Erste Anmerkungen zur Modellierung der Ontologien

Hans-Gert Gräbe

23. Februar 2021

1 Hintergrund

Wie bereits in der *Handreichung zu den Seminararbeiten* ausgeführt, schließen die auszuführenden Modellierungen an das *TRIZ Summit Ontologie-Projekt* (TOP) an. Im Gegensatz zu den dort bisher ausgeführten „großflächigen Ontologisierungen“, mit denen grundlegende Zusammenhänge verschiedener Teile der TRIZ-Theorie ontologisch „kartiert“ wurden, geht es in den Seminararbeiten darum, einzelne Teilbereiche semantisch genauer zu modellieren.

2 Abstraktonsebenen der Modellierung

Dabei geht es um eine „Modellierung der Modellierung“, denn die mit einer Ontologie angestrebte Klärung von Begriffen und Konzepten ist dazu gedacht, in realweltlichen Modellierungskontexten praktisch zur Anwendung zu kommen. Diese „Modellierung der Modellierung“ referenziert einen typischen ingenieur-technischen Kontext, in dem der *Modellierung* von Systemen als Basis des weiteren planmäßigen Vorgehens (in Projektierung, Implementierung, Betrieb, Wartung, Weiterentwicklung) eine zentrale Rolle zukommt.

Dabei sind *mehrere Abstraktionsebenen* zu unterscheiden, was im TOP allerdings nicht ausreichend geschieht. Dies sind

0. Die Ebene des *realweltlichen Systems*, auf das sich die konkrete ingenieur-technische Modellierung bezieht. Diese Ebene ist nur *praktisch* zugänglich. In der Praxis muss sich die Modellierung bewähren, alle dabei auftretenden Probleme, die inhärente Widersprüchlichkeit von Modell und Praxis eingeschlossen, sind erst und usschließlich auf der *Modellebene* sprachlich formulierbar in der *Anwendung* der dort verfügbaren Konzepte. Diese Konzepe müssen also nicht nur das System (genauer: sein Modell) selbst beschreiben können, sondern auch die erforderlichen Aspekte seines Betriebs.
1. Die Ebene der *Modellierung* eines konkreten Systems, die *Modellebene*. Die Ontologie stellt hierfür die Sprachmittel, Konzepte (RDF-Subjekte) und Eigenschaften (RDF-Prädikate) zur Verfügung, die auf dieser Ebene *angewendet* werden. Diese Ebene ist zugleich die *Ebene der methodologischen Praxis*.
2. Die Ebene des *Metamodells* als die eigentliche Ontologie-Ebene, auf der die systemischen Konzepte *vereinbart* (definert) unter Berücksichtigung und *Anwendung* der methodologischen Konzepte, deren Sprachmittel auf der Metaebene 2 zur Verfügung gestellt werden.

3. Die *Metaebene 2 der Modellierung*, auf der die methodologischen Konzepte *definiert* werden.

In den weiteren Beispielen spielen drei Namensräume eine Rolle:

- `ex:` ist der Namensraum einer konkreten Modellierung (Ebene 1).
- `tc:` ist der Namensraum der TRIZ-Konzepte (Ebene 2, Subjekte).
- `od:` ist der Namensraum der WUMM-eigenen Konzepte (methodologische Subjekte, Prädikate Ebene 2).

3 Typische Modellierungssituationen

3.1 Morphologischer Kasten

Dabei handelt es sich um ein Konzept, das eine endliche Zahl von festen Werten annehmen kann (ein `enum` Datentyp in Java).

Beispiel: Farbe (rot, grün, gelb, blau)

```
ex:MeiersAuto od:hatFarbe tc:gruen .

tc:Farbe a skos:Concept, od:AdditionalConcept ;
  skos:prefLabel "Color"@en, "Farbe"@de ;
  od:allowedValues tc:red, tc:green, tc:yellow, tc:blue .

tc:FarbValue a skos:Concept, od:AdditionalConcept ;
  skos:prefLabel "Value for Color"@en, "Farbwert"@de .

tc:green a skos:Concept, tc:FarbValue, od:AdditionalConcept ;
  skos:prefLabel "green"@en, "grün"@de .

...
```

3.2 Modellierung grundsätzlicher Konzepte

Das TOP-Modell folgt einem Vererbungskonzept, in dem sich allgemeine Unterscheidungen auf speziellere Konzepte vererben. So wird etwa für Flüsse unterschieden zwischen Flow, FlowModel, FlowAsIs, FlowAsShouldBe. Das ist einerseits eine methodische Setzung (Ebene 3), andererseits wirkt sich diese Setzung erst auf Ebene 1 aus. Deshalb soll diese Information jeder Instanz über die Eigenschaft `od:hasMode` mit dem Wertebereich `tc:Mode` zugeordnet werden.

```
ex:MeiersAuto od:hasMode tc:theModelAsIs .

tc:Mode a skos:Concept, od:AdditionalConcept ;
  skos:prefLabel "Modeling mode"@en, "Modellierungsmodus"@de ;
  od:allowedValues od:theReal, od:theModel, od:theRealAsIs,
    od:theRealAsShouldBe, od:theModelAsIs, od:theModelAsShouldBe .

tc:theModelAsIs a skos:Concept, od:AdditionalConcept ;
  od:valueOf tc:ModeValue;
```

```
  skos:prefLabel "the model as is"@en, "Modell, wie es ist"@de .
...
```

Im TOP-Modell werden Eigenschaften von Klassen nur spärlich modelliert. Zum Beispiel kann ein Fluss ein *diskreter Fluss* sein. Dies ist ein Wert einer Eigenschaft dieses Flusses. Um eine solche Beziehung zu modellieren, müssen die *Eigenschaft* selbst, der *Wertebereich der Eigenschaft* und die möglichen *Werte* dieses Wertebereichs modelliert werden.

```
ex:MeiersFluss od:hasFlowType tc:discreteFlow .

od:hasFlowType a rdfs:Property;
  rdfs:domain tc:Flow;
  rdfs:range tc:FlowType .

tc:FlowType a skos:Concept, od:AdditionalConcept ;
  od:allowedValues tc:complexFlow, tc:discreteFlow, tc:continousFlow ;
  skos:prefLabel "Type of flow"@en, "Flussart"@de ;
  skos:altLabel "Art der Strömung"@de .

tc:discreteFlow a skos:Concept, od:AdditionalConcept;
  od:valueOf tc:FlowType ;
  skos:prefLabel "discrete flow"@en, "Diskreter Fluss"@de .

...
```

Das Ganze ist schwer abzugrenzen von einer Klassenhierarchie, die mit Prädikatverfeinerungen `od:hasSubConcept` von `skos:narrower` modelliert werden sollten. Das dazu inverse Prädikat `od:subConceptOf` wird aus Gründen der Übersichtlichkeit ebenfalls verwendet; es kann aber später auch maschinell ergänzt werden.

```
ex:MeiersFluss od:hasStaticFlowComponent tc:Pump .

od:hasStaticFlowComponent a rdfs:Property;
  rdfs:domain tc:Flow;
  rdfs:range tc:StaticFlowComponent .

tc:StaticFlowComponent
  od:hasSubConcept tc:ControlUnit, tc:Receiver, tc:Source, tc:Channel ;
  a skos:Concept, od:AdditionalConcept ;
  skos:prefLabel "static components of the flow"@en,
    "statische Flusskomponenten"@de .

tc:ControlUnit
  od:subConceptOf tc:StaticFlowComponent ;
  od:hasSubConcept tc:Pump, tc:Valve ;
  skos:prefLabel "Control Unit"@en, "Steuerungssystem"@de ;
  skos:altLabel "Management System"@en, "Managementsystem"@de .

tc:Pump
  od:subConceptOf tc:ControlUnit ;
  skos:prefLabel "pump"@en, "Pumpe"@de ;
  a skos:Concept, od:AdditionalConcept .
```

Hier wäre zunächst abzugrenzen, ob `tc:Pump` mit in die Menge der Konzepte aufgenommen wird oder der Begriff zu speziell ist. Das ist natürlich eine Geschmacksfrage.

Die Modellierung über Klassenhierarchien hat den Vorteil, dass man das Objekt des Prädikats `od:hasStaticFlowComponent` mit Instanzen der gesamten Klassenhierarchie belegen kann. Dessen genauen Typ kann man dann nur über eine zweite Modellanfrage ermitteln. Alternativ hätte man in diesem Beispiel mit Prädikaten `od:hasStaticFlowComponent` und `od:hasControlUnit` arbeiten können, um den Objekttyp bereits am Prädikatbezeichner unterscheiden zu können.