

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering

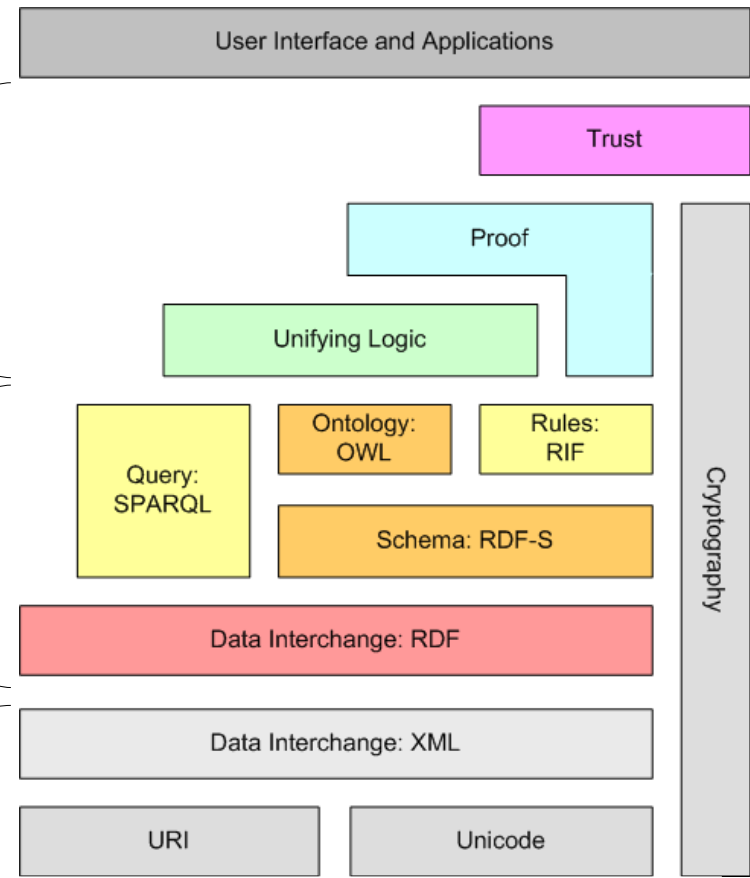
Semantic Web – Aufbau



here be dragons...

Semantic-Web-
Technologie
(Fokus der Vorlesung)

Technische
Grundlagen



Berners-Lee (2009): *Semantic Web and Linked Data*
<http://www.w3.org/2009/Talks/0120-campus-party-tbl/>

Wie baut man Ontologien?

- Wie wir bisher vorgegangen sind
 - Anforderungen durchlesen
 - einfach mal irgendwo anfangen
 - in Protégé drauflosbasteln
 - überraschen lassen, was dabei herauskommt
- Das war eher "Ontology Hacking" als "Ontology Engineering"

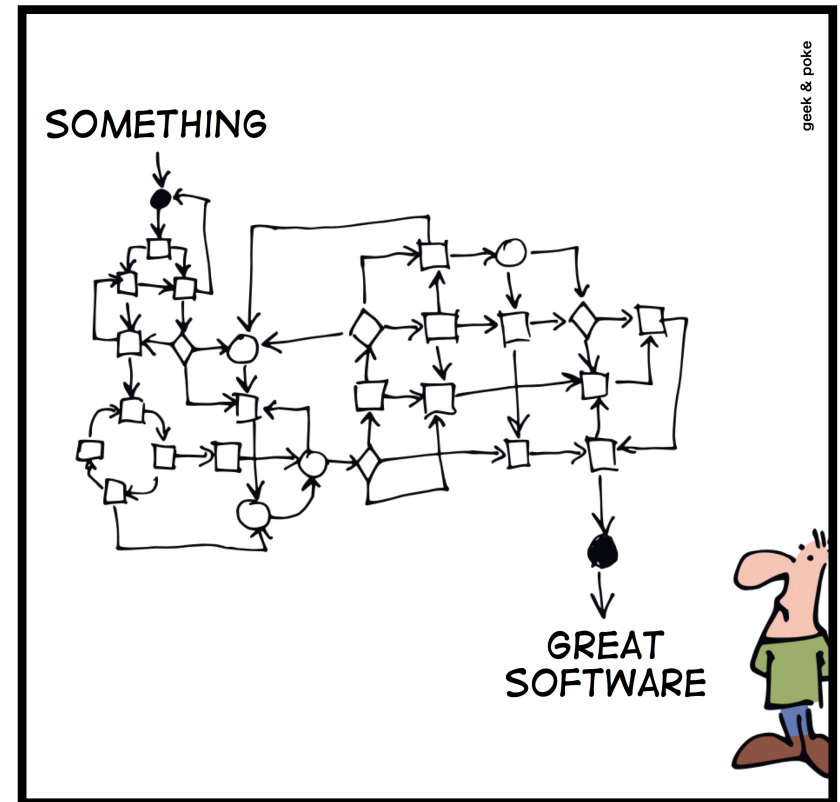


- Wie baut man Ontologien?
 - Vorgehensmodelle
- Wie baut man *gute* Ontologien?
 - Best Practices
 - Design Patterns
 - Anti-Patterns

Vorgehensmodelle

- Kennen wir aus dem Software Engineering

SIMPLY EXPLAINED



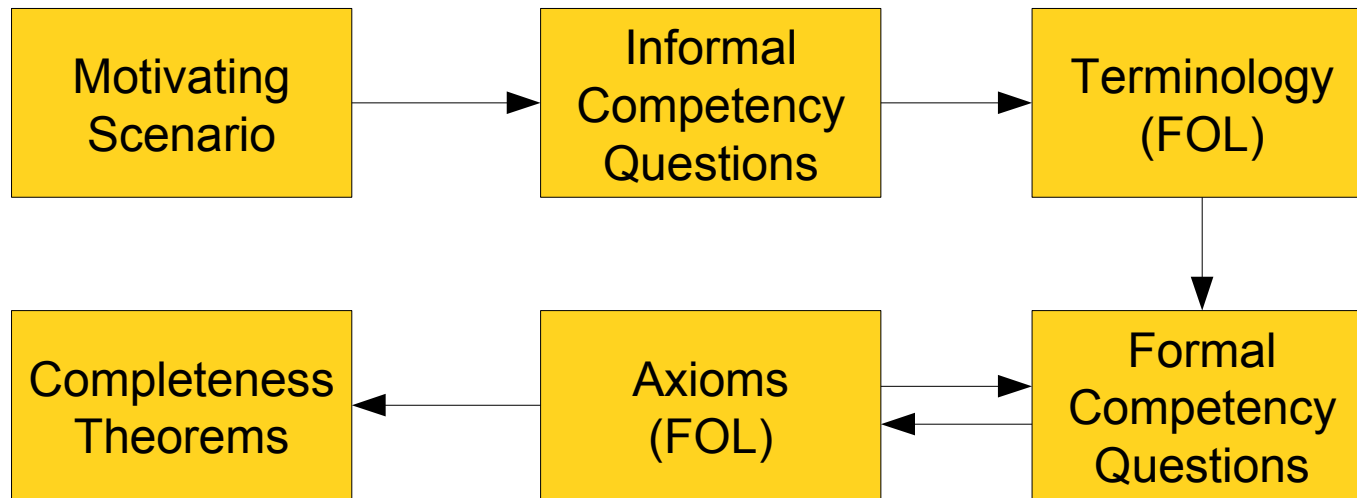
<http://geekandpoke.typepad.com/geekandpoke/2012/01/simply-explained-dp.html>

DEVELOPMENT PROCESS

Methodologie von Grüninger und Fox (1995)



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Grüninger und Fox (1995): Methodology for the Design and Evaluation of Ontologies

Methodologie von Grüninger und Fox (1995)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Motivating Scenario
 - informelle Beschreibung der Inhalte der Ontologie
 - Problemstellungen
 - möglicherweise intuitive Lösungswege

- Informal Competency Questions
 - Fragen, die später beantwortet werden können sollen
 - z.B. durch einen Reasoner
 - "Was ist die Hauptstadt von Spanien"?
 - "Welches Hobby hat die Person im blauen Haus?"
 - Auch: Einschätzung, ob eine bestehende Ontologie geeignet ist

Methodologie von Grüninger und Fox (1995)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Terminologie (FOL)
 - Definition von Klassen und Relationen
 - Nutzung einer formalen Repräsentation
 - Grüninger und Fox: First Order Logic
 - Heute: RDF(S), OWL, ...
- Formal Competency Questions
 - Übersetzung der informellen Fragen
 - z.B. nach SPARQL

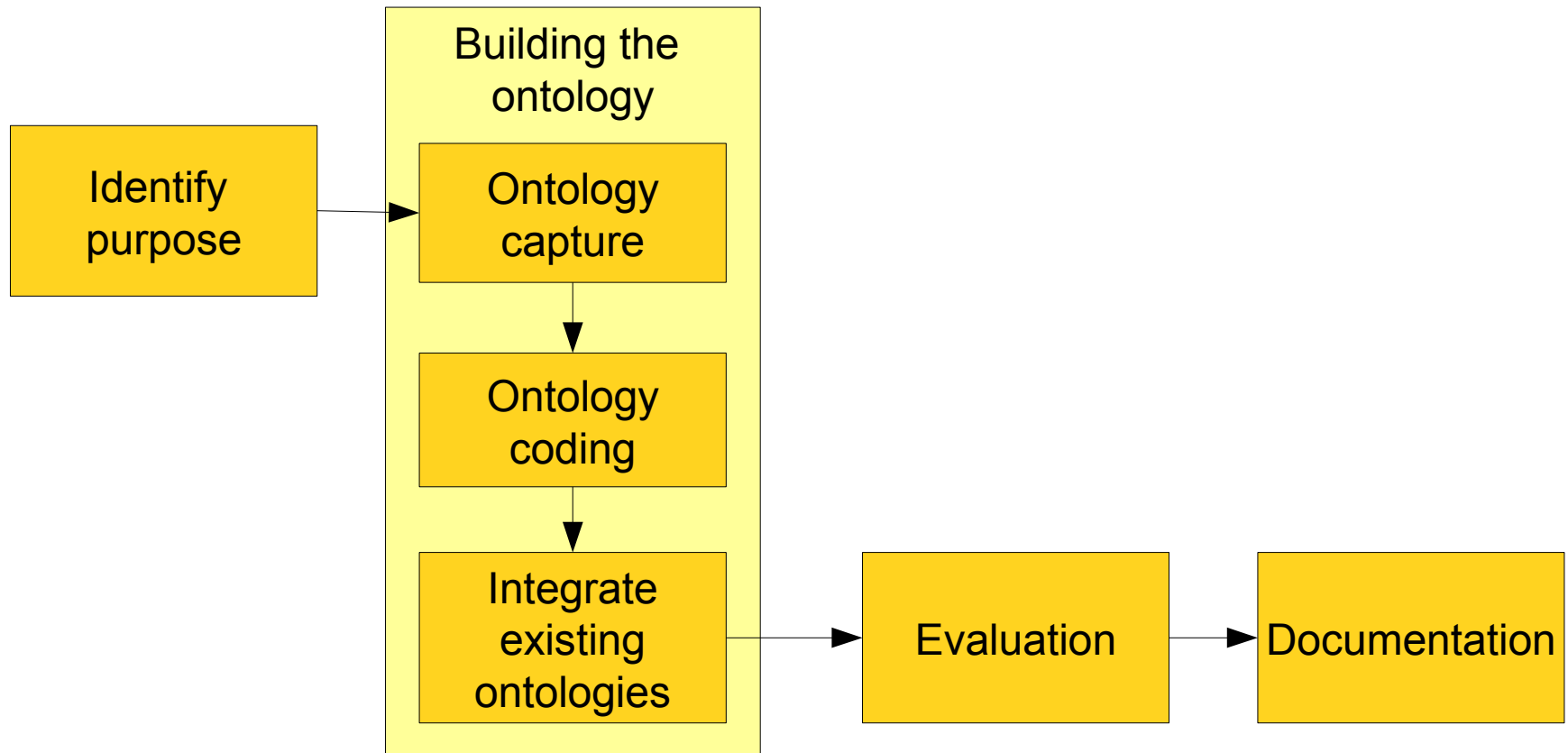
Methodologie von Grüninger und Fox (1995)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Axioms (FOL)
 - stärkere Axiomatisierung der Ontologie
 - Grüninger und Fox: First Order Logic
 - Heute: OWL-Axiome
 - Domain/Range
 - Restrictions
 - Funktionale/invers funktionale, transitive, ..., Relationen
 - Solange, bis die formalen Abfragen funktionieren
- Completeness Theorems
 - geben an, inwieweit die Folgerungen vollständig sind
 - d.h., gibt es auch unentscheidbare Aussagen?
 - "Wer rasiert den Barbier?"

Methodologie von Uschold und King (1995)



Uschold & King (1995): Towards a Methodology for Building Ontologies

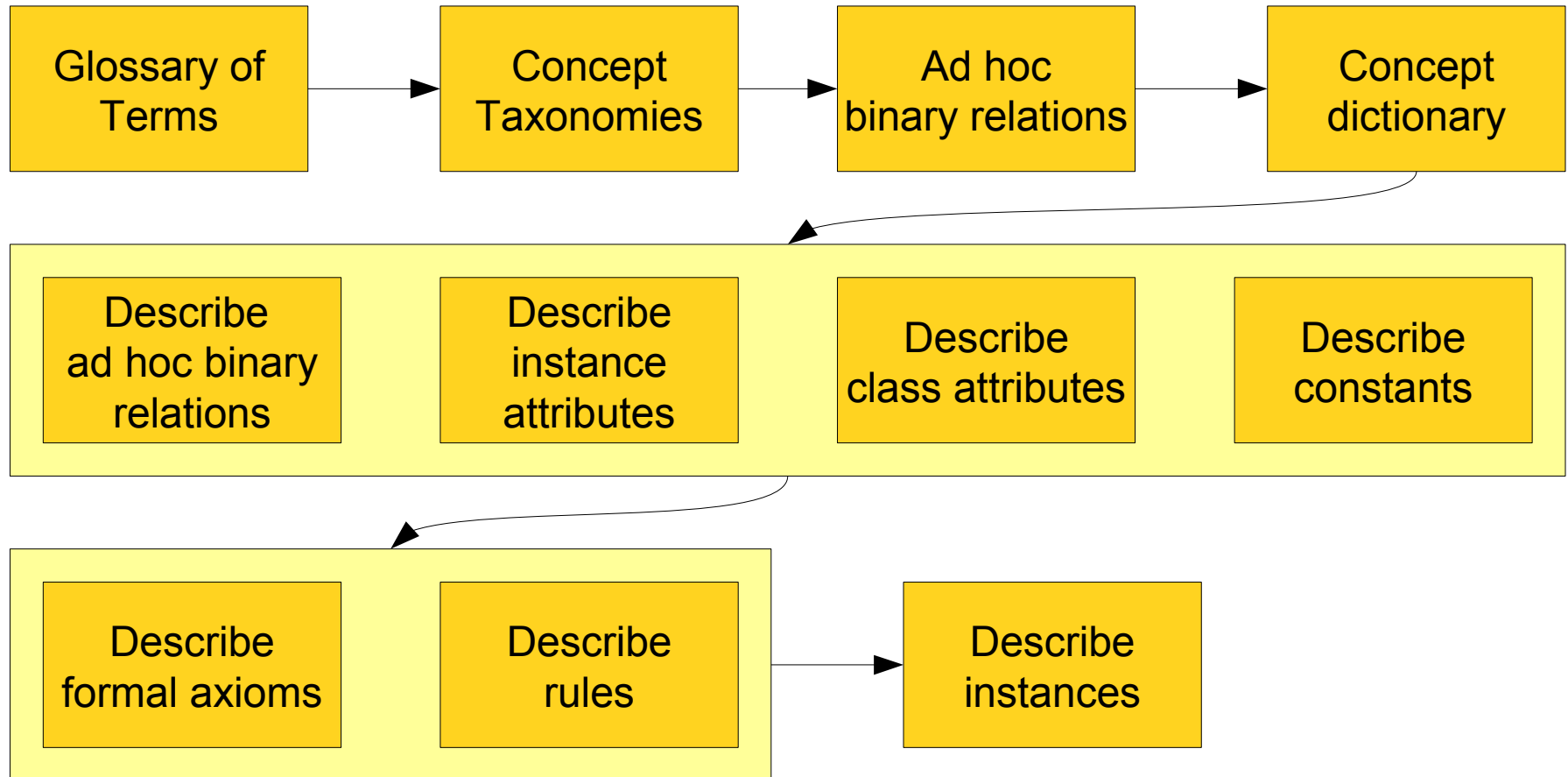
Methodologie von Uschold und King (1995)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Ähnlich zu Grüninger und Fox
- Capture und Coding werden getrennt
 - Capture: Zentrale Begriffe suchen und textuell definieren
 - Coding: formale Codierung
- Integration bestehender Ontologien als eigener Punkt
- Evaluierung
 - z.B. mit Competency Questions
 - Abgleich mit Requirements-Spezifikationen
 - ...

Methontology (Fernández et al., 1997)



Gómez-Pérez et al. (2004): Ontological Engineering

Methontology (Fernández et al., 1997)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Schrittweise von informellen zu formellen Ontologien
- Rückschritte sind erlaubt
- Dokumentation begleitend

- Glossar
 - Begriffe, Beschreibungen, Synonyme, Antonyme
- Taxonomie
 - Subklassenbeziehung
- Ad hoc binary relations
 - a.k.a. ObjectProperties
- Concept dictionary
 - enthält: Begriffe, Beschreibungen, Relationen, optional Instanzen

Methontology (Fernández et al., 1997)

- Beispiel für Concept dictionary

Concept name	Class attributes	Instance attributes	Relations
AA7462	--	--	same Flight as
American Airlines Flight	company Name	--	--
British Airways Flight	company Name	--	--
Five-star Hotel	number of Stars	--	--
Flight	--	--	same Flight as
Location	--	name size	is Arrival Place of is Departure Place of
Lodging	--	price of Standard Room	placed in
Travel	--	arrival Date company Name departure Date return Fare single Fare	arrival Place departure Place
Travel Package	--	budget final Price name number of Days travel Restrictions	arrival Place departure Place accommodated in travels in
USA Location	--	--	--

Gómez-Pérez et al. (2004): Ontological Engineering

Methontology (Fernández et al., 1997)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Detaillierte Beschreibungen
- Relationen:
 - Definitions-/Wertebereich, Kardinalitäten, Symmetrie, Transitivität...
- Instanz- und Klassenattribute, Konstanten
 - Datentyp, Einheit, Wertebereich...
 - Konstanten: z.B. Siedepunkt von Wasser

Methontology (Fernández et al., 1997)



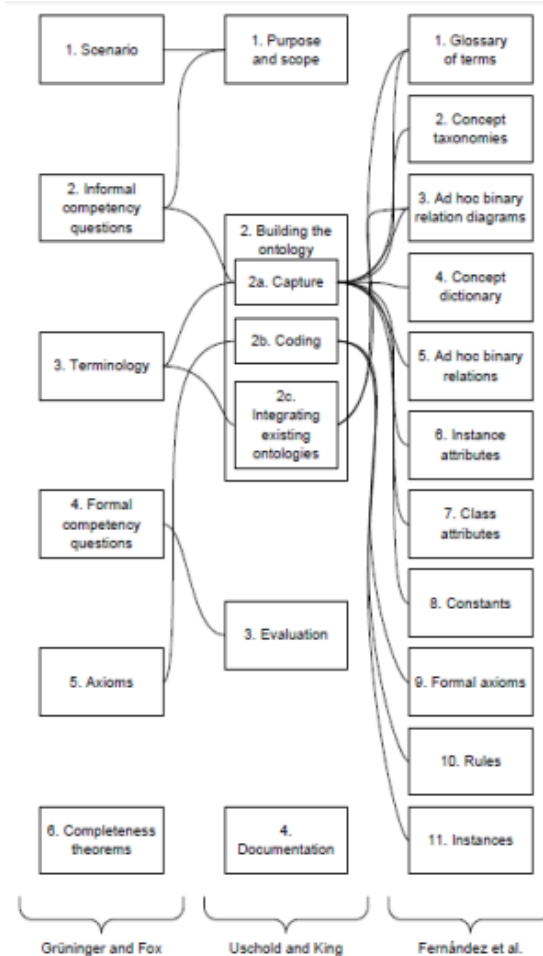
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Formale Axiome und Regeln
 - Formalisierung der textuellen Beschreibungen
 - z.B.: Europäische Hauptstädte:
 - $\text{Stadt}(x) \wedge \text{hauptstadtVon}(x,y) \wedge \text{liegtIn}(x,\text{Europa})$
- Beschreibung von Instanzen
 - Tabelle von Instanzen
 - mit Werten von allen Instanzattributen

Vergleich der Methodologien



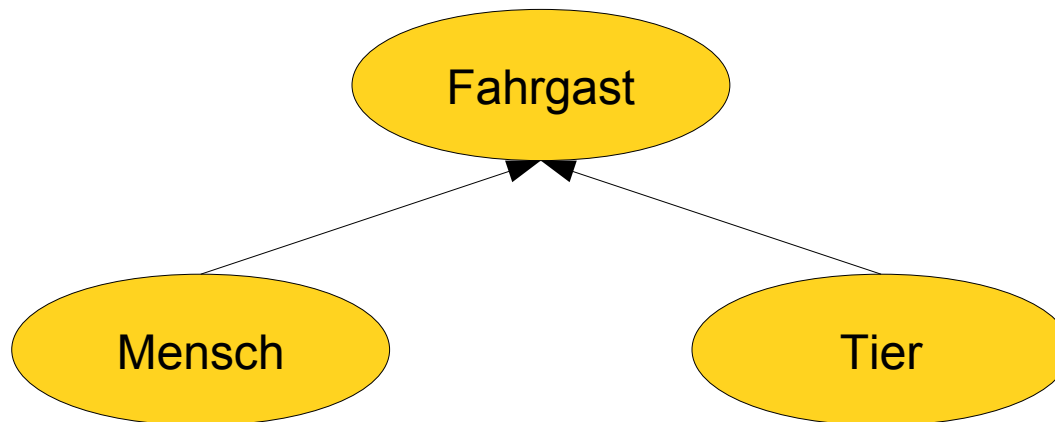
TECHNISCHE
UNIVERSITÄT
DARMSTADT



- Engineering-Methodologien
 - strukturieren der Erstellungsprozess
- Best Practices
 - sorgen für ein besseres Ergebnis

- Eine Sammlung von Analysemethoden
 - Ist meine Klassenhierarchie sinnvoll?
- Entwickelt ~2000-2004 von Guarino und Welty
- basiert auf philosophischen Grundlagen

- Betrachten wir folgende Aufgabenstellung:
 - *Bauen Sie eine Ontologie für ein U-Bahn-Ticketsystem.*
 - *"Fahrgäste der U-Bahn können Menschen und Tiere sein."*

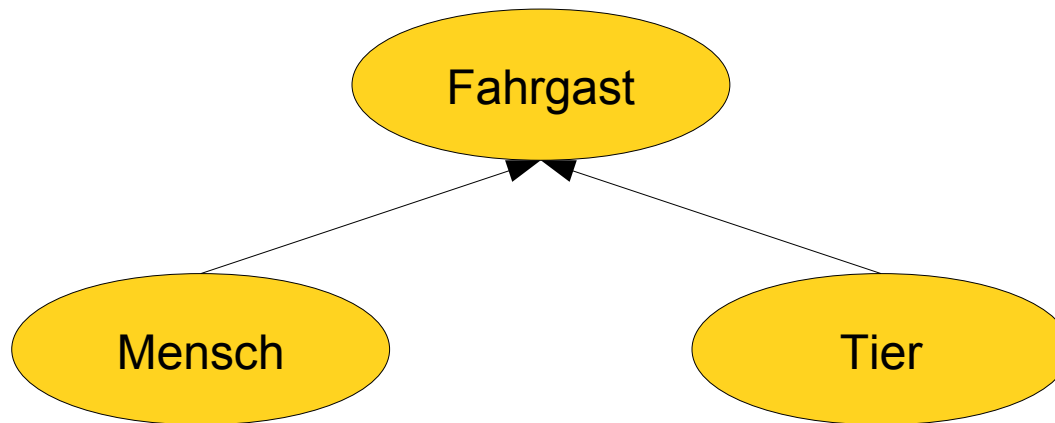


- Was halten Sie von dieser Lösung?

- Man unterscheidet *rigide* und *nicht-rigide* Klassen
 - Wenn eine Entität zu einer rigiden Klasse gehört, dann gilt das zu jedem Zeitpunkt
 - d.h.: wenn eine Entität nicht mehr zu der Klasse gehört, dann hört sie auf zu existieren
 - für nicht-rigide Klassen gilt das nicht
- Beispiele für rigide Klassen
 - Mensch, Ort, Unternehmen
- Beispiele für nicht rigide Klassen
 - Student, Aktiengesellschaft, Kleinstadt
 - Raupe und Schmetterling

Rigidität in OntoClean

- Vorschrift von OntoClean:
 - Rigide Klassen dürfen nicht Subklassen von nicht-rigiden Klassen sein

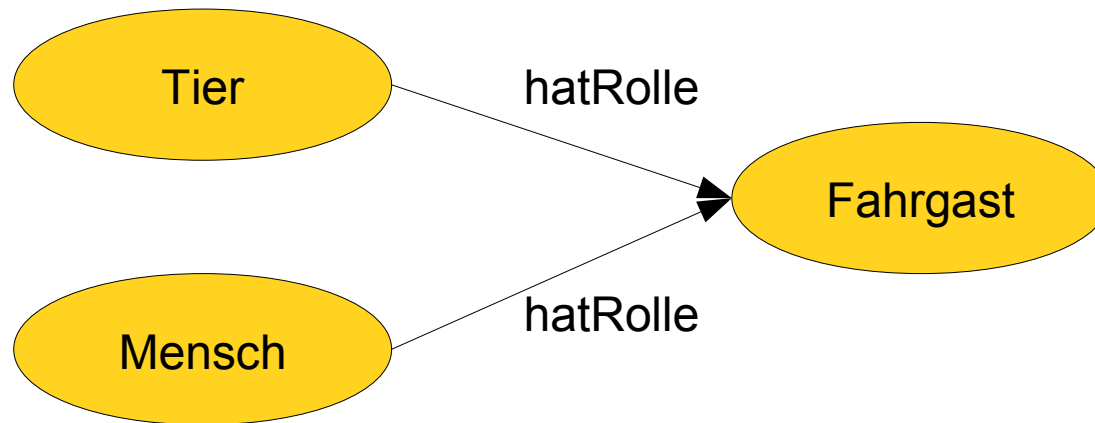


- Angenommen,
 - `:peter a :Mensch .`
 - daraus folgt: `:peter a :Fahrgast .`
 - das ist wahrscheinlich nicht immer richtig!

- Weitere typische Problemfälle
 - `PhysicalObject` > `Animal`
 - Eine Entität kann aufhören, zu leben, und ist damit kein Tier mehr
 - wenn wir Lebendigkeit als essentiell für Tiere ansehen
 - das physikalische Objekt des Körpers existiert aber weiter

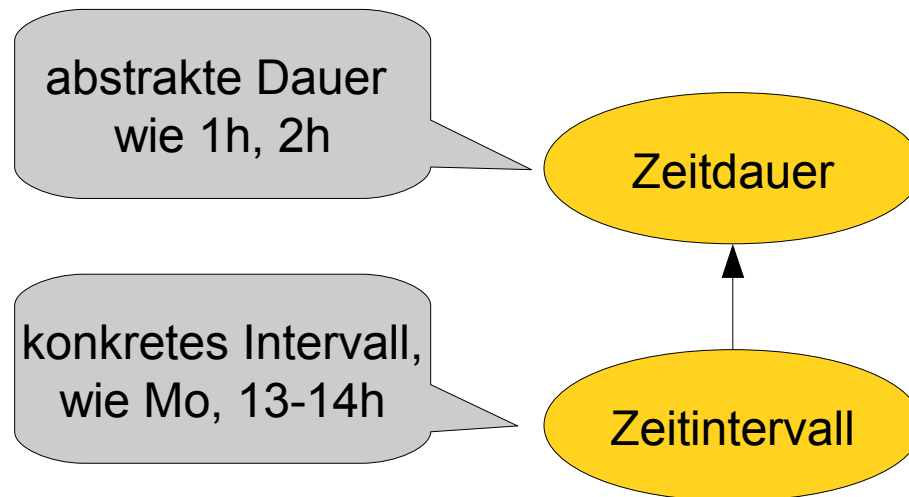
Rigidität in OntoClean

- Verbesserte Lösung:



- Woran erkennt man, dass zwei Dinge gleich sind?
 - Für manche Klassen gibt es dafür *Identitätskriterien*
 - DNA für Lebewesen
 - Matrikelnummer für Studenten
 - Ländercodes für Länder
 - ...

- Betrachten wir folgende Aufgabenstellung:
 - *Bauen Sie eine Ontologie für die Zeiterfassung.*
 - *"Zeitintervalle sind spezielle Zeitdauern: Zeitdauern sind z.B. 1h, 2h, ..., Zeitintervalle sind z.B. Mo, 13-14h, oder Di, 15-17h."*

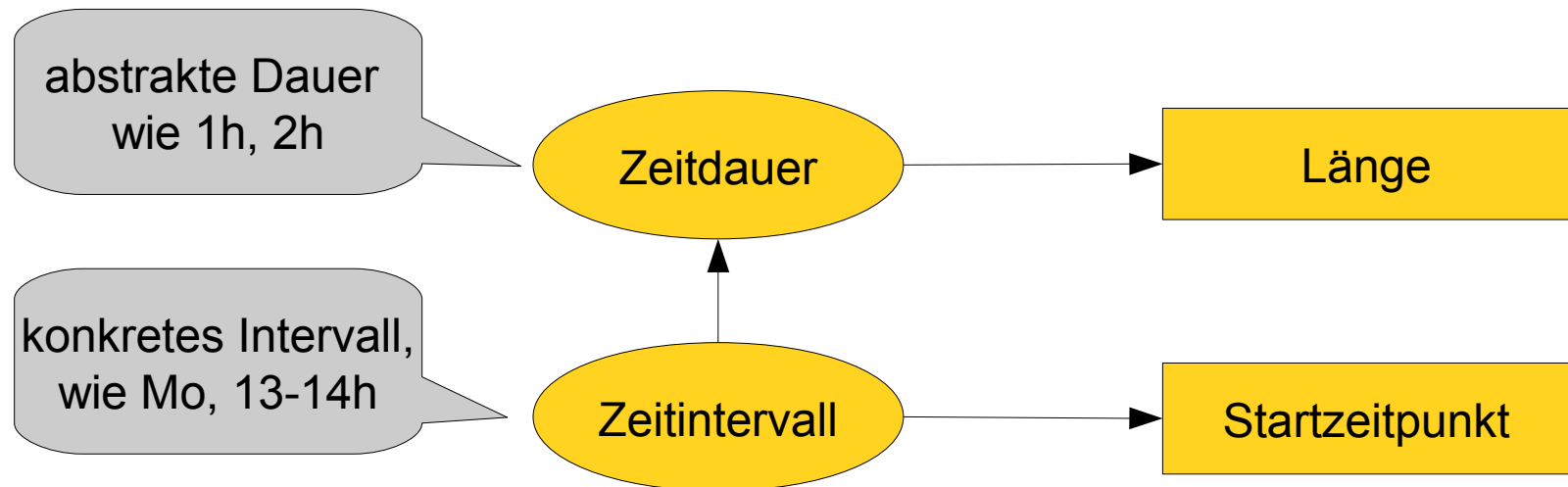


- Was halten Sie von dieser Lösung?

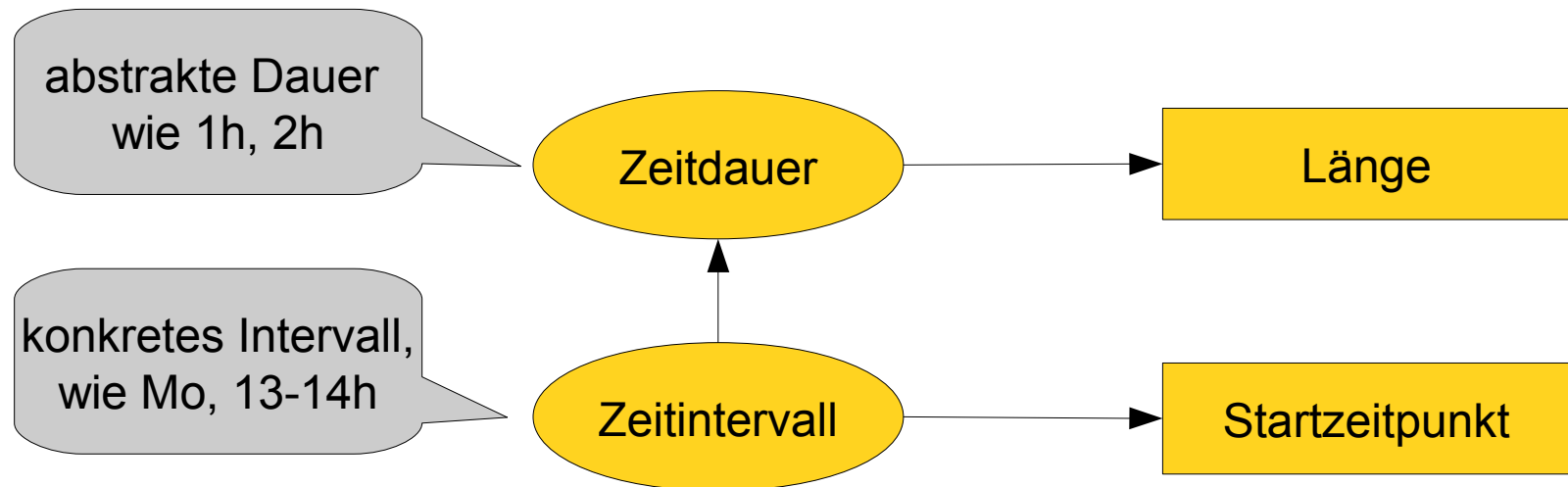
- Betrachten wir mal einige Instanzen der beiden Klassen
 - :1h a :Zeitdauer . :2h a :Zeitdauer
 - :Mo13-14 a :Zeitintervall . :Mo14-15 a :Zeitintervall
- Offenbar ist die Klasse *Zeitintervall* größer als die Klasse *Zeitdauer*...
- Was schließen wir daraus?

- Da die Subklasse nicht größer sein kann,
muss es Instanzen geben, die gleich sind
- Naheliegend wäre z.B.:
 - :Mo13-14 owl:sameAs :1h .
 - :Mo14-15 owl:sameAs :1h .
- Daraus folgt:
 - :Mo13-14 owl:sameAs :Mo14-15 .
- Wollen wir das wirklich?

- Erweitern wir unsere Ontologie noch ein wenig
- Wann sind eigentlich zwei Zeitdauern gleich?
 - wenn sie die gleiche Länge haben
 - `:1h owl:sameAs :60Min` .



- Erweitern wir unsere Ontologie noch ein wenig
- Wann sind eigentlich zwei Zeitintervalle gleich?
 - wenn sie die gleiche Länge und den gleichen Startzeitpunkt haben
 - `:Mo13-14 owl:sameAs :Mo1pm-2pm` .

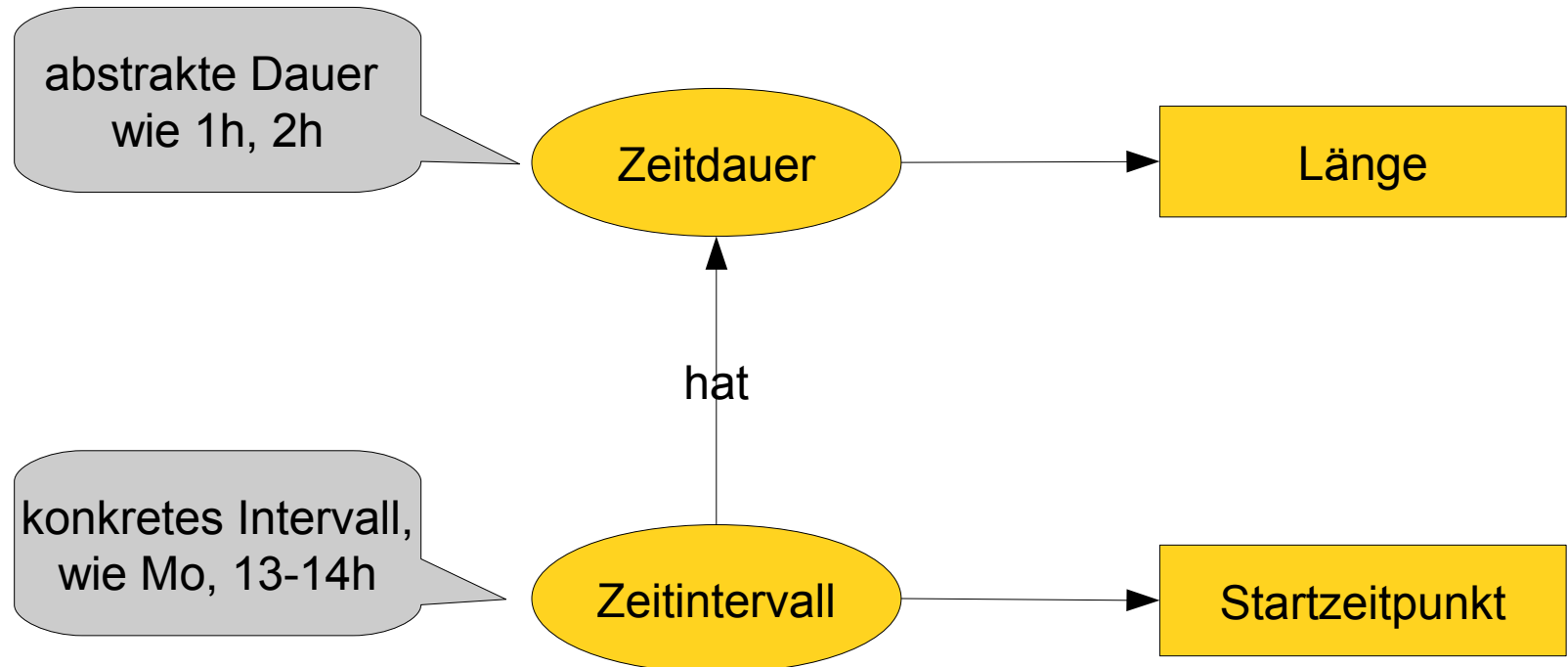


- Beobachtung:
 - Zeitdauer und Zeitintervall haben *verschiedene Identitätskriterien*
- Vorschrift von OntoClean:
 - Wenn p eine Subklasse von q ist,
darf p nicht andere Identitätskriterien besitzen als q

- Weitere typische Problemfälle
 - GeographicalObject > Country
 - Geographische Objekte und Länder haben andere Identitätskriterien
 - Geographische Objekte: Position/Polygon
 - Länder: z.B. Regierung, Verfassung
 - OntoClean zwingt zur Trennung in geographisches und soziales Konstrukt "Land"
 - Roman > Romanausgabe
 - Roman: Titel/Verfasser
 - Romanausgabe: ISBN, oder Titel/Verfasser plus Auflage

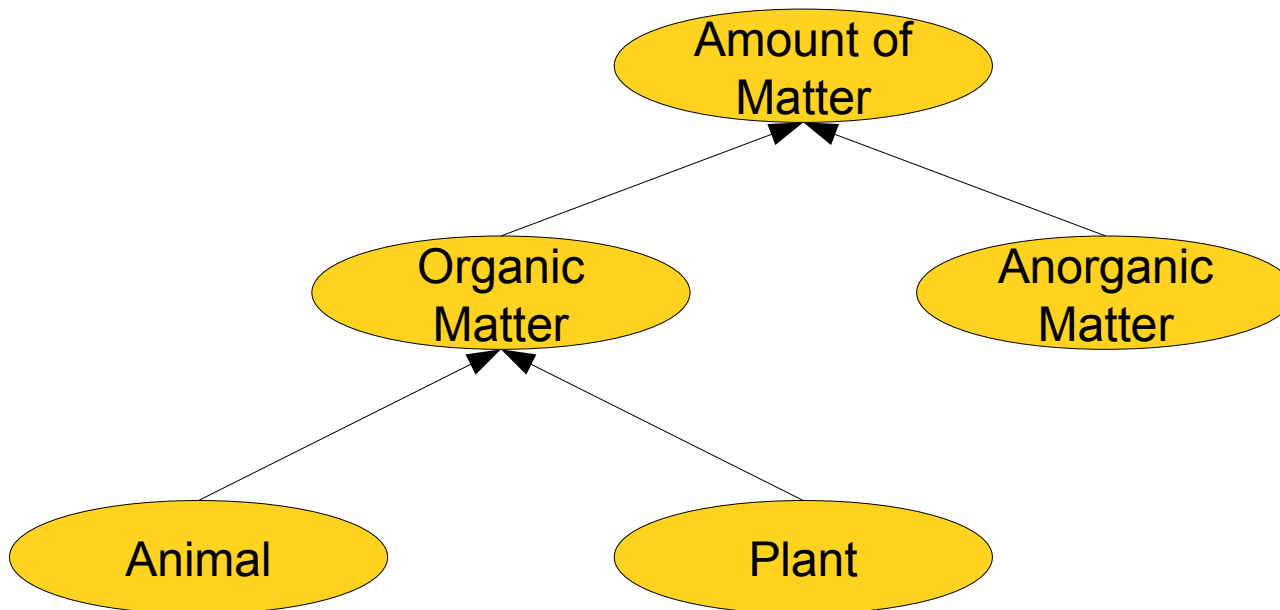
Identität in OntoClean

- Verbesserte Lösung:
 - ersetze Subklassenbeziehung durch andere Relation



- Individuen einiger Klassen lassen sich in Individuen gleicher Art "zerlegen" → Anti-Unity-Klassen
 - Eine Menge Wasser in zwei Mengen Wasser
 - Eine Gruppe in zwei Teilgruppen
 - ...
- Andere Klassen haben immer nur "ganze" Individuen → Unity-Klassen
 - die Klasse der Menschen
 - die Klasse der Städte
- Bei "ganzen" Individuen gibt es immer eine Relation, die ein Teil dem ganzen eindeutig zuordnet
 - z.B. ein Körperteil zu einem Menschen

- Angenommen, wir hätten folgendes definiert



- Weiter angenommen, wir definierten folgendes*:
 - wenn man zwei gleichartige Mengen einer Materie zusammenfügt, entsteht eine (größere) Menge derselben Materie

```
      C rdfs:subClassof AmountOfMatter
Λ      m1 a C . m2 a C . m3 hasPart m1, m2 .
→      m3 a C .
```

*Tun wir einfach mal so, als ginge das in OWL

- Dann folgt aus

```
:fluffi a :Animal .  
:schnuffi a :Animal .  
:SetOfPetersPets hasPart :fluffi, :schnuffi .
```

→ :SetOfPetersPets a :Animal .

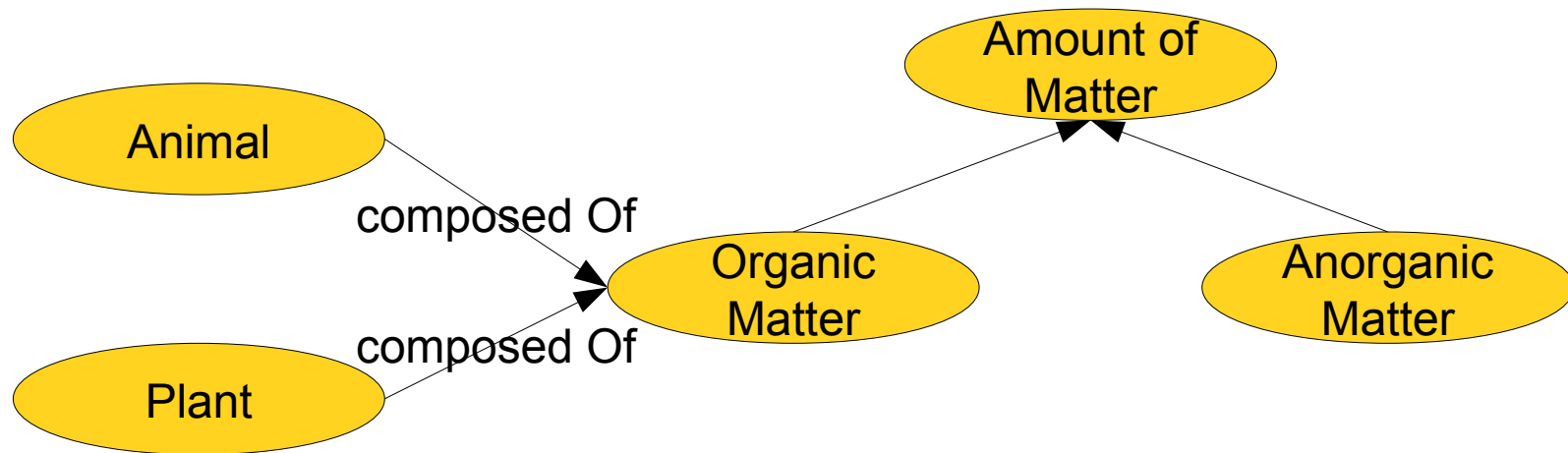
- Wollen wir das?

- OntoClean-Vorschrift:
 - Unity-Klassen dürfen nur Unity-Klassen als Subklassen haben
 - Anty-Unity-Klassen dürfen nur Anti-Unity-Klassen als Subklassen haben

- In unserem Beispiel:
 - OrganicMatter ist eine Anti-Unity-Klasse
 - Animal ist eine Unity-Klasse

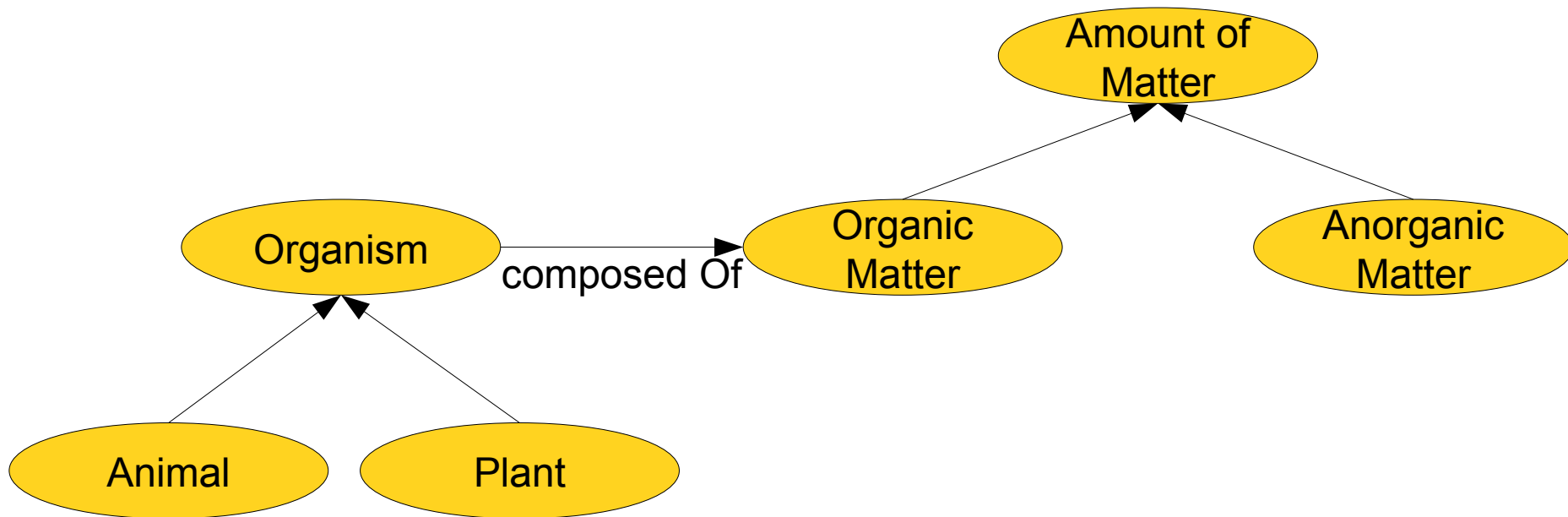
Einheit in OntoClean

- Lösung auch hier: Ersetze Subklassenbeziehung durch andere Beziehung



Einheit in OntoClean

- Das "Aufräumen" mit OntoClean deutet darauf hin, dass uns vielleicht eine zusätzliche Klasse fehlt



- Eine Reihe von Tests, die man auf Ontologien machen kann
 - Rigidität, Identität, Einheit
 - deckt Unstimmigkeiten und Probleme auf
 - bewahrt vor unsinnigen Schlüssen des Reasoners

Ontology Design Patterns

- Bestimmte Probleme treten immer wieder auf
 - wie drückt man bestimmte Probleme in einer Sprache wie OWL aus?
 - z.B. negative Aussagen
 - z.B. n-äre Relationen
 - wie modelliert man bestimmte Dinge?
 - z.B. Informationsobjekte
 - z.B. zeitlich begrenzte Rollen
- Lösung: Design Patterns

Ontology Design Patterns

- Ursprung des Begriffs
 - Christopher Alexander (*1936)
 - Buch "A Pattern Language" (1977)
- Architektur
 - wiederkehrende Probleme
 - standardisierte Lösungen
 - mit Freiheitsgraden
- Beispiel
 - Problem: es regnet ins Gebäude
 - Lösung: Dach
 - Freiheitsgrade: Flachdach, Walmdach, Pultdach...

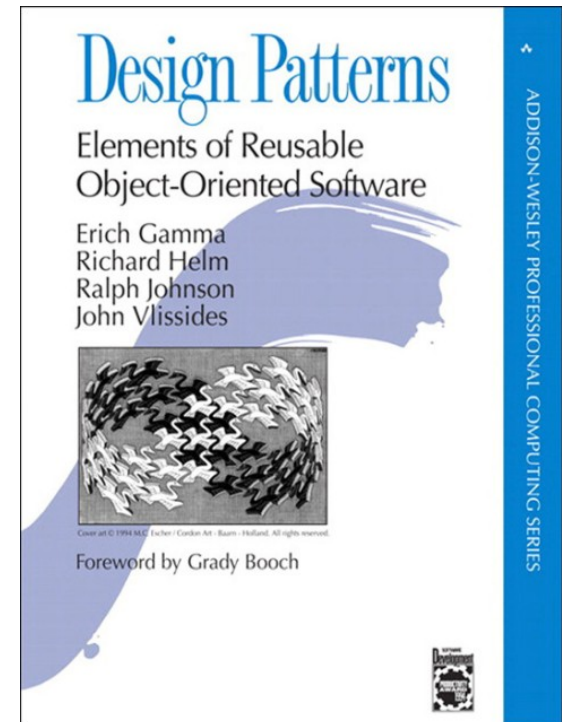


Ontology Design Patterns



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Adaption der Idee im Software Engineering
 - Lösungen für Standardprobleme
 - Gang of Four 1994
- Beispiel: Observer
 - Daten ändern sich
 - Verschiedene Verbraucher sollen darüber informiert werden
 - Entkopplung Erzeuger/Verbraucher erwünscht



Ontology Design Patterns

- Idee aufgegriffen von Aldo Gangemi
 - *Ontology Design Patterns for Semantic Web Content (2005)*
- Vision
 - wiederverwendbare Bausteine
 - durchsuchbarer Katalog
 - sortiert nach Kompetenzfragen



Unterscheidung von Ontology Design Patterns



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Presentation Patterns
 - Ontologie aus Nutzersicht
 - z.B. Namenskonventionen
- Logical Patterns
 - domänenunabhängig
 - beziehen sich auf eine Ausdruckssprache
- Content Patterns
 - domänenabhängig
 - sprachunabhängig
- Transformation Patterns
 - z.B.: wie wandelt man eine Ontologie von einer Sprache in eine andere?



Presentation Patterns

- Beispiel: Namenskonventionen
- Camel Case verwenden
 - CityInNorthernEurope
- Klassen mit Großbuchstaben beginnen, Einzahlwörter verwenden
 - City, Country
- Properties mit Kleinbuchstaben beginnen, Verb verwenden, eindeutige Leserichtung ermöglichen
 - isLocatedIn, isCapitalOf
- Instanzen mit Großbuchstaben beginnen
 - Paris, France
- Labels vorhalten
- ...

- Beispiel: negative Zuweisung vor OWL 2

- in OWL 2:

```
_:x a owl:NegativeObjectPropertyAssertion;  
    owl:sourceIndividual :Paul ;  
    owl:targetIndividual :Peter ;  
    owl:assertionProperty :vaterVon .
```

- das zugehörige Pattern für OWL 1:

```
Paul a [ owl:complementOf [  
        a owl:Restriction ;  
        owl:onProperty :fatherOf ;  
        owl:hasValue :Peter ] ].
```

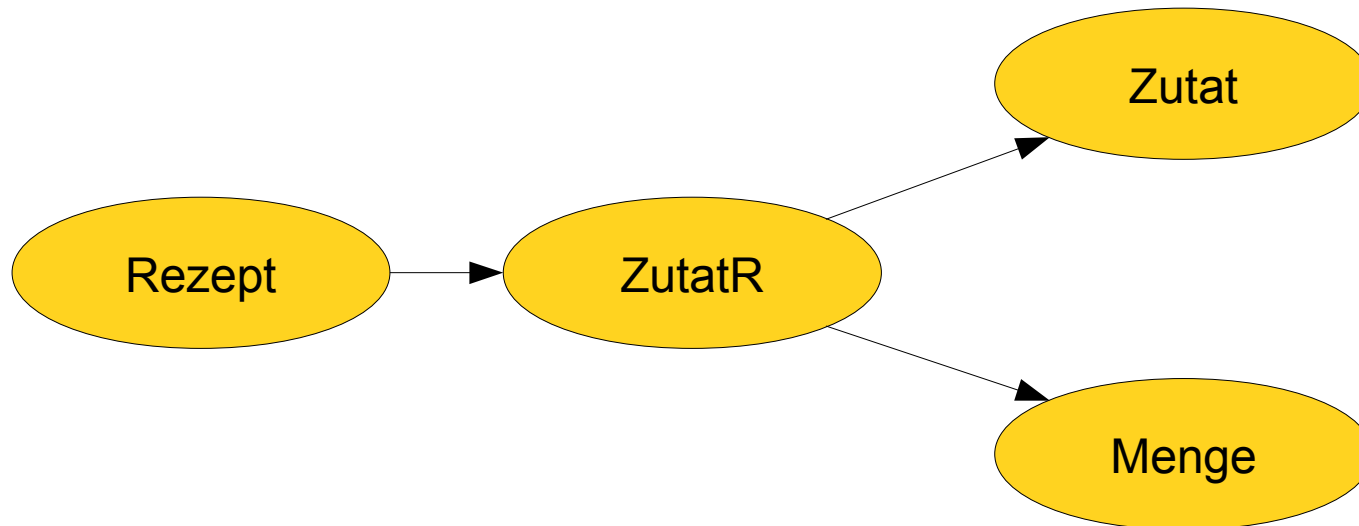

Logical Patterns

- Beispiel: negative Zuweisung, verallgemeinert
- Aussage: $\neg p(X,Y)$
- Pattern:

```
X a [ owl:complementOf [  
    a owl:Restriction ;  
    owl:onProperty P ;  
    owl:hasValue Y ] ] .
```

Logical Patterns

- n-äre Aussagen
- Beispiel Rezept: Zutaten mit Mengen
 - Problem: OWL kennt nur binäre Relationen
 - Lösung: Klasse für Relation

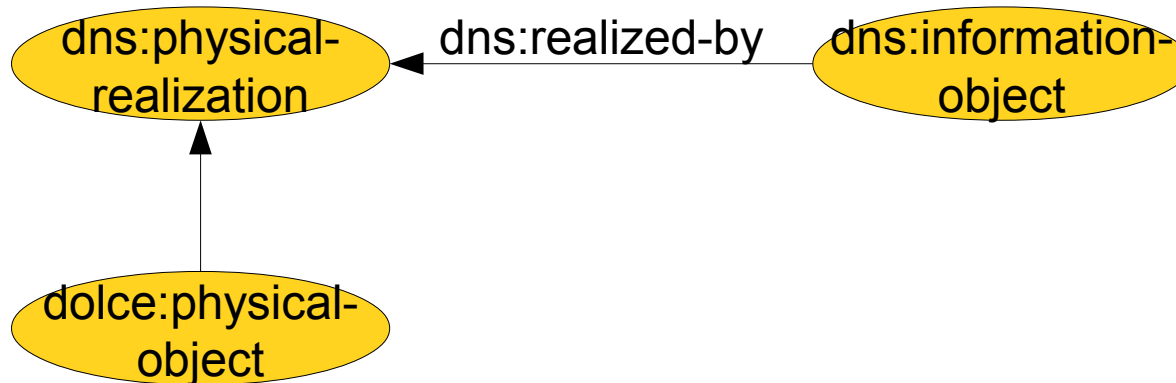


- Beispiel: n-äre Relation, verallgemeinert
- Aussage: $r(X,Y,Z)$
- Pattern:

```
R a owl:Class .
hasR a owl:ObjectProperty .
rComp1 a owl:ObjectProperty .
rComp2 a owl:ObjectProperty .
X hasR [
  a R ;
  hasComp1 Y ;
  hasComp2 Z ] .
```

Content Patterns

- Eines haben wir schon kennen gelernt
- Informationsobjekte und ihre Realisierungen

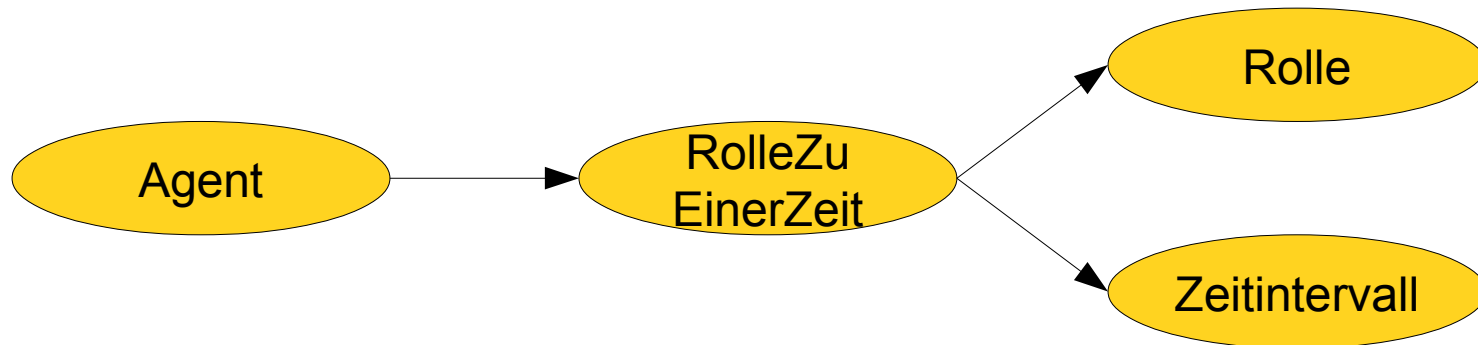


Content Pattern

- Beispiel: Information Object
- Kompetenzfragen:
 - welche Information steckt in einem Objekt?
 - wie ist diese Information realisiert?
- Lösung:
 - Information Object Ontology von DOLCE

Content Pattern

- Beispiel: Rollen zu einer Zeit
- z.B.: Gerhard Schröder war von 1998-2005 Kanzler der BRD



Content Pattern

- Beispiel: Rolle zu einer Zeit
- Kompetenzfragen:
 - wer hatte zu einer bestimmten Zeit eine bestimmte Rolle inne?
- Spezialisierung von
 - n-äre Relation

Ontology Design Patterns

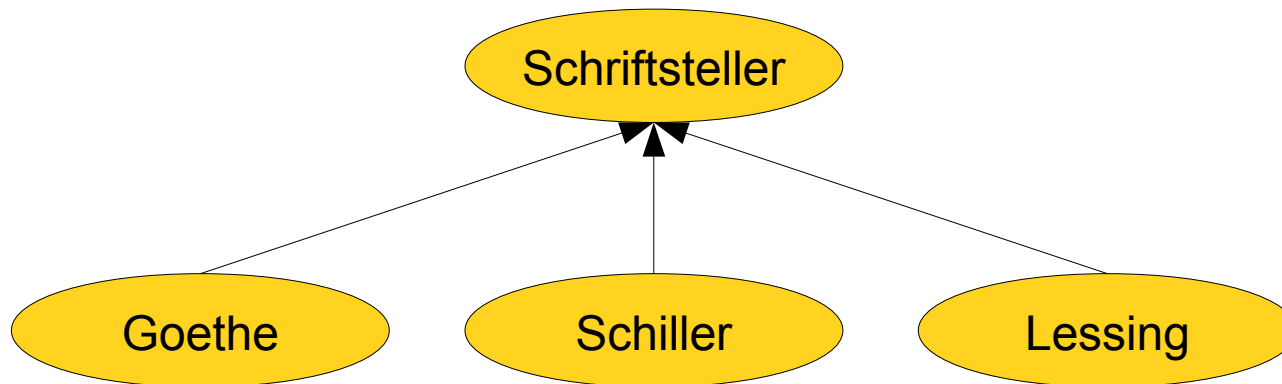


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- bieten Lösungen für bestimmte Probleme
 - logische
 - inhaltliche
- Wiederverwendbare Mini-Ontologien
- Patterns können voneinander erben

- Dinge, die man nicht machen sollte
 - die aber häufig gemacht werden
 - ...und die Probleme aufwerfen
- Mögliche Gründe
 - Folgen zu wenig bedacht
 - falsches Verständnis der Prinzipien von OWL/RDF

- Häufiges Problem:
 - wann soll etwas Klasse sein, wann Instanz?





- Betrachten wir mal diesen Extrem-Fall:

```
:Goethe rdfs:subClassOf :Schriftsteller .  
:Faust rdfs:subClassOf :Drama .  
:Goethe :urheberVon :Faust .
```

- Was folgt daraus?
- Mit einem DL-Reasoner gar nichts,
weil das nicht DL-konform ist!

Anti-Patterns: Klassenwildwuchs

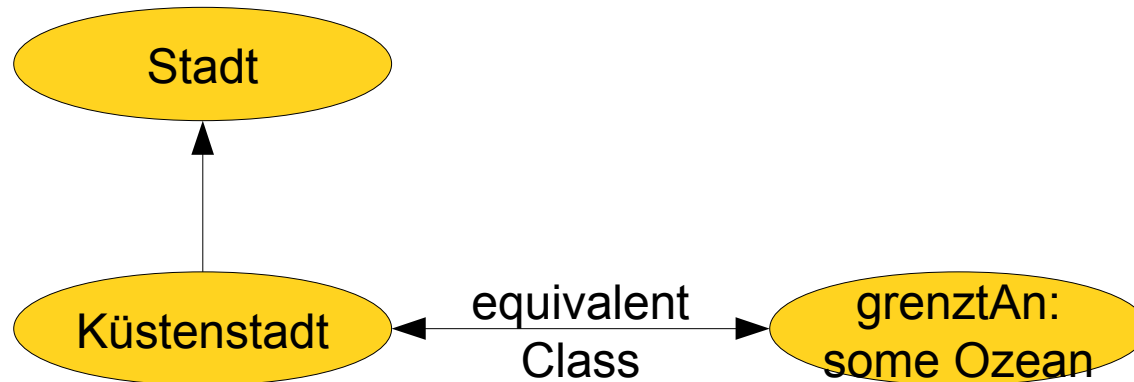


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Wie unterscheidet man Klassen und Instanzen?
- Zu jeder Klasse muss es (meist mehrere) Instanzen geben
 - Was sollen die Instanzen von *Goethe* sein?
 - Mögliche sinnvolle Sätze der Form "X ist ein Goethe"?
- Subklassenbeziehungen müssen sinnvoll sein
 - Muster: "Jede/r/s X ist ein/e Y"
 - "Jeder Goethe ist ein Schriftsteller"?

Anti-Patterns: Exklusivität

- Gegeben folgende Spezifikation:
 - *Städte, die an ein Meer grenzen, sind Küstenstädte.*
- Modellieren wir das mal:



Anti-Patterns: Exklusivität

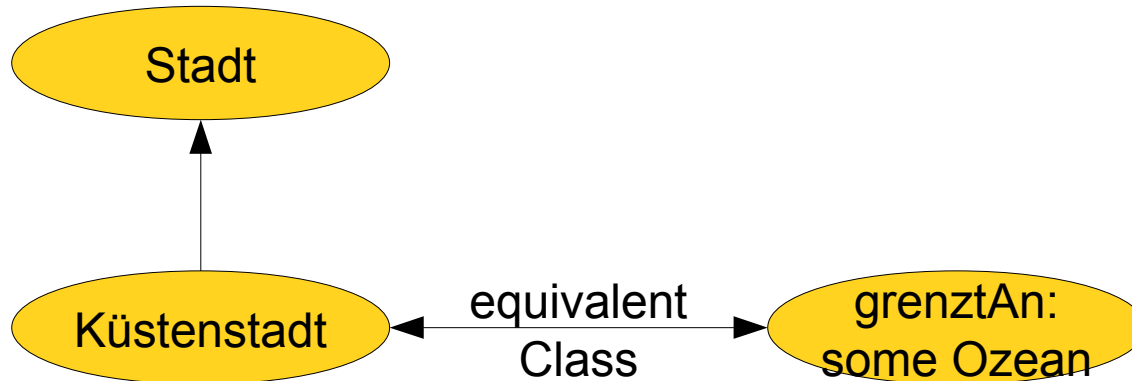


TECHNISCHE
UNIVERSITÄT
DARMSTADT

▪ In OWL:

:Küstenstadt

```
rdfs:subClassOf :Stadt ;  
owl:equivalentClass [  
  a owl:Restriction ;  
  owl:onProperty :grenztAn ;  
  owl:someValuesFrom :Ozean ] .
```



Anti-Patterns: Exklusivität



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Betrachten wir doch mal ein paar Fakten:

```
:Hamburg a :Stadt .  
:Hamburg :grenztAn :Atlantik .  
:Atlantik a :Ozean .  
→ :Hamburg a :Küstenstadt .
```

- Soweit, so gut.

```
:Deutschland a :Land .  
:Deutschland :grenztAn :Atlantik .  
:Atlantik a :Ozean .  
→ :Deutschland a :Küstenstadt .  
→ :Deutschland a :Stadt .
```

Anti-Patterns: Exklusivität

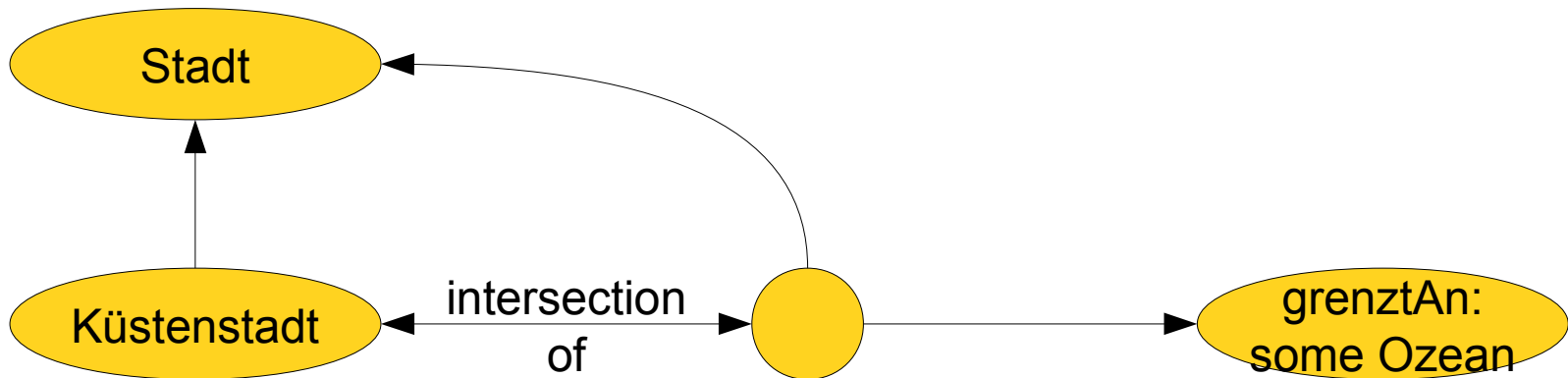


TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Was ist hier passiert?
 - Ontologie *exklusiv* für eine Domäne gebaut
 - z.B.: Städte
 - funktioniert bei unvorhergesehener Verwendung nicht mehr
- Prinzipien des Semantic Web
 - AAA (Anybody can say Anything about Anything)
 - auch unvorhergesehene Verwendungen sollten sinnvoll funktionieren!

Anti-Patterns: Exklusivität

▪ Mögliche Lösung:



```
:Küstenstadt
owl:intersectionOf
( :Stadt
  [ a owl:Restriction ;
    owl:onProperty :grenztAn ;
    owl:someValuesFrom :Ozean ] ) .
```

- *Ontology Engineering*: Entwickeln guter Ontologien
 - gemessen an Nutzwert, z.B. Reasoning-Korrektheit
- Vorgehensmodelle
- OntoClean
 - Systematisches Debugging von Ontologien
- Design Patterns
 - kleine, wiederverwendbare Ontologien
- Anti-Patterns
 - Dinge, die man vermeiden sollte

Vorlesung Semantic Web



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vorlesung im Wintersemester 2011/2012

Dr. Heiko Paulheim

Fachgebiet Knowledge Engineering