ulm university universität ulm

Semantic Web Grundlagen

Ontology Engineering

Birte Glimm
Institut für Künstliche Intelligenz | 30. Jan 2012

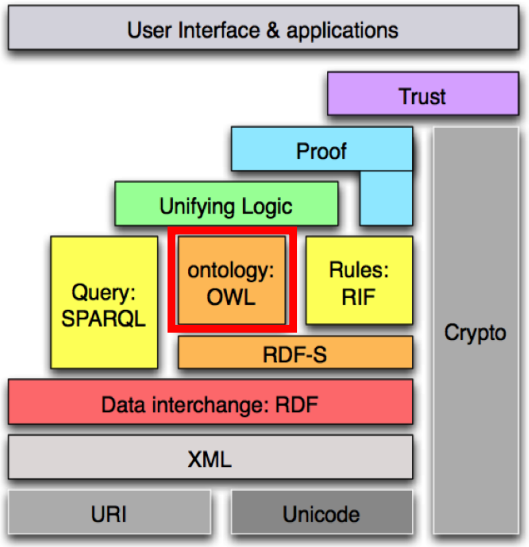
Foliensatz adaptiert und übersetzt von Eva Blomqvist “Ontology Design Patterns”.

Organisatorisches: Inhalt

Einleitung und XML	17. Okt	Hypertableau II	12. Dez
Einführung in RDF	20. Okt	Übung 4	15. Dez
RDF Schema	24. Okt	SPARQL Syntax & Intuition	19. Dez
fällt aus	27. Okt	SPARQL Semantik	22. Dez
Logik – Grundlagen	31. Okt	SPARQL 1.1	9. Jan
Übung 1	3. Nov	Übung 5	12. Jan
Semantik von RDF(S)	7. Nov	SPARQL Entailment	16. Jan
RDF(S) & Datalog Regeln	10. Nov	SPARQL Implemetierung	19. Jan
OWL Syntax & Intuition	14. Nov	Ontology Editing	23. Jan
Übung 2	17. Nov	Übung 6	26. Jan
OWL & BLs	21. Nov	Ontology Engineering	30. Jan
OWL 2	24. Nov	Linked Data	2. Feb
Tableau	28. Nov	SemWeb Anwendungen	6. Feb
Übung 3	1. Dez	Übung 7	9. Feb
Blocking & Unravelling	5. Dez	Wiederholung	13. Feb
Hypertableau	8. Dez	Übung 8	16. Feb

Abfragen und RIF wurde gestrichen

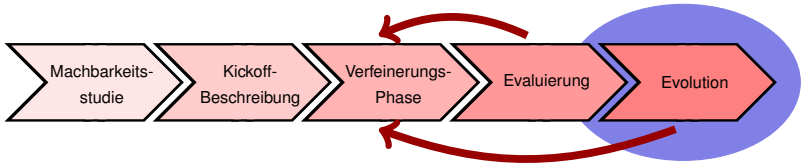
Ontology Engineering



Agenda

- Ontology Engineering: Wie erstellen wir gute Ontologien?
- Motivation für ontologische Entwurfsmuster/design patterns (ODPs)
- Logische ODPs
- Inhaltliche ODPs
- Andere ODPs und “best practices”

Der Wissensmetaprozess (1)

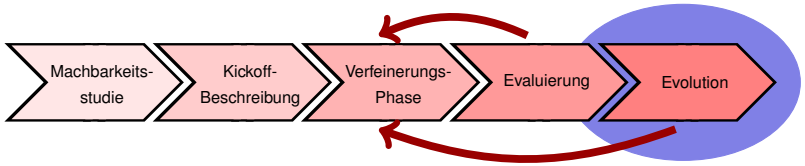


Machbarkeitsstudie: Abwägung der Rahmenfaktoren wie Beteiligte/Domäne/Zeit/zu erwartende Akzeptanz/... (unabhängig von der verwendeten Technologie)

Kickoff-Beschreibung: Ontology Requirements Specification Document (ORSD). Fragestellungen:

- Domäne und Ziel
- Entwurfsrichtlinien
- Verfügbare Wissensquellen
- Potenzielle Nutzer und Szenarien
- Breite der Anwendungen

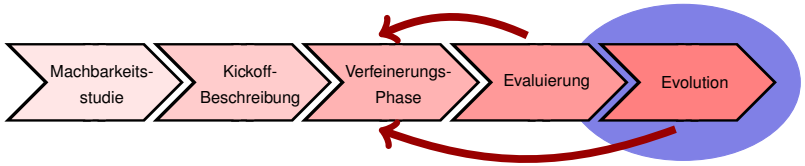
Der Wissensmetaprozess (2)



Verfeinerungsphase: Unter Verwendung der identifizierten Ressourcen:

- Wissensaquirierung mit Domänen-Experten:
  - Identifizierung relevanter Begriffe und Beziehungen
  - Formulierung typischer Axiome
- Formalisierung:
  - Z. B. in einer Beschreibungslogik
- Aufbau und sukzessive Verfeinerung der Ontologie.

Der Wissensmetaprozess (3)



Evaluierung: Gegeben: Formale Ontologien aus vorhergehender Phase.

- Überprüfung der Anforderungen (ORSD)
  - Sind alle Szenarien und Ziele abgedeckt?
  - Deckt die Ontologie den Anwendungsbereich ab?
- Tests mit Hilfe von (prototypischen) Anwendungen

Wartung & Evolution: Fehlerbereinigung und Anpassung an neue Anforderungen

Agenda

- Ontology Engineering: Wie erstellen wir gute Ontologien?
- Motivation für ontologische Entwurfsmuster/design patterns (ODPs)
- Logische ODPs
- Inhaltliche ODPs
- Andere ODPs und “best practices”

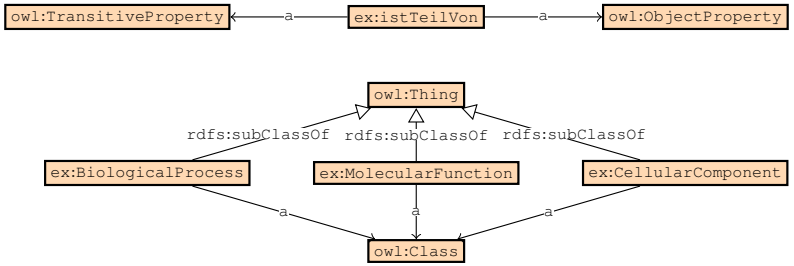
Warum brauchen wir ODPs

Was können wir mit OWL machen?

- Wir können die Ontologie auf Konsistenz prüfen, klassifizieren und Abfragen formulieren
- . . . , aber das hilft wenig für das Scarlet Beispiel
- City subClassOf Country
- Logische Konsistenz ist nicht das Hauptproblem
  - Z.B. kann `owl:sameAs` falsch benutzt werden und trotzdem ist die Ontologie konsistent
- Warum reicht OWL nicht?
  - OWL spezifiziert logische Sprachkonstrukte, aber keine Richtlinien, wie diese genutzt werden sollen, um eine bestimmte Aufgabe zu erfüllen
  - Ob etwas als Individuum oder als Klasse modelliert wird, ist z.B. oft eher willkürlich

Lösungen?

- OWL reicht nicht um automatisch gute Ontologien zu erstellen und wir können nicht erwarten, dass alle Benutzer des Webs Logik oder Ontology Engineering studieren
- Ontologie Entwurfsmuster sind wiederbenutzbare Lösungen für häufige Probleme, die keine erweiterten Kenntnisse erfordern . . .
- . . . angenommen angemessene Tool Unterstützung



Ontologie Entwurfsmuster

Ein ontologisches Entwurfsmuster ist eine wiederverwendbare bewährte Lösung für ein wiederkehrendes Modellierungsproblem.

Agenda

- Ontology Engineering: Wie erstellen wir gute Ontologien?
- Motivation für ontologische Entwurfsmuster/design patterns (ODPs)
- Logische ODPs
- Inhaltliche ODPs
- Andere ODPs und “best practices”

Logische Ontologie Entwurfsmuster

Definition

Ein logisches Entwurfsmuster ist ein formaler Ausdruck, dessen Teile Ausdrücke aus einem logischen Vokabular (z.B. OWL) sind und das ein bestimmtes Problem modelliert

- Logische ODPs sind unabhängig von einer bestimmten Anwendungsdomäne

Logische Ontologie Entwurfsmuster

Ein logisches ODP beschreibt einen formalen Ausdruck der durch Beispiele illustriert, instanziiert und angepasst werden kann, um ein Modellierungsproblem einer Domäne zu lösen

Muster

Klasse $\sqsubseteq$ Restriktion/Quantifikation Rolle Klasse

- Entzündung $\sqsubseteq \exists$befindetSichIn.Körperteil
- Kolitis $\sqsubseteq \exists$befindetSichIn.Kolon
- PetersKolitis befindetSichIn PetersKolon

Logische Makros und Transformations ODPs

Logische Makros bieten einen Shortcut, um wiederkehrende intuitive logische Ausdrücke zu modellieren

Beispiel

- Formal: $\exists r.\top \sqcap \forall r.C$
- Dinge die r, r mindestens ein Ding, dass nur C ist
- Fleischfressende Tiere fressen etwas und fleischfressende Tiere fressen nur Tiere

Transformierende ODPs

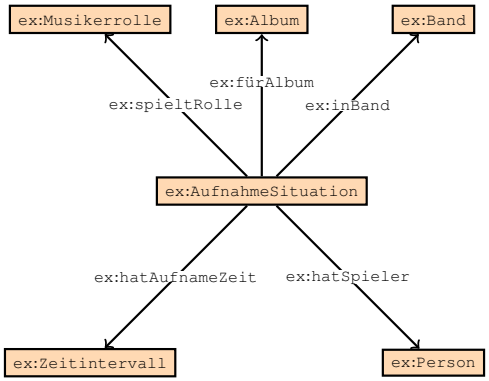
Beispiel

- Transformieren ein ausdrucksstärkes Konstrukt in etwas das in OWL modelliert werden kann

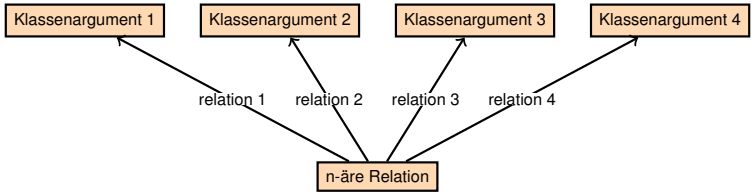
Transformations ODP: Beispiel n-äre Beziehungen

- Chad Smith war der Schlagzeuger der Red Hot Chili Peppers als sie ihr Album Stadium Arcadium von September 2004 bis Dezember 2005 aufgenommen haben
- Eine Person spielt eine bestimmte Rolle in einer Band während der Aufnahme eines Albums während eines bestimmten Zeitintervalls
- N-äre Beziehung: AufnahmeSituation(Person, Musikerrolle, Band, Album, Zeitintervall)
- Wie können wir das in OWL mit nur binären Rollen ausdrücken?

Transformations ODP: Beispiel n-äre Beziehungen



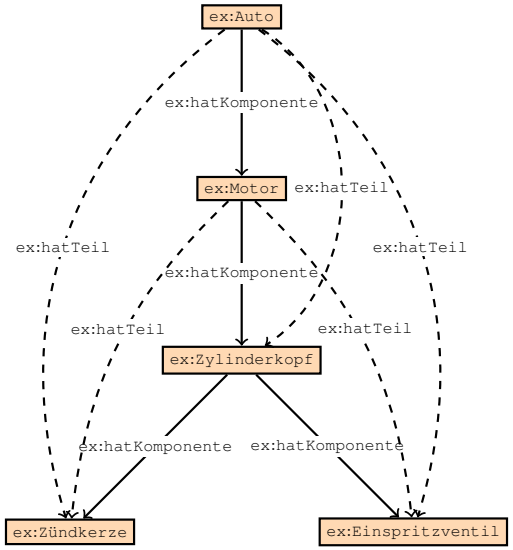
Transformations ODP: Verallgemeinerung n-äre Beziehungen



Transformations ODP: Beispiel Transitive Reduktion

- Es soll modelliert werden, dass ein Auto aus verschiedenen Teilen besteht
  - istTeilVon - hatTeil
- Es soll auch repräsentiert werden, dass ein Teil auch “direkte” Komponenten haben kann
  - istKomponenteVon - hatKomponente
- Beispiel:
  - Der Motor ist eine Komponente eines Autos, welcher wiederum einen Zylinderkopf als Komponente enthält, der die Zündkerzen und Einspritzventile aufnimmt
  - Zylinderkopf, Zündkerzen und Einspritzventile sind indirekte Teile eines Autos

Transformations ODP: Direkte und indirekte Komponenten



Transformations ODP: Beispiel Transitive Reduktion

```
T = { @prefix ...
      ex:hatTeil a owl:TransitiveProperty .
      ex:hatKomponente rdfs:subPropertyOf ex:hatTeil . }
```

Die Sub-Property `ex:hatKomponente` erbt die Transitivität nicht, aber

$T \cup \{$ `ex:Motor ex:hatKomponente ex:Zylinderkopf .` $\} \models$
$\{$ `ex:Motor ex:hatTeil ex:Zylinderkopf .` $\}$

und

$T \cup \{$ `ex:Zylinderkopf ex:hatKomponente ex:Zündkerze .` $\} \models$
$\{$ `ex:Zylinderkopf ex:hatTeil ex:Zündkerze .` $\}$

durch Transitivität von `ex:hatTeil` erhalten wir dann die Konsequenz

`ex:Motor ex:hatTeil ex:Zündkerze .`

Transformations ODP: Beispiel Transitive Reduktion

In BL Notation:

$T = \{\ \mathtt{hatTeil} \circ \mathtt{hatTeil} \sqsubseteq \mathtt{hatTeil}$
$\mathtt{hatKomponente} \sqsubseteq \mathtt{hatTeil}\ \}$

Wir erhalten:

$T \cup \{\, \mathtt{hatKomponente(Motor,\ Zylinderkopf)} \,\} \models$
$\{\, \mathtt{hatTeil(Motor,\ Zylinderkopf)} \,\}$

und

$T \cup \{\, \mathtt{hatKomponente(Zylinderkopf,\ Zündkerze)} \,\} \models$
$\{\, \mathtt{hatTeil(Zylinderkopf,\ Zündkerze)} \,\}$

durch Transitivität von `hatTeil` folgt dann

`hatTeil(Motor, Zündkerze)`

Agenda

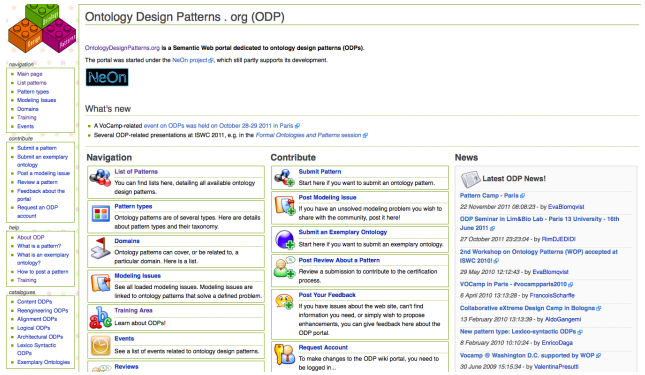
- Ontology Engineering: Wie erstellen wir gute Ontologien?
- Motivation für ontologische Entwurfsmuster/design patterns (ODPs)
- Logische ODPs
- Inhaltliche ODPs
- Andere ODPs und “best practices”

Inhaltliche Ontologie Entwurfsmuster

- Inhaltliche Ontologie Entwurfsmuster beschreiben konzeptuelle Muster
  - Logische ODPs lösen Designprobleme unabhängig von einer bestimmten Konzeptualisierung
  - Inhaltliche ODPs addressieren Designprobleme der Domänenklassen und Propertys, die in einer Ontologie vorkommen
  - Inhaltliche ODPs sind also inhaltsbezogen
- Inhaltliche ODPs sind Instantiierungen von logischen ODPs (oder von Kompositionen logischer ODPs) für eine bestimmte Signatur
  - Sie haben daher ein bestimmtes (nicht-logisches) Vokabular für die Anwendungsdomäne und sind somit inhaltsabhängig

Kataloge von Inhaltlichen Ontologie Entwurfsmustern

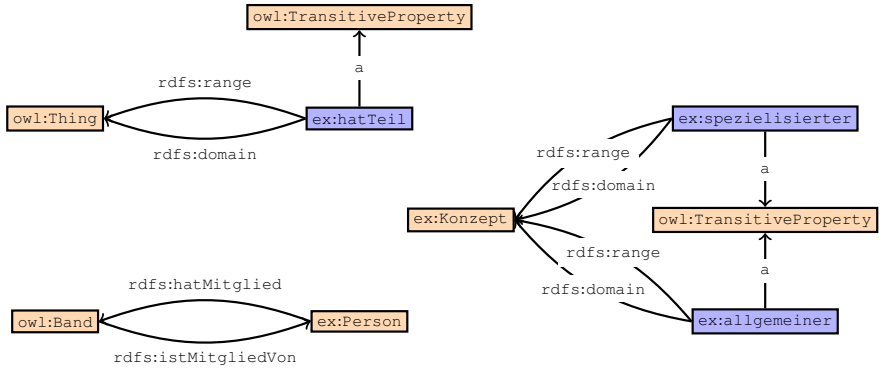
- Inhaltliche ODPs werden in Katalogen gesammelt und beschrieben und folgen einer standardisierten Vorlage
- Die Initiative ontologydesignpatterns.org verwaltet ein derartiges Repository und ein semantisches Wiki für die Beschreibung, Evaluierung, Diskussion und Zertifizierung von inhaltlichen ODPs



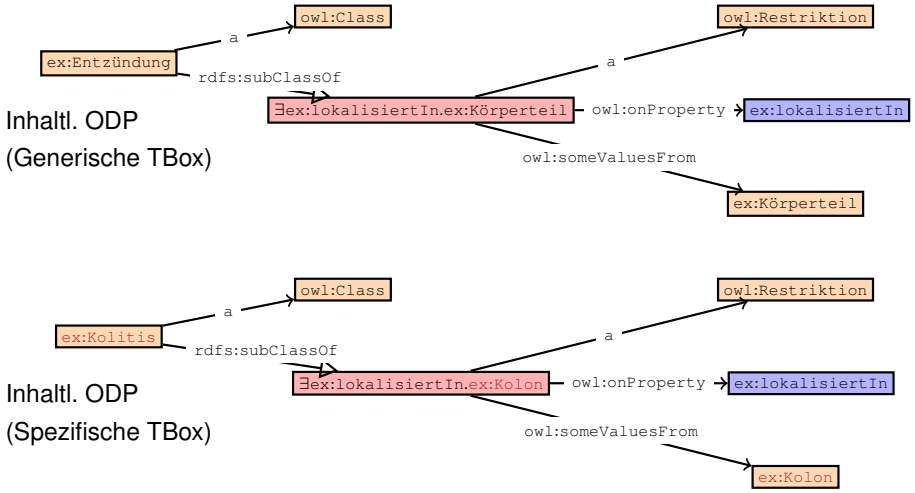
Pragmatische Charakteristiken von Inhaltlichen ODPs

- Domänenabhängig
  - Ausgedrückt mittels eines domänenabhängigen Vokabulars
- Orientiert an den Anwendungsanforderungen
  - Löst eine Modellierungsproblem in der Anwendungsdomäne (ausgedrückt als Use Case, Aufgabe oder “competency questions”)
- Komponenten die für das Reasoning relevant sind
  - Realisiert bestimmte Konsequenzen (minimale Axiomatisierung)
- Kognitätsbezogene Komponenten
  - Darstellung der wichtigsten Notationen einer Domäne und der dabei relevanten Expertise
- Linguistik-bezogene Komponenten
  - Beziehen sich auf linguistische Features, z.B., darauf wie disjunkte Klassen in natürlicher Sprache beschrieben werden

Beispiele zu Inhaltlichen ODPs



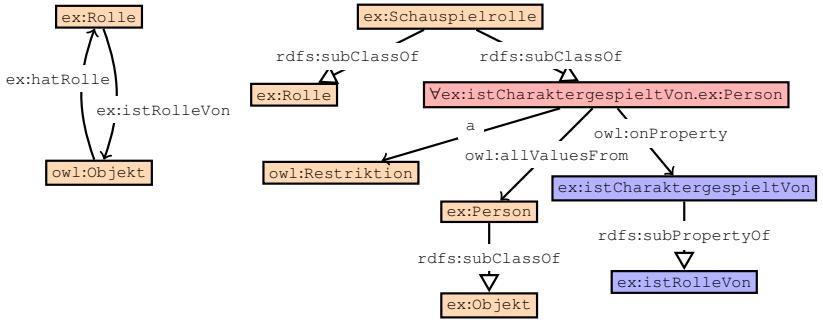
Beispiele zu Inhaltlichen ODPs



ODP Spezialisierung

Definition

Ein inhaltl. ODP P_1 spezialisiert ein ODP P_2 wenn mindestens ein ontologisches Element von P_2 eine Verallgemeinerung mittels `rdfs:subClassOf` oder `rdfs:subPropertyOf` von einem ontologischen Element in P_1 ist.



Komposition

- Eine Komposition verbindet zwei inhaltl. ODPs und hat als Ergebnis eine neue Ontologie
- Die resultierende Ontologie besteht aus der Vereinigung der ontologischen Elemente und den Axiomen aus den beiden ODPs und Axiomen, die die beiden ODPs verbinden (z.B. Äquivalenz- oder Disjunktheitsaxiome)
- Die Komposition enthält mindestens ein Axiom das ein Element auf dem ersten ODP mit einem Element aus dem zweiten ODP verbindet
- Typischerweise werden auch weitere Elemente zu der Komposition hinzugefügt

Generelle Inhaltliche ODPs

- Rollen von Objekten
- Klassifikation
- Teile-Ganzes Beziehungen
- Mitgliedschaft
- Informationen und ihre Realisierung
- Sequenzen
- Themen
- Zeit
- Plätze
- Bewegung
- Pläne
- Ereignisse
- Beschreibungen und Situationen

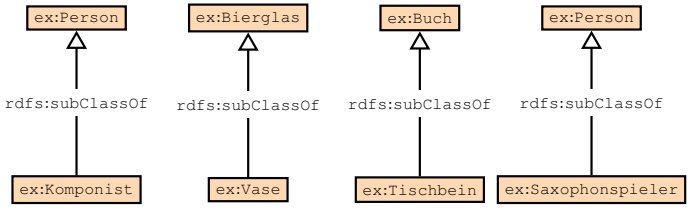
Beispiel: Rollen von Objekten

- Objekte können zu verschiedenen Zeiten verschiedene Rollen spielen
- Je nach den Anwendungsanforderungen können die Objekte und ihre Rollen unterschiedlich modelliert werden
- Wollen wir Eigenschaften von Rollen modellieren?
- Wollen wir Objekte basierend auf ihren Rollen klassifizieren?
- Wollen wir Fakten für Rollen spezifizieren?

Beispiel: Rollen von Objekten

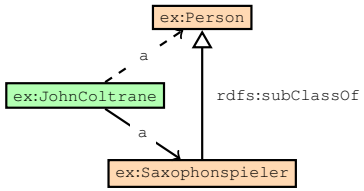
- Ein Bierglas genutzt als Vase
- Bücher verwendet als Tischbeine
- Eine Person als Saxophonspieler
- Eine Person als Komponist

1. ODP: Rollen als Klassen



1. ODP: Rollen als Klassen

- Ein Objekt und seine Rolle sind über `rdf:type` miteinander verbunden
- `rdf:type` Relationen können entweder explizit angegeben sein oder implizit durch automatisches Schlussfolgern folgen
- Um Individuen automatisch in eine bestimmte Klasse zu klassifizieren, muss die Ontologie entsprechende Axiome enthalten

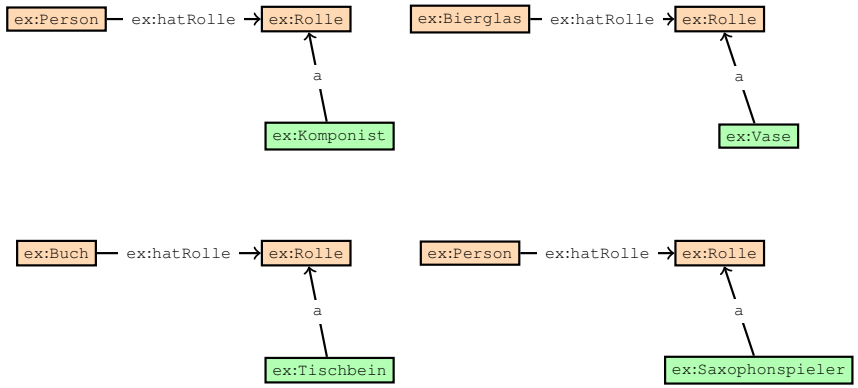


1. ODP: Rollen als Klassen

Konsequenzen

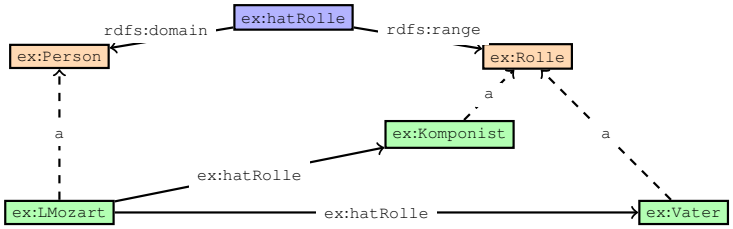
- Wenig ausdrucksstark
- Rollen werden in der TBox beschrieben
- Klassenhierarchy wird größer: eine Klasse pro Rolle
- Klassenhierarchy wird verschlungen: Mehrfachtypisierung
- ABox wird kleiner: Ein Individuum kann mehrere Rollen als Typ haben
- Automatische Klassifikation von Individuen durch `rdfs:subClassOf` (mit entsprechenden Axiomen)
- Rollen können nicht nach Zeit und Ort zugewiesen werden

2. ODP: Rollen als Individuen



2. ODP: Rollen als Individuen

- Eine Rolle und das entsprechende Objekt sind durch domänenspezifische Relationen Verbunden
- Relationen zwischen einem Objekt und seiner Rolle müssen als Fakten gegeben werden
- Automatische Schlussfolgerung der Relation zwischen einem Objekt und einer Rolle können durch Rollensubsumption erreicht werden



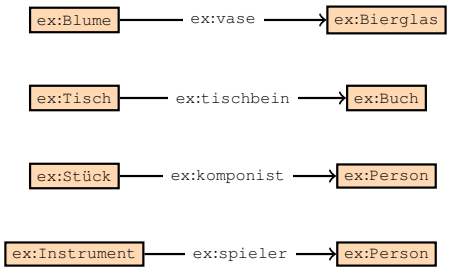
2. ODP: Rollen als Individuen

Konsequenzen

- Ausdrucksstärker
- Rollen werden in der ABox beschrieben
- Klassenhierarchy wird kleiner: Rolle sind Individuen
- ABox wird größer
- Fakten für die Rollen können spezifiziert werden
- Zeit und Ort für Rollen können durch n-äre Relationen modelliert werden
- N-äre Relationen sind nötig um ein Objekt mit seiner Rolle und einem anderen Objekt zu verbinden
  - "Birte ist die Leiterin des Kurses SemWeb":

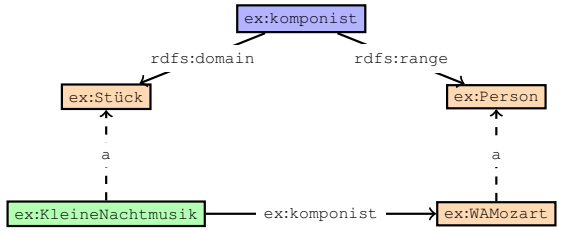
```
ex:LeitungsSituation ex:betrifftPerson ex:Birte .
ex:LeitungsSituation ex:betrifftKurs ex:SemWeb .
ex:LeitungsSituation ex:betrifftRolle ex:Leiter .
ex:Birte ex:hatRolle ex:Leiter .
```

3. ODP: Rollen als Propertys



3. ODP: Rollen als Propertys

- Die Semantik, dass eine Property eine Rolle ist, steckt nur im Namen der Property
- Objekte sind nicht explizit mit ihren Rollen verbunden. Sie sind mit anderen Dingen durch eine Property verbunden, die die Rolle spezifiziert, die gespielt wird
- Am weitesten verbreitet im Web um Rollen zu modellieren



3. ODP: Rollen als Propertys

Konsequenzen

- Kleinere Klassenhierarchy
- Größere Property-Hierarchy
- Einfachere ABoxen
  - “Birte ist die Leiterin des Kurses SemWeb”: `ex:Birte ex:leiterin ex:SemWeb .`
- Zeit und Ort für Rollen können nicht modelliert werden
- Semantik darüber was eine Rolle ist ist implizit im Namen der Property
- Fakten über Rollen können nicht spezifiziert werden

Zusammenfassung: ODPs Rollen für Objekte

- Die drei Lösungen unterscheiden sich in Ausdrucksstärke und Einfachheit
  - Rollen als Propertys ist am einfachsten
  - Rollen als Individuen ist am ausdrucksstärksten
  - Rollen als Klassen ist am ausdrucksschwächsten
- Alle Lösungen haben Vor- und Nachteile
- Die Wahl hängt von den Anforderungen ab
- Können wir die Lösungen auch kombinieren?

Inhaltliche ODPs für Rollen von Objekten

- Das generelle Muster nennt sich “Klassifikation”
- Objekt–Rolle und Agenten–Rolle
  - OWL Muster in der Rollen als Individuen repräsentiert sind
  - `http://ontologydesignpatterns.org/wiki/Submissions:Objectrole`
  - `http://ontologydesignpatterns.org/wiki/Submissions:AgentRole`
- Zeit-indexiertes Personen–Rollen Muster
  - `http://ontologydesignpatterns.org/wiki/Submissions:Time_indexed_person_role`
- Zeit-Ort-indexiertes Objekt–Rollen Muster
  - N-äre Relation repräsentieren ein Objekt, die Rollen, die es spielt an einem bestimmten Ort zu einer bestimmten Zeit
  - `http://www.ontologydesignpatterns.org/cp/owl/dul/timeplaceindexedobjectrole.owl`

Andere ODPs?

- Lexiko-syntaktische ODPs
  - Welche Muster in einem Text entsprechen welchen logischen Konstrukten, z.B., in OWL?
- Präsentations ODPs
  - Vorlagen für Annotationen und Dokumentationen
  - Namenskonventionen
    - Singular für Klassennamen
    - Lesbare Property Namen
    - URI Konventionen
- Korrespondenz ODPs
  - Reengineering ODPs yur Tranformation von Datenbanken, XML, etc. nach OWL
  - Alignment (Ausrichtungs-) Muster, die Korrespondenzen zwischen verschiedenen Modellierungsvarianten beschreiben

Agenda

- Ontology Engineering: Wie erstellen wir gute Ontologien?
- Motivation für ontologische Entwurfsmuster/design patterns (ODPs)
- Logische ODPs
- Inhaltliche ODPs
- Andere ODPs und “best practices”

ODPs versus “Best Practices”?

- ODPs repräsentieren ideallerweise “best practices”
- Dinge die in anderen Bereichen “best practices” genannte werden, werden in diesem Bereich als ODPs bezeichnet
  - Z.B. Namenkonventionen
- Wie sieht es mit automatisch generierten ODPs aus? Top-down versus bottom-up Mustererzeugung . . .
  - Encyclopedic Knowledge Patterns (knowledge patterns (KP): kleine, verbundene Bedeutungseinheiten, die 1) Aufgaben-basiert, 2) wohldefiniert und 3) fehlerfrei auf der kognitiven Ebene sind)
  - Linguistic Frames (die ein linguistisches Konstrukt in einem Zusammenhangskontext betrachten)

Zusammenfassung

- ODPs bieten dem Ontology Engineer praktische und konkrete Muster für bestimmte Szenarien
- Weitere Informationen zum Thema:
  - `http://ontologydesignpatterns.org/`
  - Kompetenz Fragen (competency questions) `http://protege.stanford.edu/publications/ontology_development/ontology101.pdf`
  - Massnahmen um die Strukturer einer Ontologie zu verbessern: OntoClean `http://www.ontoclean.org/`