From Ontology as an Art to Ontology Engineering

Fabian Neuhaus, University of Magdeburg

Bremen – July 2018

Observation

- Applied ontology is more than 20 years old
- Why is adoption in industry so sluggish?

Observation

- Applied ontology is more than 20 years old
- Why is adoption in industry so sluggish?
- Hypothesis: Applied Ontology is in a pre-engineering stage

Research interest

- What AO is lacking to become an engineering discipline?
- Two research areas
  - Modularity
  - Evaluation

Modularity

Ontologies are often big monolithic blobs

National Center for Biotechnology Information (NCBI) Organismal Classification (NCBITAXON)

projects: 12 | classes: 906,907

The NCBI Taxonomy Database is a curated classification and nomenclature for all of the organisms in the public sequence databases.

Uploaded: 6/10/15

The Drug Ontology (DRON)

classes: 408,573

An ontology of drugs

Uploaded: 5/2/15

Systematized Nomenclature of Medicine - Clinical Terms (SNOMEDCT)

notes: 2 | projects: 18 | classes: 316,031

SNOMED Clinical Terms

Uploaded: 6/10/15

Robert Hoehndorf Version of MeSH (RH-MESH)

projects: 3 | classes: 305,349

Medical Subjects Headings Thesaurus 2014, Modified version

Uploaded: 4/22/14

Cell Cycle Ontology (CCO)

projects: 2 | classes: 277,764

An application ontology integrating knowledge about the eukaryotic cell cycle.

Uploaded: 3/7/15

Obvious benefits of modular design

Modularity allows for better

- Maintainability
- Reusability
- Quality control
- Adaptability

Additional challenges

- Ontologies: modules of bigger systems
- Requirement: integration with other knowledge resources (e.g., conceptual models) often in other languages

Challenge: Where is the glue?

- The different modules need to be fitted together.

Challenge: Where is the glue?

- The different modules need to be fitted together.
- Languages may differ widely with respect to syntactic categories

Fundamental idea behind DOL

Distributed Ontology, Model, and Specification Language (DOL)
OMS = Ontology, Model, Specification

- semantics of OMS: based on various logics
- abstract from specifics of presentation (institutions)
- language which operates on abstractions

DOL metalanguage

Enables reusability and interoperability

- Literally reuse existing OMS
- Operations for modifying/reusing OMS
- Declaration of relations between OMS

Extremely Simple DOL Example

```
%prefix( : <http://example.com/> )%

logic OWL

ontology Driving =
 Class: Vehicle
 ObjectProperty: drives
  Range: Vehicle

ontology DrivingExtended =
 Driving
then
 Class: Person
 ObjectProperty: drives
  Domain: Person
```

Example for DOL Features

- Reuse of existing ontologies
- Extending ontologies
- Removal of axioms
- Blending (Colimit) of ontologies
- Translation to other languages
- Alignment of ontologies
- Switch to closed world
- Express logical relations between ontologies

DOL standardization

- Adopted as OMG Standard
- Fits in OMG standard family

OBJECT MANAGEMENT GROUP®

DOL enables modular design

- Reusability (combination, translation)
- Adaptability (renaming, filtering, reduction, minimization)
- Maintainability & quality control (proof obligations)

Generalized DOL

- Goal: Represent Ontology Design Patterns
- Means: Add parameters to DOL

Ontology Design Patterns in Generalized DOL

- Ontology Design Patterns definition includes typed parameters

```
ontology SimpleRelationODP
        [ObjectProperty: p] [Class: D][Class: R] =
  ObjectProperty: p
        Domain: D
        Range: R
```

Application of Design Pattern

```
ontology DrivingPatternInstance =
  SimpleRelationODP[drives][Person][Vehicle]
```

is computed to

```
ontology DrivingPatternInstance  =
  ObjectProperty: drives
        Domain: Person
        Range: Vehicle
  Class: Person
  Class: Vehicle
```

Evaluation / Scone

The Ontology Evaluation Puzzle

- There is no lack of research on ontology evaluation.
- Ontologists believe that quality of ontologies is important.
- Ontology evaluation plays a minor role in ontology development.

What's the approach? (1)

'Borrow' from BDD!

What's the approach? (2)

Requirements:

- Capture requirements in controlled English
- Described with the help of scenarios
- What inferences should (not) be possible?

What's the approach? (3)

Automatic generation of tests

- Semantics of controlled English: mapping to DOL / OWL
- Parse scenarios and generate behavioral tests (DOL / OWL)
- Execute tests with automatic reasoner

What's the approach? (4)

Measurement of progress

- Successful tests correlate to met requirements
- Is the ontology too narrow? (Range)
- Are all elements of the ontology tested (Coverage)

Example

Assume we want to build an ontology that covers family relationships.
How does a Scone file look like?

A Scone file – structure

```
Feature: <name>

Background:
        ...

Scenario: <name>
        ...

Scenario: <name>
        ...

Scenario: <name>
        ...
```

A Scone file – feature

```
Feature: Family relationships
The user should be able query the data using "male",
"female", "parent of", "grandparent of", "father of",
"mother of", "older than"
```

A Scone file – background

```
Feature: Family relationships
The user should be able query the data on using "male",
"female", "parent of", "grandparent of", "father of",
"mother of", "older than"

Background:
 * Language OWL
 * Test the ontology <https://example.org/family_rel.owl>
```

A Scone file – assumptions

```
Feature: ...

Background: ...

Scenario: Relative age between family members

Given Chris is a parent of Dora.
And Amy is a parent of Chris.
And Amy is a parent of Berta.
```

A Scone file – competency questions 1

```
Feature: ...

Background: ...

Scenario: Relative age between family members

Given Chris is a parent of Dora.
And Amy is a parent of Chris.
And Amy is a parent of Berta.
Then infer Chris is older than Dora.
And infer Amy is older than Dora.
```

A Scone file – competency questions 2

```
Feature: ...

Background: ...

Scenario: Relative age between family members

Given Chris is a parent of Dora.
And Amy is a parent of Chris.
And Amy is a parent of Berta.
Then infer Chris is older than Dora.
And infer Amy is older than Dora.
And don't infer Berta is older than Dora.
And don't infer Dora is older than Dora.
```

A Scone file – keywords highlighted

```
Feature: ...

Background: ...

Scenario: Relative age between family members

Given Chris is a parent of Dora.
And Amy is a parent of Chris.
And Amy is a parent of Berta.
Then infer Chris is older than Dora.
And infer Amy is older than Dora.
And don't infer Berta is older than Dora.
And don't infer Dora is older than Dora.
```

A Scone file – complex axioms

```
Feature: ...

Background: ...

Scenario:   Inferring various family relationships

Given John is a parent of Mary.
And Sue is a mother of John.
Then infer that Sue is a grandparent of Mary.
Given John is male.
Then infer that John is a father.
Given a mother is defined as a female, who is a parent of some thing.
Then infer that Sue is a mother.
```

A Scone file – inconsistency

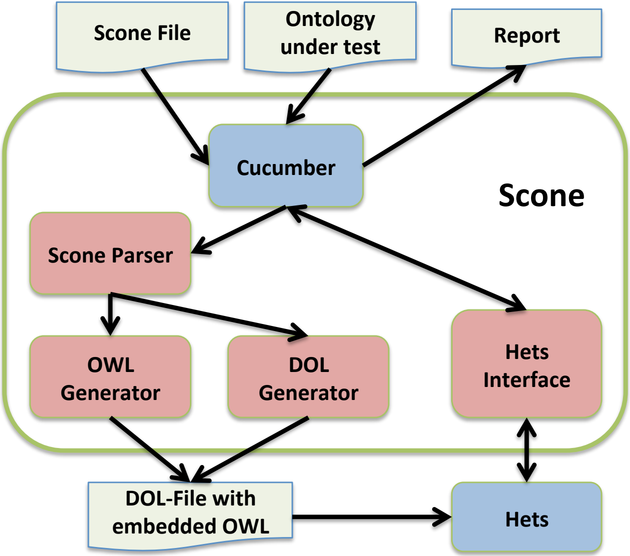
```
Feature: ...

Background: ...

Scenario: Mothers are female

Given Jill is the mother of Chris.
Given Jill is not female.
Then the scenario is inconsistent.
```

System Architecture



Created DOL File

```
logic OWL
ontology Import0 = <file:///.../familyRel.owl>
ontology CombinedImports = Import0

ontology Scenario = CombinedImports then
        Individual: Mary
        Individual: John Facts: fr:parent_of Mary
        Individual: Sue Facts: fr:mother_of John
        Individual: John Types: fr:Male
        Class: mother EquivalentTo:
    ( fr:Female and fr:parent_of some owl:Thing )
end

ontology CompetencyQuestion1 = Scenario

ontology CQ1 = CompetencyQuestion1 then %implies
        Individual: Sue Facts: fr:grandparent_of Mary
end
```

Metrics

- Ratio of scenarios successfully executed (requirements met)
- Range of ontology $= \frac{\text{classes \& properties in CQs that occur in ontology}}{\text{all classes \& properties in CQs}}$
- Coverage of scone $= \frac{\text{number of terms in ontology that are tested}}{\text{number of terms in ontology}}$

Future Work

- Make prototype solid
- Develop Atom Editor Plugin
- Apply Scone in real world
- Improve metrics
- Expand approach to wider range of behavioral tests

Conclusions

Towards ontology engineering

- DOL enables modular design of ontologies
- GDOL enables representation of ontology design patterns
- SCONE allows to track requirements, generate tests, measure progress