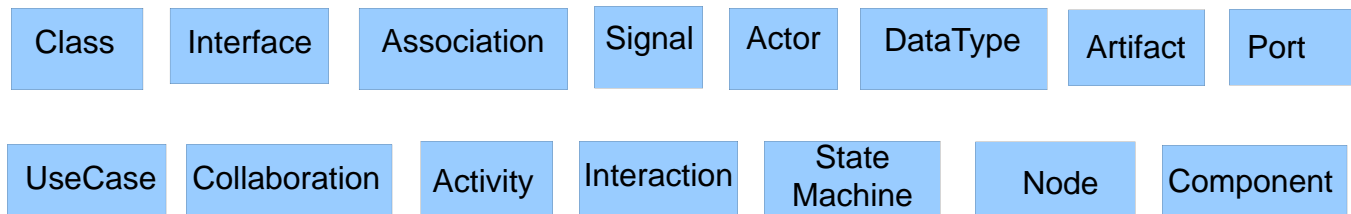
Vier Arten von Modellierungselementen/Entitäten

nämlich zur

- Bildung von Strukturen
 - Beschreibung von Verhalten
 - Bildung von Gruppierungen von Entitäten
 - Formulierung von Anmerkungen
- damit lassen sich UML-Modelle komplett beschreiben

1. Strukturelle Entitäten: Überblick

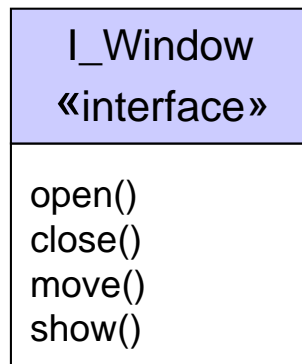
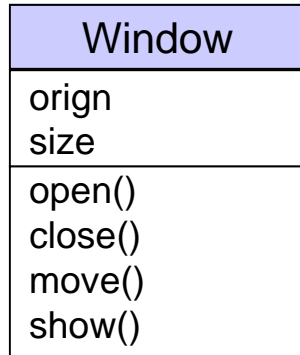
- **Substantive** in Modellen (formuliert in UML)
- bilden die Struktur eines Modells (meist statisch, aber auch dynamisch)
- Zusammenfassung als *Classifier*



z.T. mit weiteren Spezialisierungen

dafür einige Beispiele (Syntax, Semantik) ...

Strukturelle Entitäten: Klasse, Interface

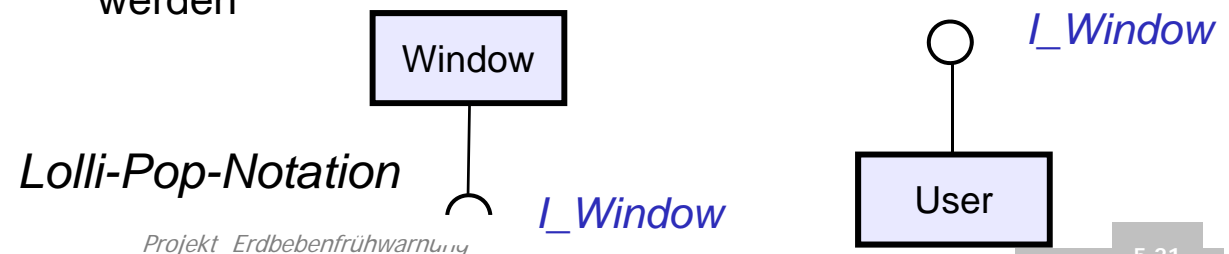


Klasse (Class)

- beschreibt Abstraktion einer Menge von Objekten
- mit gleichen Attributen, Operationen, Beziehungen, Semantik
- dargestellt als Rechteck mit Unterteilungen (Compartments)
- besitzt mindestens ein Interface

Interface (Interface)

- beschreibt Sammlung von Operationen, die durch Dienste einer Klasse oder Komponente festgelegt sind
- beschreibt mit Signaturen potentiell äußeres Verhalten einer Klasse/Komponente komplett oder partiell
- keine Implementation !!!
- Bezug zur Klasse/Komponente muss explizit dargestellt werden



Strukturelle Entitäten: Kollaboration, Anwendungsfall

Verarbeitungskette

*..in Kombination mit
Verhaltensbeschreibungen*

Anwendungsfall

*..in Kombination mit
Verhaltensbeschreibungen*

Kollaboration (Collaboration)

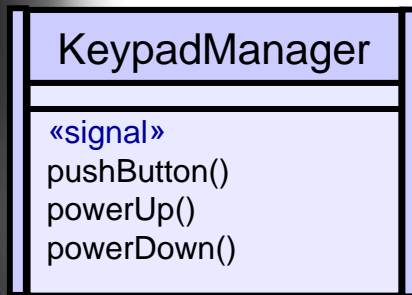
- beschreibt das Zusammenwirken von Objekten/Klassen in spezifischen Rollen als kooperatives Verhalten
- geht über Verhaltensdarstellung einer einzelnen Entität hinaus
- weist auch strukturelle Merkmale auf
- eine Entität kann an verschiedenen Kollaborationen teilhaben
- stellt eine Implementierung eines Verhaltensmusters dar

Anwendungsfall (Use Case)

- beschreibt sichtbare/messbare Handlungsfolgen eines Systems für bestimmte externe Systemnutzer (Akteure)
- dient der strukturellen Beschreibung von Einsatzfällen/Anwendungen, die über Kollaborationen realisiert werden

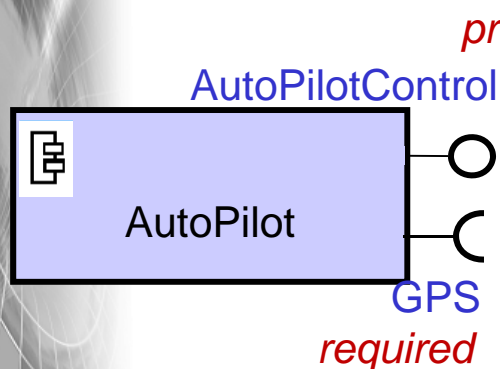
Strukturelle Entitäten: *aktive Klasse, Komponente, Artefakt, Knoten*

~ *ähnlich zum Konzept Klasse*



Aktive Klasse (active Class)

- beschreibt spezifische Klassen, deren Objekte selbst Auslöser von Aktionen sind
- Objekte können untereinander synchron oder asynchron Signale austauschen
- erlaubte Empfangssignale lassen sich spezifizieren
- geeignete Verhaltensbeschreibung: Zustandsautomat



Komponente (Component)

- modularer Teil eines Systems, dessen Implementierung sich hinter einer Menge von externen Interfaces verbirgt
- Systemkomponenten mit gleichen Interfaces sind austauschbar (Problem besteht dennoch mit Verhaltenskompatibilität !!!)
- Komponente ist die Zusammenfassung aller seiner beschreibenden und ausführbaren Artefakte

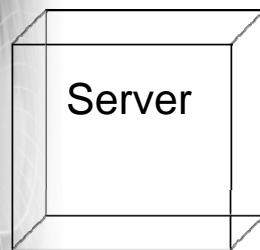
Strukturelle Entitäten: *aktive Klasse, Komponente, Artefakt, Knoten*

~ ähnlich zum Konzept Klasse



Artefakt (Artifact)

- physische Informationseinheit in Bit-Struktur: Modell, Quellcode, Script, **ausführbarer Binärcode**, Tabellen einer rel. Datenbank, Textdokument, e-Mail, ...)
- werden im Entwicklungsprozess angelegt oder aber zur Laufzeit des Systems erzeugt oder konsumiert



Knoten (Node)

- physisches Element, das eine Rechnerressource darstellt (mit Prozessor, Speicher):
- dienen der Aufnahme und Ausführung von Komponenten

Klassenähnliche Entitäten

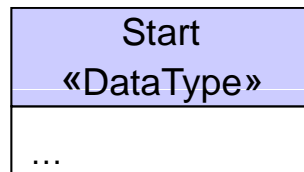
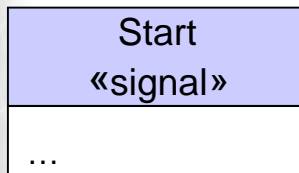
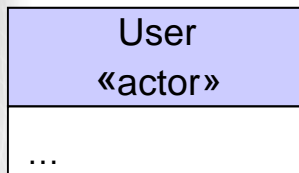
- Interface, aktive Klasse, Komponente, Artefakt, Knoten
Rechteck mit charakteristischem Icon
- allgemeine Form der spezifischen Einschränkung einer UML-Entität:

Stereotype

Grundsymbol:= Klasse (als Rechteck)
Name der Stereotype-Entität
Name der klassenähnlichen Stereotype: «name»



- weitere Stereotype



Prozess
Thread

Anwendung
Dokument
Datei

Bibliothek
Seite
Tabelle

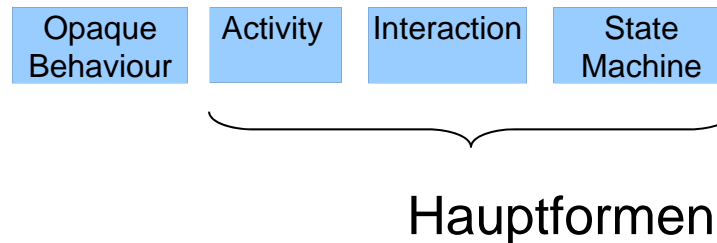
Gerät
Ausführungs-
umgebung

...weitere

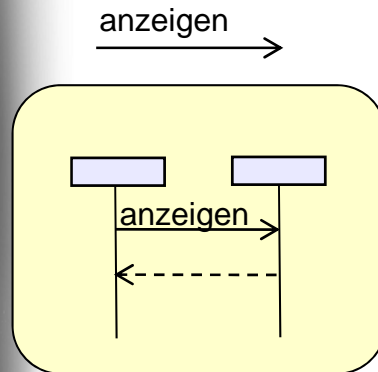
Projekt Erdbebentrühwarnung

2. Verhaltensorientierte Entitäten: Überblick

- **Verben** in Modellen (formuliert in UML), z.B. Member-Funktionen
- bestimmen das Verhalten eines Modells in Raum und Zeit
- Zusammenfassung als abstraktes Konzept *Behaviour*

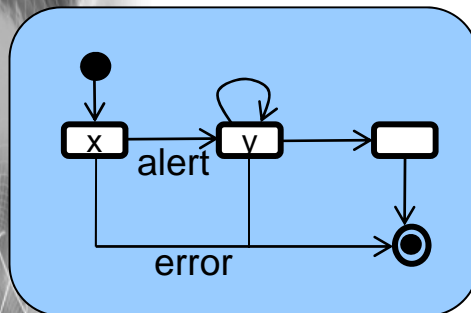


Verhaltens-Entitäten: Interaktion, Zustandsautomat



Interaktion (Interaction)

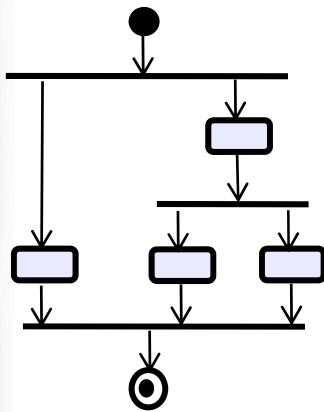
- beschreibt gerichtete Interaktion zwischen Objekten oder Rollen eines best. Kontextes zur Erbringung einer best. Funktionalität
- besteht aus Nachrichten, Aktionen und Konnektoren
Pfeilart und Linienausführung bewirken unterschiedliche Semantiken



Zustandsautomat (State Machine, State-Chart)

- beschreibt Abfolge der Zustände, die ein Objekt oder eine Interaktion während seiner Existenz als Reaktion auf Ereignisse durchläuft, wobei Ereignisreaktion ebenfalls dargestellt wird
- gestatten die Beschreibung von Verhalten von Klassen oder Kollaborationen
- Zustandsautomaten bestehen aus weiteren Entitäten: Zustände (inkl. Start, Stop), Übergänge, Ereignisse

Verhaltens-Entität: Aktivität



Synchronisation

Aktivität (Activity)

- beschreibt gerichtete Abfolge von Schritten eines Verarbeitungsvorganges
- elementare Schritte einer Aktivität heißen Aktionen (Actions)
- bei der Interaktion liegt der Schwerpunkt auf einer Reihe von Objekten, die interagieren
beim Zustandsautomaten beim Lebenslauf einer einzelnen Entität !!!
- bei der Interaktion liegt der Schwerpunkt bei den Übergängen von Schritt zu Schritt
ohne zu beachten, welches Objekt den jeweiligen Schritt realisiert

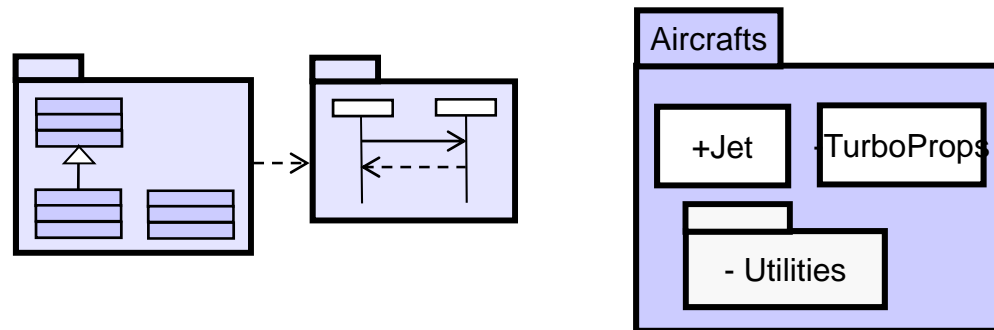
3. Gruppierungs-Entitäten: Überblick

Organisatorische Einheiten von UML-Modellen:

Paket (Package),

aber auch Paket-Variationen

(Rahmenwerken, Modellen, Subsystemen, System)



- Zusammenfassung von Struktur-, Verhaltens-, Anmerkungs- und anderen Gruppierungs-Entitäten ist möglich
- es gibt Regeln zur Abhängigkeit und zur Nutzung von Paketen
- Gruppierungs-Entitäten sind rein konzeptueller Natur
existieren nicht zur Laufzeit

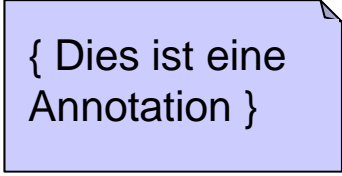
4. Anmerkungs-Entitäten: Überblick

beschreibende Anteile eines UML-Modells



Dies ist ein
Kommentar

beliebiger Kommentar



{ Dies ist eine
Annotation }

Informale oder formale Textbeschreibung
von Modelleinschränkungen und –zusätzen
(Constraints)

UML-Standard schließt **OC**L ein
(Object Constraint Language)

- können beliebigen Modellelementen zugeordnet werden
- Achtung: Anmerkungen werden von Tools auch über einblendbare Textboxen realisiert

Zwischen-Fazit

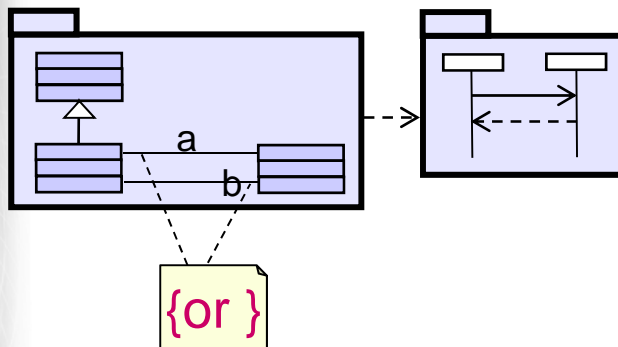
1

Konfuzius: „Wenn die Sprache nicht stimmt, ist das was gesagt wird, nicht das, was gemeint ist.“

2

UML-Modellbausteine (als Übersicht)

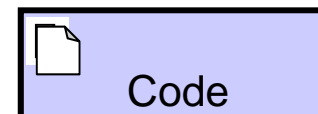
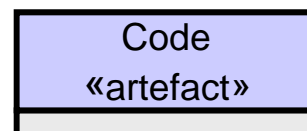
- Dinge/Entitäten (im Sinne von Abstraktionen) ✓
- Beziehungen zwischen den Entitäten
- Diagramme zur Anordnung von Entitäten



- Bildung von Strukturen
- Beschreibung von Verhalten
- Bildung von Gruppierungen von Entitäten
- Formulierung von Anmerkungen/ Einschränkungen

3

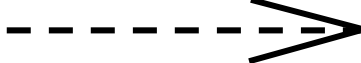




Dualität: Stereotype-Modellelemente \leftrightarrow Icons



4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. Repräsentation von UML
4. Diagrammarten
5. Diagrammrepräsentationen in UML
6. Struktur des UML-Standards
7. Beispiel: Klassendiagramm

Fünf Arten von Beziehungen in UML

- Abhängigkeit (Dependency)  weitere Spezialfälle
 - Assoziation (Association)  weitere Spezialfälle
 - Generalisierung (Generalisation) 
 - Realisierung (Realisation) 
- repräsentieren die grundlegenden Beziehungsbausteine
- weitere Spezialfälle von Assoziationen und Abhängigkeiten
- Erweiterung (Extension) 
wird für den Profilmechanismus benötigt

Spezialfälle der Assoziation

- **Aggregation: strukturelle** Beziehung
zwischen einem Ganzen und seinen Teilen

Aggregation (Aggregation)
keine Beziehung zwischen
der Lebensdauer von Teil
und Ganzem



besitzt Richtung
Teil-Entität kann gleichzeitig oder
nacheinander Teil verschiedener
Ganzer-Entitäten sein

Assoziation mit Raute als Zusatz

stärkste Beziehung=
Komposition (Composition)
besteht über die Lebensdauer
(Teil und Ganzes haben i.d.R.
gleiche Lebensdauer)



besitzt Richtung
Teil-Entität kann nur Teil von einem
Ganzen sein

Unterscheidung bei der Behandlung von
statisch existenten und
dynamisch erzeugten Teilen

Beziehungen: Generalisierung, Realisierung

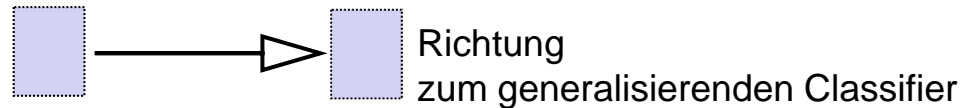
- **Generalisierung** (Generalisation)

Strukturelle Beziehung zwischen Classifier-Entitäten,

für diese
sind Spezialisierung bzw. Generalisierung erlaubt
(z.B. Klasse, Signal, Zustandsautomat, ...)

die spezialisierte Entität übernimmt Struktur und Verhalten der
generalisierenden Entität

Generalisierung



- **Realisierung** (Realisation)

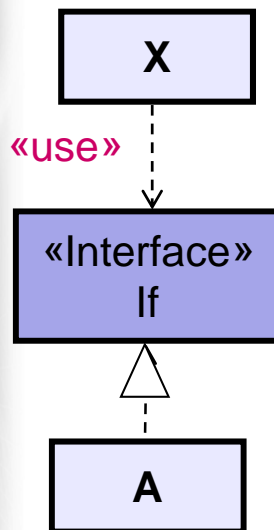
Semantische Beziehung zwischen Classifier-Entitäten,

Realisierung



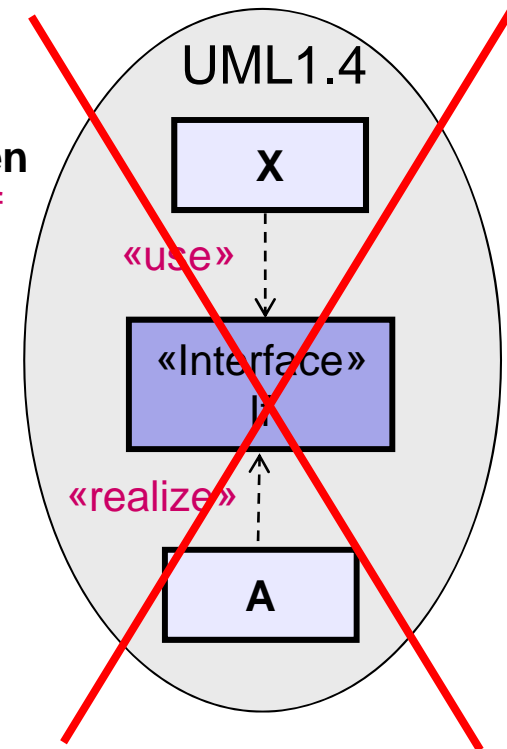
Realisierung als spezielle semantische Abhängigkeit

- Beispiel von Realisierung und Abhängigkeit (Alternative zur Lolli-Pop-Notation)



Objekte irgendeiner Klasse **X** **benutzen** Funktionalität, die über das Interface **If** zur Verfügung gestellt wird

Klasse **A** **realisiert** Interface **If**



- dennoch breites Spektrum von Abhängigkeitsvariationen in UML 2.0

Typ	Variation	Schlüsselwort /Symbol - - ->
Abstraktion (abstraction)	Ableitung (derivation)	«derive»
	Festlegung (manifestation)	«manifest»
	Verfeinerung (refinement)	«refine»
	Trace-Abhängigkeit	«trace»
Bindung (binding)	-	«bind» (parameter _{list,})
Deployment (depolyment)	-	«deploy» oder explizite Verschachtelung
UseCase-Include	-	«include»
UseCase-Erweiterung	-	«extend» (extension pont _{list,})
Paketimport	privat (private)	«access»
	öffentlich (public)	«import»
Paketverschmelzung	-	«merge»
Erlaubnis (permission)	-	«permit»
Informationsfluss	-	«flow»
Ersetzung (substitution)	-	«substitute»
Benutzung (usage)	Benutzung (allgemein)	«use»
	Ruf (call)	«call»
	Erzeugung (creation)	«create»
	Instanziierung	«instantiate»
	Verantwortung (responsibility)	«responsibility»
	Senden (send)	«send»

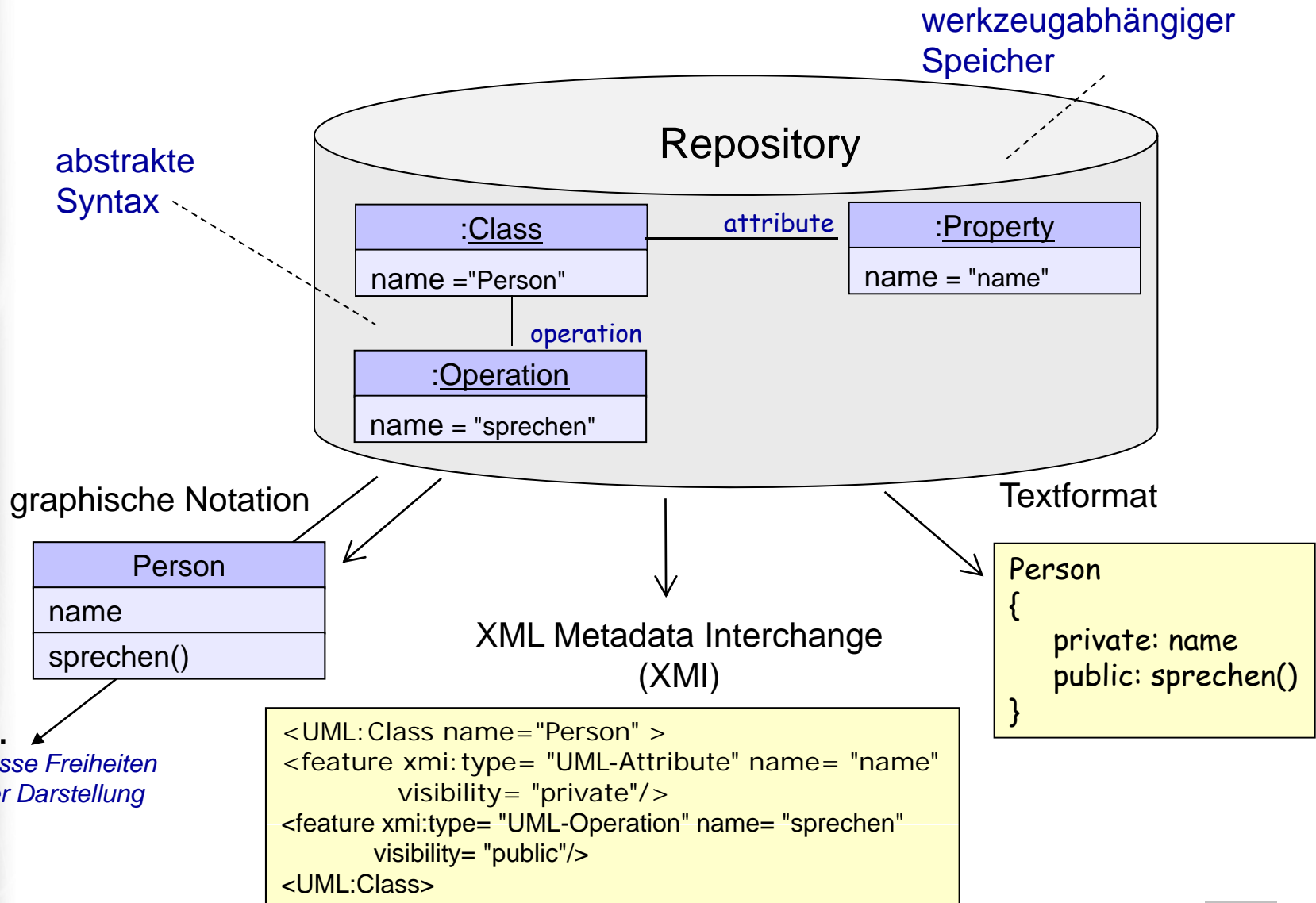
4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. Repräsentation von UML-Modellelementen
4. Diagrammarten
5. Diagrammrepräsentationen in UML
6. Struktur des UML-Standards
7. Beispiel: Klassendiagramm

Repräsentation von UML-Elementen

- ... erfolgt
mittels grafischer Symbole (z.B. Rechteck für eine Klasse)
- aber ohne fest genormte Darstellung,
sie müssen lediglich den Regeln des UML-Metamodells
(abstrakte Grammatik) folgen
- alternative Repräsentationen zur graphischen Notation:
 - XML-konforme Strukturen
 - Codedateien
 - abstrakter Syntaxbaum des UML-Metamodells

Illustration: UML-Modell- Repräsentation



Wo
stehen wir ?

UML-Modellbausteine

- Dinge/Entitäten (im Sinne von Abstraktionen)
✓
- Beziehungen zwischen den Entitäten
✓
- Diagramme zur Anordnung von Entitäten

2. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. UML-Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards
6. Beispiel: UML-Klassendiagramm

Diagrammsprachen der UML 2.0

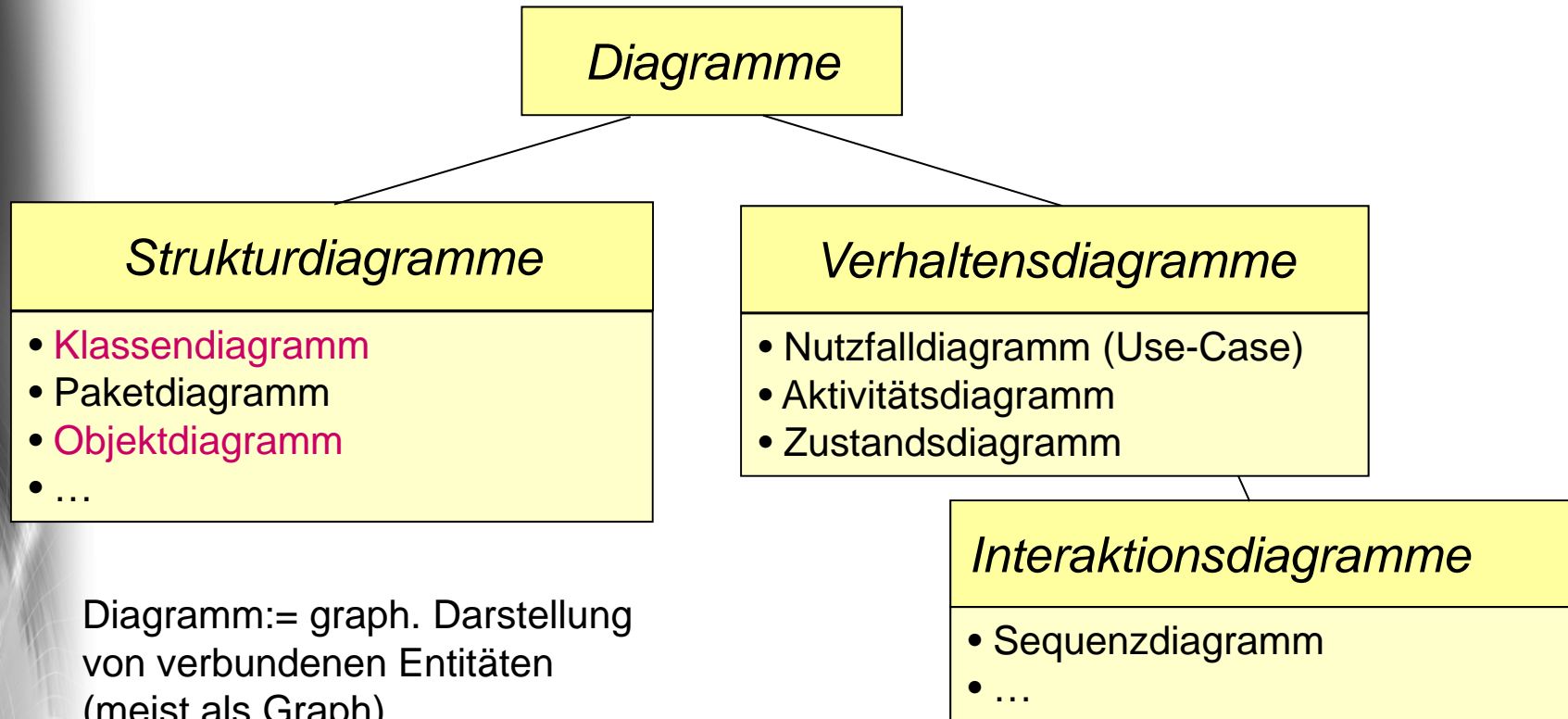
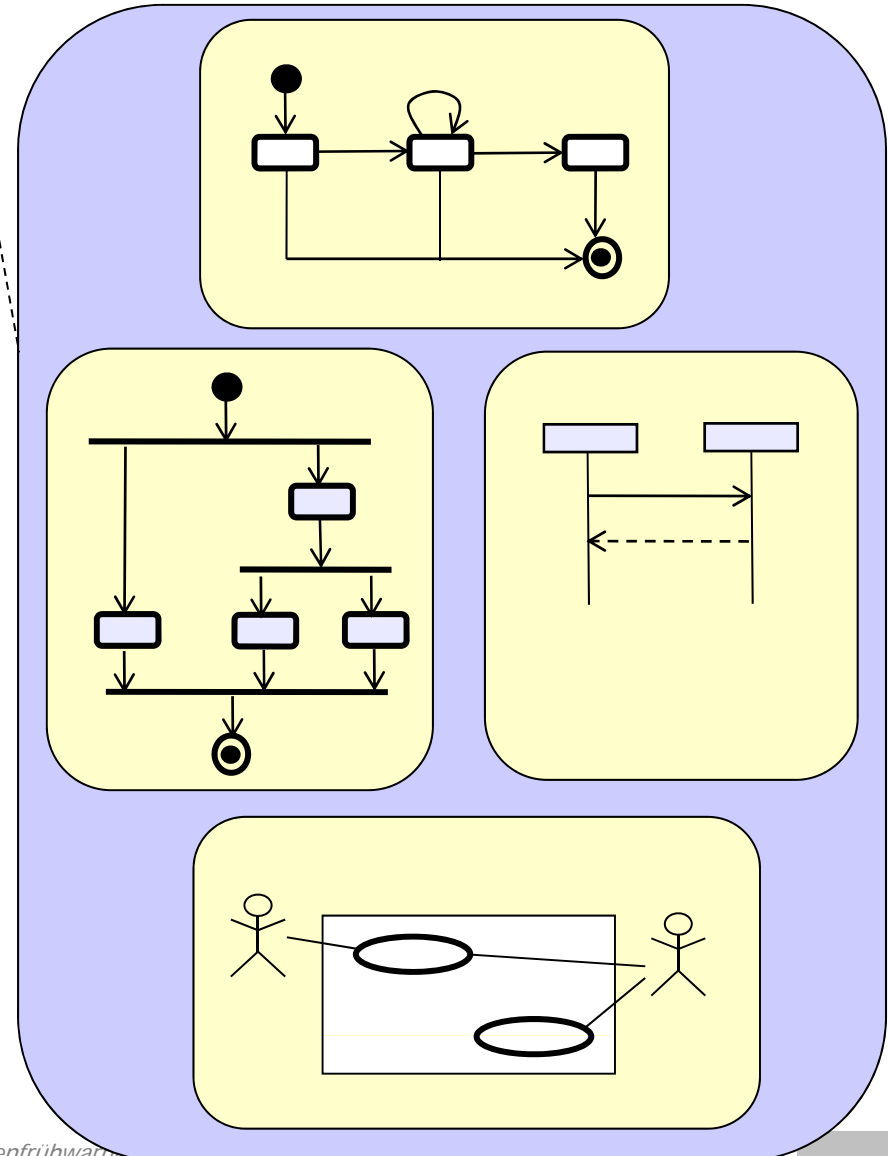
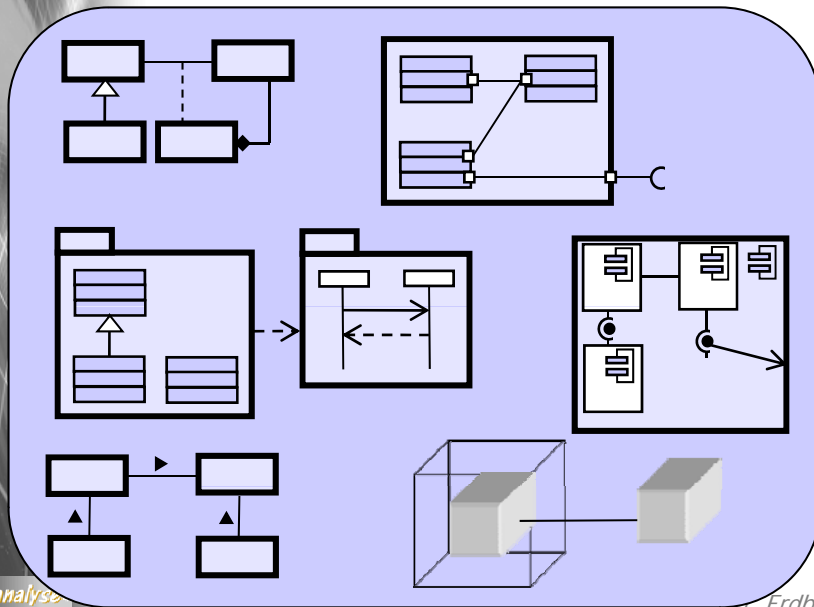


Diagramm:= graph. Darstellung von verbundenen Entitäten (meist als Graph)

Ziel: Modellensemble eines Systems aus verschiedenen Blickwinkeln in separaten Diagrammen

typisch: mehrfacher Bezug auf bestimmte Modellentitäten

Diagrammarten zur Anordnung von Entitäten: **Struktur und Verhalten**



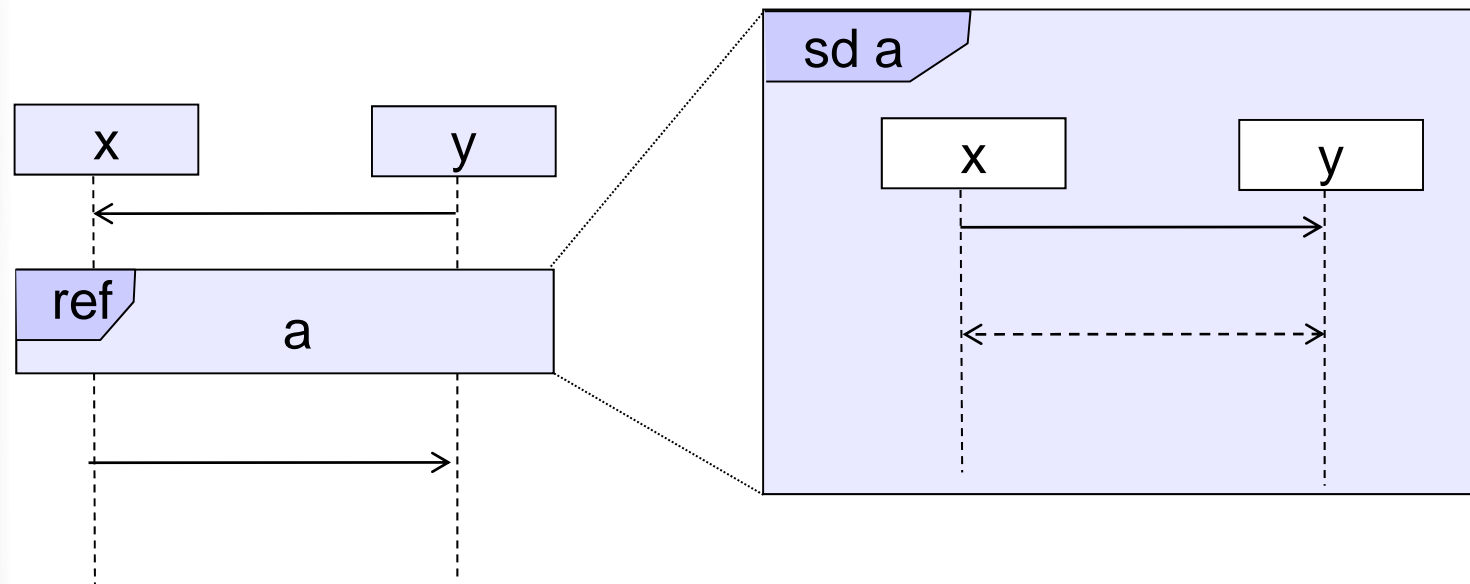
4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. UML-Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards
6. Beispiel: UML-Klassendiagramm

Diagramm, Diagrammtyp

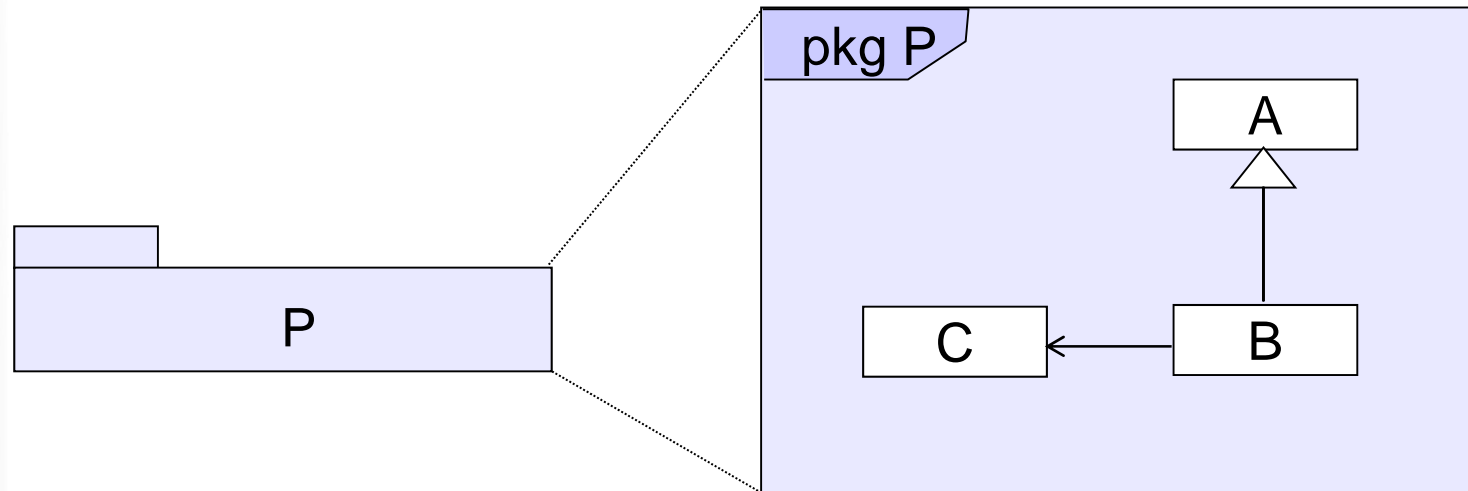
- bestimmte Modellelemente können geeignet in Kombination angewendet werden (andere schließen sich aus)
- Elemente vordefinierter **Diagrammtypen**:
 - **Klassendiagramm**:
Klassen, Attribute, Operationen, Assoziationen
 - **Zustandsdiagramm**:
Zustände, Transitionen, Ereignisse
→ aber: es nicht exakt festgelegt,
welche Elemente zulässig sind
- Definition nutzereigener Diagrammtypen ist möglich
- Anwendungen:
 - i. allg. mehrere **Diagramme** eines Diagrammtyps

Auslagerung von Diagrammteilen



Auslagerung und
mögliche Mehrfacheinbindung von Rahmen

Auslagerung von Diagrammteilen (2)



Paketinhalt: Klassendiagramm

Referenzsymbol

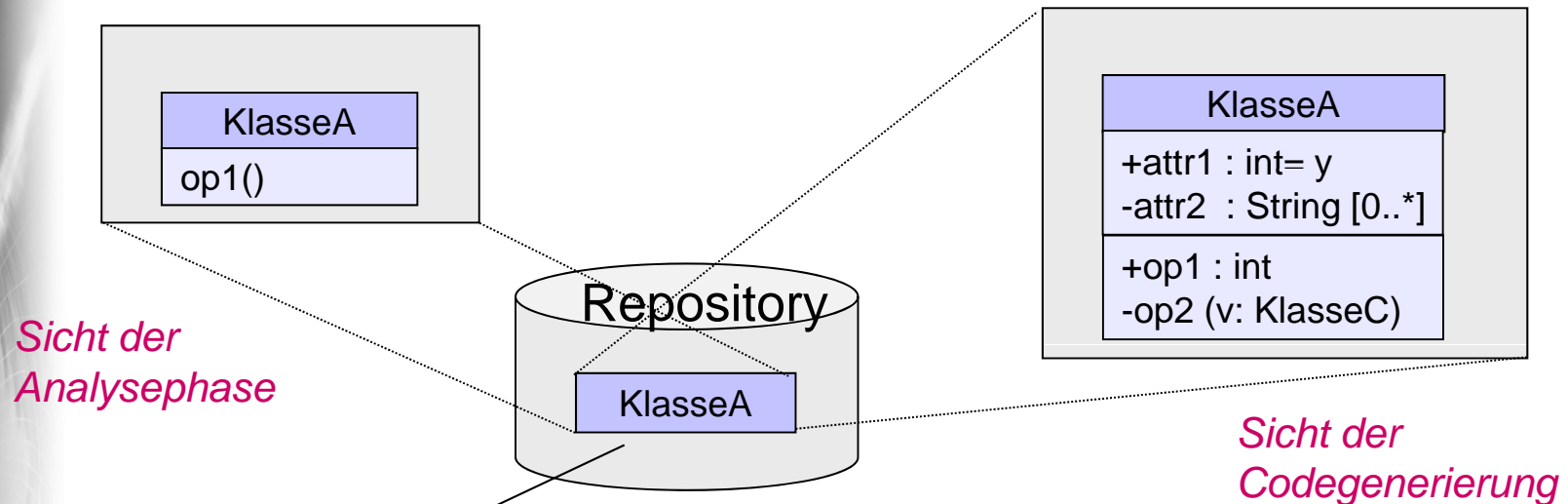
Deklaration/
Referenz eines
Modellelementes

definierendes Diagramm

Definition eines
Modellelementes

Sichten

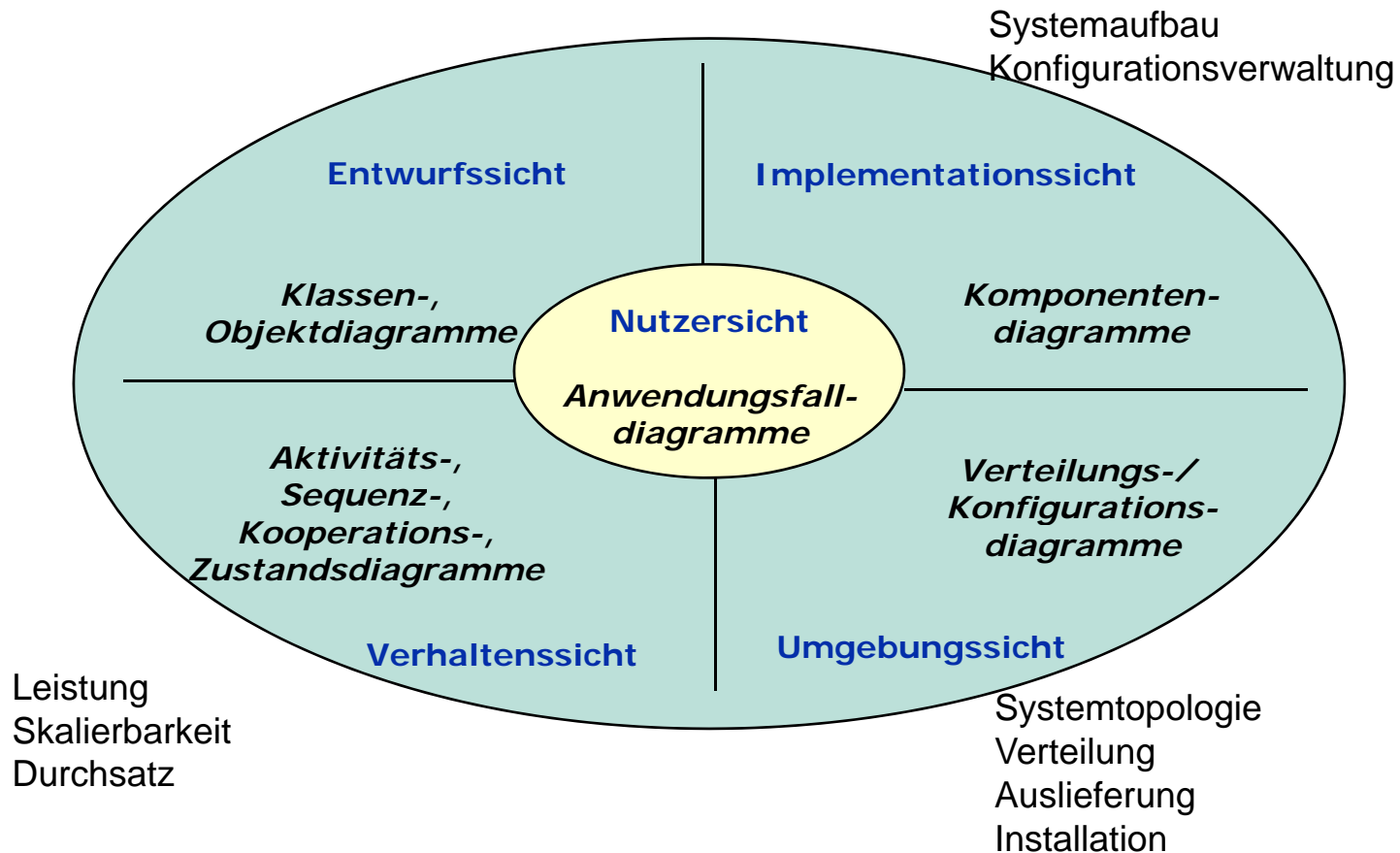
- ein und dasselbe Modellelement kann unterschiedlich detailliert für unterschiedliche Sichten dargestellt werden



Klasse **A** ist in einem Namensraum nur einmal im Repository definiert

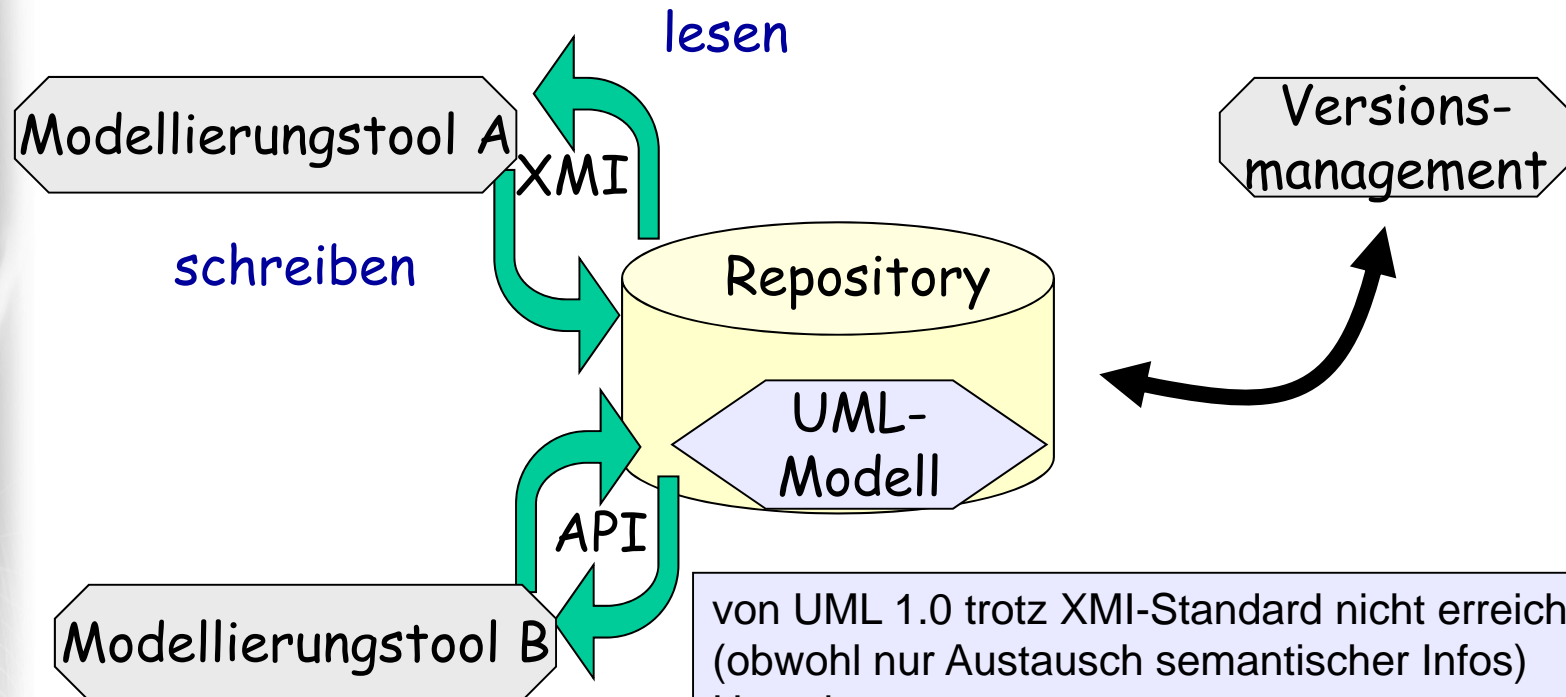
Systemarchitekturmodell eines Systems

- ... unter Verwendung von UML



Austausch von Diagrammen

- uraltes Problem der Informatik: Dokumentenaustausch



von UML 1.0 trotz XMI-Standard nicht erreicht (obwohl nur Austausch semantischer Infos)

Ursache:

- mehrere Standards: UML, MOF, XMI (Versionskonflikte)
- nicht behobene Fehler im Standard
- unpräzise Spezifikation

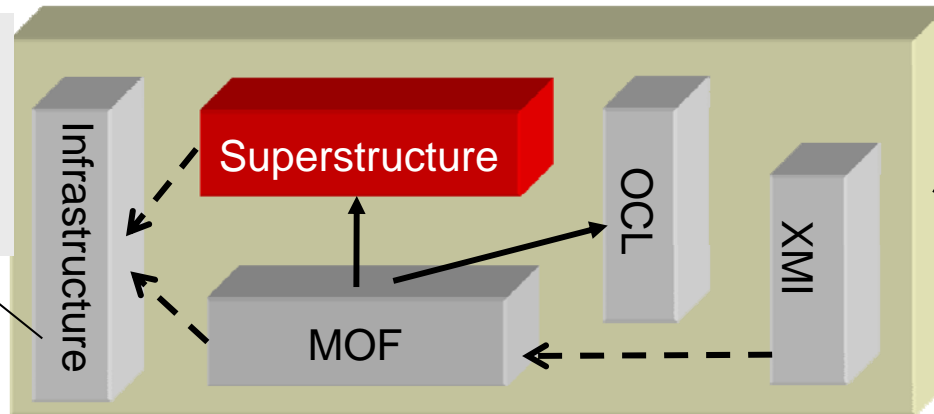
- UML 2.0 fordert sogar Diagramm-Austausch (XMI-Datei mit Teilbäumen von Layout-Repräsentationen)

4. UML-Überblick

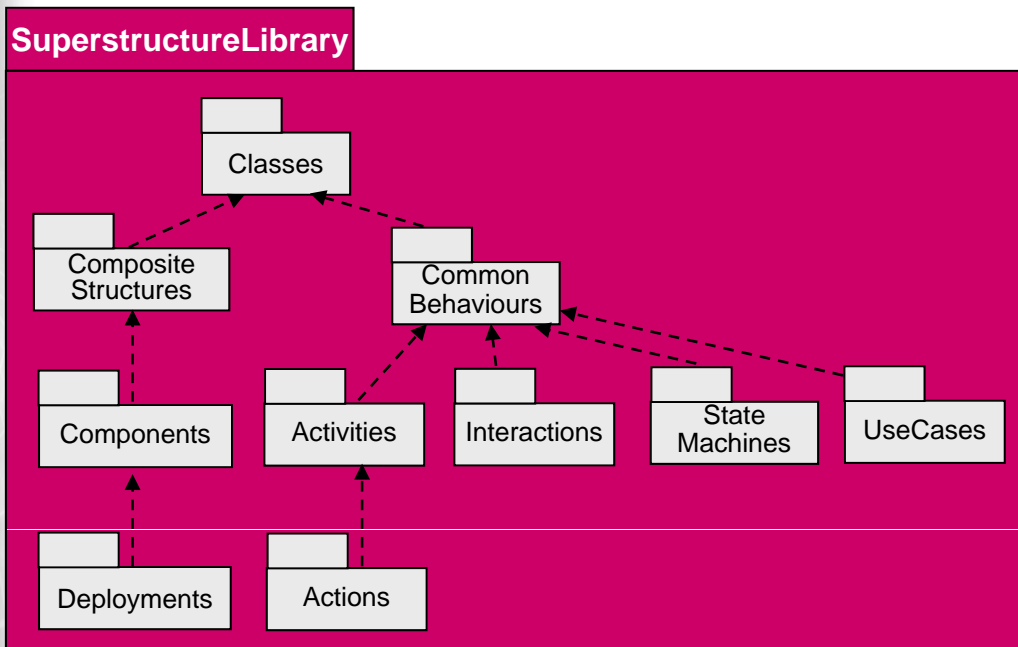
1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. UML-Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards
6. Beispiel: UML-Klassendiagramm

Struktur des Standards

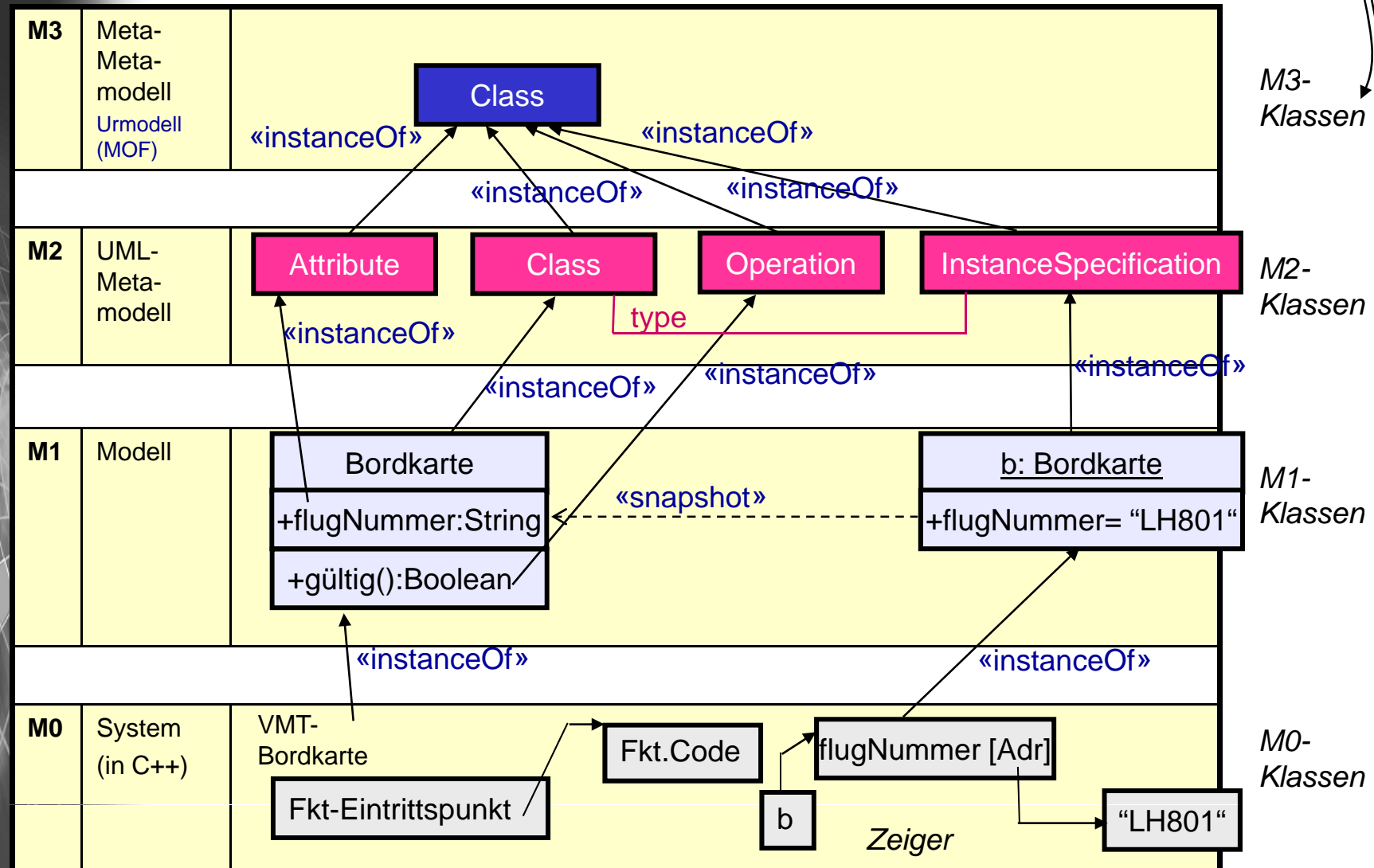
grundlegende Sprachkonstrukte in abstrakter Syntax (Klasse, Assoziation, ...)



- Modellierungskonzepte
- Sprachdefinitions-konzepte
- Transformations-konzepte



Die UML-Spracharchitektur



UML-Modifikationen (Aufbau eigener Sprachen)

- im Prinzip kann unter Verwendung von UML ein eigenes Metamodell (für eine DSL) aufgebaut werden

Wege, UML zu erweitern

- a) direkte Erweiterung des UML-Metamodells:
Einführung weiterer Metaklassen, die von UML-Metaklassen erben
- b) Profil-Mechanismus

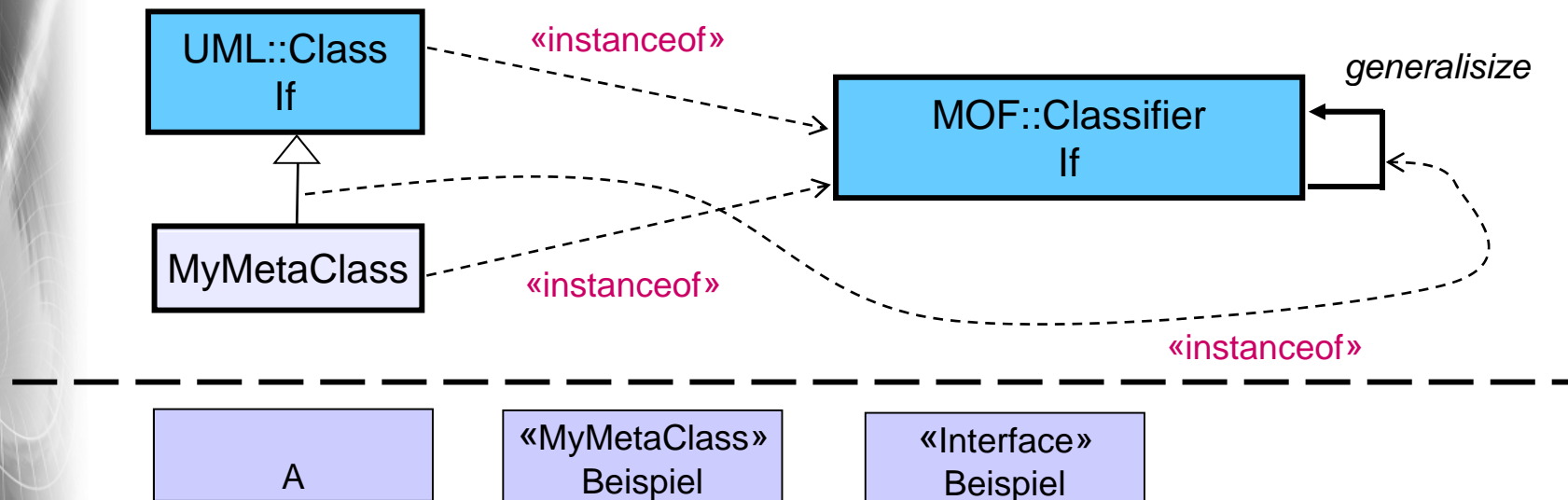
Achtung:

- beide Ansätze erlauben es aber **nicht**, Teile der UML auszublenden
- das komplette UML-Metamodell steht im Hintergrund, das nur mit OCL in der Instanziierung eingeschränkt werden kann

Direkte Erweiterung des UML-Metamodells

- Resultat: ein erweitertes Metamodell
→ eine erweiterte UML-Sprache

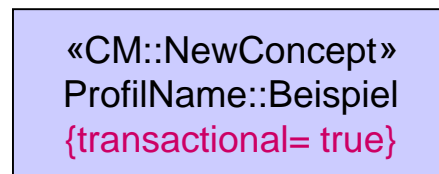
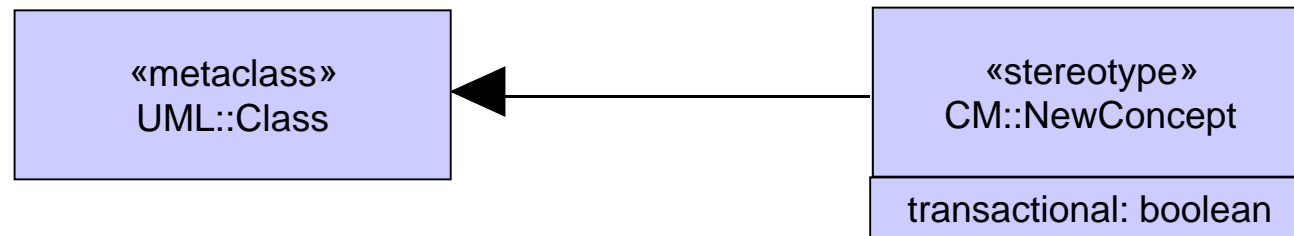
im UML'-Modell lässt sich nun **MyMetaClass** benutzen



Achtung: Ansatz funktioniert für jede MOF-basierte Sprache (z.B. UML,
aber ohne Tool-Unterstützung (komplett neue Sprache!!!))

Profile und Stereotype

- Definition einer Stereotype als Extension (nur für UML erlaubt) im Kontext eines Profils



Einschränkung in der Erweiterung:
Reduktion auf Attribute
(Tagged Values)

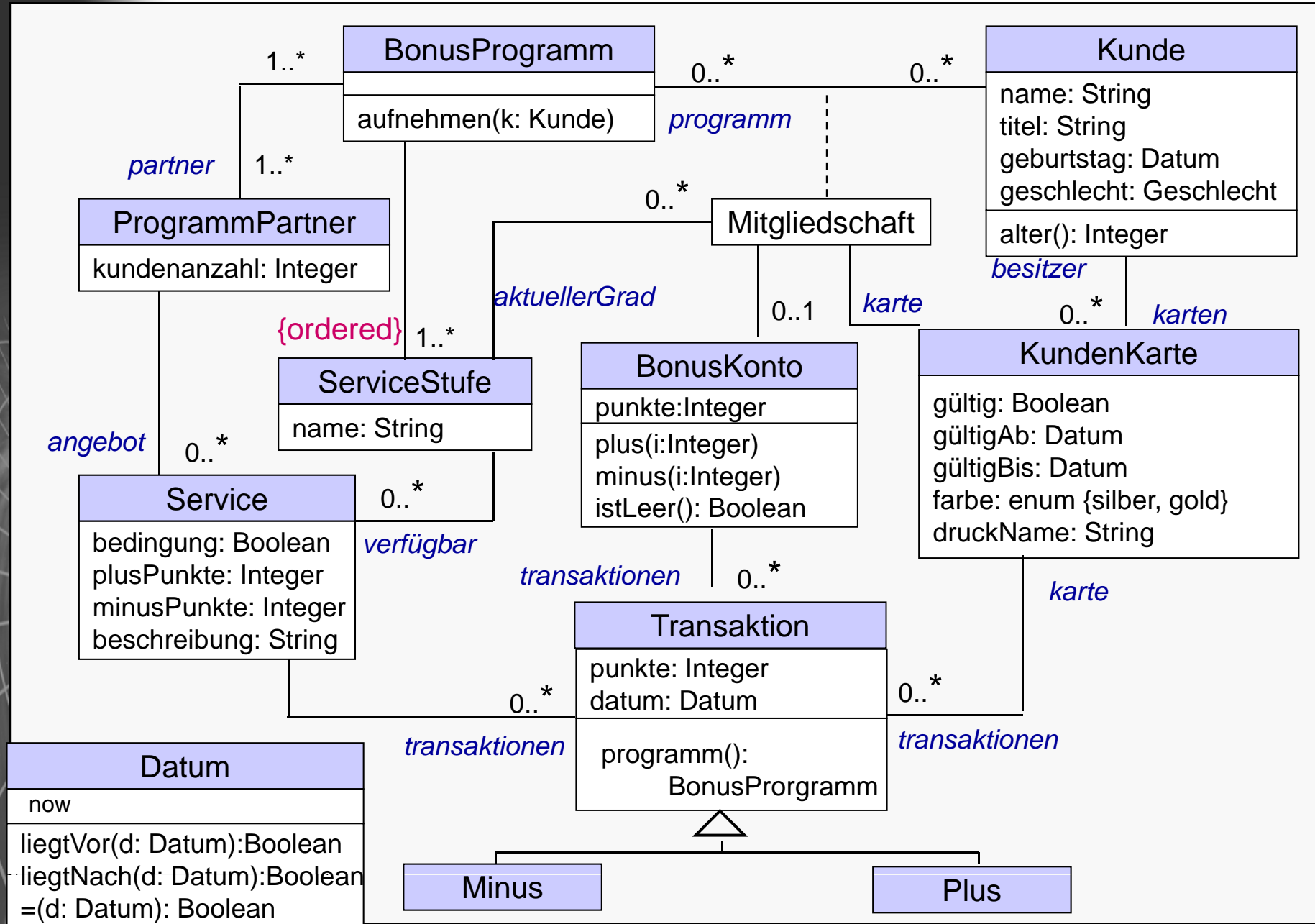
Vorteil:

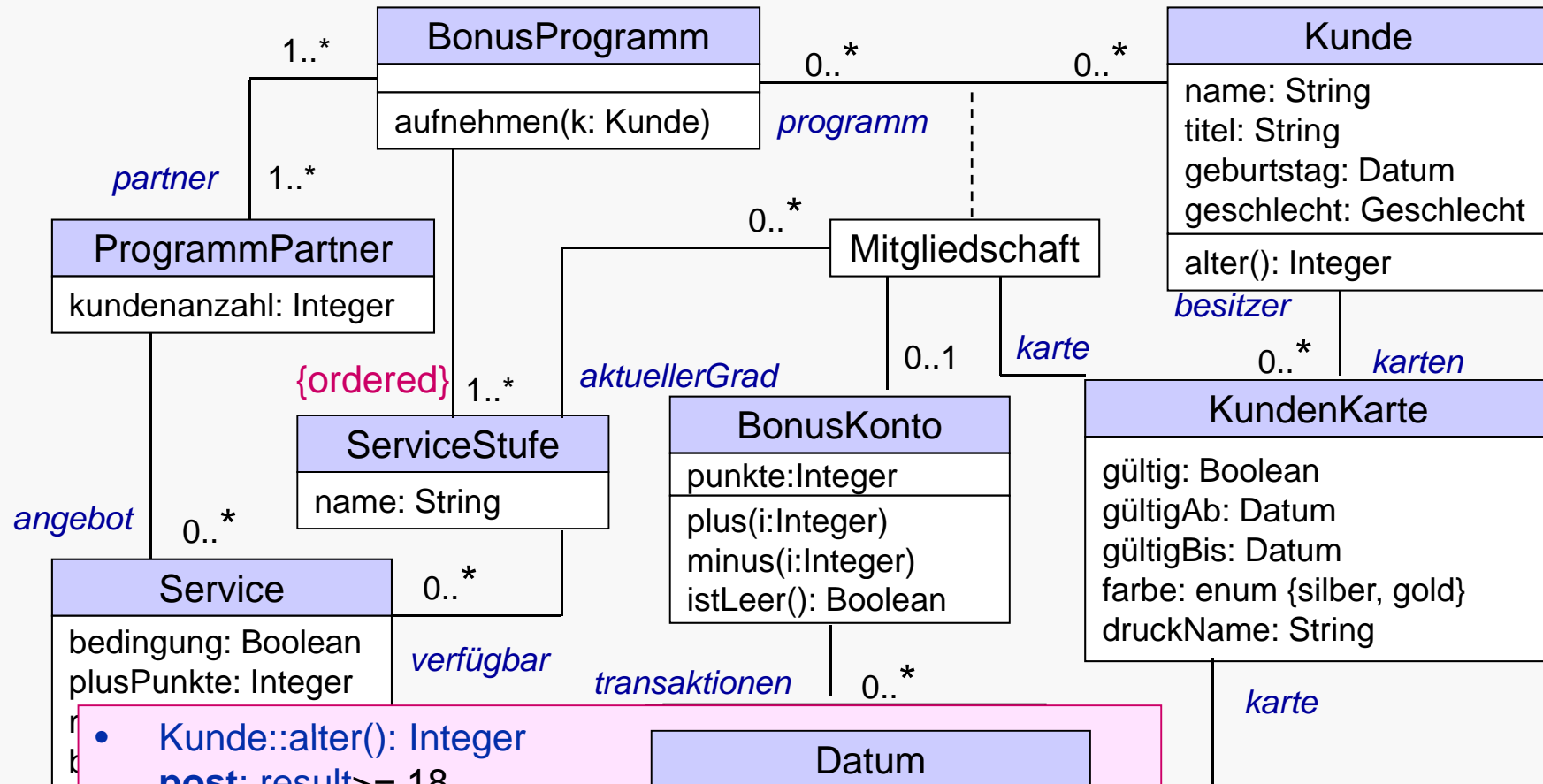
Kann Erweiterung mit UML-Tool vornehmen
und über spezielle Pakete benutzen

Problem: spezielle Symbole

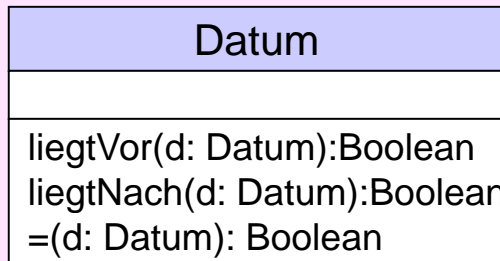
4. UML-Überblick

1. Historie von UML
2. Modellierungselemente von UML im Überblick
3. UML-Diagrammarten
4. Diagrammrepräsentationen in UML
5. Struktur des UML-Standards
6. Beispiel: UML-Klassendiagramm

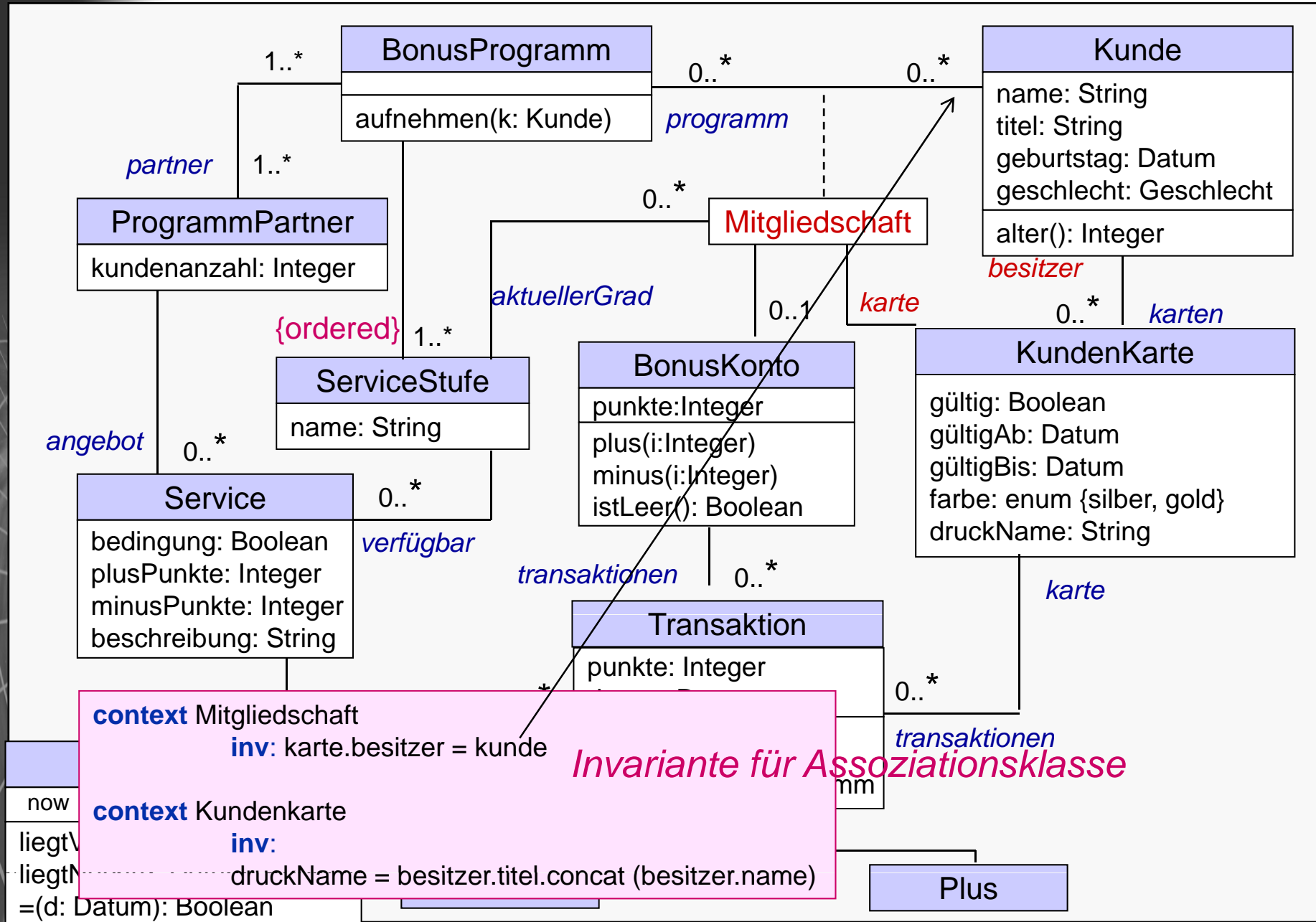


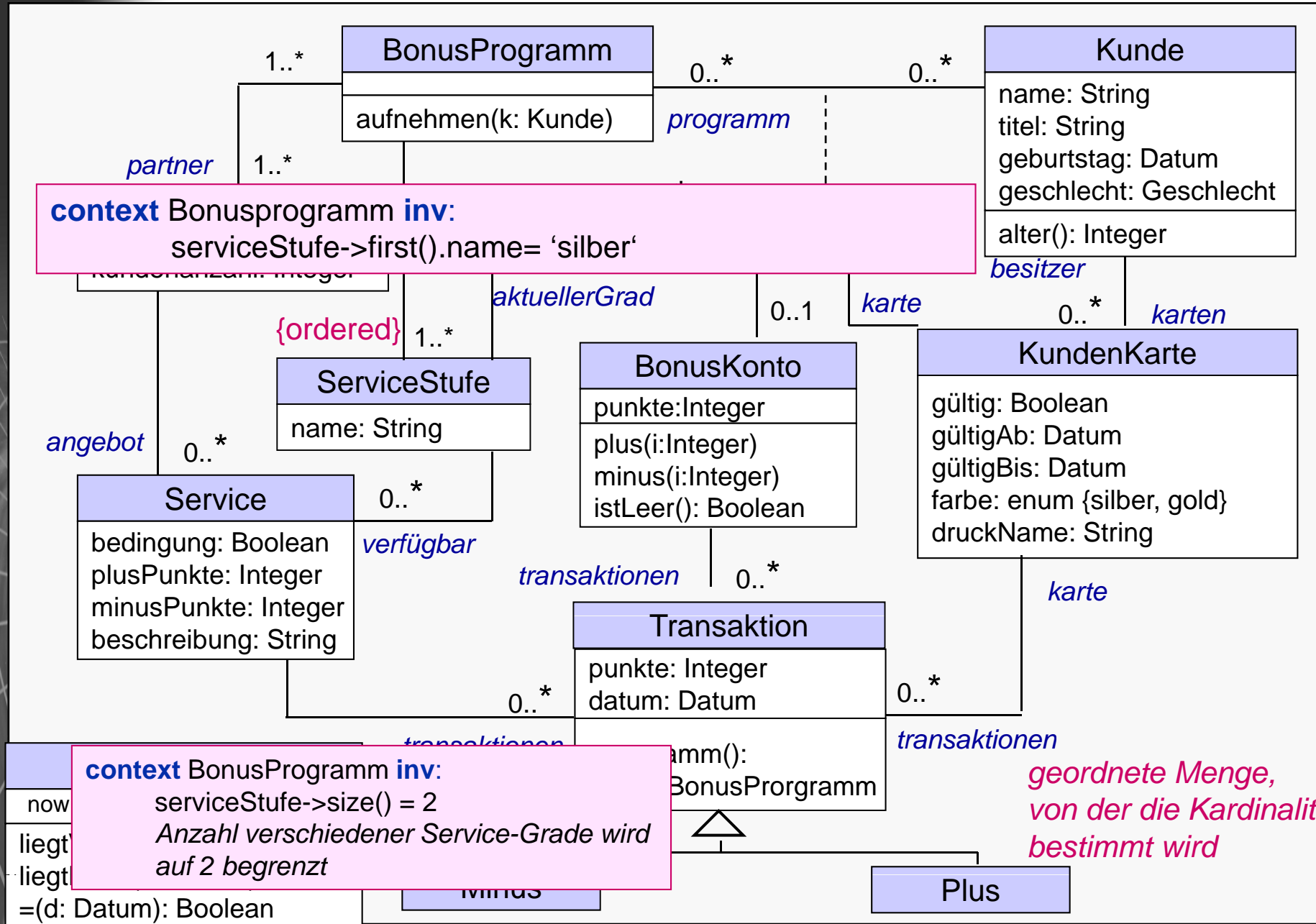


- Kunde::alter(): Integer
post: result >= 18
- **context** KundenKarte **inv:**
gültigAb.liegtVor(gültigBis)



Invariante für Attribut





*geordnete Menge,
von der die Kardinalität
bestimmt wird*