Ontology Composition using a Role Modeling Approach

Michael Pradel

TU Dresden
Lehrstuhl Softwaretechnologie

Supervisor: MSc. Jakob Henriksson
Professor: Dr. rer. nat. habil. Uwe Aßmann
An ontology is an explicit specification of a conceptualization.  

(Gruber)

- Set of axioms containing
  - Individuals
  - Classes built from class expressions
  - Properties
- Web Ontology Language OWL based on descriptions logics
- Applications
  - Semantic web
  - Domain modeling
  - Data integration
- Main advantage: Reasoning
Wine Ontology:
Building Ontologies from Components

- **Goal:** Create new ontology from existing building blocks
- **What is the reuse unit?**
  - OWL *import* statement
    - One ontology imports another
    - No partial reuse
  - Approaches to modular description logics
- **Problems with partial reuse**
  - Classes depend on each other
  - No separation of concerns
A New Reuse Abstraction: Collaborations

- Natural types vs. role types
- Idea: Introduce roles into ontologies
  - Role as ontological primitives
  - Split class description into parts
  - Collaborations of individuals expressed by role models
- Benefits
  - Role models (= collaborations) as reuse abstraction
  - Different contexts in which classes appear are distinguished
  - Better modeling through distinction of natural types and role types
Wine Ontology with \textit{Product} Role Model

\textbf{Consumer} \ \exists \ \text{consumes.Product (Product)}

\textbf{Producer} \ \exists \ \text{produces.Product (Product)}

\textbf{Winery}

\textbf{Wine}

\textbf{Food}

\textbf{WineTaste}

consumes \ \rightarrow

produces \ \rightarrow

hasTaste \ \rightarrow

accompanies \ \rightarrow
What Do Roles Mean?

Additional syntax:

- Role types and role properties: $R$, $p$
- Role binding axiom: $R \triangleright C$
- Role assertion axiom: $R(a)$

Meaning? Translational semantics!

- Algorithm (sketch):
  1. Role types and role properties $\leadsto$ Classes and properties
  2. For each role type $R$:
     - If $R$ bound to $C_1, \ldots, C_n$, then $R \sqsubseteq C_1 \sqcup \cdots \sqcup C_n \sqcup \bot$
  3. Role assertion $\leadsto$ Class assertion

- Note: Unbound roles lead to unsatisfiable classes
Implementation with Reuseware

Reuseware: Invasive composition for arbitrary languages

- **Modularity Extension**
  - Role modeling syntax, hooks & slots

- **Core Language**
  - Manchester OWL

- **Reuse Language**
  - Manchester OWL with roles

- **Composers**
  - import, plays

- **Components**
  - Role models

- **Result**
  - Class-based ontology

- **Composition Program**
  - Role-based ontology

written in language
Composers

import http://ontology-rolemodels.org/product.rm

Class: Wine
  Plays: Product
Class: Winery
  Plays: Producer
Class: Food
  Plays: Product
Composers

Class: Product

Class: Producer
  EquivalentTo: produces some Product

Class: Consumer
  EquivalentTo: consumes some Product

Class: Wine
  Plays: Product
Class: Winery
  Plays: Producer
Class: Food
  Plays: Product
Composers

Class: Product
  SubClassOf: owl:Nothing or <<ProductSubClassHook>>

Class: Producer
  EquivalentTo: produces some Product
  SubClassOf: owl:Nothing or <<ProducerSubClassHook>>

Class: Consumer
  EquivalentTo: consumes some Product
  SubClassOf: owl:Nothing or <<ConsumerSubClassHook>>

Class: Wine
  Plays: Product

Class: Winery
  Plays: Producer

Class: Food
  Plays: Product
Composers

Class: Product
   SubClassOf: owl:Nothing or Wine or <<ProductSubClassHook>>

Class: Producer
   EquivalentTo: produces some Product
   SubClassOf: owl:Nothing or <<ProducerSubClassHook>>

Class: Consumer
   EquivalentTo: consumes some Product
   SubClassOf: owl:Nothing or <<ConsumerSubClassHook>>

Class: Wine

Class: Winery
   Plays: Producer

Class: Food
   Plays: Product
Composers

Class: Product
   SubClassOf: owl:Nothing or Wine or <<ProductSubClassHook>>

Class: Producer
   EquivalentTo: produces some Product
   SubClassOf: owl:Nothing or Winery or <<ProducerSubClassHook>>

Class: Consumer
   EquivalentTo: consumes some Product
   SubClassOf: owl:Nothing or <<ConsumerSubClassHook>>

Class: Wine

Class: Winery

Class: Food
   Plays: Product
Composers

Class: Product
  SubClassOf: owl:Nothing or Wine or Food <<ProductSubClassHook>>

Class: Producer
  EquivalentTo: produces some Product
  SubClassOf: owl:Nothing or Winery or <<ProducerSubClassHook>>

Class: Consumer
  EquivalentTo: consumes some Product
  SubClassOf: owl:Nothing or <<ConsumerSubClassHook>>

Class: Wine

Class: Winery

Class: Food
Composers

Class: Product
  SubClassOf: owl:Nothing or Wine or Food

Class: Producer
  EquivalentTo: produces some Product
  SubClassOf: owl:Nothing or Winery

Class: Consumer
  EquivalentTo: consumes some Product
  SubClassOf: owl:Nothing

Class: Wine

Class: Winery

Class: Food
Composers

Class: Product
  SubClassOf: Wine or Food

Class: Producer
  EquivalentTo: produces some Product
  SubClassOf: Winery

Class: Consumer
  EquivalentTo: consumes some Product
  SubClassOf: owl:Nothing

Class: Wine

Class: Winery

Class: Food
Implications of Translation

- Multiple uses of one role type translated into the same class
  → An individual of class *Product* can be a wine, a food, or both
  → Get all products of the ontology

- Open role types are a subclass of *owl:Nothing*
  → Individuals of *Consumer* make ontology inconsistent
  → Individuals need a natural type
Demo...
Integration into Protégé

The image shows a screenshot of the Protégé 3.3 interface with the "Role Model Browser" and "Role Editor" panels open. The browser includes role models such as "Product" and "Meal," while the editor is focused on the "Product" role, with options to add conditions and create new role expressions. This demonstrates the integration of role models into the Protégé platform for ontology composition with roles.
Outlook: Other uses of roles in ontologies

- Goal: Relate and combine independently developed ontologies
- Alignment & Merging
- What if classes semantically overlap?

- Idea: Compose role types
  → Only one concern: More precise matching

- Process:
  1. Align role models
  2. Compose classes based on role type alignment
Example: Pizza and Pasta Ontologies
Example: Pizza and Pasta Ontologies
Example: Pizza and Pasta Ontologies
Example: Pizza and Pasta Ontologies

- Country
  - Origin
    - hasOrigin
  - Thing
    - hasOrigin.Origin
- Pizza
  - Meal
    - accompanies
- Pasta
  - Product
    - produces
  - Consumer
    - consumes
- Wine
  - Drink
    - accompanies.Meal
Conclusion

- Ontologies need roles as first class concept
  - Role models = Ontological components
  - Reusable, intuitive unit of abstraction
  - Separation of concerns
  - More natural modeling
- Translational semantics
  - Compatible with existing tools
  - Reuseware-based implementation
- Open questions
  - Different semantics (must-play?)
  - Multiple uses of one role model

See also:

A Good Role Model for Ontologies: Collaborations
M. Pradel, J. Henriksson, U. Aßmann

Workshop on Semantic-Based Software Development at OOPSLA'07
Thanks!