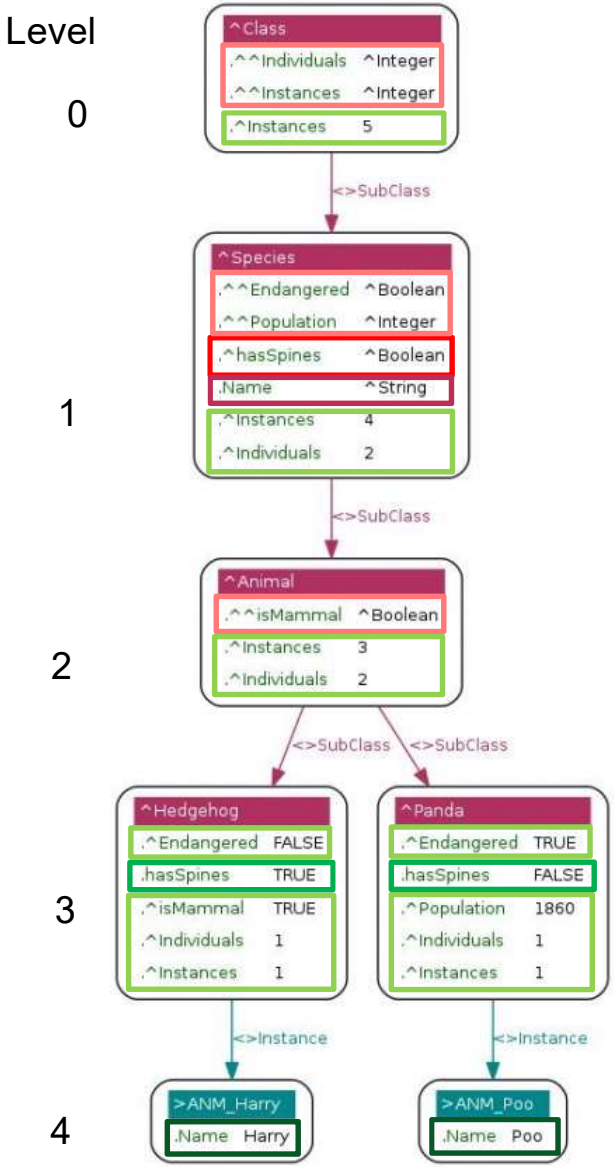
Neue Aspekte des Conceptual (Meta-)Modeling

- Dipl.-Inform. Hermann Bense
  hb@bense.com

  ontology4.us
  schematik.de
  predicator.name
  schreib-maschine.info

  [bense.com] Verlagsgesellschaft
  für Digitales Publizieren GmbH
  Schwarze-Brüder-Straße 1
  44137 Dortmund

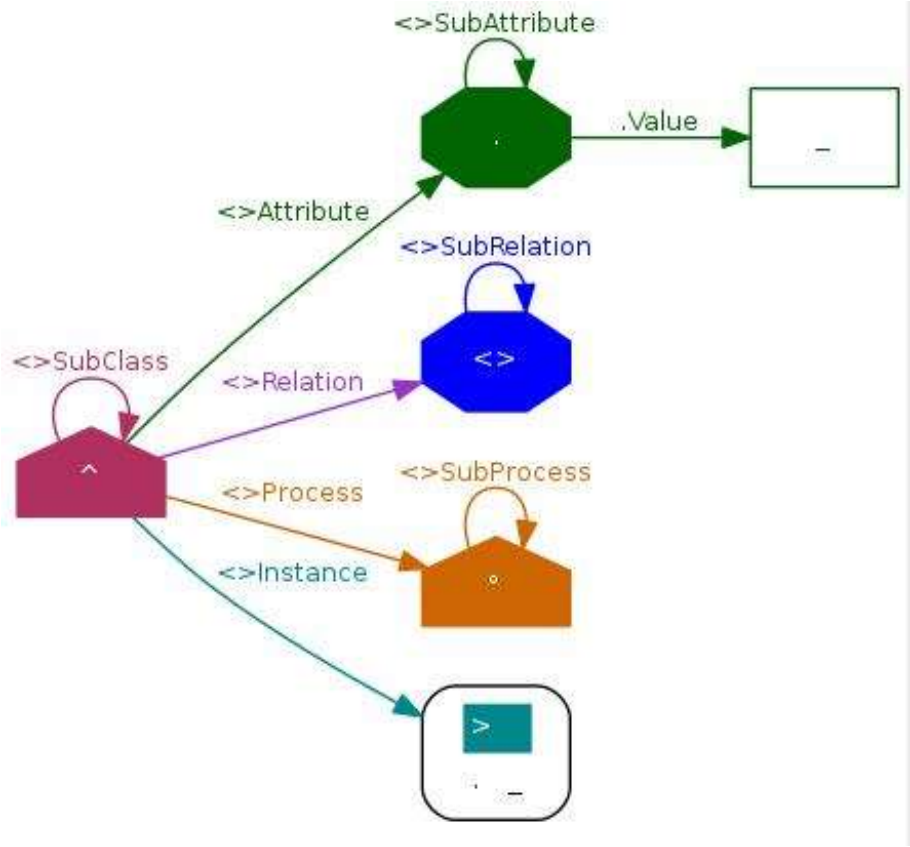


Ziele des neuen Conceptual (Meta-) Modeling

- Vereinfachung der konzeptuellen Modellierung und Modelle
  - Maximal 2 - 3 Abstraktionsebenen
- Reduktion der Anzahl der benötigten Wissenselemente (Tripel in der Ontologie)
  - unter Beibehaltung des semantischen Gehaltes
- Verbindlichere Richtlinien für
  - Namensgebung (Semiotik, Prädikatoren)
    - Vermeidung von Mehrdeutigkeiten (Homonyme)
  - Die Methodik der Modellierung
    - Duale Modellierung von Konzepten (Ausprägung als Unterklassen vs. Ausprägung als DataProperties)
- Bessere Unterstützung bei der Qualitätssicherung der Modelle und Ontologien durch
  - Vermeidung von „layer mistakes“
  - Visualisierung der Knowledge-Graphen
  - klare Unterscheidung zwischen Instanzen und Individuen
- Klare Regeln für die
  - Instanziierung
  - Verbindung zwischen Klassen und Individuen

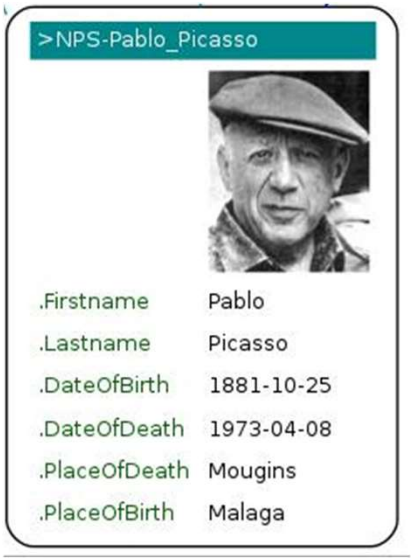
Namenskonventionen und graphische Notationen

- **^** ^Class, ^Category, ^Universal (something, which can have attributes)
- Properties
  - **<>** <>Relation, <>Relativum, <>Nexus, ObjectProperty (OP) (connects different knowledge subjects)
  - **.** .Property, Data Property (DP), internal Attribute (e.g. .Name, .Length)
  - **.^** .^ClassDataProperty (CDP, e.g. .^hasSpines)
  - **.^^** .^MetaClassDataProperty (CDP, e.g. .^^Endangered)
- **_** PropertyValue (e.g. "January", TRUE, "solid")
- **>** >Instance, >Particular, >Perdurant
- **°** °Process, °Activity, °Occurent (Prozess, Algorithmus, Verfahrensvorschrift)
- **~** ~Persistence (ongoing Activity, true fact)
- **>°** >ProcessInstance (e.g. building the Eiffeltower)
- **>>** >>Relator, represents n-ary relationships between >Instances and other knowledge subjectss;
  - Spezialfall: Reifizierte Wissenssubjekte (knowledge subjects)



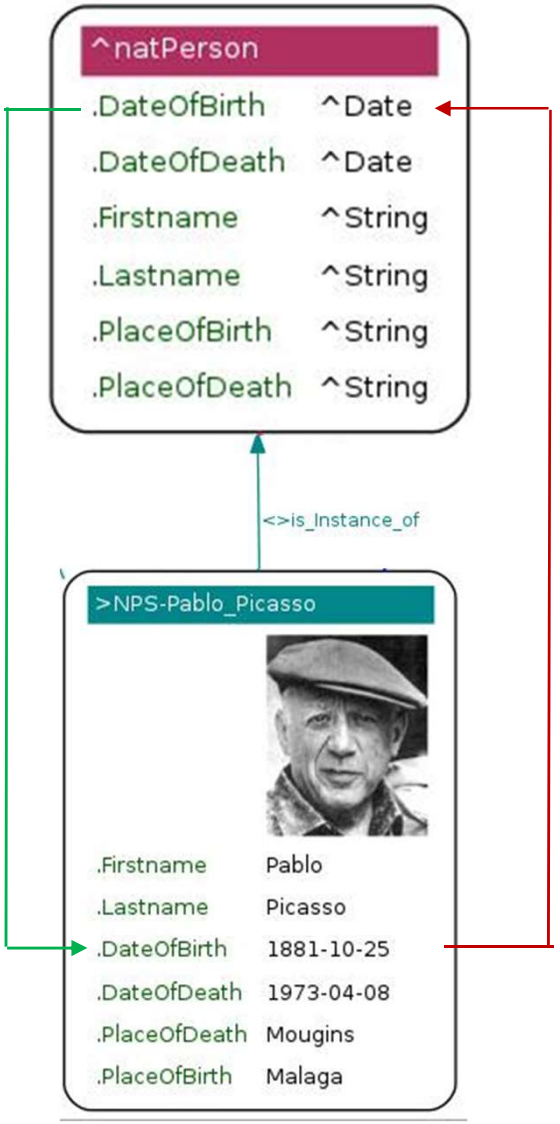
Beispiel Ontologie
Individuum „Pablo Picasso“

- Individual >NPS-Pablo_Picasso
  - Mit Merkmalen (Data Property)
    - .Firstname, .Lastname, .DateOfBirth, .DateOfDeath, .PlaceOfDeath, .PlaceOfBirth
    - .Image
- Speicherung in der Ontologie als Tripel
    - (>NPS-Pablo_Picasso, .Firstname, Pablo)
    - (>NPS-Pablo_Picasso, .Lastname, Picasso)
    - …
    - Allgemein (s, p, o)



Beispiel Ontologie Schematisierung

- Mengen von Individuen mit gleichen Merkmalen werden durch eine Klasse beschrieben
  - Beispiel ^natPerson (natürliche Person)
  - Eine Klasse enthält u.a. DataProperty-Definitionen der Art
    - (^natPerson, .Firstname, ^String)
    - …
    - (^natPerson, .DateOfBirth, ^Date)
    - …
  - Def. DPD = DataPropertyDefiniton
    - (s, p, o) mit s ist Klassenbezeichner, p ist DataProperty-Bezeichnung und o = AtomicDataType wird als DPD bezeichnet, mit
    - AtomicDataType := {^String, ^anyURI, ^Boolean, ^Date, ^DateTime, ^Integer, ^Decimal, ^Float}
  - Instanziierung
    - (>NPS-Pablo_Picasso, <>is_Instance_of, ^natPerson) oder
    - (^natPerson, <>has_Instance, >NPS-Pablo_Picasso)
    - (^natPerson, .DateOfBirth, ^Date)
      (>NPS-Pablo_Picasso, .DateOfBirth, 1881-10-25)
  - Klassifizierung / Schematisierung

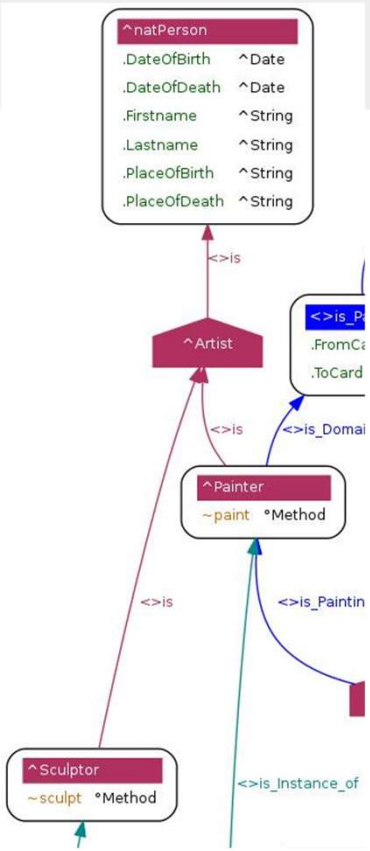


- Diskussion/Abstimmung:
  - Entspricht eine Klasse bzw. ein Universal einem *Begriff* (Englisch: *concept*), bzw. spezieller, einem Begriff der FBA?

Beispiel Ontologie
Klassenhierarchie und Vererbung

- **Beispiele**
  - (^Artist, <>is, ^natPerson)
  - (^Sculptor, <>is, ^Artist)
  - (^Painter, <>is, ^Artist)
  - **Wobei semantisch equivalent gilt:**
    - (^natPerson, <>SubClass, ^Artist)
    - (^Artist, <>SubClass, ^Sculptor)
    - (^Artist, <>SubClass, ^Painter)
  - **Mit**
    - (<>is, <>is_Inverse_of, <>SubClass)
    - (<>SubClass, <>is_Inverse_of, <>is)
    - Und
    - (<>Inverse, <>is_Inverse_of, <>is_Inverse_of)

- **Daher gilt**
  - **Eine Oberklasse ist allgemeiner als jede ihrer Unterklassen**
  - **Eine Klasse erbt alle Data-Property-Definitionen aller ihrer Oberklassen**
    - Z.B. ^Painter erbt von ^Artist, ^Artist erbt von ^natPerson



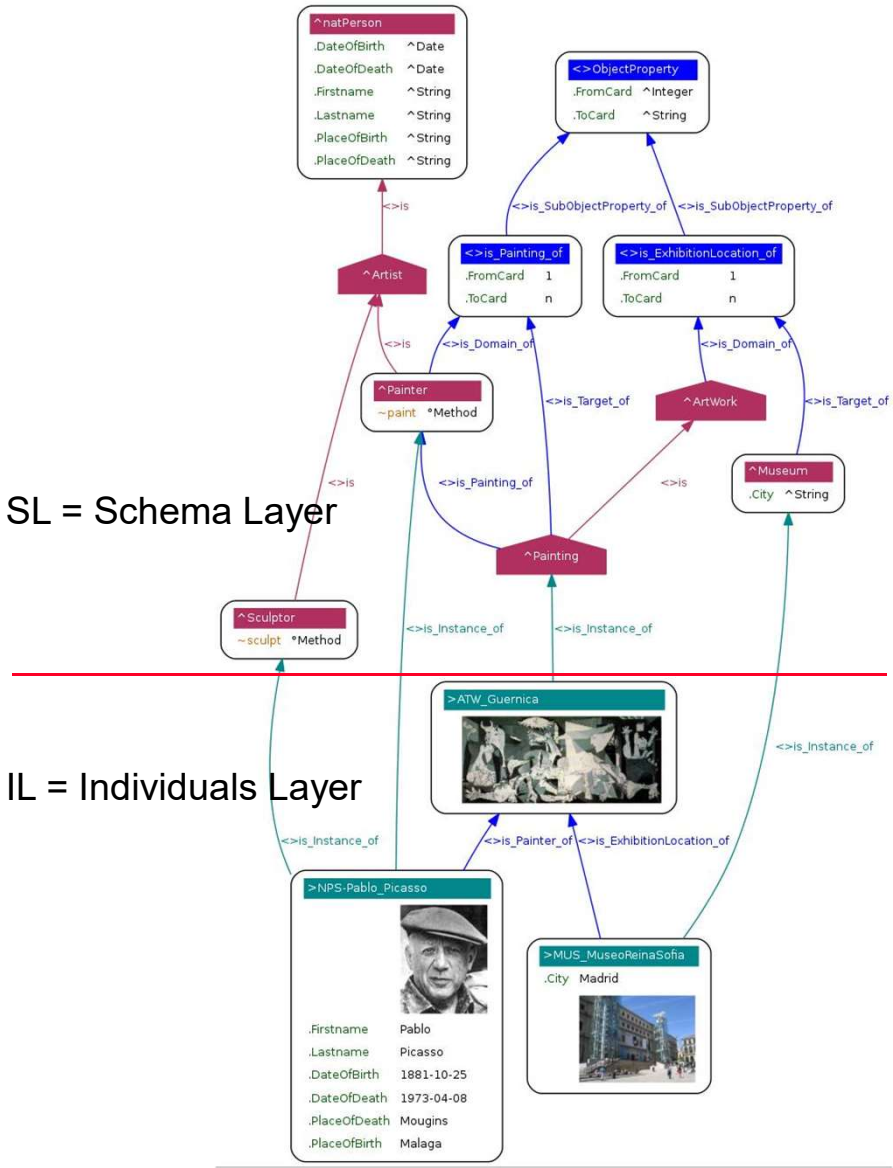
- ClassHierarchy (s) == ch(s) =
  {s} U sbc(s) U spc(s)
  - Die Klassenhierarchie von s ist die Vereinigungsmenge aus der Klasse s und ihren Ober- und Unterklassen.

Beispiel Ontologie „Pablo Picasso“

- **Beispiel Mehrfachvererbung**
  - (^NPS-Pablo_Picasso, <>is_Instance_of, ^Sculptor)
  - (^NPS-Pablo_Picasso, <>is_Instance_of, ^Painter)

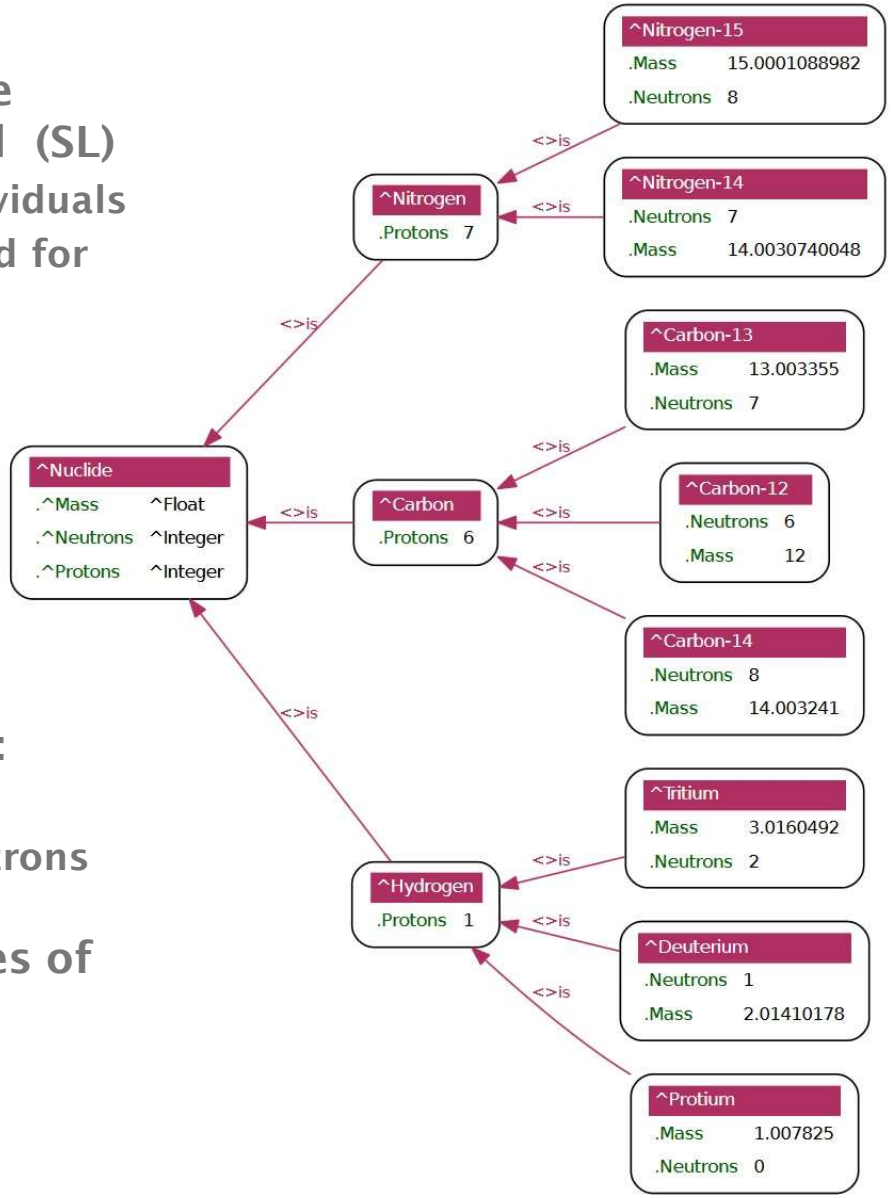
- **Beispiel ObjectPropertyDefinition**
  - **(^Museum, <>is_ExhibitionLocation_of, ^Artwork)**
  - **mit**
    - (^Museum, <>is_Domain_of, <>is_ExhibitionLocation_of)
    - (^Artwork, <>is_Target_of, <>is_ExhibitionLocation_of)
    - (<>is_ExhibitionLocation_of, .FromCard, 1)
    - (<>is_ExhibitionLocation_of, .ToCard, n)

  - **Def. OPD = ObjectPropertyDefinition (Relationship Type)**
    - Eine OPD definiert externe Beziehungen zwischen Wissenssubjekten.
    - .FromCard = Minimale/Maximale Zahl von Domain-Subjekten
    - .ToCard = Minimale/Maximale Zahl von Range-Subjekten
    - Mit .FromCard <= .ToCard
  - **Beispiele:**
    - (>MUS_MuseoReinaSofia,<>is_ExhibitionLocation, ^ATW_Guernica)
    - (>NPS-Pablo_Picasso, <>is_Painter_of, ^ATW_Guernica)



Nuclides Meta Class Hierarchy using Class Data Properties (CDP)

- Class Data Properties (CDP) are restricted to the instantiation of sub-classes on the schema level (SL)
  - they implement kind of virtual properties for individuals
  - Advantage: the property value has not to be stored for every individual
  - CDPs have .^ as prefix for property names, e.g. .^Mass, .^Protons, .^Neutrons
- Type definitions of a CDP:
  - (^Nuclide, .^Mass, ^Float)
  - (^Nuclide, .^Neutrons, ^Integer)
  - (^Nuclide, .^Protons, ^Integer)
- Instantiation of a CDP on the schemal layer (SL):
  - (^Carbon-14, .Neutrons, 8)
  - => each instance of a ^Carbon-14 atom has 8 neutrons
- Classes under ^Nuclide form the set of instances of the ^Nuclide class
  - Sample Query: get all nuclides with 8 neutrons => {^Nitrogen-15, ^Carbon-14}

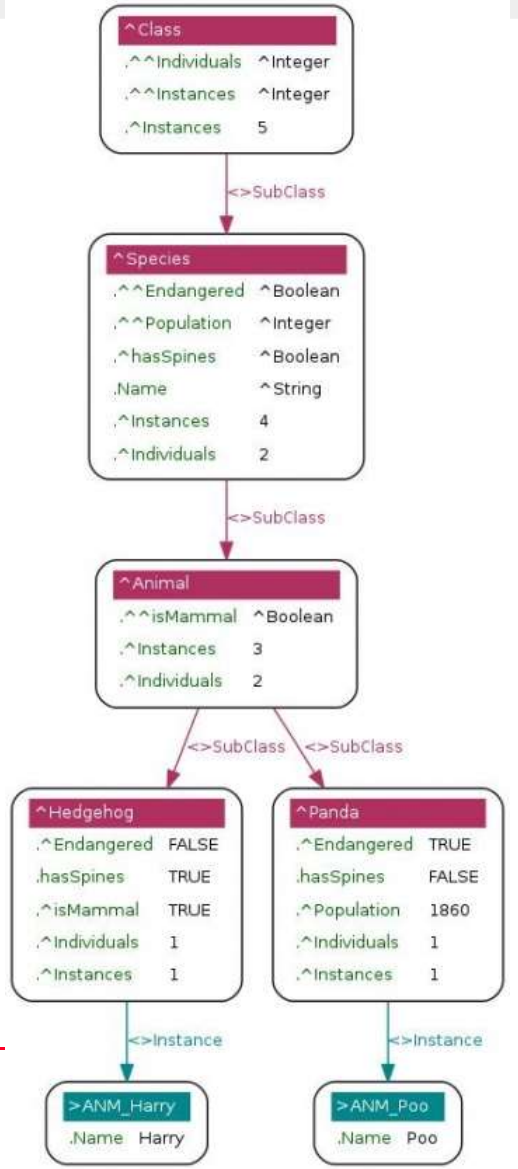


Class Data Properties (CDP) & Meta Class Data Properties (MCDP)

- Meta Class Data Properties (MCDP) are also restricted to the instantiation of sub-classes on the schema level (SL) and implement properties of meta classes
- MCDPs have .^^ as prefix for property names, e.g. .^^Endangered, .^^Population …
- Type definition of a MCDP:
  - (^Species, .^^Endangered, ^Boolean)
  - (^Species, .^^Population, ^Integer)
- Instantiation of a MCDP on SL:
  - (^Panda, .^Endangered, TRUE)
  - => only the species ^Panda is endangered and not any instance of a ^Panda
- Instantiation of a CDP on SL:
  - (^ Hedgehog, .hasSpines, TRUE)
  - => every ^Hedgehog has spines

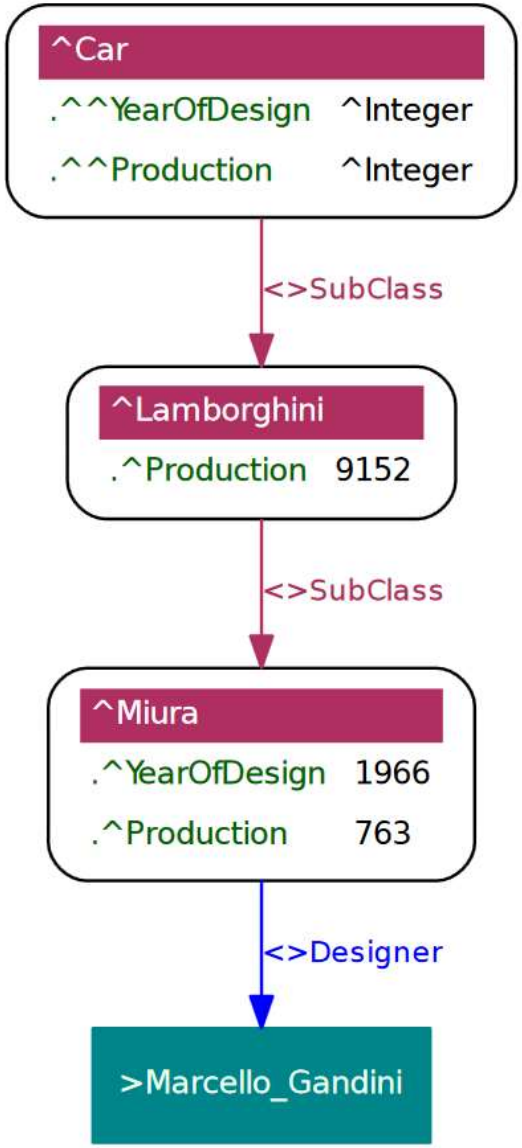
SL = Schema Layer

IL = Individuals Layer



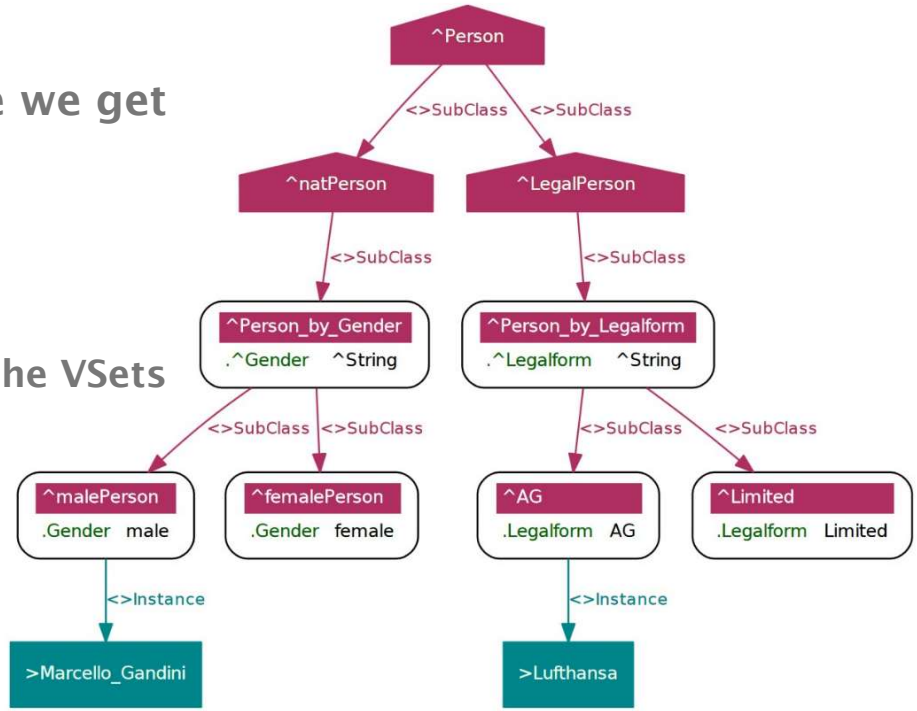
Relationships instances between different classes, meta classes and individuals

- Relationships between class, meta-classes and individuals are allowed
- .^^YearOfDesign is a MCDP (Meta Class Data Property) of type ^Integer
  - (^Car, .^^YearOfDesign , ^Integer)
- ^Miura is a meta class instance of ^Lamborghini with the instantiation of the MCDP .^^YearOfDesign
  - (^Miura, .^YearOfDesign , 1966)
- The designer of the car model ^Miura is the individual >Marcello_Gandini
  - (^Miura, <>Designer, >Marcello_Gandini)
- More Examples
  - (>Yo-Yo_Ma, <>is_Expert_of, ^Violin)
  - (>Albert_Einstein, <>is_Creator_of, ^Special_Theory_of_Relativity)



Partitioning of Classes by data properties

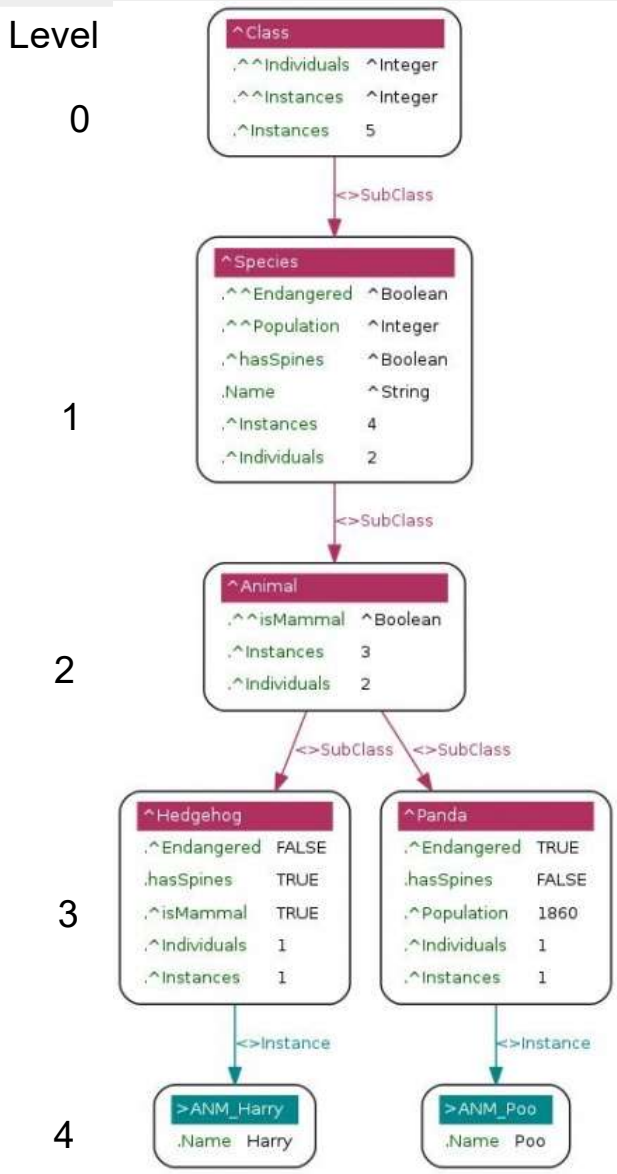
- Example candidates for the partitioning of class ^Person are the data properties .Gender and .Legalform
  - VSet(.Gender) = {male, female}
  - VSet(.Legalform) = {Limited, AG, GmbH … }
- With c as class name and dp as property name we get for the name of a fully partitioning class:
  - ^Person_by_Gender
  - ^Person_by_Legalform
  - with the naming schema ^c_by_dp
  - Where names of sublasses include values form the VSets e.g. ^malePerson, ^femalePerson
- Type definition and instantiation of a CDP:
  - (^Person_by_Gender, .^Gender, ^String)
  - (^malePerson, .Gender, male)
    =>
  - >Marcello_Gandini is a male Person
- Advantage:
  - it is not neccessary to have extra relationship instances like (^Person_by_Gender, <>partitions, ^Person)



Summary for
A „property-centric“ approach for Multi Level Meta Modelling (MLMM)

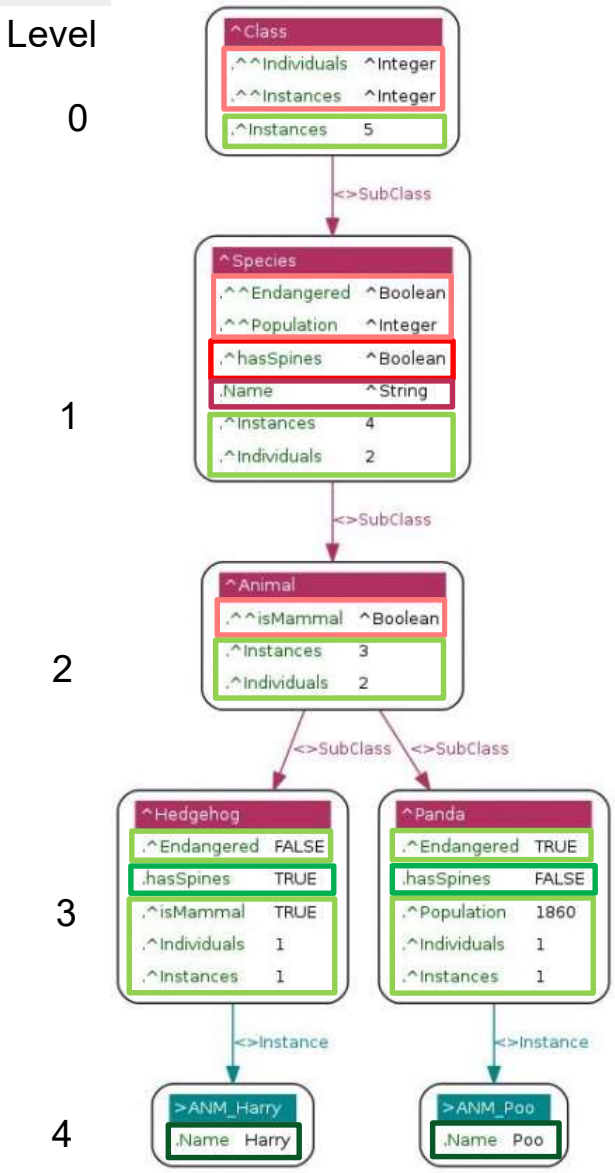
[ontology4.us]

- New types of data properties introduced with new instantiation rules for classes and meta classes
- Naming conventions introduced for
  - data property names
  - the partitioning of classes
- Meta Class Data Properties
  - .^^Individuals: gibt die Anzahl der Instanzen auf dem Individuals Layer (IL) an
  - .^^Instances: gibt die Anzahl der Instanzen von Klassen auf dem Schema Layer (SL) an
  - Dual Facette Behaviour: Eine Klasse kann gleichzeitig Unterklasse und Instanz einer anderen Klasse sein
- Advantages for conceptual modeling:
  - lesser classes needed without losing semantics
  - conceptual models easier to understand
  - only 2 abstraction layers needed (Data Property Defintions & Data Property Instances)



Summary for
Layers and Levels

- Only 2 Layers of Abstraction
- 1: Property Definitions
  - DP-Definitions
  - CDP-Definitions
  - MCDP-Definitions
- 2: Property Instantiations
  - DP-Instantiations
  - CDP-Instantiations
  - MCDP-Instantiations
- Instantiation Restrictions
  - Data Properties (DP) can only be instantiated into Individuals
  - Class Data Properties (CDP) can only be instantiated into Classes AND Individuals
  - Meta Class Data Properties (MCDP) can only be instantiated into Classes

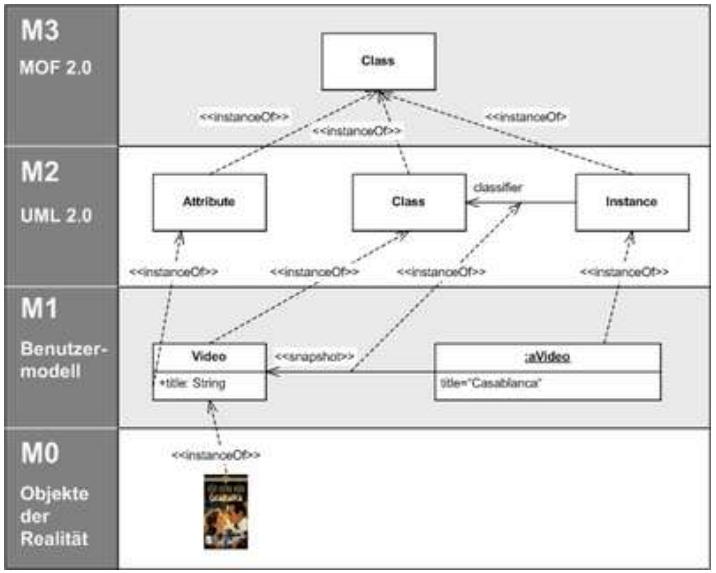


Maximale Ebenen (Layers) der Abstraktion = 2

- **Allgemeine DataProperty-Definition (DPD)**
  - (^Class, $Property, ^AtomicDataType)
  - Wobei $Property ein einfaches DataProperty (DP), ein ClassDataProperty (CDP) oder ein MetaClassDataProperty (MCDP) sein kann, also $ ∈ {., .^, .^^}

- Beispiele DPD und Instanziierung =>
  - DP: **(^natPerson, .DateOfBirth, ^Date) => (>NPS-Pablo_Picasso, .DateOfBirth, 1881-10-25)**
  - CDP: **(^Nuclide, .^Protons, ^Integer) => (^Carbon-14, .Protons, 8)**
  - MCDP: **(^Species, .^^Endangered, ^Boolean) => (^Panda, .^Endangered, TRUE)**

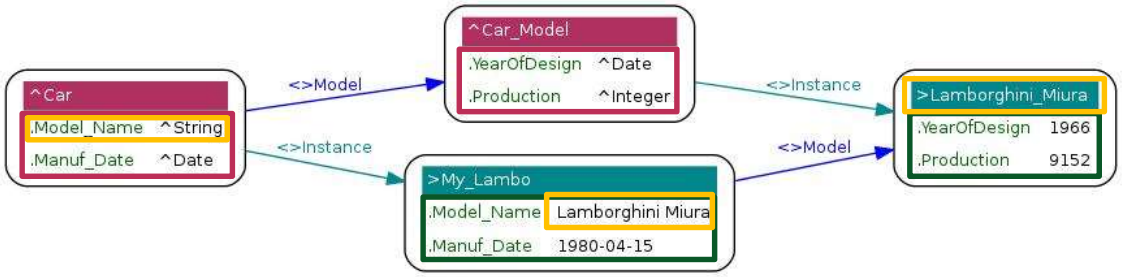
- **Daher: Maximale Ebenen der Abstraktion = 2**
  - **Es gibt also nur den Layer der DataProperty-Definitions und den Layer der DataProperty-Instanzen.**
  - **Die meisten anderen Ansätze wie UML (MOF = Meta Object Facility) gehen von 3 oder 4 Leveln der Abstraktion aus (0..3)**
    - Warum sind im MOF Attribute, Class und Instance Instanzen von Class?



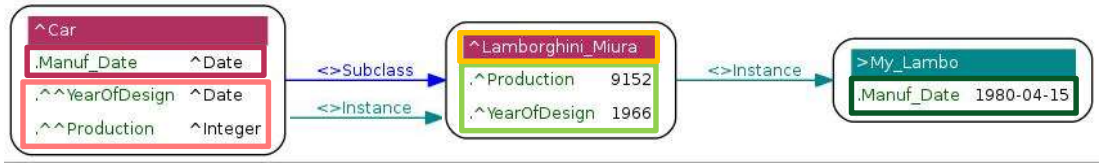
https://de.wikipedia.org/wiki/Meta_Object_Facility

Consolidation of conceptual modeling with respect to class pairs of type (^Car, ^Car_Model)

- Examples for class/meta class pairs
  - (^Car, ^Car_Model)
  - (^Panthera, ^Panthera_Species)
  - (^Phone, ^Phone_Model)
  - (^Person, ^Person_by_Gender)
  - Design choice for data properties:
    - (^Person, .Gender, ^String)
    - (^Person_by_Gender, .^Gender, ^String)



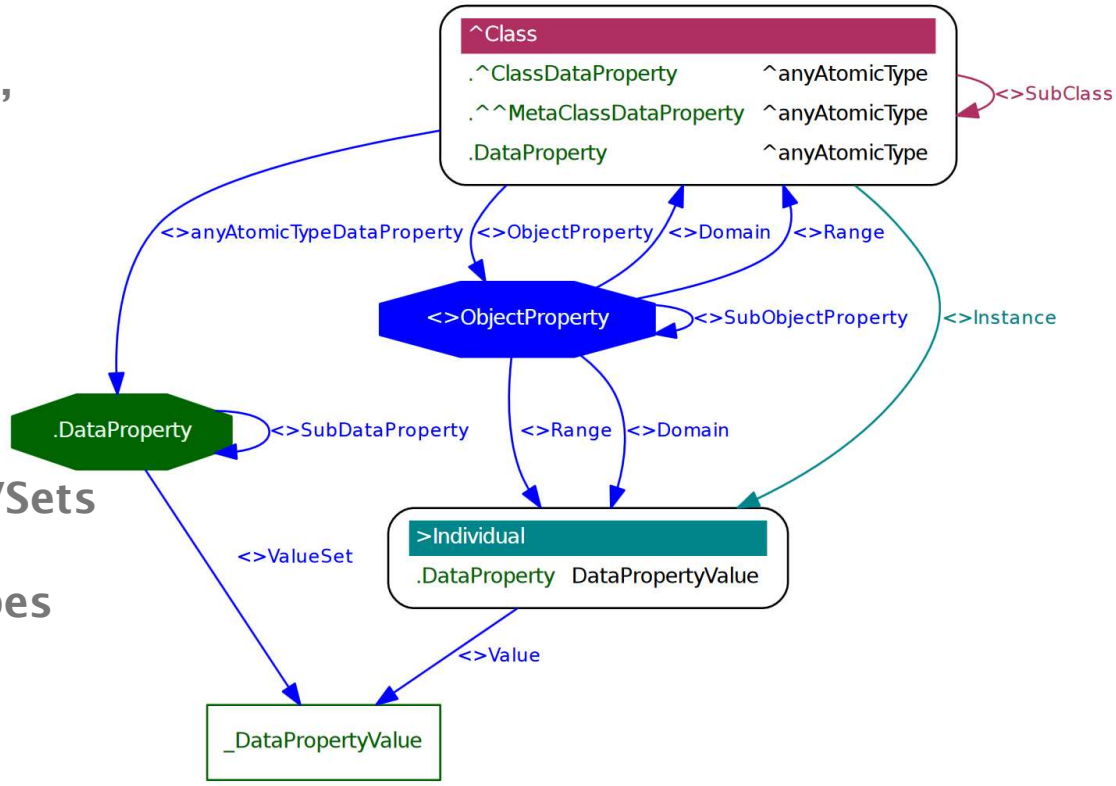
- Consolidation scheme for class pair (^Car, ^Car_Model) [PiZi1994]
  - (^Car, .DateOfProduction, ^Date)
  - (^Car_Model, .YearOfDesign, ^Integer)
  - (^Car_Model, .Production, ^Integer)
  - are merged into one class
    - (^Car, .DateOfProduction, ^Date)
    - (^Car, .^^YearOfDesign, ^Integer)
    - (^Car, .^^Production, ^Integer)
  - by the transformation of data properties of the model class ^Car_Model into class data properties of the base class ^Car
  - Advantages:
    - lesser classes, individuals, data properties and object properties (<>Model) needed
    - conceptual models easier to understand



[PiZi1994] *Alain Pirotte, Esteban Zimanyi, David Assart, Tatiana Yakusheva, Materialization: a powerful and ubiquitous abstraction pattern, VLDB, Morgan Kaufmann, 1994 , pp. 630-641*

AMO - Abstract Meta Ontology

- Classes are the prime modeling entities and aggregate Data- & Object properties
- AtomicType: String, Number, Boolean, Date, Time, anyURI …
  - .Cityname: ^String
- VSet (.CityName) = {Berlin, Hamburg, London, Tokio …}
- Individuals are instantiated from Classes and aggregate Values from VSets
- Hierarchies built by Relationship Types <>SubClass, <>SubObjectProperty, <>SubDataProperty
- Domain and Ranges of Relationship-Types can be Classes or Individuals



ACHTUNG: Die AMO dient ausschließlich dazu, die syntaktische Korrektheit von konkreten Ontologien zu definieren und zu überprüfen. Sie ist also sozusagen eine Blaupause für die Ausprägung von Ontologien.

Danke für das Interesse!

Referenzen

[AtKu2003] Colin Atkinson, Thomas Kühne, **Model-Driven Development: A Metamodeling Foundation**, IEEE Software 20, **2003** , **pp. 36-41**

•[Bens2014] Hermann Bense, **The Unique Predication of Knowledge Elements and their Visualization and Factorization in Ontology Engineering**, **2014** IOS Press, Amsterdam, in [GaKu2014] , **pp. 241-250**

•[BrAl2016] Freddy Brasileiro, Joao Paulo A. Almeida, Victorio A. Carvalho, Giancarlo Guizzardi, **Expressive Multi-Level Modeling for the Semantic Web**, **2016**

•[BrAl2016b] Freddy Brasileiro, Joao Paulo A. Almeida, Victorio A. Carvalho, Giancarlo Guizzardi, **Applying a Multi-Level Modeling Theory to Assess Taxonomic Hierarchies in Wikidata**, Wiki Workshop at 25th Int. Conference Companion World Wide Web, **2016** , **pp. 975-980**

•[CaAl2016] Victorio A. Carvalho, Joao Paulo A. Almeida, **Toward a Well-Founded Theory for Multi-Level Conceptual Modeling**, **2016** (Download: http://nemo.inf.ufes.br/mlt)

•[Chen1976] Peter Pin-Shan Chen, **The Entity-Relationship Model - Toward a Unified View of Data**, ACM Transactions on Database Systems, Vol.1, No.1, March 1976, **1976** , **pp. 9-36** (Download: http://csc.lsu.edu/news/erd.pdf)

•[GaKo2009a] Emden R. Gansner,Eleftherios Koutsofios, Stephen North, **Drawing graphs with dot**, **2009** (Download: http://www.graphviz.org/pdf/dotguide.pdf)

•[GaKu2014] Pawel Garbacz, Oliver Kutz, **Formal Ontology in Information Systems, Proceedings of the Eighth International Conference (FOIS 2014), Rio de Janeiro, Brazil, Sept. 22-25, 2014**, IOS Press, Amsterdam, ISBN: 978-1-61499-437-4, **2014**

•[GuAl2015] Giancarlo Guizzardi, Joao Paulo A. Almeida, Nicola Guarino, Victorio A. Carvalho, **Towards an Ontological Analysis of Powertypes**, International Workshop on Formal Ontologies for Artificial Intelligence (FOFAI), **2015** (Download: https://www.researchgate.net/publication/279178175_Towards_an_Ontological_Analysis_of_Powertypes)

•[NeSc2009] Bernd Neumayr, Katharina Grün, Michael Schrefl, **Multi-Level Domain Modeling with M-Objects and M-Relationships**, 6th Asia-Pacific Conference on Conceptual Modeling, **2009** (Download: https://www.researchgate.net/publication/220268481_Multi-Level_Domain_Modeling_with_M-Objects_and_M-Relationships)

•[Odel1994] James J. Odell, **Power Types**, Journal of Object-Oriented Programming, Volume 7(2), **1994** , **pp. 8-12**

•[PaHo2001] Jeff Z. Pan, Ian Horrocks, **Metamodeling Architecture of Web Ontology Languages**, Proc. of the 2001 International Semantic Web Working Symposium, **2001** , **pp. 131-149**

•[PiZi1994] Alain Pirotte, Esteban Zimanyi, David Assart, Tatiana Yakusheva, **Materialization: a powerful and ubiquitous abstraction pattern**, VLDB, Morgan Kaufmann, **1994** , **pp. 630-641**

Related Work from RDF/RDFS, OWL, UML, OMG

The Unified Modelling Language (OMG) is a standard object-oriented design language and has a four-layer metamodeling architecture. The lowest layer of *User Objects* are instances of classes, which we call individuals.
Dies ist im Widerspruch zu den folgenden Definitionen:

- An individual is a First-Order-Type/Class (1stOrderClass),
- a Second-Order-Type/Class (2ndOrderClass) instantiates entities of 1stOrderClass,
- a 3rdOrderClass instantiates entities of a 2ndOrderClass

[AtKu2003] Colin Atkinson, Thomas Kühne, ***Model-Driven Development: A Metamodeling Foundation****, IEEE Software 20,* ***2003*** *,* ***pp. 36-41***
•*[BrAl2016] Freddy Brasileiro, Joao Paulo A. Almeida, Victorio A. Carvalho, Giancarlo Guizzardi,* ***Expressive Multi-Level Modeling for the Semantic Web****,* ***2016***

- In the Semantic Web, a metaclass is a class whose instances are classes.
- Classes are divided into "fixed order classes" and "variable order classes".
- In the case of the former, an order is attributed for metaclasses by measuring the distance to individuals with respect to the number of **"instance of" triples** that are necessary to find an individual.
- Classes that are not metaclasses are classes of individuals, so their order is "1".
- Metaclasses that are classes of first order classes' order is "2", and so on.
- Variable order metaclasses, on the other hand, can have instances, one example of variable order metaclass is the class of all fixed order classes.

https://en.wikipedia.org/wiki/Metaclass_(Semantic_Web)

Pan and Horrocks critizise, that RDF Schema, as a schema layer Semantic Web Language, has a non-standard metamodeling architecture and lacks clear semantics and lends itself to "**layer mistakes**":

> *"Although you can define class and subClassOf as resources in RDF, RDF provides no standard mechanism for declaring classes and (global) properties, nor does it provide any mechanisms for defining the relationships between properties and between classes."*

[PaHo2001] Jeff Z. Pan, Ian Horrocks, ***Metamodeling Architecture of Web Ontology Languages****, Proc. of the 2001 International Semantic Web Working Symposium,* ***2001*** *,* ***pp. 131-149***

Requirements for Multi Level Meta Modeling (MLMM) defined in MLT (Multi-Level Modeling Theory)

- **R1: Represent entities of multiple (related) classification levels capturing chains of instantiations between the involved entities**
  - Classification levels are defined by the 3 different type of data properties:
    - DP = base data property
    - CDP = class data property
    - MCDP = meta class data property
- **R2: define principles for the organization of entities into levels**
  - IL = Individuals Layer
  - SL = Schema Layer (SL classes may be 1stOrderClasses or 2ndOrderClasses)
  - AMO = Abstract Meta Ontology
- **R3: capture rules for the instantiation of entities into levels**
  - For each type of data properties different instantiation rules are defined
- **R4: allow the representation of domain relations between entities in different classification levels**
  - There are no restrictions regarding the representation of domain relations between entities in different levels.

Ontology Layers

- **Def. AKE = Atomic Knowledge Entity = Wissensatom**
  - Jedes Tripel t_3 =(s, p, o) ∈ KB wird als atomare Wissenseinheit (AKE = Atomic Knowledge Entity = Wissensatom) bezeichnet.
- **Def. KS = KnowledgeSubject**
  - Ein KnowledgeSubject ist die Menge aller atomaren Wissenseinheiten (AKE) mit gleichem Subjekt:
    KS (x) = { (s, p, o) ∈ KB | s = x},
  - z.B. KS (>NPS_Pablo_Picasso); KS(^natPerson)
- SL == Schema Layer := Menge aller Universals/Klassen = {KS (n) | n ist Bezeichner einer Klasse oder Property}
- IL == Individuals Layer := Menge aller Individuals = {KS (i) | i ist Bezeichner eines Individuals}
- I == Individual := Instanz einer Klasse im IL
- CI == Class Individual = Instance einer Klasse im SL
- **Ontology** KG := SL ∪ IL, mit SL ∩ IL = ∅,
  wobei KG ⊆ S x P x O