Programmierung und ontologische Semantik

Juli 2018

Dr. Reinhold Kloos

Dr. Reinhold Kloos

Email: reinhold.kloos@gmail.com

Kurzvita

Staatl. Gepr. Technischer Assistent
Informatik,
Berufsfachschule b.i.b., Paderborn

Systemprogrammierer/analytiker
BOSCH-Telenorma GmbH,
Frankfurt am Main

Diplom-Informatiker
FernUniversität Hagen

Wissenschaftlicher Mitarbeiter
Fraunhofer Institut SIT, Darmstadt

Doktor der Informatik, Ph.D.
City University of London, UK
ACTAS – Adaptive Composition and Trading with Agents for Services

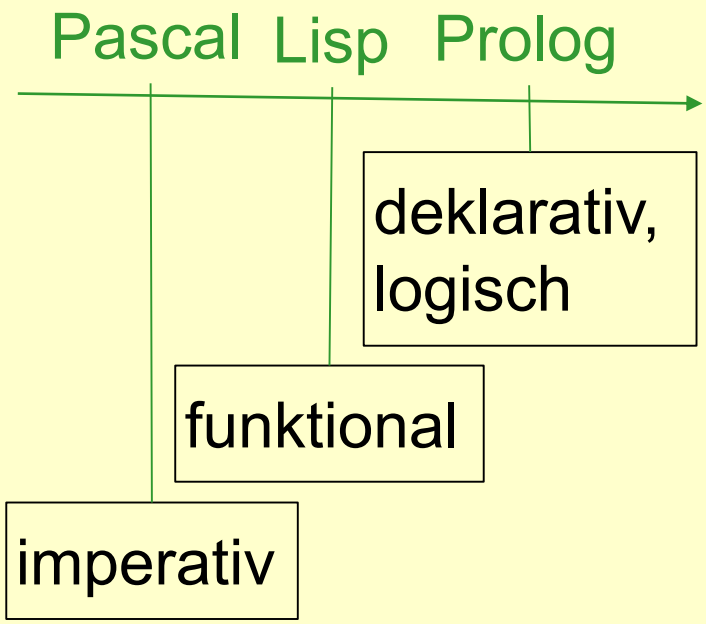
Wissenschaftlicher Mitarbeiter
Universität Duisburg-Essen

Lehraufträge

Schwerpunkte:
CSCW, Software Agenten, Web Services, E-Learning

Lehre:
Software Engineering, Schüler AG

Eine mögliche Einteilung (erste Dimension)



(*function par-1 … par-n*)

(+ 3 (* 4 5)) => 23

´(+ 3 (* 4 5)) => (+ 3 (* 4 5))

(first ´(a b c)) => a (rest ´(a b c)) => (b c)

Spezielle Ausdrücke erlauben die Programmsteuerung:

(progn *ausdruck-1 … ausdruck-n*)

(if *then-form [else-form]*)

```
(defun anfang-und-ende (l)
        (let ( (anfang (first l)) (ende (last l)) )
                (cons anfang ende) ) )
```

Ein gültiger ausführbarer LISP-Ausdruck (form) ist gegeben durch: Zahl, Character, String, Liste, Symbol (Variable), Funktionsaufruf, Makro, spezieller Ausdruck.

PROLOG (logische und deklarative Programmierung)

GESELLSCHAFT FÜR INFORMATIK

Knowledge Base (KB)

% Prolog Text mit Fakten

```
mann(adam).
mann(tobias).
mann(frank).
frau(eva).
frau(daniela).
frau(ulrike).
vater(adam,tobias).
vater(tobias,frank).
vater(tobias,ulrike).
mutter(eva,tobias).
mutter(daniela,frank).
mutter(daniela,ulrike).
```

% Prolog Text mit Regel

```
grossvater(X,Y) :- vater(X,Z), vater(Z,Y).

elternteil(X,Y) :- mutter(X,Y); vater(X,Y).

vorfahr(X,Z) :- elternteil(X,Z).
vorfahr(X,Z) :- elternteil(X,Y), vorfahr(Y,Z).

weg(Ziel,Ziel,Ortsliste):- write(Ortsliste),nl.  /* Ziel erreicht */
weg(Start,Ziel,Ortsliste):-
        strasse(Start,Neu), /* ... Strasse zu einem neuen Ort ... */
        not(member(Neu,Ortsliste)),    /* ... noch unbekannt  ... */
        weg(Neu,Ziel,[Neu|Ortsliste]). /* ... Rekursion */
```

% Prolog Abfrage mit Query

```
?- frau(X).
X=eva
X=daniela
X=ulrike
no
?- frau(heinrich).
no
```

```
?- grossvater(adam,frank).
yes.
?- grossvater(X,ulrike).
X=adam
```

```
?- vorfahr(Y).
Y=tobias
Y=daniela
Y=adam
Y=eva
no
```

```
?- weg(warstein, soest,_).
[belecke,lippstadt,soest]
[belecke,allagen,soest]
[hirschberg,allagen,soest]
```

GESELLSCHAFT FÜR INFORMATIK

Nutzen einer Typisierung einer Programmiersprache

- **(Hauptnutzen) Der Compiler kann bei statischer Analyse Fehler in der Programmierung erkennen**
- **(Weitere Nutzen) Unterstützung bei der Modellierung, der Dokumentation und der Abstraktion (Abstract Data Type, ADT)**
- **In der logischen Programmierung werden Typen zusätzlich genutzt, um sogenannte Constraints aufzustellen (Constraint Logic Programming, CLP(X))**

Diplomarbeit am Lehrstuhl Beierle, FernUniversität Hagen

„Erweiterung eines Typ-Konzepts für PROLOG zur Unterstützung der modularen Software-Entwicklung und Meta-Programmierung in der KI“

Eine mögliche Einteilung (zweite Dimension)

objekt-orientiert

modula-risiert

strukturiert

Java

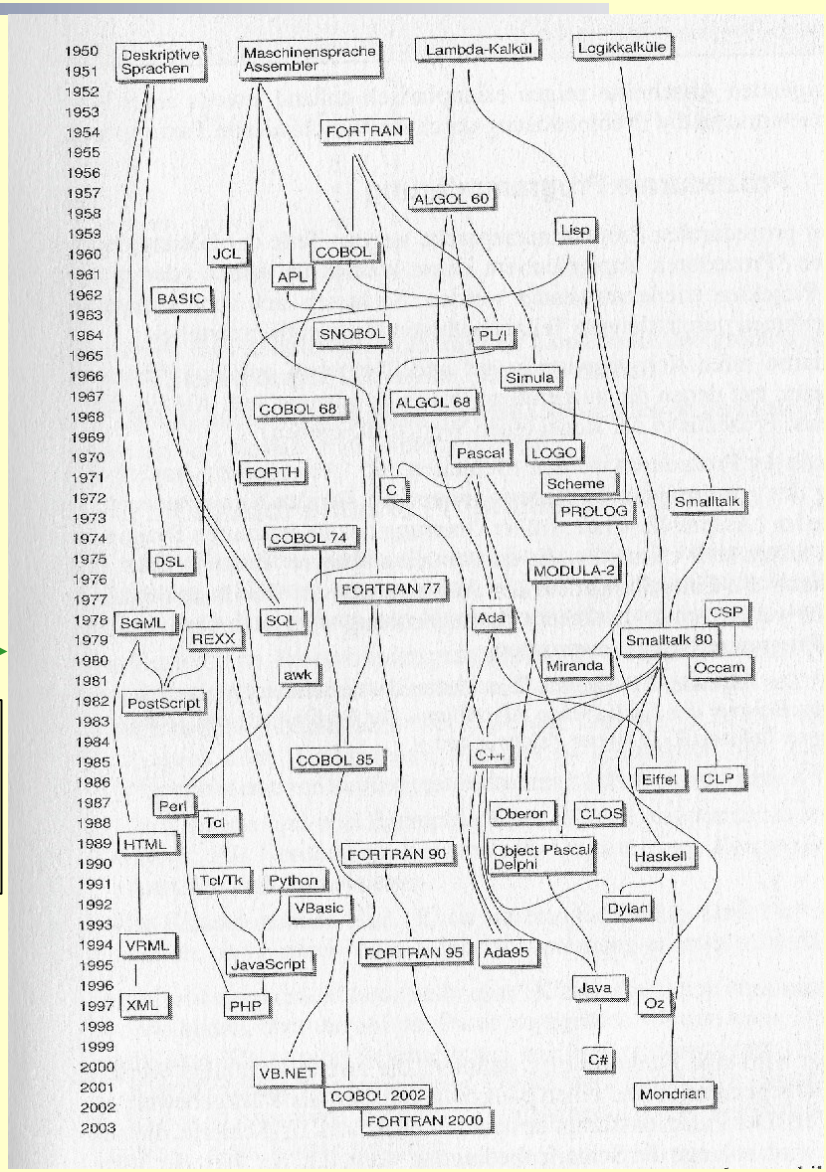
Modula2

Pascal Lisp Prolog

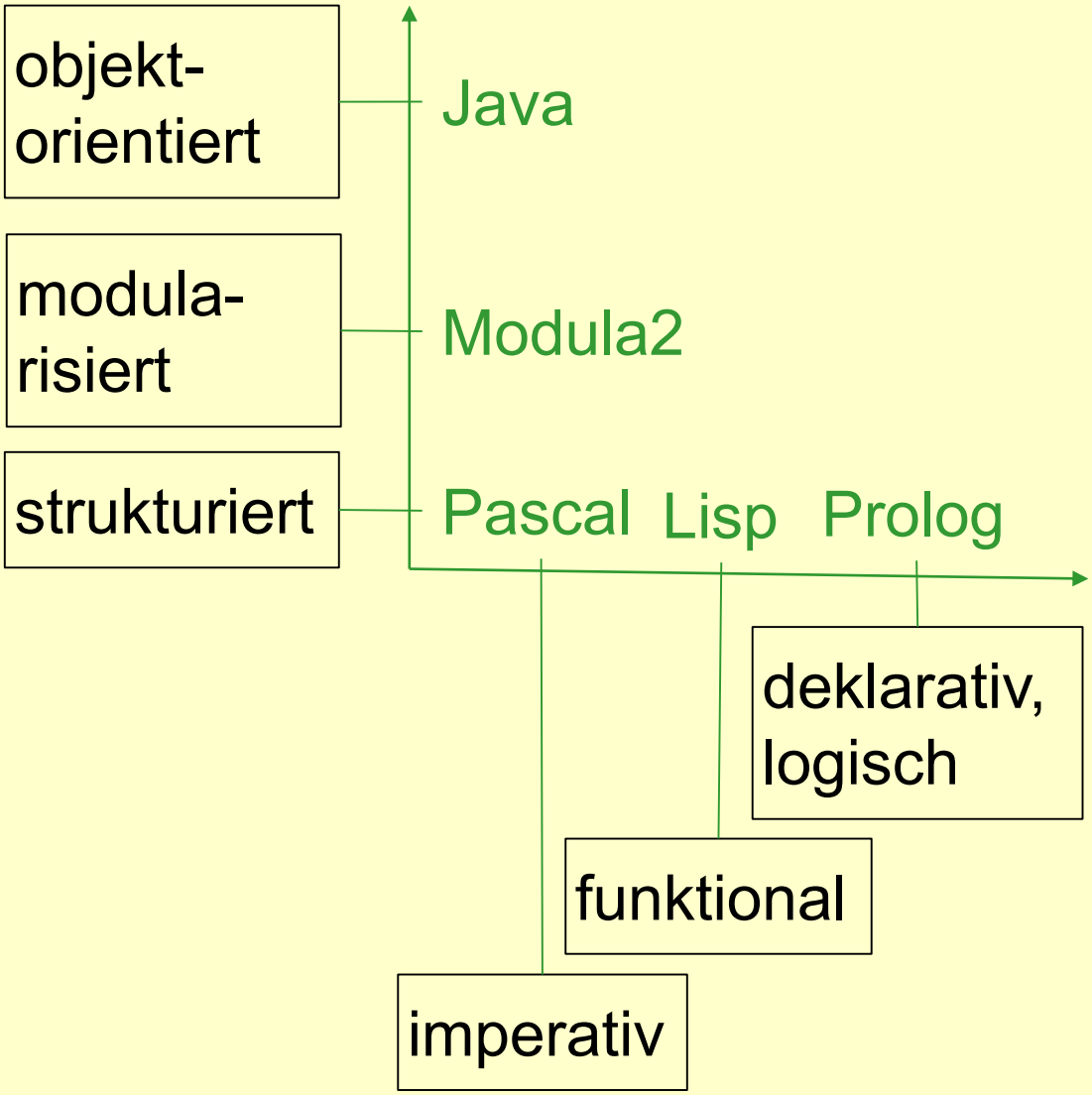
deklarativ, logisch

funktional

imperativ

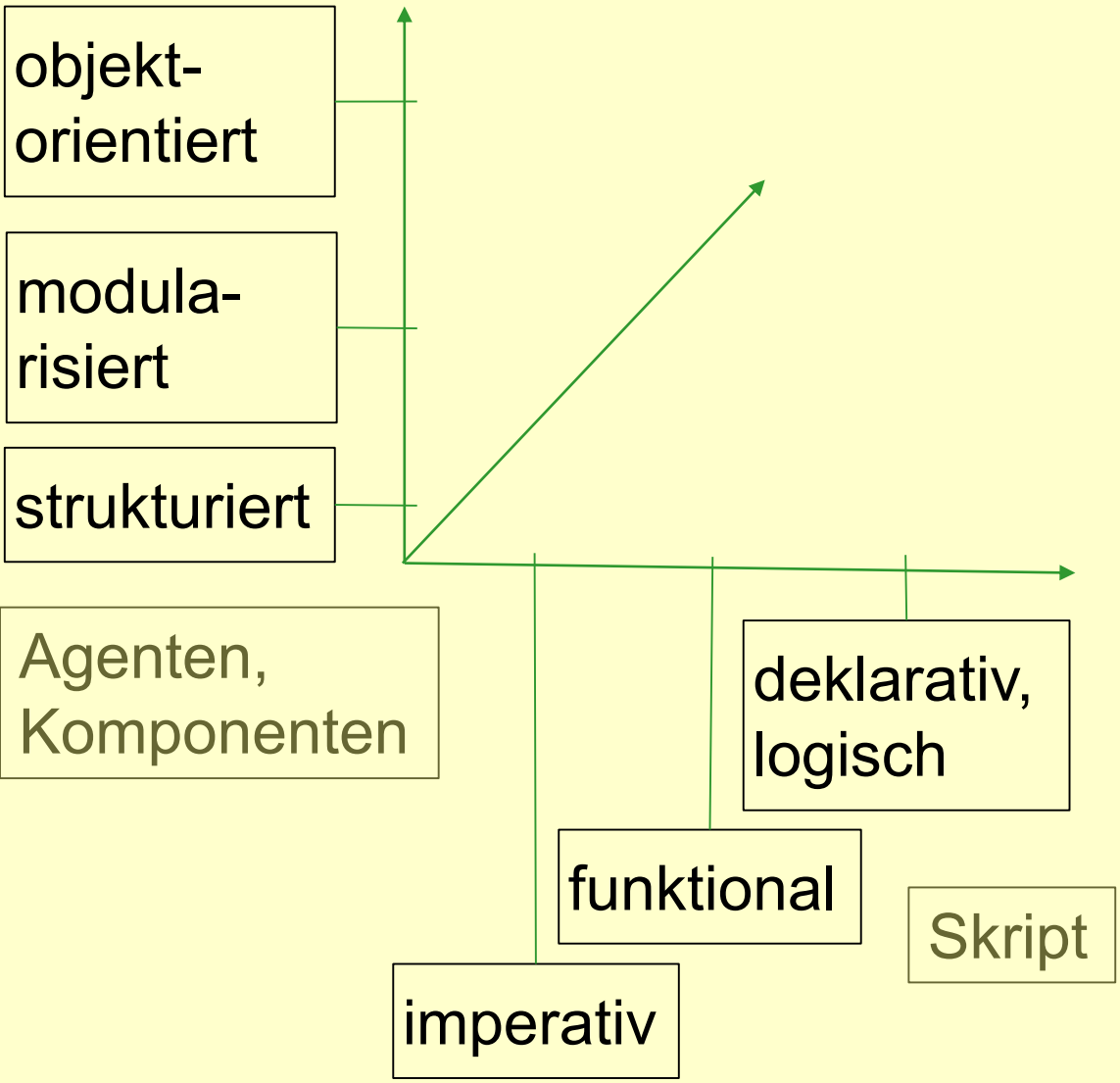


Eine mögliche Einteilung (eine dritte Dimension?)



Wie hat es sich weiter entwickelt?

Zweck-Orientierte Aspekte als mögliche dritte Dimension



Schnittstellen-orientiert Software-Engineering

Optimierung

- Parallelrechnen
- System-nah / −unabhängig
- reduzierter Code
- Realtime, Prozess

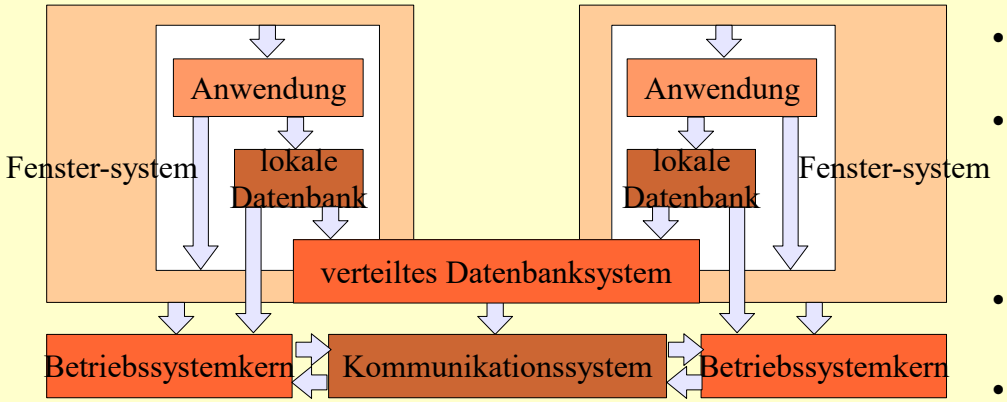
Verteilte Programmierung

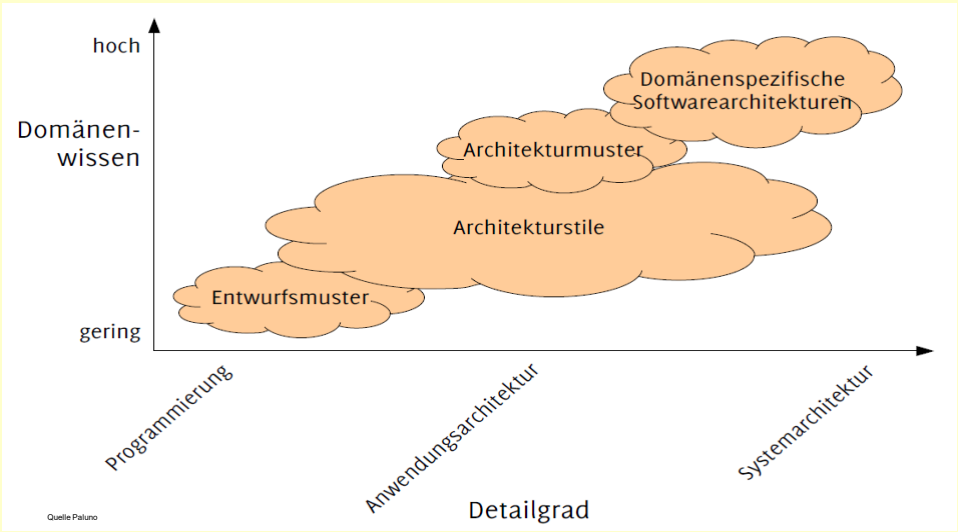
- Client-Server u.a.
- Multi-Agenten-Systeme (MAS)
- Service Oriented Architectures (SOA)

KI (Neuronal, Lernen …)

Makroarchitektur – Design auf Komponentenebene (Bausteinsicht)

Architekturstile und –muster auf Komponentenebene:





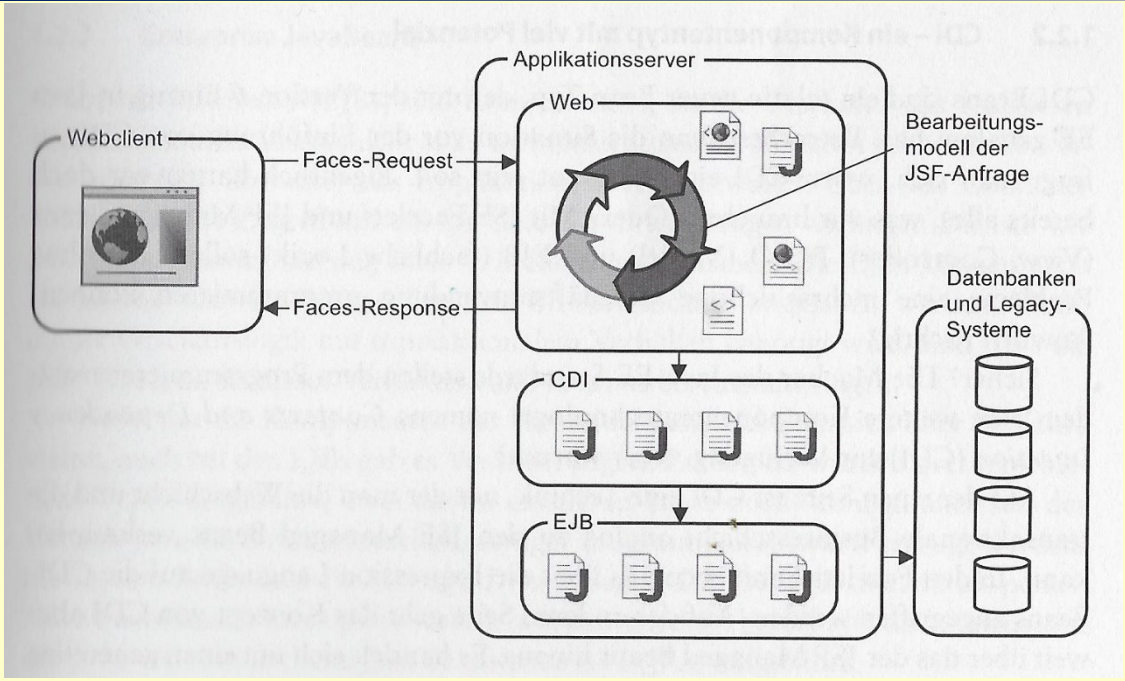
- Komponentenkonzepte sind für den Entwurf (z.B. UML-Komponentendiagramme)
- Theorie z.B. nachschlagbar in: *Richard N. Taylor, Nenad Medvidovic, Eric M. Dashofy: Software Architecture – Foundations, Theory, and Practice, Wiley, 2010*
- Komponenteninstanzen werden durch ggf. mehreren Klassen realisiert
- Realisierung in der Praxis durch Standards für verschiedene Komponentenmodelle (Beispiele: EJB für Java und Open Service Gateway Initiative (OSGi))

- Beispiele für Architekturstile: „Publish-Subscribe“, „Event-based“, „Batch-sequential“, „Pipes-and-Filters“

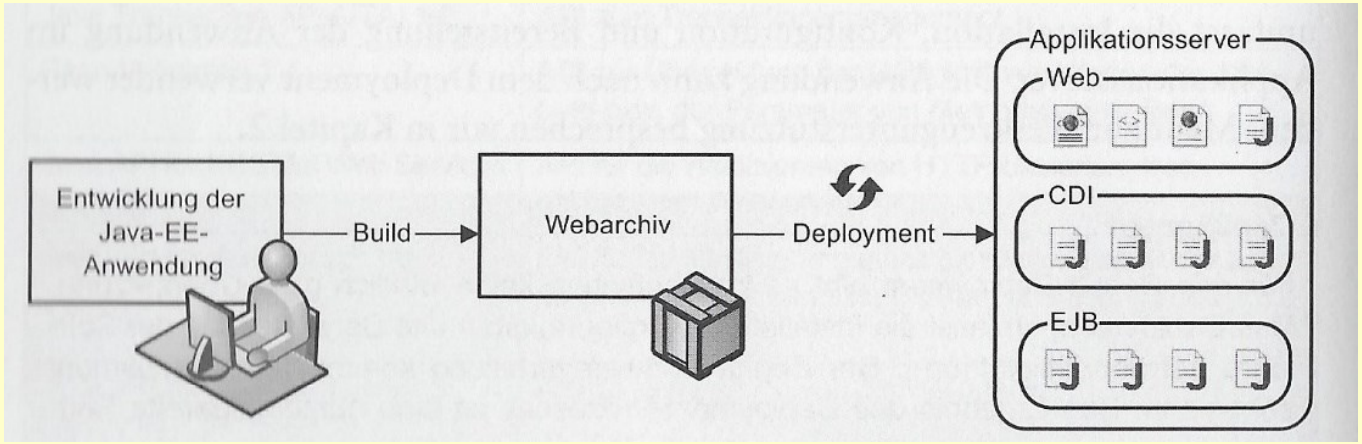
- Beispiele für Architekturmuster:
  - „Three-Tier“ – Dreischichten-Architektur
  - „Model-View-Controller“ (MVC)
  - „Sense-Compute-Control“ (SCC)

Alternative Architekturmuster: Multi-Agenten Systeme (MAS) sowie Web Services und SOA

EJB – ein Komponentenmodell und MVC

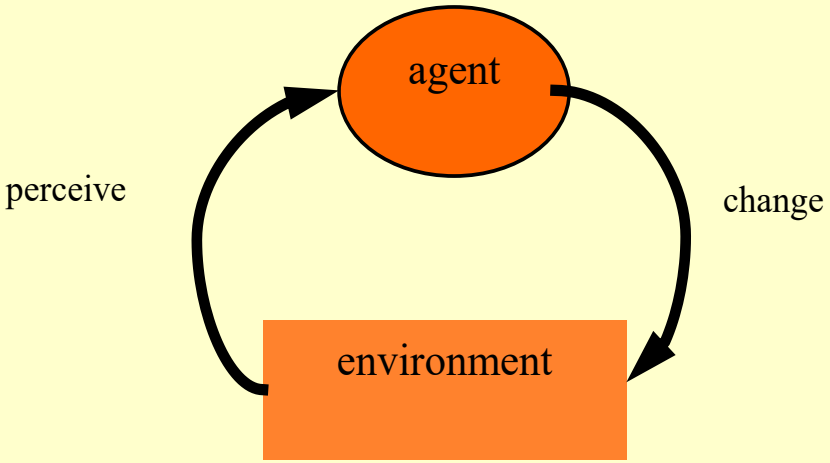


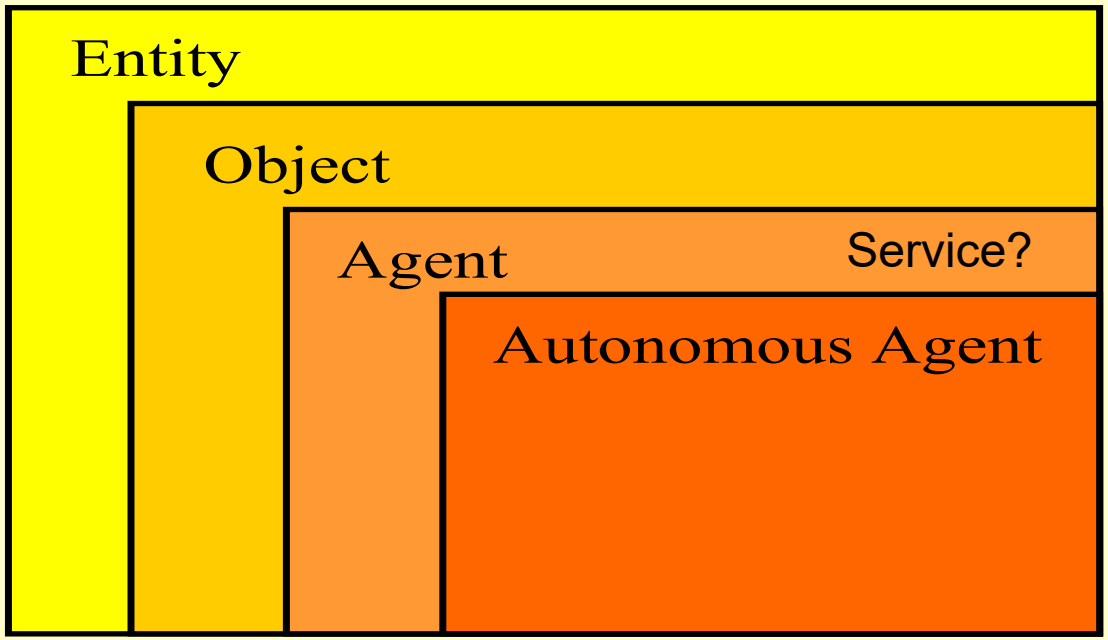
MVC mit EJB



Arbeiten
mit EJB

Software Agent – An Alternative Software Paradigm

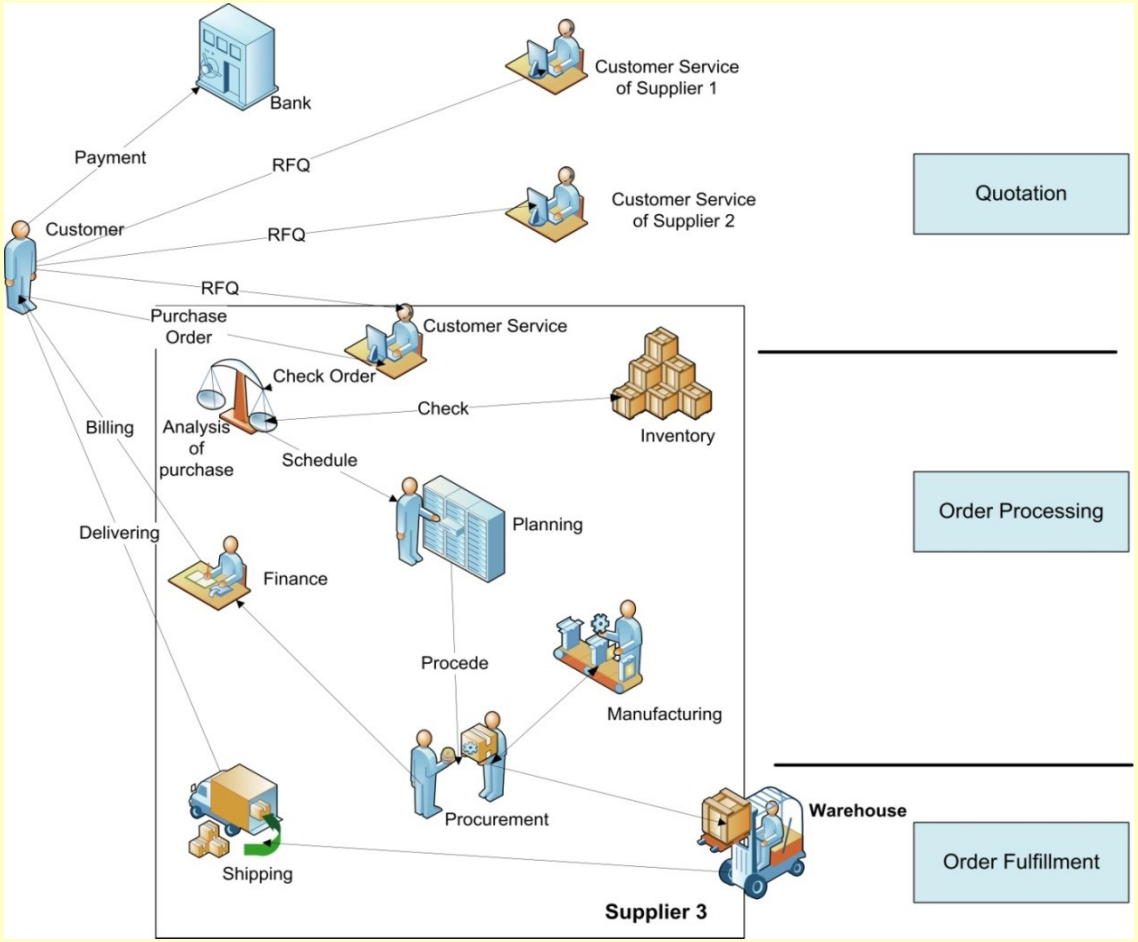




Merkmale von Agenten

- Autonom
- Re-Aktiv
- Pro-Aktiv
- Kommunikativ
- Weitere Merkmale:
  z.B.:
  intelligent,
  lernend,
  migrierend,
  sicher …

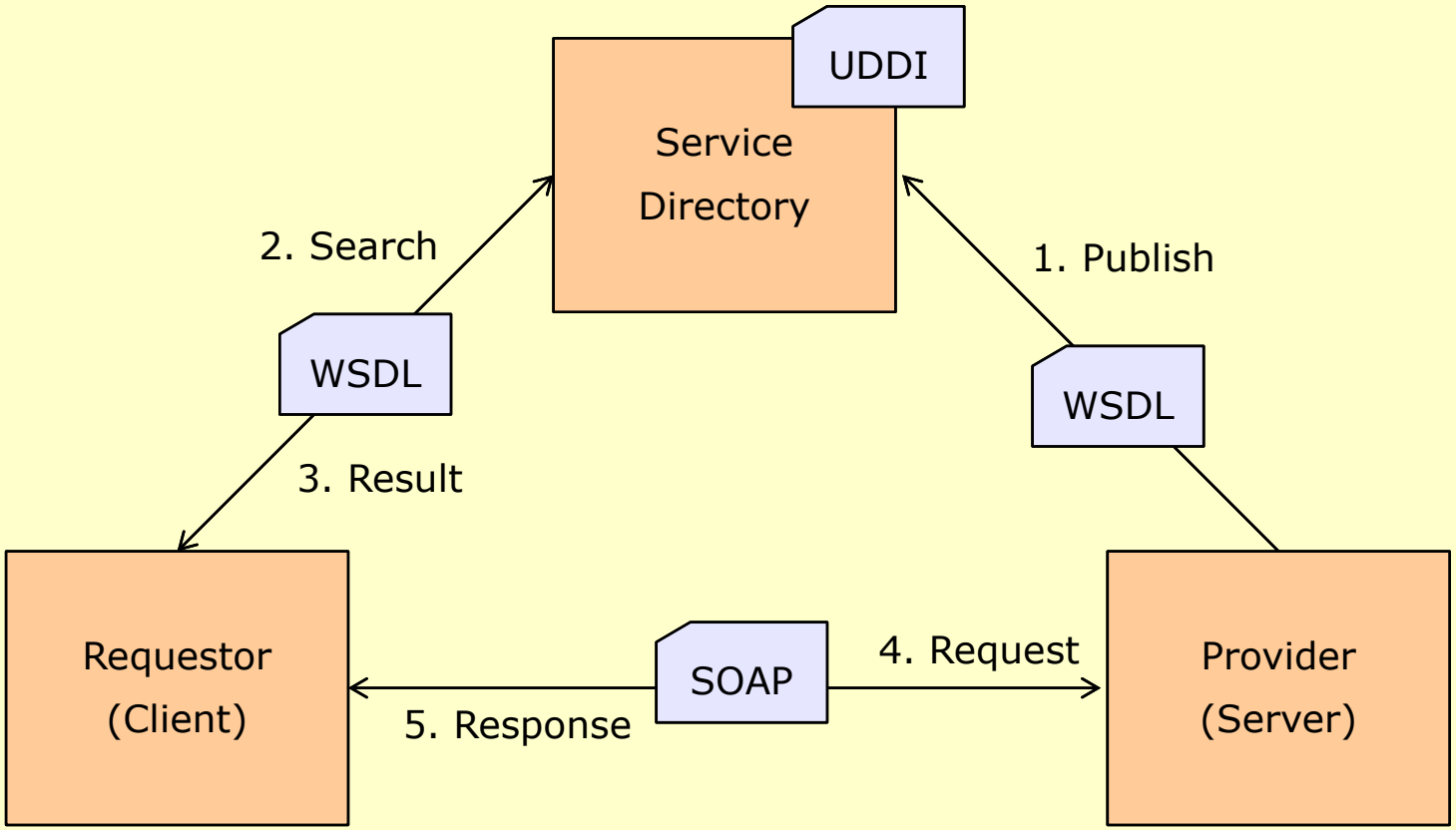
Verteilter Geschäftsprozess und seine Herausforderungen



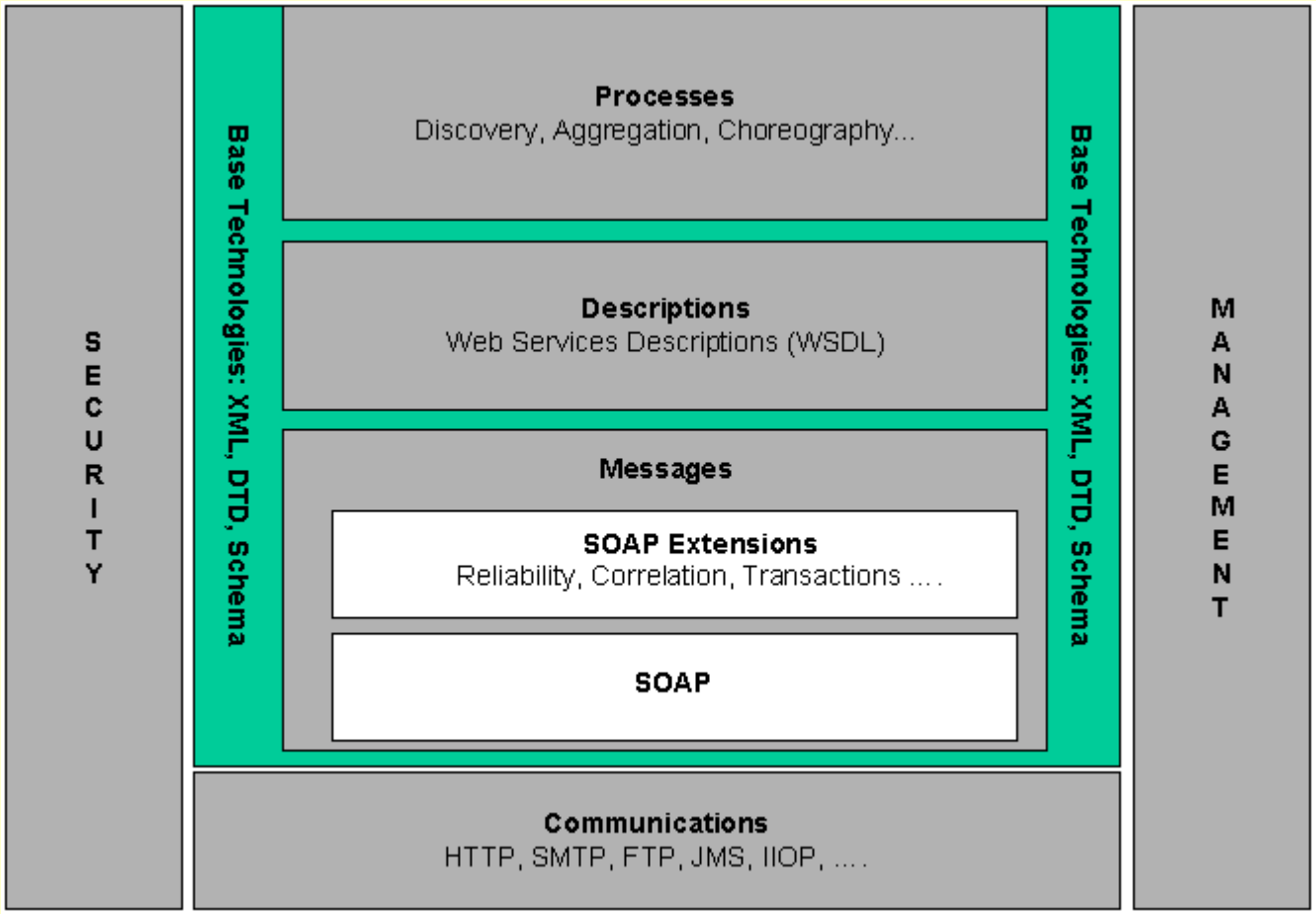
Herausforderungen

- Integration verschiedener Umgebungen (DIS -> EAI, BPM)
- Semantik – Vereinbarungen auf die angebotene Funktionalität
- Dienstleistungsabarbeitung ist hybrid (Nicht-funktionale Kriterien einbeziehen)
- Autonome Service-Ausführung / Discovery

Web Services und Service Oriented Architecture (SOA)

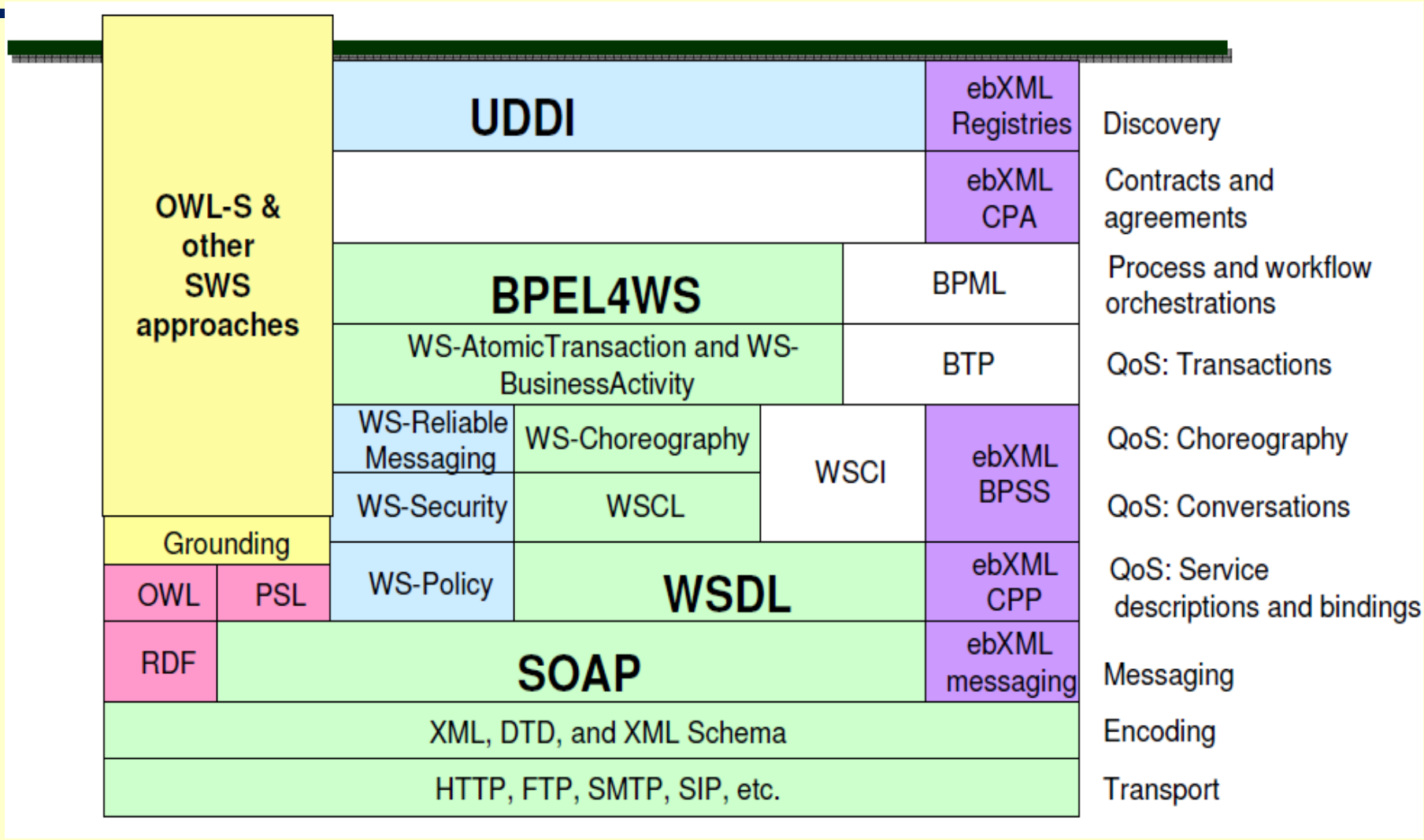


Architektur – Stack



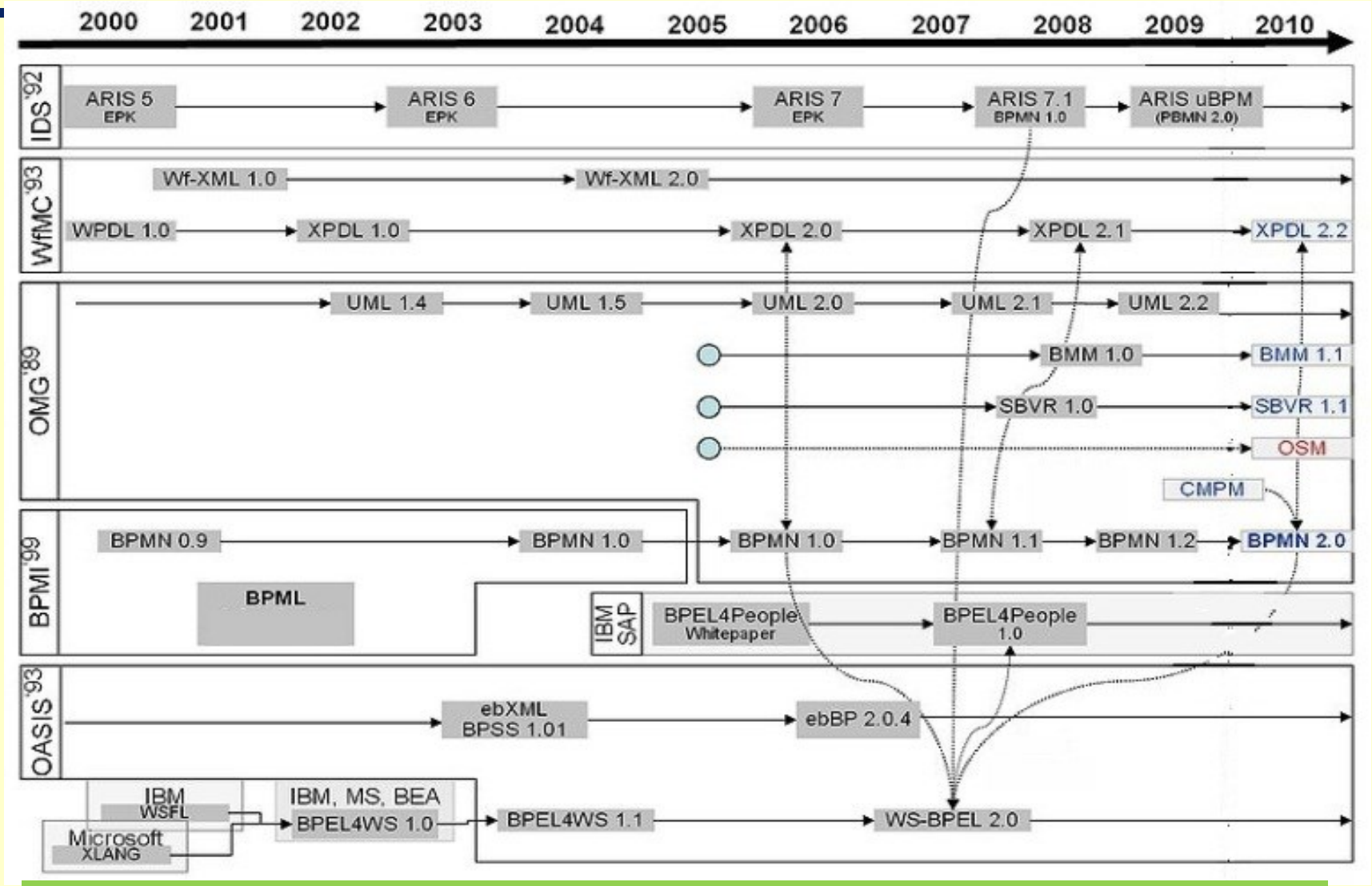
Quelle: [W3C-WS]

Web Service Standards



Choreography, Orchestration, Coordination, Transaction

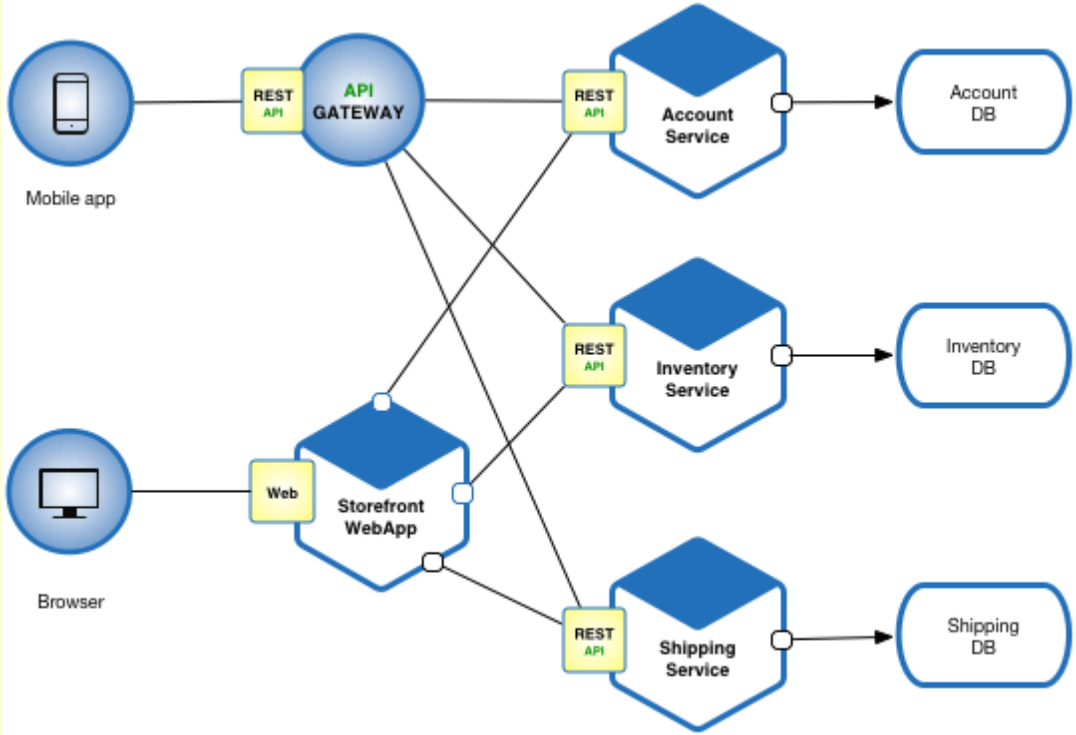
Web Service Standards (2/2)



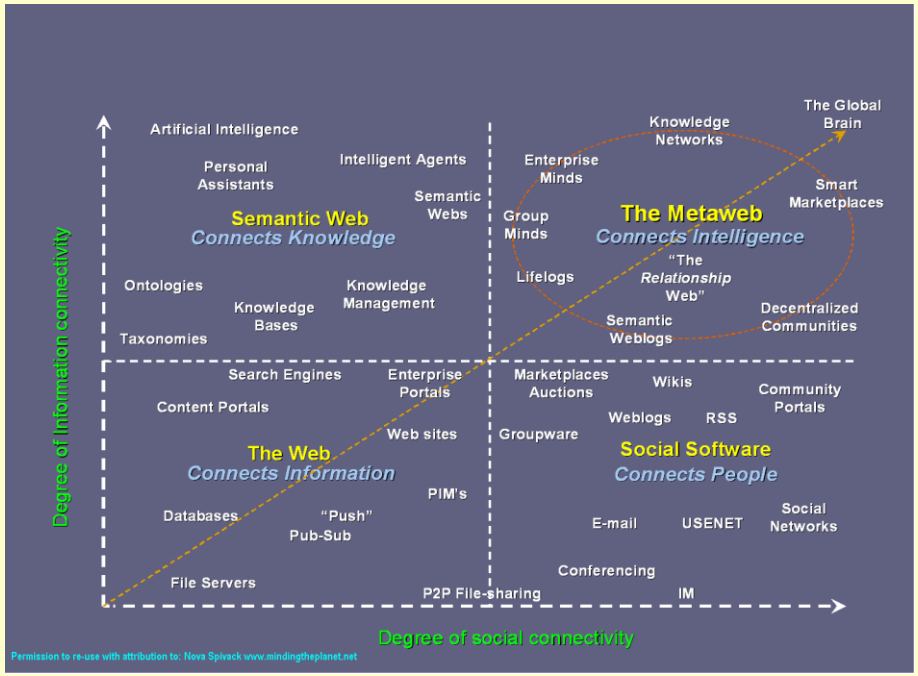
Business-Process-Oriented Service Composition ➔ WS-BPEL

Micro-Services

- Micro-Services sollen klein sein
- Sie kennen keine (automatische) Service-Composition (Composite Service)
- Erledigen eine spezifische Aufgabe
- Es wird immer die API zum Zugriff genutzt.
- Die Realisierung bleibt bei einem Entwicklerteam



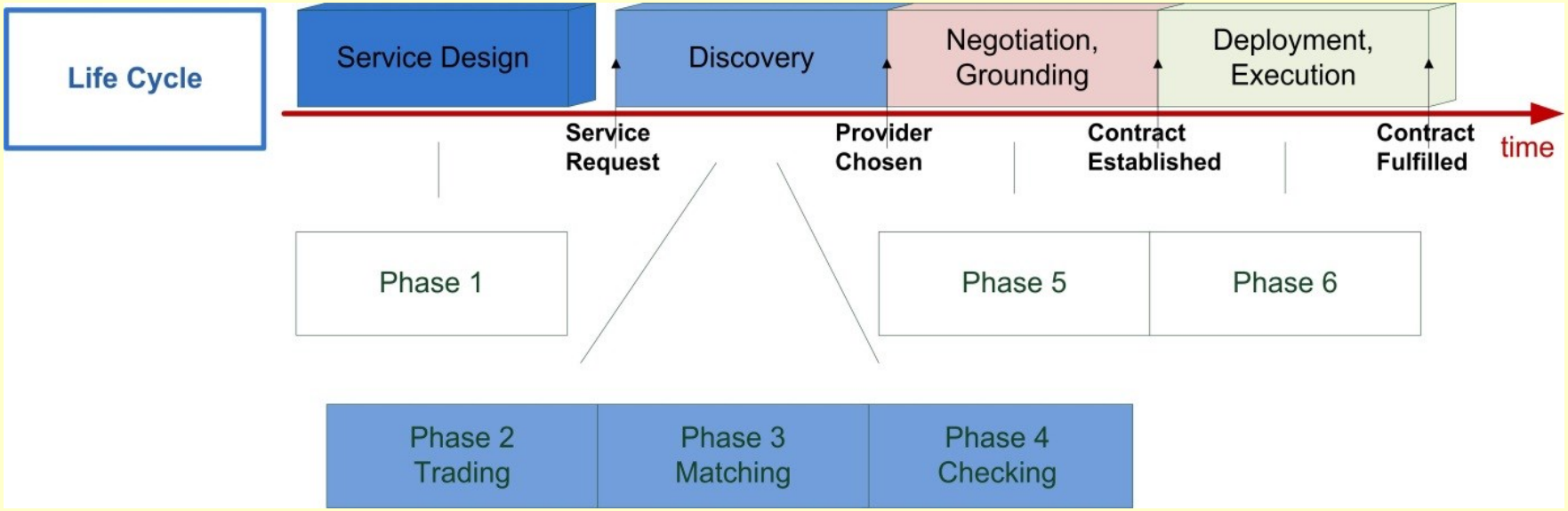
Ziel ist ein skalierbares, verteiltes, verlässliches System, dass aber nicht so dynamisch ist wie SOA.



(Semantic) Web Services Erweitert und Ontologie

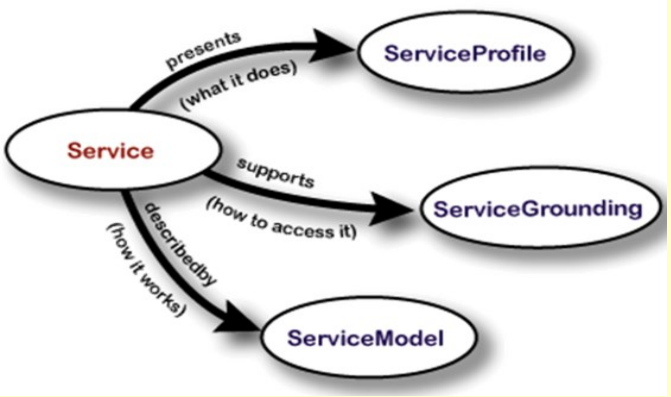
Was ist ein Service überhaupt?

Functional Domain and view on service	Non-Functional Properties (nfp)	Inherent Complexity Policies, Choreography, Orchestration	Service Design, Trading, Matching and other phases of Life Cycle
First Aspect	Second Aspect	Third Aspect	Fourth Aspect

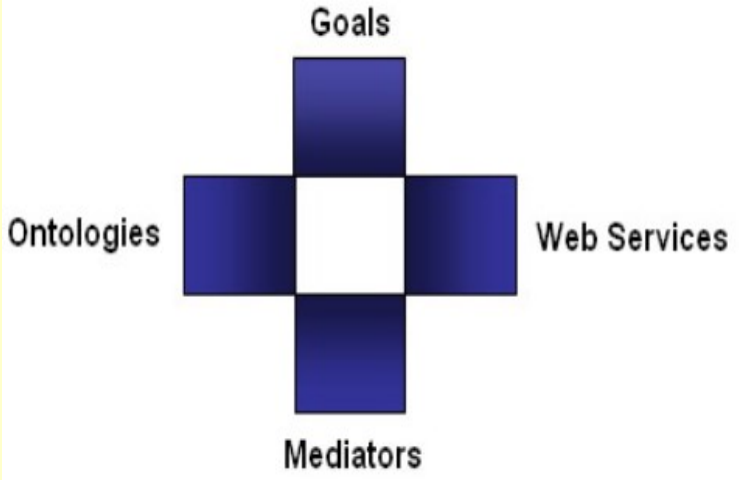


Ontology-Languages in Semantic Web

- RDF(S) – Erweiterung des Resource Description Frameworks (RDF) <W3C standard>
- Web Ontology Language (OWL) – genutzt in OWL-S für Web Services <W3C standard>
- WSML – Teil der Web Service Modelling Ontology (WSMO) <WSMO initiative>



OWL-S and seine Spezialisierung für Geschäftsprozesse SWSF

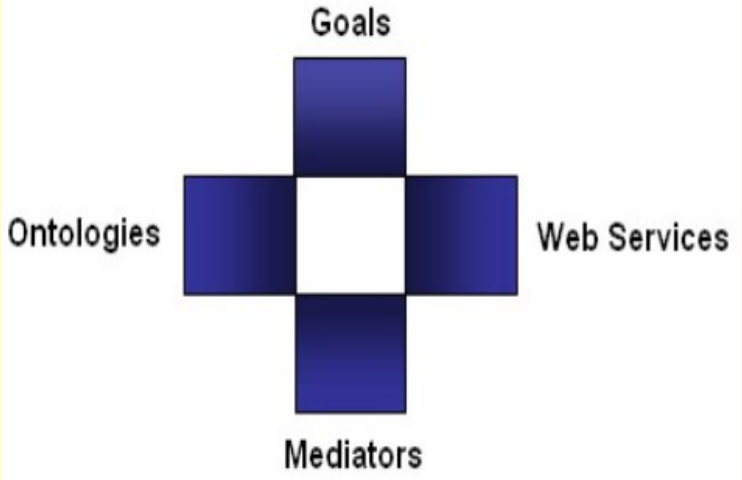


WSMO

Alternativer Ansatz: Semantische Erweiterung des WSDL-Standards: SAWSDL (formerly WSDL-S)

Top-Level Elemente als Hauptkonzepte zur Beschreibung eines Semantic Web Services

- Ontologien
  Stellen die Terminologie der Domäne zur Verfügung
- Services
  Dienste werden mit Capabilities (P[Q]R, e.g. IOPE), Interfaces und Internal Working mit der Terminologie von den Ontologien beschrieben
- Goals
  Modellierung der Benutzersicht
- Mediatoren
  Unterschiedliche Mediatoren setzen zwischen den anderen Top-Level Elementen um:
  Konzepte von Ontologien kommen in Relation
  Goals werden mit Dienstleistungen verbunden



WSMO

OWL-S definiert eine höhere Ontologie für service mit vier Hauptkonzepten

- Service

  Zentrales organisatorisches Konzept
  (Ähnlich WSMO, aber keine Unterscheidung der Benutzersicht (Goals))

- Service Profile

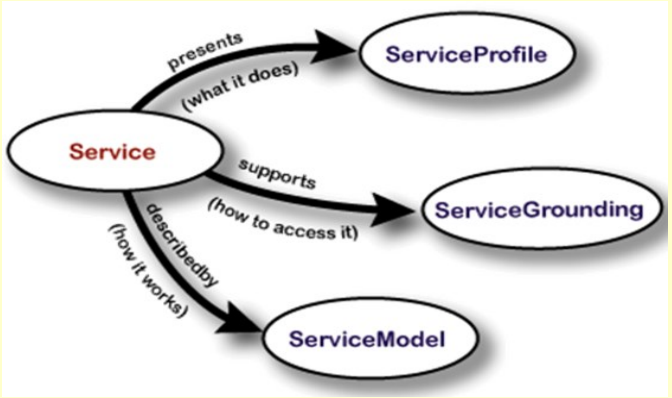
  Festhalten von funktionalen und nicht-funktionalen Eigenschaften für semantisches Service Discovery

- Service Model

  Modellierung der funktionalen Eigenschaften mit den einzelnen Prozessen

- Service Grounding

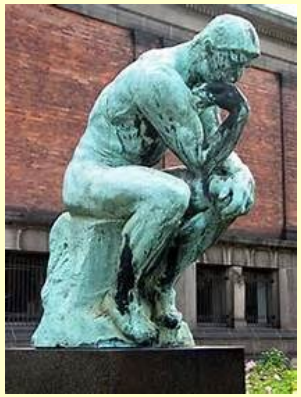
  Festhalten wie der Service Client den Dienst nutzen kann.
  (Verknüpfung mit WSDL-Beschreibungen)



OWL-S

Was ist eine Ontologie?

Theoretische Philosophie:



- Einteilung des Seienden
- Metaphysische Studie für Grundstukturen von Wirklichkeit und Möglichkeit (im Speziellen von existierenden Entitäten eines Interessensgebiet)

Informatik:

- Ein Artefakt
- Wissen über ein Interessensgebiet in einer maschinell-verarbeitbaren Form repräsentieren

Je nach Anwendungsgebiet existieren verschiedene Definitionen

Allgemeine Eigenschaften:

- Formalität – verarbeitbare Wissens-Repräsentations-Sprache
- Ausdrücklichkeit – Wissen muss ausdrücklich sein (Regeln)
- Geteilte Übereinkunft – Kein Standard, sondern begrenzt auf eine Gruppe
- Konzeptionelle Erfassung – intuitiv begreifbare Konzepte und Relationen
- Spezifisch für ein Interessensgebiet

Nutzung der Ontologie

Generelle Nutzung

- Klassifikation von Konzepten
- Kategorisierung von Instanzen

Unterscheidung nach der Wissensverknüpfung

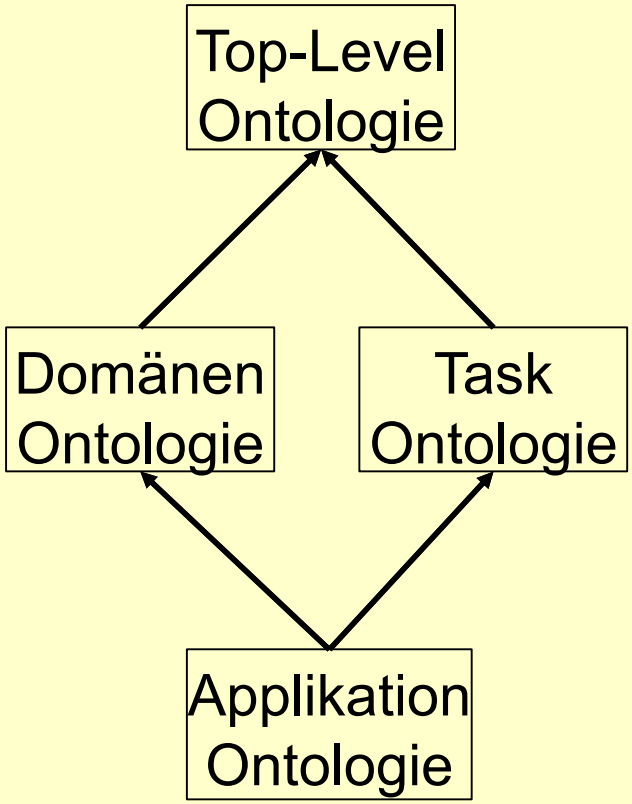
- Ontologie nur für ein System
  (z.B. Expertensystem)
- Ontologie zur Verbindung verschiedener Systeme

Unterscheidung nach der Wissensabstraktion

- Wird mit Konzepten auf Meta-Level oder Instanzen gearbeitet

Unterscheidung nach automatisierter Nutzung

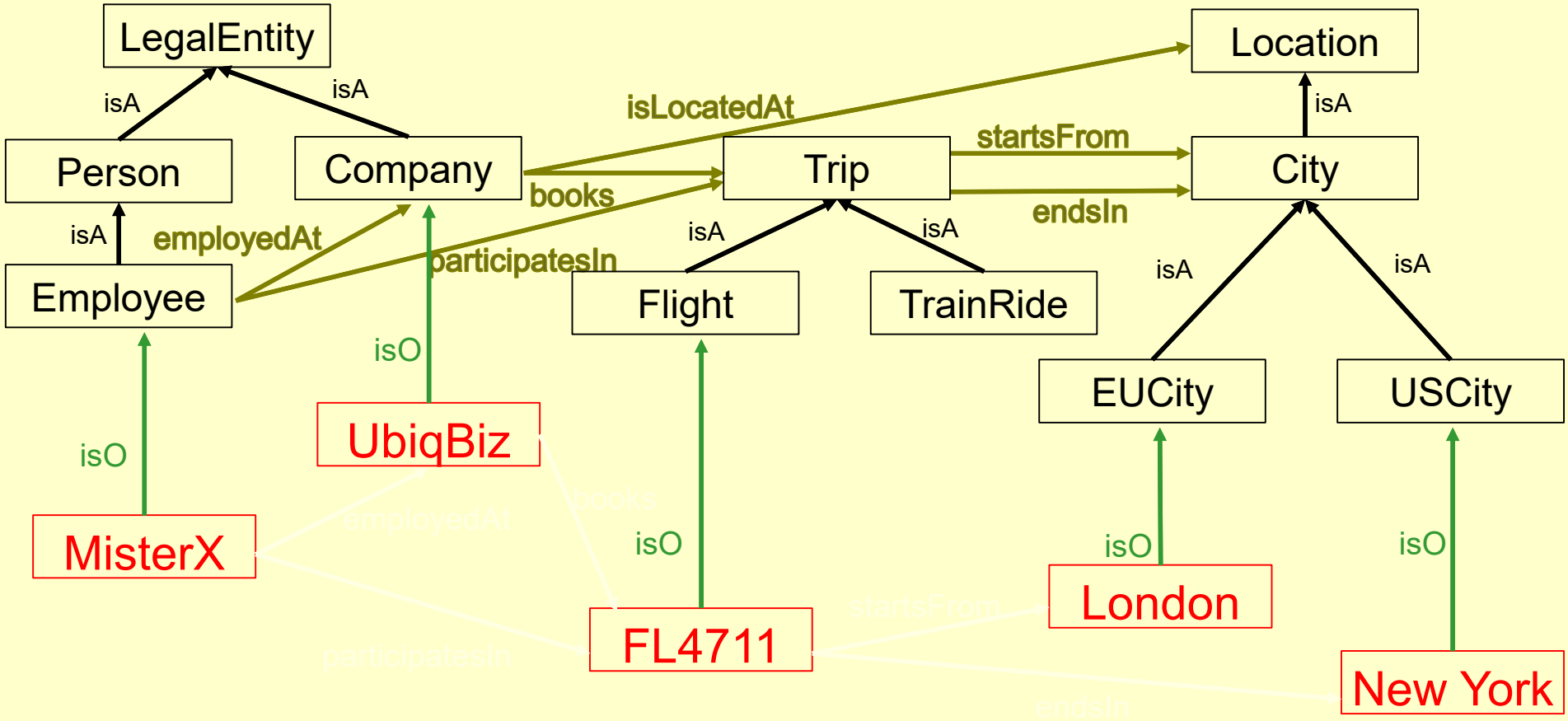
- Wieviel implizites Wissen kann gewonnen werden?



Typen von Ontologien

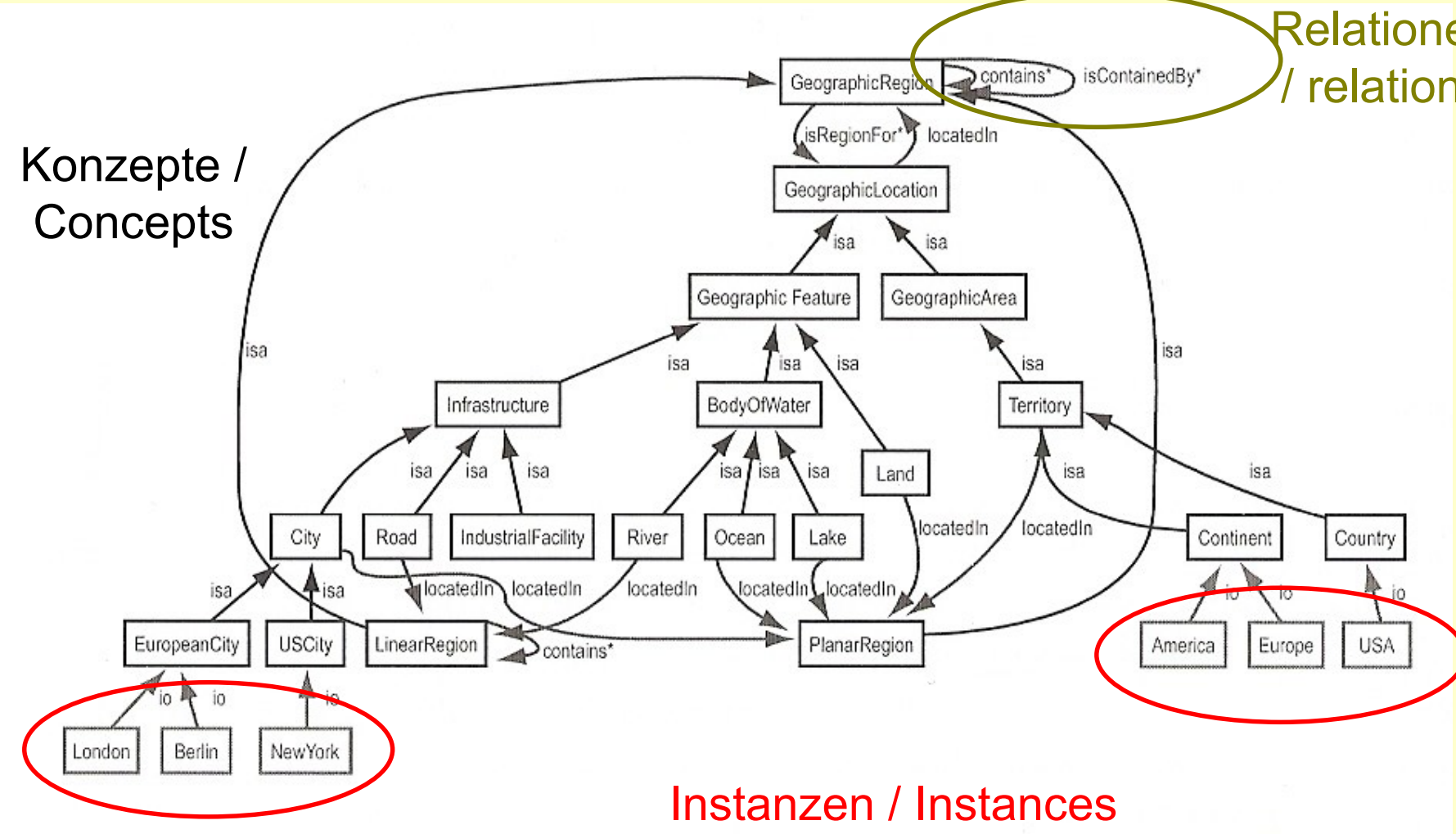
Ontologie-Beispiel für Reiseerfassung

Eine Ontologie besteht aus Konzepten, Relationen und Instanzen.

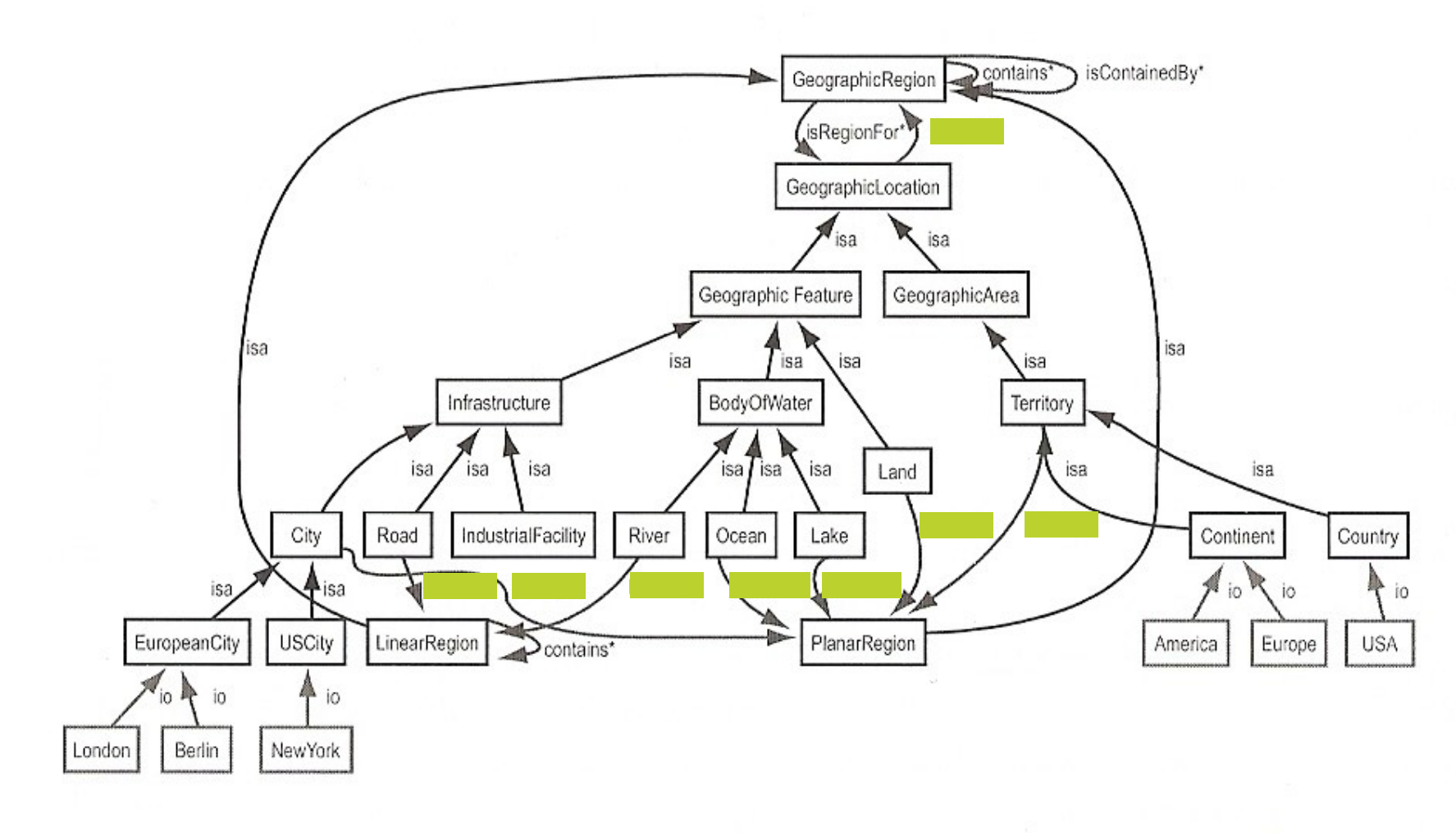


Beispiel entlehnt aus: Studer, Rudi; Grimm, Stefan; Abecker, Andreas: Semantic Web Services, Springer 2007

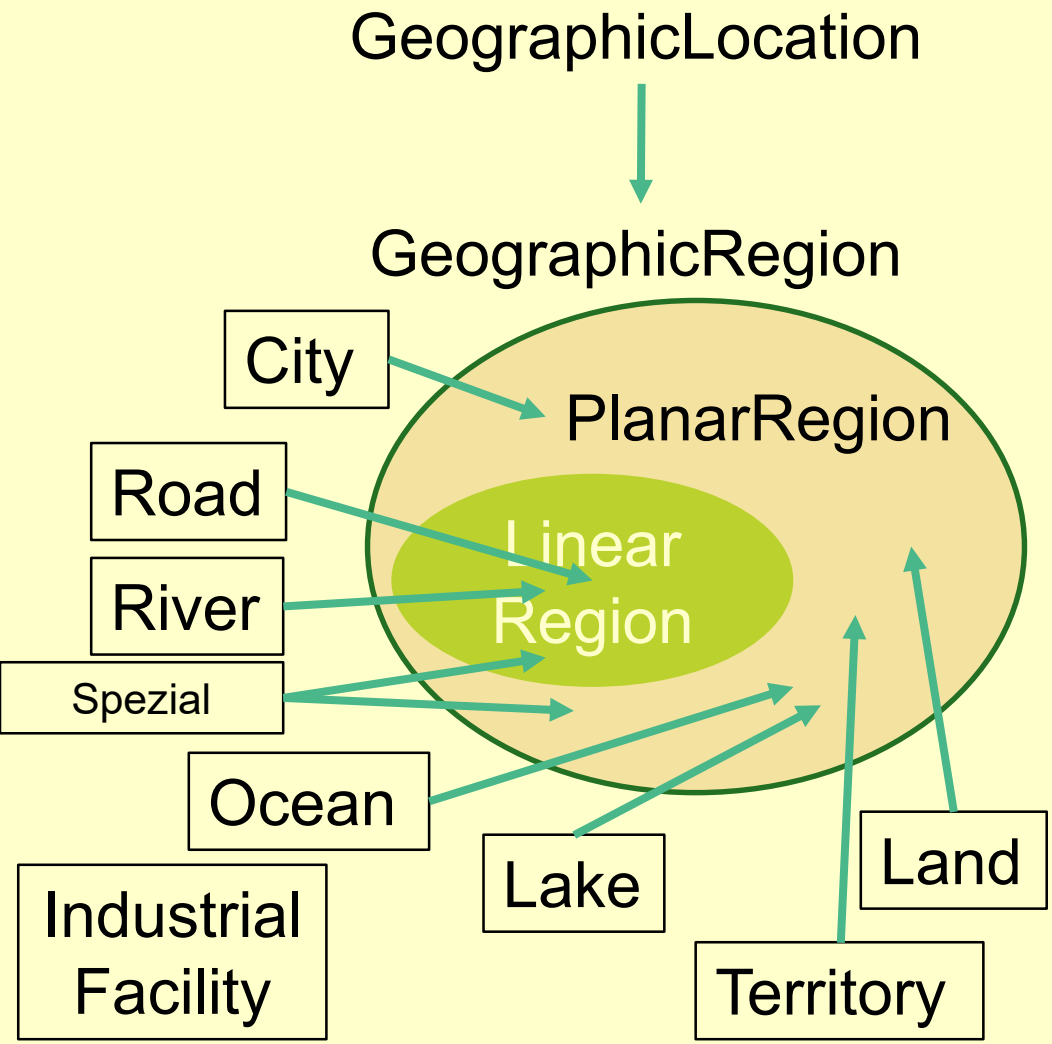
Ontologie-Beispiel Erweiterung für geografische Begriffe



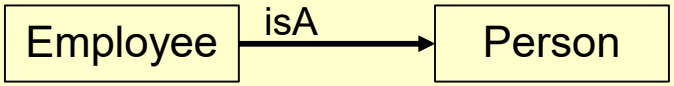
Ontologie-Beispiel Erweiterung für geografische Begriffe



Relation locatedIn

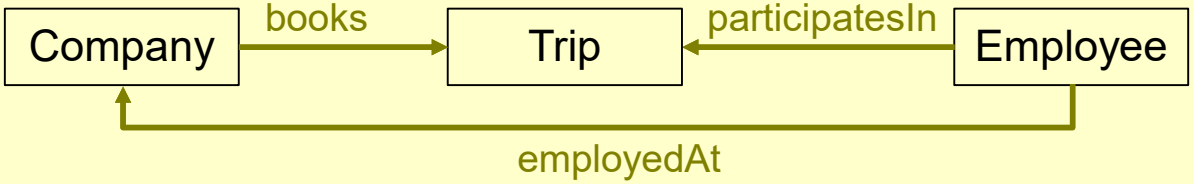


Formulierungen in First-Order-Logik (FOL)



$$\forall\, x : (Employee(x) \rightarrow Person(x))$$

$$\forall x, y : \big(books(x, y) \rightarrow Company(x) \wedge Trip(y)\big)$$
$$\forall x\colon \exists y\colon (Trip(x) \rightarrow Company(y) \wedge \text{books(y,x)})$$



Zusätzliche Restriktion

$$\forall x, z\colon \exists y\colon : (Trip(x) \rightarrow Employee(y) \wedge \text{participatesIn(y,x)} \wedge \text{employedAt(y,z)} \wedge \text{books(z,x)})$$

IF Start- und Zielort eines Trips nah beieinander sind
THEN der Trip ist per Zug

Zusätzliche Regel

$$\forall x, y, z\colon (\text{Trip(x)} \wedge startsFrom(x, y) \wedge \text{endsIn(x,}z) \wedge \text{close(y,z)} \rightarrow \text{TrainRide(x)})$$

Die Grafen-Notation von Semantischen Netzwerken ist durch Description Logik (DL) formalisiert worden.

Reasoning mit Knowledge Base (KB)

$$KB = \left\{ \begin{array}{c} Person(MisterX), participates(MisterX, FL4711), Flight(FL7411), books(UbiqBiz, FL4711), \\ \forall x, y, z: (Flight(y) \wedge participates(x, y) \wedge books(z, y) \rightarrow employedAt(x, z)) \\ \forall x, y: (employedAt(x, y) \rightarrow Company(y) \wedge Employee(x)) \\ \forall x: (Person(x) \rightarrow \neg Company(x)) \end{array} \right\}$$

Ableitung impliziter Informationen von der KB:

$ask(KB, employedAt(MisterX, UbiqBiz))$ $\quad KB \vDash employedAt(MisterX, UbiqBiz)$

$ask(KB, Company(UbiqBiz))$ $\quad KB \vDash Company(UbiqBiz)$

Open World Assumption (OWA) vs. Closed World Assumption (CWA)	
Decidability	Algorithmus antwortet immer richtig nach einer gegebenen Zeit
Monotones vs. Nicht-Monotones-Schließen	
Logiken höherer Ordnung für Metamodellierungen von Konzepten	
Unsicherheit in einer Logik, z.B. Fuzzy-Logik	

Description Logic (DL) ($\mathcal{ALC}$-Version)

- **Einteilung der Logik in zwei Teile**
  - Klassifikation der (bezeichneten und anonymen) Konzepte (T-Box)
  - Kategorisierung der Instanzen (A-Box)

Description Logic (DL) ($\mathcal{ALC}$-Version)

- **Einteilung der Logik in zwei Teile**
  - Klassifikation der (bezeichneten und anonymen) Konzepte (T-Box)
  - Kategorisierung der Instanzen (A-Box)
- **Operatoren auf Konzepte in der $\mathcal{ALC}$-Version von DL:**

$\sqcap$, Konjunktion	$\sqsubseteq$, Inklusion
$\sqcup$, Disjunktion	$\sqsupseteq$, Inverse Inklusion
$\neg$, Negation (unär)	$\equiv$, Äquivalenz
$\bot \equiv C \sqcap \neg C$, Leeres Konzept	$\top \equiv C \sqcup \neg C$, Globales Konzept

Description Logic (DL) ($\mathcal{ALC}$-Version)

- **Einteilung der Logik in zwei Teile**
  - Klassifikation der (bezeichneten und anonymen) Konzepte (T-Box)
  - Kategorisierung der Instanzen (A-Box)
- **Operatoren auf Konzepte in der $\mathcal{ALC}$-Version von DL:**

$\sqcap$, Konjunktion	$\sqsubseteq$, Inklusion
$\sqcup$, Disjunktion	$\sqsupseteq$, Inverse Inklusion
$\neg$, Negation (unär)	$\equiv$, Äquivalenz
$\bot \equiv C \sqcap \neg C$, Leeres Konzept	$\top \equiv C \sqcup \neg C$, Globales Konzept

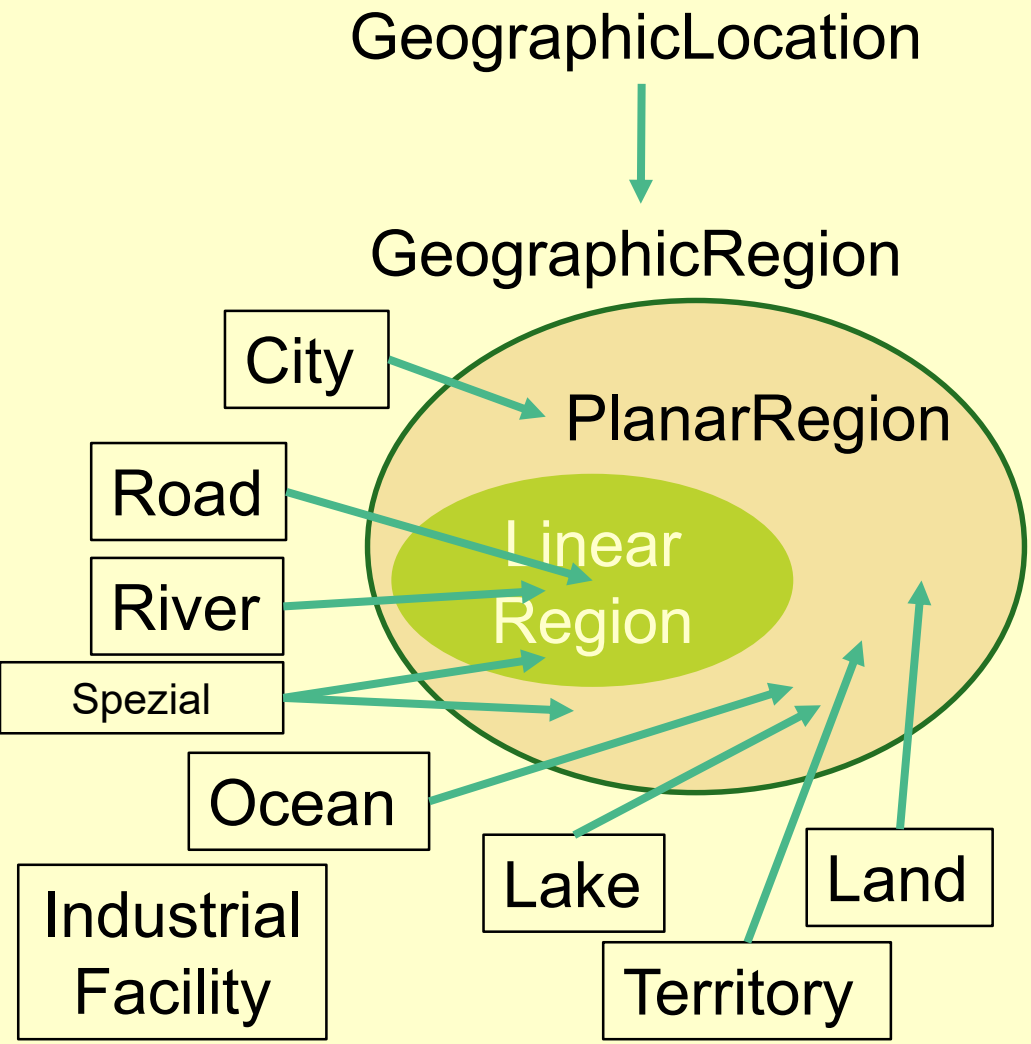
- **Restriktive Nutzung von Quantoren:**

Geg. sei eine Rolle r (vergleichbar einer Relation) und ein bezeichnetes Konzept C (C kann auch das leere oder globale Konzept sein oder anonym erzeugt werden)

$\forall r.C$	$\forall y : (r(x,y) \rightarrow C(y))$ resultiert im Model zu einem **anonymen Konzept** über die freie Variable x, das nach **OWA** alle Instanzen erfasst, die auf C abbilden
$\exists r.C$	$\exists y : (r(x,y) \wedge C(y))$, das resultierende anonyme Konzept enthält alle Instanzen, die wirklich echt auf C abbilden.

Einführung der Rollen in Description Logik

$\forall r.C$	$\forall y:\ (r(x,y) \rightarrow C(y)\ )$
$\exists r.C$	$\exists y:\ (r(x,y) \wedge C(y)\ )$



$\exists locatedIn.\top \sqsubseteq GeographicLocation$
$\top \sqsubseteq \forall locatedIn.GeographicRegion$
Ausgangsmenge und Zielmenge

$X \sqsubseteq \forall locatedIn.PlanarRegion$
$X = \{x \mid \forall y:\ (locatedIn(x,y) \rightarrow PlanarRegion(y)\ )\}$
$X = \{City, Ocean, Lake,$
$Land, Territory, Industrial Facility\}$

$X \sqsubseteq \exists locatedIn.PlanarRegion$
$X = \{x \mid \exists y:\ (locatedIn(x,y) \wedge PlanarRegion(y)\ )\}$
$X = \{City, Ocean, Lake,$
$Land, Territory, Spezial\}$

Description Logic (DL) ($\mathcal{ALC}$-Version)

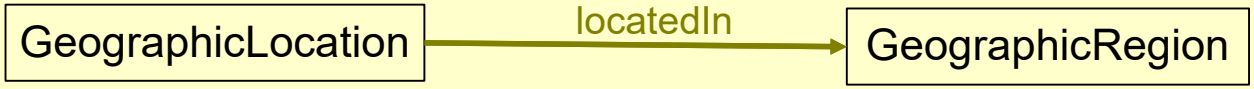
$\sqcap$, Konjunktion	$\sqsubseteq$, Inklusion
$\sqcup$, Disjunktion	$\sqsupseteq$, Inverse Inklusion
$\neg$, Negation (unär)	$\equiv$, Äquivalenz
$\bot \equiv C \sqcap \neg C$, Leeres Konzept	$\top \equiv C \sqcup \neg C$, Globales Konzept

T-Box Aussagen über Konzepte
A-Box Aussagen über Instanzen

$\forall r.C$	$\forall y : (r(x,y) \rightarrow C(y))$
$\exists r.C$	$\exists y : (r(x,y) \wedge C(y))$

Die Knowledge Base bei DL besteht aus Aussagen in der T-Box und der A-Box

T-Box:



$\exists locatedIn.\top \sqsubseteq GeographicLocation$ Festhalten der Ausgangsmenge

$\top \sqsubseteq \forall locatedIn.GeographicRegion$ Festhalten der Zielmenge

$Continent \sqsubseteq GeographicLocation$ isA-Relation

$City \sqsubseteq GeographicLocation \sqcap \forall locatedIn.PlanarRegion$

$Trip \sqsubseteq \exists bookedBy.(Company \sqcup Person)$

A-Box:

$Continent(Europe)$
$Flight(FL4711)$
$bookedBy(FL4711, UbiqBiz)$

Description Logic (DL) ($\mathcal{ALC}$-Version erweitert)

$GeographicLocation \sqsubseteq \ =1locatedIn$	Rolle als Funktion durch nummerische Restriktion erlaubt folgende Aquivalenz: $\exists r \equiv \geq 1r$
$isRegionFor \equiv locatedIn^-$	Rolle definiert als inverse einer anderen Rolle
$r \sqsubseteq s$	hierarchische Anordnung von Rollen weitere Eigenschaften von Rollen (z.B. transitiv)

Eine europäische Stadt muss in einer Region mit dem Kontinent Europa liegen

$$EuropeanCity \equiv City \sqcap \forall locatedIn.\exists isContainedBy.\exists isRegionFor.\{Europe\}$$

In FOL:
$$\forall x: \left(EuropeanCity(x) \leftrightarrow City(x) \wedge \forall y: \left(locatedIn(x,y) \rightarrow \exists z: \left(isContainedBy(y,z) \wedge isRegionFor(z, Europe) \right) \right) \right)$$

$$EuropeanCity \equiv City \sqcap \forall locatedIn.\exists contains^-.\exists locatedIn^-.\{Europe\}$$

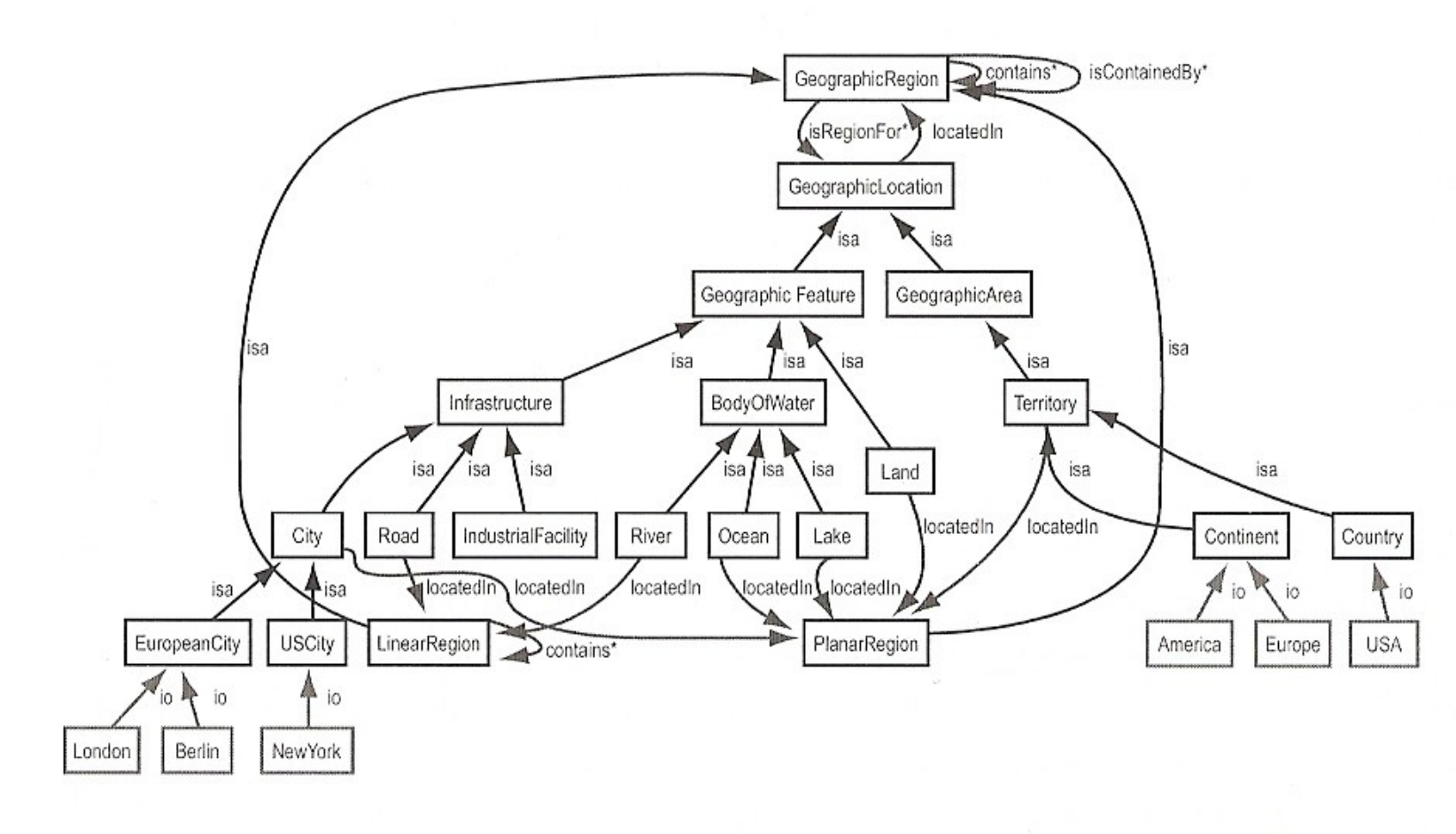
In FOL:
$$\forall x: \left(EuropeanCity(x) \leftrightarrow City(x) \wedge \forall y: \left(locatedIn(x,y) \rightarrow \exists z: \left(contains(z,y) \wedge locatedIn(Europe,z) \right) \right) \right)$$

DL Umsetzung in OWL-S

OWL abstract syntax	DL syntax
Axioms	
`Class(A partial` $C_1 \ldots C_n$`)`	$A \sqsubseteq C_1 \sqcap \ldots C_n$
`Class(A complete` $C_1 \ldots C_n$`)`	$A \equiv C_1 \sqcap \ldots C_n$
`EnumeratedClass(A` $a_1 \ldots a_n$`)`	$A \equiv \{a_1\} \sqcup \cdots \sqcup \{a_n\}$
`SubClassOf(C D)`	$C \sqsubseteq D$
`EquivalentClasses(`$C_1 \ldots C_n$`)`	$C_1 \equiv \cdots \equiv C_n$
`DisjointClasses(`$C_1 \ldots C_n$`)`	$C_i \sqsubseteq \neg C_j, (1 \leq i < j \leq n)$
`ObjectProperty(`r `super(`r_1`)...super(`r_n`)`	$r \sqsubseteq r_1 \sqcap \cdots \sqcap r_n$
`domain(`C_1`) ... domain(`C_n`)`	$\exists r.\top \sqsubseteq C_1 \sqcap \cdots \sqcap C_n$
`range(`C_1`) ... range(`C_n`)`	$\top \sqsubseteq \forall r.C_1 \sqcap \cdots \sqcap \forall r.C_n$
`[inverseOf(`s`)]`	$r \equiv s^-$
`[Symmetric]`	$r \equiv r^-$
`[Functional]`	$\top \sqsubseteq \leq 1\, r$
`[InverseFunctional]`	$\top \sqsubseteq \leq 1\, r^-$
`[Transitive])`	Trans(r)
`SubPropertyOf(`r s`)`	$r \sqsubseteq s$
`EquivalentProperties(`$r_1 \ldots r_n$`)`	$r_1 \equiv \cdots \equiv r_n$
`Individual(`a `type(`C_1`) ... type(`C_n`)`	$C_1 \sqcap \cdots \sqcap C_n(a)$
`value(`r_1 a_1`) ... value(`r_n a_n`))`	$r_1(a, a_1), \ldots, r_n(a, a_n)$
`SameIndividual(`$a_1 \ldots a_n$`)`	$a_1 = \cdots = a_n$
`DifferentIndividuals(`$a_1 \ldots a_n$`)`	$a_i \neq a_j, (1 \leq i < j \leq n)$
Descriptions	
`Class(`A`)`	A
`Class(owl:Thing)`	$\top$
`Class(owl:Nothing)`	$\bot$
`intersectionOf(`C_1 C_2 `...)`	$C_1 \sqcap C_2$
`unionOf(`C_1 C_2 `...)`	$C_1 \sqcup C_2$
`complementOf(`C`)`	$\neg C$
`oneOf(`a_1 a_2 `...)`	$\{a_1\} \sqcup \{a_2\}$
`restriction(`r `someValuesFrom(`C`))`	$\exists r.C$
`restriction(`r `allValuesFrom(`C`))`	$\forall r.C$
`restriction(`r `hasValue(`a`))`	$\exists r.\{a\}$
`restriction(`r `minCardinality(`n`))`	$\geq n\, r$
`restriction(`r `maxCardinality(`n`))`	$\leq n\, r$

Erstellen und Arbeiten mit der geografischen Ontologie

Geografische Ontologie (erneut)



Erstellen der Ontologie in OWL und DL (1/4)

① "*A continent is a geographic location different from a country.*"	
`Class(`*Continent* `partial` `intersectionOf(`*GeographicLocation* `complementOf(`*Country*`))`	*Continent* $\sqsubseteq$ *GeographicLocation* $\sqcap \neg$*Country*

② "*Europe is a particular continent.*"	
`Individual(`*Europe* `type(`*Continent*`))`	*Continent*(*Europe*)

③ "*The continents are America, Europe, Africa, Asia and Australia.*"	
`EnumeratedClass(`*Continent* *America Europe Africa Asia Australia*`)`	*Continent* $\equiv$ {*America*, *Europe*, *Africa*, *Asia*, *Australia*}

Erstellen der Ontologie in OWL und DL (2/4)

④ "*Geographic regions in general contain geographic regions.*"

```
ObjectProperty(contains
    domain(GeographicRegion)
    range(GeographicRegion)
    inverseOf(isContainedBy)
    Transitive)
```

$\exists\, contains.\top \sqsubseteq GeographicRegion$,
$\top \sqsubseteq \forall\, contains.GeographicRegion$,
$contains \equiv isContainedBy^{-}$
$\mathsf{Trans}(contains)$

⑤ "*A planar region only contains planar or linear regions.*"

```
SubClassOf(PlanarRegion
    restriction(contains
allValuesFrom(
        unionOf(PlanarRegion
LinearRegion)))
```

$PlanarRegion \sqsubseteq \forall\, contains.(PlanarRegion \sqcup LinearRegion)$

Erstellen der Ontologie in OWL und DL (3/4)

⑥ "*A European city is a city whose geographic region is contained in that of Europe.*"

```
EquivalentClasses(EuropeanCity
intersectionOf(
    City
    restriction(locatedIn
allValuesFrom(
     restriction(isContainedBy
someValuesFrom(
       restriction isRegionFor
someValuesFrom(
         oneOf(Europe)))))))
```

$$EuropeanCity \equiv City \sqcap \forall\, locatedIn.\exists\, isContainedBy.\exists\, isRegionFor.\{Europe\}$$

⑦ "*A city is a geographic location governed by a single country.*"

```
SubClassOf(City
    restriction(governedBy
maxCardinality 1))
```

$$City \sqsubseteq \leq 1\ governedBy$$

Erstellen der Ontologie in OWL und DL (4/4)

⑧ "*Munich is a German city with 1288307 inhabitants.*"	

`Individual(`*Munich* `type(`*City*`)`
`value(`*governedBy Germany*`)`
`value(`*numberOfInhabitants* `1288307))`
	City(*Munich*),
	governedBy(*Munich*, *Germany*)
	numberOfInhabitants(*Munich*, 1288307)

Anwendung der Ontologie (1/4)

GESELLSCHAFT FÜR INFORMATIK

Inkonsistenz einer Ontologie erkennen

⑦ "*A city is a geographic location governed by a single country.*"

```
SubClassOf(City
    restriction(governedBy
maxCardinality 1))
```

$City \sqsubseteq \leq 1\ governedBy$

$$Individual\left(\begin{matrix}\textbf{\textit{Nicosia}}\ type(City)\ value(governedBy, Greece)\\ value(governedBy, Turkey)\end{matrix}\right),$$

$$DifferentIndividuals(Greece, Turkey)$$

Mit diesen Regeln wird die KB inkonsistent.

Inkohärenz einer Ontologie erkennen

⑦ "*A city is a geographic location governed by a single country.*"

```
SubClassOf(City
    restriction(governedBy
maxCardinality 1))
```

$$City \sqsubseteq \leq 1\ governedBy$$

$$class\left(\begin{array}{c}\textbf{SplitCity}\ complete\ intersectionOf\\ \left(City\ restriction\left(governedBy\ minCardinality(2)\right)\right)\end{array}\right)$$

Mit dieser Regel wird die KB nur inkohärent.

Es entsteht kein Widerspruch zum Konzept *City*, wenngleich *SplitCity* leer bleibt.

Anfragen auf Ebene der Konzepte lösen (Querying for Subsumption)

$$class\left(\begin{array}{c}\boldsymbol{SplitCity}\ complete\ intersectionOf\\ \left(City\ restriction\left(governedBy\ minCardinality(2)\right)\right)\end{array}\right),$$

$$class\left(\begin{array}{c}\boldsymbol{GreekTurkishCity}\ partial\\ intersectionOf\left(\begin{array}{c}City\\ restriction\left(governedBy\ someValuesFrom\left(oneOf(Greece)\right)\right)\\ restriction\left(governedBy\ someValuesFrom\left(oneOf(Turkey)\right)\right)\end{array}\right)\end{array}\right),$$

$$DifferentIndividuals(Greece, Turkey)$$

Das Konzept ***City*** wird nicht eingeschränkt in Bezug auf die Anzahl der regierenden Länder.

Ein DL-Reasoner wird bei dieser KB automatisch erkennen:
$GreekTurkishCity \sqsubseteq SplitCity$

Anfragen auf Ebene der Instanzen lösen (Querying for Assertion)

$$subClassOf\left(\boldsymbol{EUCountry}\ restriction(officialCurrency\ hasValue(Euro))\right),$$

$$class\left(\begin{array}{c}\boldsymbol{GermanCity}\ partial \\ intersectionOf\left(\begin{array}{c}City \\ restriction\left(governedBy\ hasValue(Germany)\right)\end{array}\right)\end{array}\right),$$

$$Individual(Germany, EUCountry),$$
$$Individual(Munich, GermanCity)$$

Ein DL-Reasoner wird bei dieser KB automatisch für Instanzen erkennen:

$$Individual(Munich, value(governedBy\ Germany))$$

und

In München existiert die Restriktion, dass mit Euro gezahlt wird.

Er könnte auch alle weiteren Städte so erkennen, in denen mit Euro gezahlt wird.

Ende

GESELLSCHAFT FÜR INFORMATIK

Service Discovery / Trading

- Web Service Description Language (WSDL)
  erlaubt das Suchen aufgrund von Messages/RPCs
- Ausfallsicherheit
- UDDI veraltet, es gibt unterschiedliche Discovery / Trading Lösungen

Service Composition

- Kombination mit anderen Services
- Beschreibungen erlauben weitergehende Algorithmen

Einbeziehung der Semantik eines Services für genaueres Suchen => Semantic WebServices

- Auffinden anhand semantischer Kriterien oft mit Ontologien
- Möglichkeit, Semantische Web Service Beschreibungen zu nutzen:
  z.B. OWL-S, WSMO

WSDL Beispiel


```
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://test/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://person.webservice.fom.de/"
name="PersonServiceService">

[…]

</definitions>
```

WSDL-Struktur
Datentypen
Nachrichten
Port-Typen
Bindings
Services

```
<types>

  <xsd:schema>

        <xsd:import
        namespace="http://person.webservice.fom.de/"
        schemaLocation="http://localhost:8080/person?xsd=1"/>

</xsd:schema></types>
```

Entlehnt aus der Lehre an der Universität Duisburg-Essen

WSDL Beispiel

Nachrichten: Request und Response getrennt beschrieben

```
<message name="getPersonByName">
        <part name="arg0" type="xsd:string"/>
</message>
<message name="getPersonByNameResponse">
        <part name="return" type="tns:person"/>
</message>
```

WSDL-Struktur
Datentypen
Nachrichten
Port-Typen
Bindings
Services

Port-Typen: Die (statische) Schnittstellenbeschreibung

```
<portType name="PersonService">
       <operation name="getPersonByName">
              <input message="tns:getPersonByName"/>
              <output message="tns:getPersonByNameResponse"/>
       </operation>
</portType>
```

WSDL Beispiel

Bindung: Beschreibt den Transport der Nachrichten

(Hier: SOAP über HTTP als RPC)

```
<binding name="PersonServicePortBinding" type="tns:PersonService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
        <operation name="getPersonByName">
                […]
                <input><soap:body use="literal"
namespace="http://person.webservice.fom.de/"/></input>
                <output> <soap:body use="literal"
namespace="http://person.webservice.fom.de/"/></output>
        </operation>
</binding>
```

WSDL-Struktur
Datentypen
Nachrichten
Port-Typen
Bindings
Services

Services: Bieten Ports (Instanzen der Schnittstellen) an einer Adresse an

```
<service name="PersonServiceService">
        <port name="PersonServicePort" binding="tns:PersonServicePortBinding">
                <soap:address location="http://localhost:8080/person"/>
        </port>
</service>
```

Java Beispiel - Service Provider (1/2)

```
public class PersonBean
{
        public Person getPersonByName(String name)
        {
                return new Person(name);
        }
}
```

Service

```
public class Person
{
        private String name;

        public String getName()
        {
                return name;
        }

        public Date getDateOfBirth()
        {
                return new Date();
        }
        [...]   // Konstruktoren
}
```

Objekt mit Nutzdaten

Java Beispiel - Service Provider (2/2)

Libraries aus dem JDK machen das Leben leicht

```
@WebService
@SOAPBinding(style = Style.RPC)
public class PersonService
{
        public Person getPersonByName(String name)
        {
                return new Person(name);
        }
}
```

Web Service als RPC

POJO (Plain Old Java Object)

```
public class PersonServiceServer {
        public static void main(String[] args) {
                PersonService server = new PersonService();
                Endpoint endpoint =
                    Endpoint.publish("http://localhost:8080/person",
                                server);
        }
}
```

Veröffentlichung

GESELLSCHAFT FÜR INFORMATIK

Aufbau einer SOAP-Nachricht

- Nachrichten ebenfalls XML-basiert
- Nutzdaten der Nachrichten verpackt in „Umschlägen“
- Zuordnung der Nutzdaten zu Definitionen aus WSDL
- Technik dazu: Simple Object Access Protocoll (SOAP)

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:personservice="http://person.webservice.fom.de/">
   <soapenv:Header/>
   <soapenv:Body>
      <personservice:getPersonByName>
         <personservice:arg_0>MeinName</personservice:arg_0>
      </personservice:getPersonByName>
   </soapenv:Body>
</soapenv:Envelope>
```

GESELLSCHAFT FÜR INFORMATIK

SOAP-Nachrichten durch einen Stub erzeugen

wsimport -keep http://localhost:8080/person?wsdl

Erzeugt:

PersonService.java (Das Interface)

PersonServiceService.java (Der Service-Agent des Clients)

Nutzung:

```
public class PersonServiceClient {
    public static void main(String args[]) {
        PersonServiceService serviceDesc = new PersonServiceService();
        PersonService service = serviceDesc.getPersonServicePort();
        System.out.println(service.getPersonByName(„Peter“));
    }
}
```

OWL - Beispiel

```
...
<owl:Class rdf:ID="City">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#locatedIn"/>
            <owl:allValuesFrom rdf:resource="#PlanarRegion"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Infrastructure"/>
    <owl:disjointWith rdf:resource="#Road"/>
    <owl:disjointWith rdf:resource="#IndustrialFacility"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#GeographicLocation"/>
    <rdfs:range rdf:resource="#GeographicRegion"/>
    <owl:inverseOf rdf:resource="#isRegionFor"/>
</owl:ObjectProperty>
<EuropeanCity rdf:ID="London"/>
...
```

Konzept

Relation

Instanz

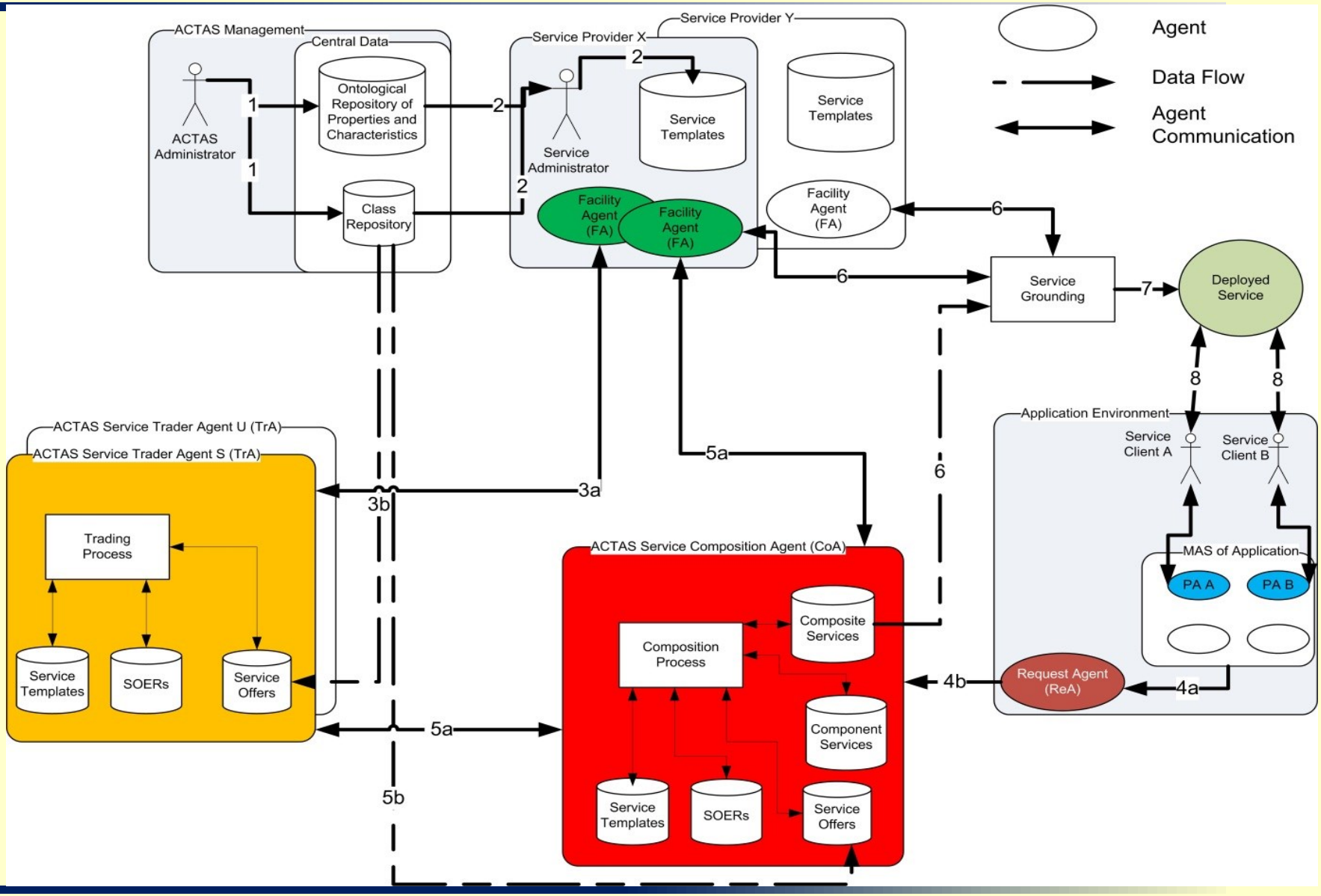
WSML - Beispiel

```
concept GeographicRegion
    isRegionFor inverseOf(locatedIn) ofType GeographicLocation
    contains inverseOf(isContainedBy) transitive impliesType GeographicRegion
    boundedBy ofType (2 *) SurfacePoint

concept SurfacePoint
    hasLongtitude ofType _float
    hasLatitude ofType _float

concept City subConceptOf Infrastructure
    locatedIn ofType PlanarRegion
    officialName ofType _string
    numberOfInhabitants ofType _integer

concept EuropeanCity subConceptOf City

instance Europe memberOf Continent

instance Munich memberOf City
    officialName hasValue "M ünchen"
    numberOfInhabitants hasValue 1288307

axiom EuropeanCity_sufficient_condition definedBy
    ?c memberOf EuropeanCity :– ?c memberOf City and
                ?c[locatedIn hasValue ?rc] and
                ?rc[containedBy hasValue ?re] and
                ?re[isRegionFor hasValue Europe].
```

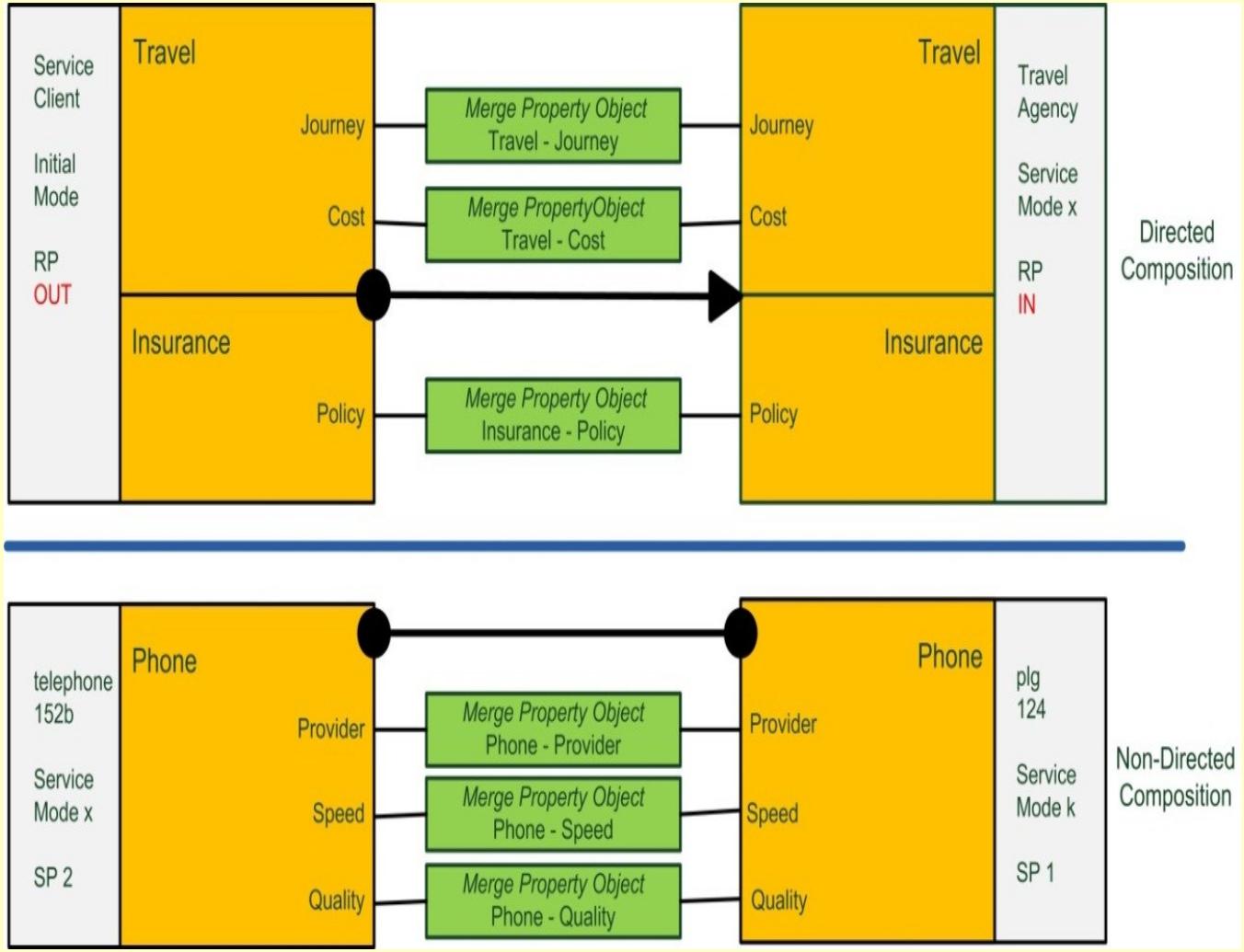
- Konzepte mit Eigenschaften
- Relationen sind Eigenschaften
- Kategorisierung von Instanzen
- Restriktionen, aber auch automatische Kategorisierung mit Axiomen

Axiom arbeitet auf Instanzen

System Environment



Gastvortrag SE / Abschluss und Idee aus der Dissertation



Abschluss

Principal Compatibility and Merge Constraint (Me-Co)