Semantics in Software Engineering - Towards Ontology-Driven Software Development

Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Innovationsforum Software Saxony
April 18, 2008
What does Dresden have to do with Ontologies?
Outline

Introduction

Differences of ontology languages and UML/MOF/OOP

Advantages of ontologies in product-line engineering
   1) Future domain models will be ontologies
   2) Ontologies as constraints in product-lines

Integration technologies

[Challenges for integration and the EU MOST project]
Analysis and Design in Modern Product-Line Engineering (PLE)

Domain Model

Analysis Model

Product Line Design Model (Framework)

Product Design Model

Implementation

Domain vocabulary of the product family

Specifications for requirements of a product family

Technical design of the product family (commonalities)

Technical design of the products

Implementations of products

Design Variants

Code Components
Product-Line Engineering with Ontologies

- Domain Ontology
- Analysis Model
- Product Line Design Model (Framework)
- Product Design Model
- Implementation
- Code Constraints
- Design Constraints
- Business Rules
- Configuration Constraints
- Design Variants
- Code Components

Variants

Analysis Model

Product Line Design Model (Framework)
Considerable Differences of Ontology and Modeling languages

OWL vs UML

Technological spaces
OWL

Classes are sets

Classes and relations can be defined by *expressions*
- with set union, difference, intersection (Venn diagrams)
**UML vs OWL**

- **Extensional, explicit specification of classes and class hierarchy (UML, OWL)**
- **Intensional, implicit specification of classes and class hierarchy (OWL class calculus)**
- **Reasoning about inheritance (subsumption)**
- **Integrity constraints (OWL constraints)**
- **Integrity constraints (OCL)**
- **Behavior (Methods) (UML-Statecharts,..)**
- **no implicit classes**
- **implicit classes**
**OWL uses Sets**

**Intensional specification**
- Creating new classes by expressions (class calculus)
- Father = Male $\cap$ Parent

**Reasoning about the inheritance relations in the resulting lattice**
- disjointness of classes
- finding out whether a specific class exists (concept satisfiability)

**Integrity constraints**
- All Father: Male
Advantages of Ontology Languages in Product-Line Engineering
Product-Line Engineering with Ontologies as Domain Models

- Domain Ontology
- Analysis Model
- Product Line Design Model (Framework)
- Product Design Model
- Implementation
- Configuration Constraints
- Design Variants
- Code Components
- Business Rules
- Design Constraints
- Code Constraints
1) Future Domain Models will be Ontologies

A domain may be very large and complicated
- Expressions describe it
- Integrity constraints are checked by reasoner

Domain experts aren't software engineers
This influences Product Data Engineering

- Intensional, implicit specification of classes and class hierarchy (OWL class calculus)
- Integrity constraints (OWL constraints)
- Reasoning about inheritance (subsumption)
Example: Car Industry

A Phaeton has 10000 parts

- Life-time tracked
- Many different variants (individualized), many integrity constraints
  - “diesel ∩ catalysator” does not exist
  - “gas ∩ russfilter” does not exist
  - “diesel ∩ 10-cylinder ∩ cabriolet” exists

How to model this appropriately?

Ontologies are good for product data engineering (PDE) in supply chains
Ex.: Gene Ontology

Saxony wants to be a Biotechnology region

Checkout www.geneontology.org

- Component Ontology:
  - Rules governing content and stylistic aspects of GO terms in the cellular component ontology.
  - The Cell Protein Complexes Membranes and Envelopes ..

- Function Ontology:
  - Rules governing content and stylistic aspects of GO terms, standard definitions and term relationships in the molecular function ontology.

- Process Ontology:
  - Rules governing content and stylistic aspects of GO terms, standard definitions and term relationships in the biological process ontology.
  - The Cell Cycle The Development Node Interaction Between Organisms ...
2) Product-Line Constraints in Ontologies

Diagram:
- Domain Ontology
- Analysis Model
- Product Line Design Model (Framework)
- Product Design Model
- Implementation
- Business Rules
- Design Constraints
- Code Constraints
- Configuration
- Constraints
- Design
- Variants
- Components
- Code
- Variants
Examples: Integrity Constraints in Phaeton Product Line

Business rules
- FavoredCustomer = Customer and Customer.turnaround > 50000
- “John Silver” instanceof FavoredCustomer?

Configuration constraints
- CheapVersion = Phaeton ∩ 6-cylinder
- ExpensiveVersion = Phaeton ∩ 10-cylinder ∩ Diesel ∩ Cabriolet

Design and code constraints
- #Component.neighbors < 10
- #Class.methods < 20
Unfortunately:
Two Separate Technological Spaces

- M3 metametamodel level
  - OWL

- M2 metamodel level
  - Upper ontologies (basic description concepts)
    - SUMO GFO BFO CYC
  - UML-based metamodels (languages)

- M1 model level
  - Domain ontologies
  - UML Models

- M0 object level
  - Real world objects
  - Software objects

U. Aßmann, TU Dresden
Integration Technologies for Ontologies and System Models
How to Access the Domain Model and the Constraint Ontologies?

- Domain Model (Domain ontology)
- Analysis Model
- Product Line Design Model (Framework)
  - Design Constraints
  - Product Constraints
  - Business Rules

- Configuration Constraints
- Design Variants
- Code Components

- Implementation
  - Product Design Model
Black-Box Integration of Ontologies

Using an ontology as data base

SPARQL, RDQL querying as with Embedded SQL

```java
// Create a new query passing a String containing the RDQL to execute
Query query = new Query(queryString);
// Set the model to run the query against
query.setSource(model);
// Use the query to create a query engine
QueryEngine qe = new QueryEngine(query);
// Use the query engine to execute the query
QueryResults results = qe.exec();
```


Problem: Speed
A tight integration of ontologies into programs would be much faster
Tight Integration of Ontologies

A tight integration of ontologies into programs would be much faster

Solution: Prova (Prolog+Java)
- Prof. Michael Schröder, TU Dresden
- http://prova.ws
- Java classes can contain Prolog rules

Solution 2: Language Integration by metamodel Integration
- e.g., as sublanguages for data definition and integrity
- There should be one universal metalanguage for ontologies and system models
- --> MOST Project
Challenges for Integration of Ontology and System Modeling Languages
Models vs Ontologies – A Big Difference

Description or Control

Ontologies need the open-world assumption
- Analysis perspective
- Anything not explicitly expressed is unknown
- Ontologies use a form of partial description to abstract

System models need closed-world assumption
- Design perspective
- Anything not explicitly expressed is wrong
- System models specify completely

A model can be descriptive or prescriptive. [Seidewitz CACM 03]

Models describe or control reality.
If they describe, they monitor reality and form true, or faithful, abstractions (Analysis, Reengineering)
If they control, they prescribe reality (Construction, Specification)
Analysis with Ontologies, Specification with System Models

**An ontology:**

a standardized, descriptive model,
representing reality by a set of concepts, their interrelations, and constraints under open-world assumption.

**A system model:**

a non-standardized, prescriptive model,
representing a set of systems by a set of concepts, their interrelations, and constraints under closed-world assumption.
What to Do with What

With Closed World Reasoning

- Querying
  - needs CWA to exclude erroneous data
- Metamodelling:
  - needs CWA to exclude erroneous programs
- Integrity constraints
  - needs CWA to exclude erroneous models

With Open World Reasoning

- Domain modeling
  - needs OWA because of partial specification of domain
Integration with a Universal Metalanguage

M3 metametamodel level

M2 metamodel level

M1 model level

M0 object level

Universal CWA Ontology
Metamodelling language

<<instance-of>>
Upper ontologies
<<is-a>>
Domain ontologies
<<described-by>>
Real world objects

<<instance-of>>
Metamodelling language
(languages, language concepts)
<<instance-of>>
Models
<<instance-of>>
Software objects
Conclusions

Ontologies are advantageous in PLE for
- domain ontologies
- integrity constraint ontologies in product lines

but...
- Ontologies should not be misused as system models
- Ontologies *complement* system models
- Ontologies in OWA for domain modeling, CWA for the rest

Integration technology and tools needed!
Solution to the Riddle

Dresden wants to be a factory automation region

- see Track 3 of Innovationsforum

Ontologies are good for product data engineering (PDE) in supply chains

Dresden needs ontologies in domain models and PLE
Looking for Partners

MOST www.most-project.eu
- Comarch, SAP, BOC

Integration into PLE
- Metamodel integration
- Process guidance with ontologies
- Ontology-aware software development (ODSD)
References


www.reworse.net/i3

www.most-project.eu
The End