Ontologies and Taxonomies

Modellbildung in der Entwicklung
Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München
SS 2007

TECHNISCHE UNIVERSITÄT MÜNCHEN

Inhalte

Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
3. Anwendung
   1. Ontologien
   2. Requirements Engineering
   3. Architekturen
   4. Qualitätssicherung
   5. Modellqualität
4. Modellgetriebene Entwicklung
5. Domänenspezifische Modellierung

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies

- **In philosophy ...**
  - are branch of metaphysics, concerned with analyzing various types of existence
- **In computer science ...**
  - are branch of knowledge representation, concerned with the storage of knowledge that computers can process
  - are used in:
    - information retrieval and classification
    - natural language processing/understanding
    - knowledge management and organization
    - language engineering
    - knowledge engineering
    - software engineering
    - ...
  - since mid '90 ontologies took a new advent especially in the context of **semantic web**
    - most of the current driving force behind ontologies

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in the Semantic Web

- **Semantic Web is an extension of the current web where the information is given well defined meaning**
  - it should enable people and computers to work in cooperation
- **For realising the semantic web, computers need access to structured and well defined information and sets of inference rules that automate the reasoning**
  - instead of natural language as in the current web, the semantic web should contain machine procesable data
- **Ontologies should enable the communication between software agents and to provide support for large scale knowledge management**
  - agents are programs that collect web content from more sources, process their information and exchange results with other programs (also agents)

TECHNISCHE UNIVERSITÄT MÜNCHEN

A Typical Case for Ontologies

- **A personalized wine agent can provide a number of services for a human by using the internet:**
  - Recommend the most appropiate wine for a specific meal given a set of constraints
    - e.g. type of food, wine produces, price
  - Find and compare informations about particular wines
  - Look for related apparel for a specific wine
    - e.g. glass types

TECHNISCHE UNIVERSITÄT MÜNCHEN

A Typical Case for Ontologies

- **A personalized wine agent can provide a number of services for a human by using the internet:**

Keywords based searching is clearly not enough

We need ontological knowledge about the wines domain
- we need agreed concepts about wine sorts, meals, prices
- all parties should agree on this knowledge

This knowledge should be in a machine processable format

TECHNISCHE UNIVERSITÄT MÜNCHEN

A Typical Case for Ontologies

- **A personalized wine agent can provide a number of services for a human by using the internet:**

Keywords based searching is clearly not enough

We need ontological knowledge about the wines domain
- we need agreed concepts about wine sorts, meals, prices
- all parties should agree on this knowledge

This knowledge should be in a machine processable format

Wine Agent 1.0: http://onto.stanford.edu:8080/wino/index.jsp

TECHNISCHE UNIVERSITÄT MÜNCHEN

Some Software Engineering Challenges

- **Integrate different views produced by different stakeholders in the development process**
- **Bridge the abstraction gap between the domain and the code**
- **Modelling a common understanding of application domains through formal and semi-formal methods**
- **Achieve automation in the SE process**
- **Web Service concepts and technologies also pose the challenge of managing the knowledge – finding services, etc.**
  - also an older problem of finding reusable components

TECHNISCHE UNIVERSITÄT MÜNCHEN

Some Software Engineering Challenges

Manage and disseminate the knowledge in software projects

- a great potential for application of knowledge engineering techniques in software engineering

- **Web Service concepts and technologies also pose the challenge of managing the knowledge – finding services, etc.**
  - also an older problem of finding reusable components

TECHNISCHE UNIVERSITÄT MÜNCHEN

Some Software Engineering Challenges

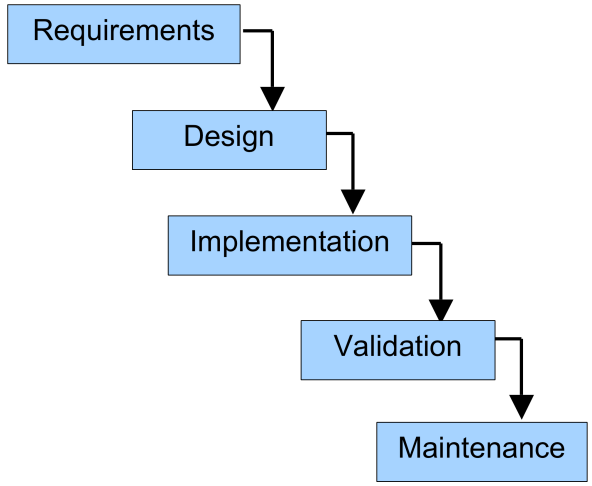
Manage and disseminate the knowledge in software projects

- a great potential for application of knowledge engineering techniques in software engineering

Discover and use the knowledge at run-time

TECHNISCHE UNIVERSITÄT MÜNCHEN

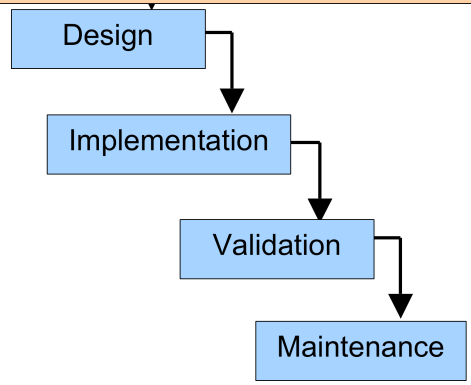
Ontologies in Software Development Process



TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers



TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Implementation

Validation

Maintenance

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Augment sharing and interoperation of models
- define semantics of interfaces

Validation

Maintenance

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Augment sharing and interoperation of models
- define semantics of interfaces

Check the consistency between different artefacts

Maintenance

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Augment sharing and interoperation of models
- define semantics of interfaces

Check the consistency between different artefacts

Manage the knowledge during maintenance
- semantic search capabilities in the code
- help identifying and reusing existing components

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Augment sharing and interoperation of models
- define semantics of interfaces

Check the consistency between different artefacts

Manage the knowledge during maintenance
- semantic search capabilities in the code
- help identifying and reusing existing components

Sustain the knowledge flow in the project

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies in Software Development Process

Formally represent the domain through ontologies
- build a conceptual model of the domain
- represent the domain in a form understandable by both domain experts and the engineers

Use ontologies as the entry point in the design
- the central domain concepts represent the main design abstractions

Augment sharing and interoperation of models
- define semantics of interfaces

Check the consistency between different artefacts

Manage the knowledge during maintenance
- semantic search capabilities in the code
- help identifying and reusing existing components

Sustain the knowledge flow in the project

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies Definition

- **Ontologies are "explicit, formal specification of a share conceptualization" [Gruber]**
  - formal means that the ontology is represented using a formal language
    - formal language means that it is machine processable
  - explicit means that the type of concepts used, and the constraints on their use are explicitly defined
    - everything that is important in the domain must be explicitly state in the ontology
  - shared means that the ontology mirrors a common understanding of the modelled domain being the result of a consensus achieved within a community of ontology users
    - ontologies are like contracts between their providers and users that specify an agreed semantics
  - specifies a conceptualization: the ontology is used to model a specific view over some domain in the world
    - the view is obtained through certain simplifications, abstractions, omissions or other modelling decisions

TECHNISCHE UNIVERSITÄT MÜNCHEN

Formalization spectrum

- **Controlled vocabulary**
  - finite lists of terms (e.g. catalogs)
- **Glossary**
  - list of terms and meanings expressed in natural language
- **Thesauri**
  - words and basic relations (e.g. synonymy)
- **Taxonomies**
  - terms with minimal hierarchy (e.g. parent/child structure)
- **Frames**
  - classes with information about properties
- **Values restriction**
  - define restrictions on the values that can fill a property
- **General logical constraints and complex relations**
  - define constraints on terms and relations in FOL
  - define complex relations (e.g. disjoint classes, inverse relationships)

Infor

Form

TECHNISCHE UNIVERSITÄT MÜNCHEN

Formalization spectrum

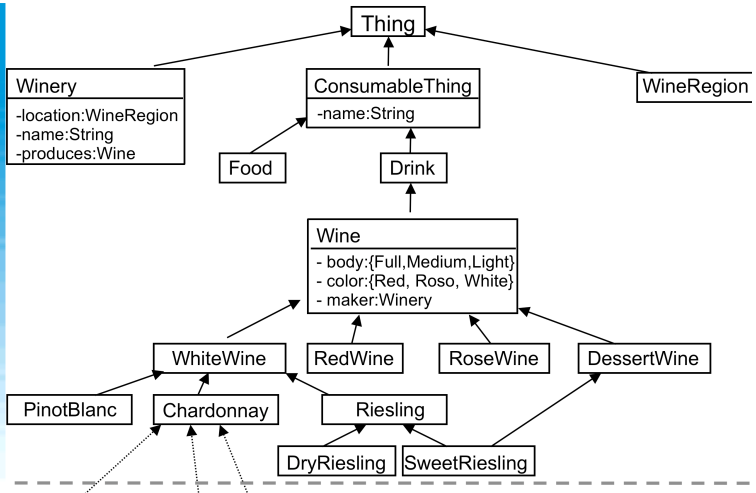
Infor

- **Controlled vocabulary**
  - finite lists of terms (e.g. catalogs)
- **Glossary**
  - list of terms and meanings expressed in natural language
- **Thesauri**
  - words and basic relations (e.g. synonymy)
- **Taxonomies**
  - terms with minimal hierarchy (e.g. parent/child structure)
- **Frames**
  - classes with information about properties
- **Values restriction**
  - define restrictions on the values that can fill a property
- **General logical constraints and complex relations**
  - define constraints on terms and relations in FOL
  - define complex relations (e.g. disjoint classes, inverse relationships)

Forı

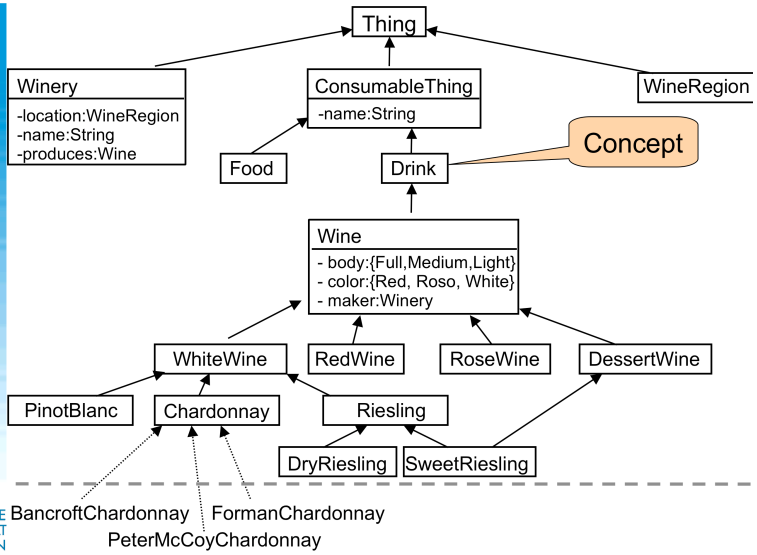
TECHNISCHE UNIVERSITÄT MÜNCHEN

Example: The Wine Ontology



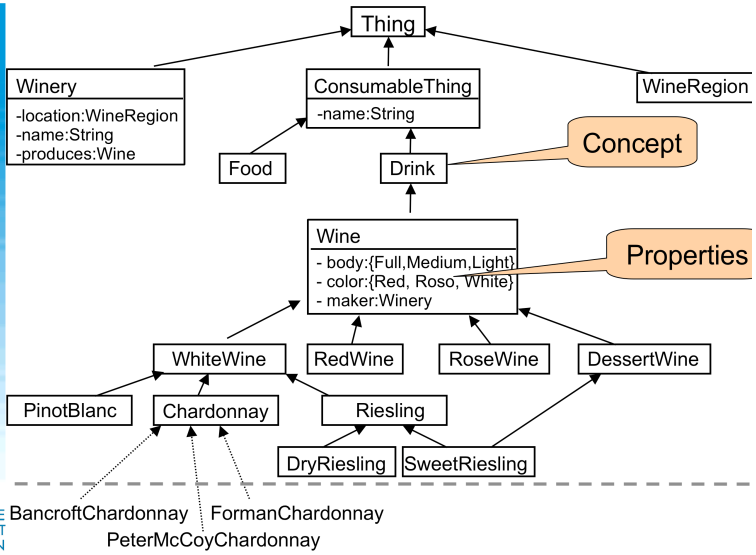
TECHNISCHE UNIVERSITÄT MÜNCHEN

Example: The Wine Ontology



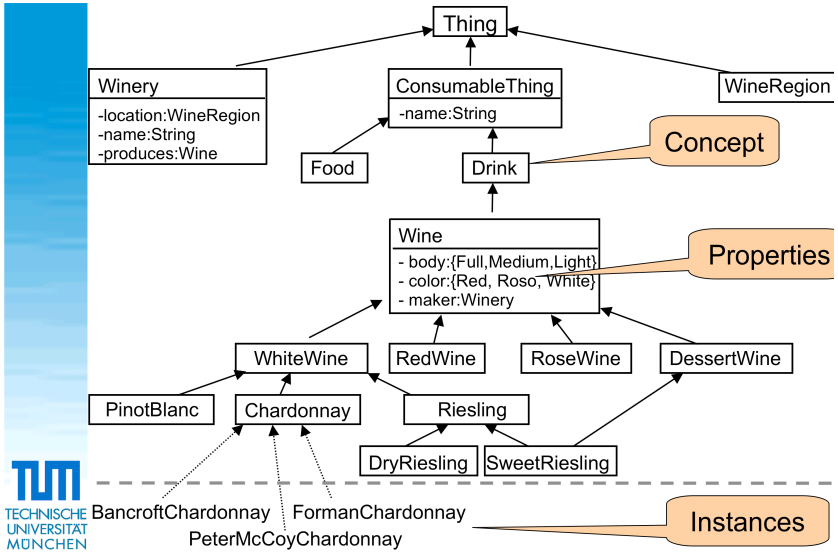
TECHNISCHE UNIVERSITÄT MÜNCHEN

Example: The Wine Ontology



TECHNISCHE UNIVERSITÄT MÜNCHEN

Example: The Wine Ontology



TECHNISCHE UNIVERSITÄT MÜNCHEN

Typical Operations on Ontologies

- **Ontologies construction and management**
  - support for building and managing large ontologies
    - thousands of concepts
- **Reasoning tasks**
  - classification of concepts
    - place a concept in a proper place in a taxonomic hierarchy of concepts
  - instance checking
    - verifies if an individual is an instance of a specific concept
- **Integrity checks**
  - consistency checking
    - does every concept allow individuals? (no contradictions)
- **Ontologies mapping and alignment**
  - given two distinct ontologies, identify the common concepts

TECHNISCHE UNIVERSITÄT MÜNCHEN

Sources of Ontologies

- **General purpose ontologies**
  - high-level classifications of topics in the web
    - e.g. Open Directory Project (www.dmoz.org)
      - over 500.000 categories that describe web-pages
  - ontological dictionaries
    - e.g. WordNet (http://wordnet.princeton.edu/)
      - over 110.000 concepts denoted through english words
      - contains information about synonyms, part of relations, etc.
  - formalization of the common-sense knowledge
    - e.g. CyC (www.cyc.com)
      - hundreds of thousands of terms
      - millions of assertions relating the terms to each other
- **Domain ontologies**
  - Bio-medical
    - e.g. Disease Ontology (http://diseaseontology.sourceforge.net/)
      - over 90.000 concepts and 140.000 relations
    - e.g. Systematized Nomenclature of Medicine (http://snomed.net/
      - over 350,000 health care concepts organized in hierarchies

TECHNISCHE UNIVERSITÄT MÜNCHEN

Sources of Ontologies

- **General purpose ontologies**
  - high-level classifications of topics in the web
    - e.g. Open Directory Project (www.dmoz.org)
      - over 500.000 categories that describe web-pages
  - ontological dictionaries
    - e.g. WordNet (http://wordnet.princeton.edu/)

If no ontology is available then we have to build one

    - e.g. CyC (www.cyc.com)
      - hundreds of thousands of terms
      - millions of assertions relating the terms to each other
- **Domain ontologies**
  - Bio-medical
    - e.g. Disease Ontology (http://diseaseontology.sourceforge.net/)
      - over 90.000 concepts and 140.000 relations
    - e.g. Systematized Nomenclature of Medicine (http://snomed.net/
      - over 350,000 health care concepts organized in hierarchies

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontological Engineering

- **Ontological engineering is a branch of knowledge engineering concerned with the development of metho and tools to build, maintain and reuse ontologies**
  - Knowledge engineering is focused on developing knowledge-based systems
    - represents the general process of knowledge base construction
    - involves methodologies and knowledge representation techniqu

TECHNISCHE UNIVERSITÄT MÜNCHEN

Activities for Ontologies Building

- **Domain analysis**
  - analyse the application domain with respect to a set of predefined requirements that the ontology need to fulfill
    - make up a list of questions that the ontology needs to answer
    - acquire the knowledge that needs to be represented in the ontolc
    - study the possibility of reusing (parts of) already existing ontologi
- **Build the conceptualization**
  - model the application domain in terms of ontological primitives (e.g. concepts, relations and axioms)
    - the result of this activity is a language-independent conceptual model of the domain
- **Implement the conceptualization**
  - the conceptual model is implemented with the help of a formal language
    - the result of this step is an ontolgy expressed in a machine processable language

TECHNISCHE UNIVERSITÄT MÜNCHEN

Activities for Ontologies Building (2)

- **Evaluation**
  - the ontology is evaluated against the requirements
    - does the ontology answer the required questions?
    - is the ontology consistent (free of contradictions)?
- **Population**
  - align the instances of the application data to the ontology
    - identify the instances of concepts described in the ontology
- **Evolution and maintenance**
  - evolve the ontology to fulfill new user requirements
  - adapt the ontology to changes in the modeled domain

TECHNISCHE UNIVERSITÄT MÜNCHEN

Web Ontology Languages (OWL)

- **OWL is an expressive ontology definition language**
  - Since 2004 is a standard of W3C
  - Syntax based on RDF, semantic based on description logic
  - Balancing expressivity and complexity:
    - OWL language layers: OWL Lite, OWL DL, OWL Full
- **OWL is a full-fledged knowledge representation language that allows to define:**
  - Classes, class hierarchies
  - Properties, property hierarchies
  - Domain and range restrictions
  - Logical expressions (and, or, not)
  - (in)equality
  - local properties
  - required/optional properties
  - required values of properties
  - enumerated classes
  - transitivity, symmetry, inverse of relations

TECHNISCHE UNIVERSITÄT MÜNCHEN

OWL Language Layers

- **OWL Lite**
  - Classification hierarchy, simple constraints
  - features:
    - (sub)classes, individuals
    - (sub)properties, domain, range
    - conjunction
    - (in)equality
    - cardinality constraints 0/1
    - datatypes
    - inverse, transitive, symmetric properties
    - someValuesFrom
    - allValuesFrom
- **OWL DL**
  - Maximal expressiveness while maintaining tractability
  - direct correspondence wit a Description Logic
  - features:
    - Negation
    - Disjunction
    - Full cardinality
    - Enumerated types
- **OWL Full**
  - Very high expressiveness by losing tractability
  - No restriction on use of vocabulary (as long as it complies RDF)
  - features:
    - Meta-classes
    - Modifiying language

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements

- **Simple classes and individuals**
  - Defining classes:
    - E.g. defining the Region concept
      - <owl:Class rdf:ID="Region"/>
  - Defining individuals
    - E.g. defining the CentralCoastRegion as an instance of Region
      - <Region rdf:ID="CentralCoastRegion" />
  - Defining a taxonomy:
    - E.g. Wine is a kind of Drink
      - <Class rdf:ID="Wine">
        <subClassOf rdf:resource="#Drink"/>
        </Class>

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements (2)

- **Simple properties (properties = binary relations)**
  - Defining relations between instances of classes and RDF literals and XML Schema datatypes (xsd namespace)
    - E.g. the VintageYear is a positive integer
      - <**owl:DatatypeProperty** rdf:ID="yearValue">
        <rdfs:domain rdf:resource="#VintageYear" />
        <rdfs:range rdf:resource="&xsd;positiveInteger"/>
        </owl:DatatypeProperty>
  - Defining relations between instances of two classes
    - E.g. the relation between a Wine and its Grapes
      - <**owl:ObjectProperty** rdf:ID="madeFromGrape">
        <rdfs:domain rdf:resource="#Wine"/>
        <rdfs:range rdf:resource="#WineGrape"/>
        </owl:ObjectProperty>
  - Defining properties of individuals
    - E.g. the "locatedIn" property of a particular CabernetSauvignon
      - <CabernetSauvignon
        rdf:ID="SantaCruzMountainVineyardCabernetSauvignon" >
        <locatedIn rdf:resource="#SantaCruzMountainsRegion"/>
        </CabernetSauvignon>

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements (3)

- **Properties Characteristics**
  - Transitive properties - "P(x,y) and P(y,z) implies P(x,z)"
    - E.g. the "locatedIn" property is transitive
      - <owl:ObjectProperty rdf:ID="locatedIn">
        <rdf:type rdf:resource="&**owl;TransitiveProperty**" />
        <rdfs:domain rdf:resource="&owl;Thing" />
        <rdfs:range rdf:resource="#Region" />
        </owl:ObjectProperty>
  - Symmetric properties - "P(x,y) iff P(y,x)"
    - E.g. the "adjacentRegion" property is symmetric
      - <owl:ObjectProperty rdf:ID="adjacentRegion">
        <rdf:type rdf:resource="&**owl;SymmetricProperty**" />
        <rdfs:domain rdf:resource="#Region" />
        <rdfs:range rdf:resource="#Region" />
        </owl:ObjectProperty>
  - Inverse properties - "P1(x,y) iff P2(y,x)"
    - E.g. the "producesWine" property is the inverse of "hasMaker"
      - <owl:ObjectProperty rdf:ID="producesWine">
        <**owl:inverseOf** rdf:resource="#hasMaker" />
        </owl:ObjectProperty>

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements (4)

- **Properties Restrictions**
  - Cardinality constraints (minimum, maximum, exact)
    - E.g. every Wine is made at least from one Grape

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger"
          1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements (5)

- **Property restrictions**
  - owl:allValuesFrom – for every instance of the class that has instances of the specified property, the values of the property are all members of the class indicated by the owl:allValuesFro clause
    - E.g. for all wines, if they have makers, all the makers are wineries

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:allValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Basic OWL Elements (6)

- **Property restrictions**
  - owl:someValuesFrom – for every instance of the class that has instances of the specified property, some of the values of the property are members of the class indicated by the owl:someValuesFrom clause.
    - E.g. for all wines at least one of the hasMaker properties of a Wir must point to an individual that is a Winery.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:someValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Advanced OWL Elements

- **Complex classes**
  - Define a class by directly enumerating its members
    - E.g. the “WineColor” can be only “White”, “Red” or “Rose”

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor"/>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White"/>
    <owl:Thing rdf:about="#Rose"/>
    <owl:Thing rdf:about="#Red"/>
  </owl:oneOf>
</owl:Class>
```

  - Defining disjoint classes – whenever an individual is member of a class, it can not be also member of the others
    - E.g. “Pasta”, “Meat”, “Seafood”, “Dessert” and “Fruit” are disjoint concepts

```
<owl:Class rdf:ID="Pasta">
  <owl:disjointWith rdf:resource="#Meat"/>
  <owl:disjointWith rdf:resource="#Seafood"/>
  <owl:disjointWith rdf:resource="#Dessert"/>
  <owl:disjointWith rdf:resource="#Fruit"/>
</owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Advanced OWL Elements (2)

- **Complex Classes**
  - Define a class by specifying its complement
    - E.g. “NonConsumableThing” are all the instances that do not belong to “ConsumableThing”

```
<owl:Class rdf:ID="NonConsumableThing">
  <owl:complementOf rdf:resource="#ConsumableThing" />
</owl:Class>
```

  - Define a class exactly as the intersection between two class
    - E.g. “WhiteWine” is the intersection of “Wine” and “White”

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Wine" />
   <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#White" />
   </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Advanced OWL Elements (3)

- **Complex Classes**
  - Define a class as the union between two classes
    - E.g. “Fruit” is the union of “SweetFruit” and “NonSweetFruit”

```
<owl:Class rdf:ID=“Fruit">
  <owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#SweetFruit" />
   <owl:Class rdf:about="#NonSweetFruit" />
   </owl:intersectionOf>
 </owl:Class>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Advanced OWL Elements (4)

- **Ontology mapping**
  - Equivalence between classes
    - E.g. the class “Wine” is equivalent with “Wine” from the “vin” namespace

```
<owl:Class rdf:ID=“Wine">
  <owl:equivalentClass rdf:resource=“&vin:Wine"/>
</owl:Class>
```

  - Equivalence between properties
    - E.g. the property “producesWine” is equivalent with property “producer” from the “vin” namespace

```
<owl:ObjectProperty rdf:ID="producesWine">
  <owl:equivalentProperty rdf:resource=“&vin:producer" />
</owl:ObjectProperty>
```

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies vs. Data-modelling

- **Origins**
  - Ontologies: knowledge representation
  - Data-models: data-bases (ER diagrams)
- **Typical application areas**
  - Ontologies: knowledge formalization and sharing
  - Data-models: data modelling
- **Open vs. Closed World**
  - Ontologies: reasoning based on open-world assumption
    - OWA – assumes by default that the **knowledge is incomplete**
      - what is not stated is considered unknown (and not wrong)
  - Data-models: reasoning based on the closed-world assumptio
    - CWA - assumes by default that the **knowledge is complete**
      - what is not currently known to be true is false

TECHNISCHE UNIVERSITÄT MÜNCHEN

Ontologies vs. Data-modelling (2)

- **Common modelling constructs**
  - Ontologies
    - building taxonomies
    - instances
    - properties
    - property restrictions
  - Data-modelling
    - specialization, generalizati
    - instances
    - relations
    - constraints on relations

TECHNISCHE UNIVERSITÄT MÜNCHEN

Summary

- **SE World**
  - buid software systems
  - use modeling and programming languages
  - involve software engineers
- **KE World**
  - build knowledge-based syste
  - use knowledge representatio languages (e.g. OWL)
  - involve knowledge engineers
- **Ontologies can be used for...**
  - modeling rigorous semantics during specification and design stages of the software lifecycle
  - describing, identifying, discovering and sharing artefacts amongst subsystems during design
    - e.g. knowledge management in a project
  - specifying, discovering artefacts amongst subsystems at runtime
    - e.g. web-services

TECHNISCHE UNIVERSITÄT MÜNCHEN

Literature

- **Ontology Development 101: A Guide to Creating Your First Ontolog**
  - http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf
- **Applications of Ontologies in Software Engineering**
  - http://km.aifb.uni-karlsruhe.de/ws/swese2006/final/happel_full.pdf
- **OWL Web Ontology Language Guide**
  - http://www.w3.org/TR/2004/REC-owl-guide-20040210/

TECHNISCHE UNIVERSITÄT MÜNCHEN