


## Applied Ontology Engineering (703817, VU2)

### Unit 2: Ontology Formalisms and Ontology Tools

Dr. Martin Hepp

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/




## Course Logistics (1)

<http://www.heppnetz.de/teaching/aoe/>

- **Lecturer:** Dr. Martin Hepp
- Lecture with Hands-on Sessions (VU2) for Master Students
- **Classroom:** 3W03 (Video conference room, TransIT building, level "02" (3rd floor in American counting))
- **Time:** starting March 6th, Mon, 13:00 - 15:00 plus **one extra unit July 3, 12.00 noon – 6.00 p.m.**
- **Registration:** <https://www.uibk.ac.at/zid/basicauth/validate.pl>


© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/



## Course Logistics (2)

- **Classes**
  - Mar 6: Unit 1
  - Mar 13: **no classes – make-up on July 3**
  - Mar 20: **no classes – make-up on July 3**
  - Mar 27: Unit 2
  - April 3: **no classes – make-up on July 3**
  - April 24: Unit 3
  - May 8: Unit 4
  - May 15: Unit 5
  - May 22: Unit 6
  - May 29: Unit 7
  - Jun 12: **no classes – make-up on July 3**
  - Jun 19: Unit 8
  - Jun 26: Unit 9
  - July 3: Unit 10: Final Presentations (mandatory) **12.00 noon – 6.00 p.m.** (also make-up for missed classes)


© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/



## Preliminary Syllabus

- Unit 1:** Ontology Engineering Fundamentals
- Unit 2:** Ontology Formalisms: **RDF-S and WSML / Ontology Tools: WSMT and SWEDE**
- Unit 3:** Ontology Engineering Methodologies
- Unit 4:** OntoClean including Exercises
- Unit 5:** Introduction to Description Logics / Pitfalls – The OWL Pizza Experiments
- Unit 6:** Ontology Mapping, Merging, Alignment, and Exercises
- Unit 7:** Ontology Engineering Exercises 1
- Unit 8:** Ontology Engineering Exercises 2
- Unit 9:** Ontology Engineering Exercises 3
- Unit 10:** Group Presentations


© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/



## Assignments and Grading

- **Two Quizzes (25 %)**
  - Covers mandatory reading
  - Quiz 1: March 27
  - Quiz 2: to be announced
  - No make-up possible (except in case of absence due to medical reasons, to be certified by a medical doctor)
- **Individual Assignments (40 %)**
  - One or more (small) WSML ontology/ontologies plus documentation
- **Group Assignment (35 %)**
  - One (big) WSML ontology plus documentation plus presentation


© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/



## Resources

- **Textbook (mandatory)**  
Gómez-Pérez/Fernández-López/Corcho:  
*Ontological Engineering*  
Springer 2004
- **Additional Materials (mandatory)**  
Will be made available or announced in class
- **Course Web Page**  
<http://www.heppnetz.de/teaching/aoe>

Slides will be made available on the Web page **after (!)** the respective unit.





© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/




# Ontology Formalisms: RDF-S

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 7






# RDF-S as an Ontology Formalism

- Important RDF-S Classes
  - **rdfs:Resource**
    - All things described by RDF are called resources, and are instances of the class **rdfs:Resource**. This is the class of everything. All other classes are subclasses of this class. **rdfs:Resource** is an instance of **rdfs:Class**.
  - **rdfs:Class**
    - This is the class of resources that are RDF classes. **rdfs:Class** is an instance of **rdfs:Class**.
  - **rdfs:Literal**
    - The class **rdfs:Literal** is the class of literal values such as strings and integers. Property values such as textual strings are examples of RDF literals. Literals may be plain or typed. A typed literal is an instance of a datatype class.
  - **rdfs:Datatype**
    - **rdfs:Datatype** is the class of datatypes. All instances of **rdfs:Datatype** correspond to the RDF model of a datatype described in the RDF Concepts specification [RDF-CONCEPTS]. **rdfs:Datatype** is both an instance of and a subclass of **rdfs:Class**. Each instance of **rdfs:Datatype** is a subclass of **rdfs:Literal**.
  - **rdf:Property**
    - **rdf:Property** is the class of RDF properties. **rdf:Property** is an instance of **rdfs:Class**.

Source: <http://www.w3.org/TR/rdf-schema/>

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 8






# RDF-S: Properties

- **rdfs:range**
  - **rdfs:range** is an instance of **rdf:Property** that is used to state that the values of a property are instances of one or more classes.
  - The triple **P rdfs:range C** states that **P** is an instance of the class **rdf:Property**, that **C** is an instance of the class **rdfs:Class** and that the resources denoted by the objects of triples whose predicate is **P** are instances of the class **C**.
  - Where **P** has more than one **rdfs:range** property, then the resources denoted by the objects of triples with predicate **P** are instances of all the classes stated by the **rdfs:range** properties.
  - The **rdfs:range** property can be applied to itself. The **rdfs:range** of **rdfs:range** is the class **rdfs:Class**. This states that any resource that is the value of an **rdfs:range** property is an instance of **rdfs:Class**.
  - The **rdfs:range** property is applied to properties. This can be represented in RDF using the **rdfs:domain** property. The **rdfs:domain** of **rdfs:range** is the class **rdf:Property**. This states that any resource with an **rdfs:range** property is an instance of **rdf:Property**.

Source: <http://www.w3.org/TR/rdf-schema/>

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 9






# RDF-S: Properties (2)

- **rdfs:domain**
  - **rdfs:domain** is an instance of **rdf:Property** that is used to state that any resource that has a given property is an instance of one or more classes.
  - A triple of the form **P rdfs:domain C** states that **P** is an instance of the class **rdf:Property**, that **C** is an instance of the class **rdfs:Class** and that the resources denoted by the subjects of triples whose predicate is **P** are instances of the class **C**.
  - Where a property **P** has more than one **rdfs:domain** property, then the resources denoted by subjects of triples with predicate **P** are instances of all the classes stated by the **rdfs:domain** properties.
  - The **rdfs:domain** property may be applied to itself. The **rdfs:domain** of **rdfs:domain** is the class **rdf:Property**. This states that any resource with an **rdfs:domain** property is an instance of **rdf:Property**.
  - The **rdfs:range** of **rdfs:domain** is the class **rdfs:Class**. This states that any resource that is the value of an **rdfs:domain** property is an instance of **rdfs:Class**.
- **rdf:type**
  - **rdf:type** is an instance of **rdf:Property** that is used to state that a resource is an instance of a class.
  - A triple of the form **R rdf:type C** states that **C** is an instance of **rdfs:Class** and **R** is an instance of **C**.
  - The **rdfs:domain** of **rdf:type** is **rdfs:Resource**. The **rdfs:range** of **rdf:type** is **rdfs:Class**.

Source: <http://www.w3.org/TR/rdf-schema/>

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 10






# RDF-S Properties (3)

- **rdfs:subClassOf**
  - The property **rdfs:subClassOf** is an instance of **rdf:Property** that is used to state that all the instances of one class are instances of another.
  - A triple of the form **C1 rdfs:subClassOf C2** states that **C1** is an instance of **rdfs:Class**, **C2** is an instance of **rdfs:Class** and **C1** is a subclass of **C2**. The **rdfs:subClassOf** property is transitive.
  - The **rdfs:domain** of **rdfs:subClassOf** is **rdfs:Class**. The **rdfs:range** of **rdfs:subClassOf** is **rdfs:Class**.

Source: <http://www.w3.org/TR/rdf-schema/>

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 11

# RDF-S Properties (4)

- **rdfs:subPropertyOf**
  - The property **rdfs:subPropertyOf** is an instance of **rdf:Property** that is used to state that all resources related by one property are also related by another.
  - A triple of the form **P1 rdfs:subPropertyOf P2** states that **P1** is an instance of **rdf:Property**, **P2** is an instance of **rdf:Property** and **P1** is a subproperty of **P2**. The **rdfs:subPropertyOf** property is transitive.
  - The **rdfs:domain** of **rdfs:subPropertyOf** is **rdf:Property**. The **rdfs:range** of **rdfs:subPropertyOf** is **rdf:Property**.

Source: <http://www.w3.org/TR/rdf-schema/>

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 12

## RDF-S Properties (5)

- **rdfs:label**
  - rdfs:label is an instance of **rdf:Property** that may be used to provide a human-readable version of a resource's name.
  - A triple of the form R rdfs:label L states that L is a human readable label for R.
  - The **rdfs:domain** of rdfs:label is **rdfs:Resource**. The **rdfs:range** of rdfs:label is **rdfs:Literal**.
  - Multilingual labels are supported using the **language tagging** facility of RDF literals.

Source: <http://www.w3.org/TR/rdf-schema/>

## RDF-S Properties (6)

- **rdfs:comment**
  - rdfs:comment is an instance of **rdf:Property** that may be used to provide a human-readable description of a resource.
  - A triple of the form R rdfs:comment L states that L is a human readable description of R.
  - The **rdfs:domain** of rdfs:comment is **rdfs:Resource**. The **rdfs:range** of rdfs:comment is **rdfs:Literal**.
  - **A textual comment helps clarify the meaning of RDF classes and properties.** Such in-line documentation complements the use of both formal techniques (Ontology and rule languages) and informal (prose documentation, examples, test cases). A variety of documentation forms can be combined to indicate the intended meaning of the classes and properties described in an RDF vocabulary. Since RDF vocabularies are expressed as RDF graphs, vocabularies defined in other namespaces may be used to provide richer documentation.
  - Multilingual documentation is supported through use of the **language tagging** facility of RDF literals.

Source: <http://www.w3.org/TR/rdf-schema/>

## RDF & RDF-S Validators

- <http://www.w3c.org/RDF/Validator/>
- <http://www.swi.psy.uva.nl/projects/SWI-Prolog/packages/sgml/online.html>
- <http://zoe.mathematik.Uni-Osnabrueck.DE/RDF/parser.html>
- <http://www.proflum.com/gb/products/developers>
- <http://www.redland.opensource.ac.uk/demo>
- <http://rdfstoredemo.lrc.it/>

Source: <http://demo.heywow.com/schema/validator.html>

## Ontology Formalisms: WSML

## WSML: Overview

- WSML provides a formal syntax and semantics for the conceptual elements of WSMO, based on:
  - Description Logics
  - Deductive Databases
  - First-Order Logic
- For application in SWS, many current SW languages have
  - undesirable computational properties
  - unintuitive conceptual modeling features for SWS applications
  - inappropriate language layering
    - RDFS/OWL
    - OWL Lite/DL/Full
    - OWL/SWRL

Slide prepared by Jos de Bruijn ©

## Basic WSML Syntax

- Layered syntax
- Inspired by OIL/OWL, F-Logic and First-Order Logic
- Two flavors:
  - Conceptual syntax
    - goals, web services, ontologies, concepts, relations, etc...
    - Corresponds to WSMO conceptual modeling elements
  - Logical Expression Syntax
    - Based on FOL/F-Logic: function symbols, variables, predicate symbols, molecules, logical connectives, etc...

Slide prepared by Jos de Bruijn ©

## WSML-Flight example

Conceptual Syntax

```

wsmlVariant _ "http://www.wsmo.org/2004/wsml/wsml-flight"
ontology _ "http://www.example.org/myOntology"

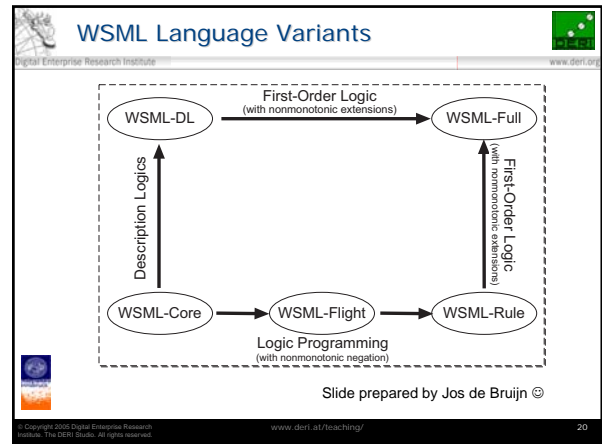
concept ticket
  nfp
  dc:relation hasValue validDates
endnfp
origin ofType location
destination ofType location
departure ofType _dateTime
arrival ofType _dateTime

axiom validDates
  definedBy

false impliedBy ?x memberOf ticket[arrival hasValue ?y,
  departure hasValue ?z] and ?y < ?z.
  
```

Logical Expression Syntax

Slide prepared by Jos de Bruijn ©



## WSML Syntaxes – Human-readable syntax

```

wsmlVariant _ "http://www.wsmo.org/2004/wsml/wsml-flight"
ontology _ "http://www.example.org/myOntology"

concept ticket
  nfp
  dc:relation hasValue validDates
endnfp
destination ofType location
  
```

21

## WSML Syntaxes - XML

```

<wsm! xmlns="http://www.wsmo.org/2004/wsml#"
  variant="http://www.wsmo.org/2004/wsml/wsml-flight">
  <ontology name="http://www.example.org/myOntology">
    <concept name="http://www.example.org/myOntology#ticket">
      <nonFunctionalProperties>
        <attributeValue
          name="http://purl.org/dc/elements/1.1#relation">
          <value>validDates</value>
        </attributeValue>
      </nonFunctionalProperties>
      <attribute
        name="http://www.example.org/myOntology#name"
        type="constraining">
        <range>http://www.w3.org/2001/XMLSchema#string</range>
      </attribute>
    </concept>
  </ontology>
</wsm!>
  
```

22

## WSML Syntaxes - RDF

```

<http://www.example.org/myOntology> rdf:type wsm!:ontology
<http://www.example.org/myOntology> wsm!:variant
<http://www.wsmo.org/2004/wsml/wsml-flight>
<http://www.example.org/myOntology> wsm!:hasConcept
<http://www.example.org/myOntology#ticket>
<http://purl.org/dc/elements/1.1#relation> rdf:type wsm!:nfp
<http://www.example.org/myOntology#ticket>
  <http://purl.org/dc/elements/1.1#relation>
  <http://www.example.org/myOntology#validDates>
<http://www.example.org/myOntology#ticket> wsm!:hasAttribute _X
_:X wsm!:ofType <http://www.w3.org/2001/XMLSchema#string>
_:X wsm!:attribute <http://www.example.org/myOntology#name>
  
```

23

## WSML: Web Semantics Modeling Language + Web Service Modeling Language

- **Web Semantics Modeling Language**
  - Part of WSML that covers the specification of ontologies
    - Concepts
    - Relations
    - Instances
    - Axioms
- **Web Service Modeling Language**
  - WSML elements that are specific for modeling aspects of Semantic Web services
    - Capabilities
    - Mediators
    - Interfaces

24

## WSML Syntax Basics

- The conceptual syntax for WSML has a **frame-like style**. The information about a class and its attributes, a relation and its parameters and an instance and its attribute values is specified in one large syntactic construct, instead of being divided into a number of atomic chunks.
- It is possible to spread the information about a particular class, relation, instance or axiom over several constructs, but this is not recommended.
- Nonetheless, attribute names are global and it is possible to specify global behavior of attributes through logical expressions. However, we do not expect this to be necessary in the general case and we strongly advise against it. In case one needs to model a property which is independent of the concept definition, this property is most likely a relation rather than an attribute and thus should be modeled as a relation.
- It is often possible to specify a list of arguments, for example for attributes. Such argument lists in WSML are separated by commas and surrounded by curly brackets.
- Statements in WSML start with a keyword and can be spread over multiple lines.
- A WSML specification is separated into two parts.
  - The first part provides meta-information about the specification, which consists of such things as WSML variant identification, namespace references, non-functional properties (annotations), import of ontologies, references to mediators used and the type of the specification. This meta-information block is strictly ordered.
  - The second part of the specification, consisting of elements such as concepts, attributes, relations (in the case of an ontology specification), capability, interfaces (in the case of a goal or web service specification), etc., is not ordered.

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 25

## WSML Syntax Basics

**wsml = wsmlvariant? namespace? definition\***

**definition =** **goal**  
**ontology**  
**webservice**  
**mediator**

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 26

## Namespaces in WSML

- Namespaces can be used to syntactically distinguish elements of multiple WSML specifications and, more generally, resources on the Web. A namespace denotes a syntactical domain for naming resources.
- WSML adopts the namespace mechanism of RDF.
- A namespace can be seen as part of an IRI.
- The WSML keywords themselves belong to the namespace <http://www.wsmo.org/wsml/wsml-syntax#> (commonly abbreviated as 'wsml').
- Whenever a WSML specification has a specific identifier which corresponds to a Web address, it is good practice to have a relevant document on the location pointed to by the identifier. This can either be the WSML document itself or a natural language document related to the WSML document. Note that the identifier of an ontology does not have to coincide with the location of the ontology. It is good practice, however, to include a related document, possibly pointing to the WSML specification itself, at the location pointed to by the identifier.

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 27

## Identifiers in WSML

- An identifier in WSML is either a data value, an IRI [Duerst & Suignard, 2005], or an anonymous ID.
- Data values**
- WSML has direct support for different types of concrete data, namely, strings, integers and decimals.
- WSML uses datatype wrappers to construct data values based on XML Schema datatypes. The use of datatype wrappers gives more control over the structure of the data values than the lexical representation of XML. For example, the date: 3rd of February, 2005, which can be written in XML as: '2005-02-03', is written in WSML as: `_date(2005,2,3)`. The arguments of such a term can be either strings, decimals, integers or variables. No other arguments are allowed for such data terms.
- Example of data values:
  - `_date(2005,3,12)`
  - `_boolean("true")`
- The following are example attribute definitions which restrict the range of the attribute to a particular datatype:
  - `age ofType _integer`
  - `location ofType _iri`
  - `hasChildren ofType _boolean`
- Anonymous identifiers are supported in WSML but not covered in this unit.

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 28

## WSML: Shortcuts for Datatypes

- Data values of type *string* can be written between double quotes `"`. Double quotes inside a string should be escaped using the backslash (`\`) character: `"\"`.
- Integer values can simply be written as such. Thus an integer of the form *integer* is a shortcut for `_integer("integer")`. For example, 4 is a shortcut for `_integer("4")`.
- Decimal values can simply be written as such, using the `.` as decimal symbols. Thus a literal of the form *decimal* is a shortcut for `_decimal("decimal")`. For example, 4.2 is a shortcut for `_decimal("4.2")`.

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 29

## WSML: Full IRIs

- Examples of full IRIs in WSML:
  - `_"http://example.org/PersonOntology#Human"`
  - `_"http://www.uibk.ac.at/"`

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 30

## WSML: Serialized Qnames (sQNames)

- Examples of sQNames in WSML (with corresponding full IRIs)
  - dc stands for <http://purl.org/dc/elements/1.1#>
  - foaf stands for <http://xmlns.com/foaf/0.1/>
  - xsd stands for <http://www.w3.org/2001/XMLSchema#>
  - We assume the default namespace <http://example.org/#>
- dc#title -> <http://purl.org/dc/elements/1.1#title>
- foaf#name -> <http://xmlns.com/foaf/0.1/name>
- xsd#string -> <http://www.w3.org/2001/XMLSchema#string>
- Person -> <http://example.org/#Person>
- hasChild -> <http://example.org/#hasChild>

## Comments in WSML

- A WSML file may at any place contain a comment.
- A single line comment starts with `comment` or `//` and ends with a line break.
- Comments can also range over multiple lines, in which they need to be delimited by `/*` and `*/`.
- It is recommended to use non-functional properties for any information related to the actual WSML descriptions; comments should be only used for meta-data about the WSML file itself. Comments are disregarded when parsing the WSML document.
- Examples:
 

```
/* Illustrating a multi line
 * comment */
// a one-line comment
comment another one-line comment
```

## WSML Elements: WSML Variant

- Every WSML specification document may start with the **wsmVariant** keyword, followed by an identifier for the WSML variant which is used in the document.
- Example:
 

```
wsmVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsm-flight"
```

WSML Variant	IRI
WSML-Core	<a href="http://www.wsmo.org/wsml/wsml-syntax/wsm-core">http://www.wsmo.org/wsml/wsml-syntax/wsm-core</a>
WSML-Flight	<a href="http://www.wsmo.org/wsml/wsml-syntax/wsm-flight">http://www.wsmo.org/wsml/wsml-syntax/wsm-flight</a>
WSML-Rule	<a href="http://www.wsmo.org/wsml/wsml-syntax/wsm-rule">http://www.wsmo.org/wsml/wsml-syntax/wsm-rule</a>
WSML-DL	<a href="http://www.wsmo.org/wsml/wsml-syntax/wsm-dl">http://www.wsmo.org/wsml/wsml-syntax/wsm-dl</a>
WSML-Full	<a href="http://www.wsmo.org/wsml/wsml-syntax/wsm-full">http://www.wsmo.org/wsml/wsml-syntax/wsm-full</a>

## WSML Elements: Namespace References

```
namespace
{_"http://www.example.org/ontologies/example#", dc
_"http://purl.org/dc/elements/1.1#", foaf
_"http://xmlns.com/foaf/0.1/", wsm
_"http://www.wsmo.org/wsml-syntax#", loc
_"http://www.wsmo.org/ontologies/location#", oo
_"http://example.org/ooMediator#"}
```

```
namespace
_"http://www.example.org/ontologies/example#"
```

## WSML Elements: Header (1)

- Any WSML specification may have
  - non-functional properties,
  - may import ontologies and
  - may use mediators.
- The import of ontologies and the usage of mediators are not covered in this unit.
- non-functional property blocks are delimited with the keywords **nonFunctionalProperties** and **endNonFunctionalProperties** or the short forms **nfp** and **endnfp**.
- Following the keyword is a list of attribute values, which consists of the attribute identifier, the keyword **hasValue** and the value for the attribute, which may be any identifier and can thus be an IRI, a data value, an anonymous identifier or a comma-separated list of the former, delimited with curly brackets.

## WSML Elements: Header (2) – NFP Example

```
nonFunctionalProperties
dc#title hasValue "WSML example ontology"
dc#subject hasValue "family"
dc#description hasValue "fragments of a family ontology to provide
WSML examples"
dc#contributor hasValue {
_"http://homepage.uibk.ac.at/~c703240/foaf.rdf",
_"http://homepage.uibk.ac.at/~csaa5569/",
_"http://homepage.uibk.ac.at/~c703239/foaf.rdf",
_"http://homepage.uibk.ac.at/homepage/~c703319/foaf.rdf"}
dc#date hasValue _date("2004-11-22")
dc#format hasValue "text/html"
dc#language hasValue "en-US"
dc#rights hasValue _"http://www.deri.org/privacy.html"
wsm#version hasValue "$Revision: 1.191 $"
endNonFunctionalProperties
```

## Ontology Specification in WSML

```

wsm:Variant _ "http://www.wsmo.org/wsm/wsm-syntax/wsm-rule"
name:space ( "http://www.example.org/ontologies/example#" , dc
  _ "http://purl.org/dc/elements/1.1/#" foaf _ "http://xmlns.com/foaf/0.1/" , wsm
  _ "http://www.wsmo.org/wsm/wsm-syntax#" , loc
  _ "http://www.wsmo.org/ontologies/locations#" , oo _ "http://example.org/ooMediator#" )

ontology _ "http://www.example.org/ontologies/example#"
nfp
  dc:title hasValue "WSML example ontology"
  dc:subject hasValue "family"
  dc:description hasValue "fragments of a family ontology
    to provide WSML examples."
endnfp
concept Human
nonFunctionalProperties
  dc:description hasValue "concept of a human being"
endNonFunctionalProperties
  hasName ofType foaf#name
  hasParent impliesType Human
  hasChild impliesType Human
relation ageOfHuman (ofType Human, ofType _integer)
instance Susan memberOf Woman
  hasName hasValue "Susan Jones"
  hasBirthdate hasValue _date(1976,08,16).

```

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 37

## WSML: Concepts

- A concept definition starts with the concept keyword, which is optionally followed by the identifier of the concept.
- This is optionally followed by a superconcept definition which consists of the keyword **subConceptOf** followed by one or more concept identifiers (as usual, if there is more than one, the list is comma-separated and delimited by curly brackets).
- This is followed by an optional nonFunctionalProperties block and zero or more attribute definitions.
- Note that WSML allows inheritance of attribute definitions, which means that a concept inherits all attribute definitions of its superconcepts. If two superconcepts have a definition for the same attribute *a* but with a different range, these attribute definitions are interpreted conjunctively. This means that the resulting range of the attribute *a* in the subconcept is the conjunction (intersection) of the ranges of the attribute definitions in the superconcepts.

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 38

## WSML: Concepts - Example

```

concept Human subConceptOf (Primate, Mammal)
nonFunctionalProperties
  dc:description hasValue "concept of a human being"
  dc:#relation hasValue humanDefinition
endNonFunctionalProperties
  hasName ofType foaf#name
  hasParent impliesType Human
  hasChild impliesType Human
  hasAncestor impliesType Human
  hasWeight ofType _float
  hasWeightInKG ofType _float
  hasBirthdate ofType _date
  hasObit ofType _date
  hasBirthplace ofType loc#location
  isMarriedTo impliesType Human
  hasCitizenship ofType oo#country

```

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 39

## WSML: Attributes

- WSML allows two kinds of attribute definitions, namely,
  - constraining definitions with the keyword **ofType** and
  - inferring definitions with the keyword **impliesType**.
- Several WSML variants, namely, WSML-Core and WSML-DL, do not allow constraining attribute definitions.

```

concept Human subConceptOf (Primate, Mammal)
  hasName ofType foaf#name
  hasParent impliesType Human
  hasChild impliesType Human
  hasCitizenship ofType oo#country

```

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 40

## WSML: Relations

- A relation definition starts with the relation keyword, which is followed by the identifier of the relation.
- WSML allows the specification of relations with arbitrary arity.
- The domain of the parameters can be optionally specified using the keyword **impliesType** or **ofType**. Note that parameters of a relation are strictly ordered.
- A relation definition is optionally completed by the keyword **subRelationOf** followed by one or more identifiers of superrelations. Finally an optional nonFunctionalProperties block can be specified.
- Relations in WSML can have an arbitrary arity and values for the parameters can be constrained using parameter type definitions of the form ( **ofType type-name** ) and ( **impliesType type-name** ). The definition of relations requires either the indication of the arity or the parameter definitions.
- Example:**

```

relation distance (ofType City, ofType City, impliesType _decimal)
subRelationOf measurement

```

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 41

## WSML: Instances

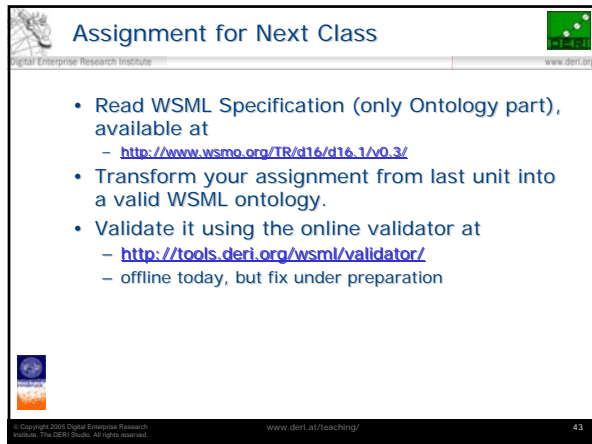
- An instance definition starts with the **instance** keyword, (optionally) followed by the identifier of the instance, the **memberOf** keyword and the name of the concept to which the instance belongs.
- The memberOf keyword identifies the concept to which the instance belongs. This definition is followed by the attribute values associated with the instance. Each property filler consists of the property identifier, the keyword **hasValue** and the value(s) for the attribute.
- Example:**

```

instance Mary memberOf (Parent, Woman)
nfp dc:description hasValue "Mary is parent of the twins Paul and Susan"
endnfp
  hasName hasValue "Maria Smith"
  hasBirthdate hasValue _date(1949,9,12)
  hasChild hasValue { Paul, Susan }

```

© Copyright 2005 Digital Enterprise Research Institute. The DERI Studio. All rights reserved. www.deri.at/teaching/ 42



Assignment for Next Class

- Read WSM Specification (only Ontology part), available at
  - <http://www.wsmo.org/TR/d16/d16.1/v0.3/>
- Transform your assignment from last unit into a valid WSM ontology.
- Validate it using the online validator at
  - <http://tools.deri.org/wsm/validator/>
  - offline today, but fix under preparation

© Copyright 2005 Digital Enterprise Research Institute, The DERI Studio. All rights reserved. www.deri.at/teaching/ 43



Thank you!

The slides will be available on the Internet at <http://www.heppnetz.de/teaching/aoe>

© Copyright 2005 Digital Enterprise Research Institute, The DERI Studio. All rights reserved. www.deri.at/teaching/ 44