TOWARDS AN EFFECTIVE METHODOLOGY FOR RAPIDLY DEVELOPING COMPONENT-BASED DOMAIN ONTOLOGIES

Dave Kolas Troy Self
Today’s Talk

- Why Composite Domain Ontologies?
- Software Engineering vs Ontology Engineering
- Ontology Engineering Methodology
- Compositeness Metric
  - Desirable Properties
  - Definition
  - Examples
- Going Forward
Composite Domain Ontology Engineering

Problem

- Existing research into ontology engineering focuses on:
  - Creating new ontologies
  - Aligning existing, redundant ontologies
  - Interacting with ontologies using software tools
- No real effort spent on high-level engineering of cohesive ontologies

Goal

- Develop a methodology for linking and re-purposing ontologies in an agile manner to enable better interoperability between ontology-based applications and services
Composite Domain Ontologies

Definition

- Domain ontology developed by linking multiple ontologies about various topics through imports
- Analogue to component-based software development or Service-Oriented Architecture (SOA)
As the amount and complexity of software systems has increased, so has the amount of effort dedicated to the study of software engineering.

- More time and effort are spent attempting to make software reusable.
- Metrics have been developed to track the quality of software in many different ways.
- These techniques enhance the understanding of the process and reduce cost and risk.

As ontologies become more widely used, the same changes will happen.
Ontologies and Software Components share some common attributes:

- Used in multitude of applications, including some unanticipated by designer
- Likely to evolve after initial creation
- Cost to replicate is minimal compared to the cost to develop
- Reuse has potential to save considerable engineering resources
- Must be specific enough to be useful, yet general enough to be reusable
Software Metrics vs. Ontology

**Software Metrics**

- Cyclomatic Complexity
- # of classes and interfaces
- Program Size
- Coupling
- Cohesion

**Ontology Metrics**

- Level of Expressiveness?
- # of classes and properties
- # of statements, axioms
- Compositeness?
- ?
Composite Domain Ontologies

- Why/when to use composite ontologies?
  - Time savings
  - Leverage existing software
  - Take advantage of existing reasoning and translation
  - Scope vs Applicability
Building a Composite Domain Ontology

- Start With the Application Domain
  - What questions need to be asked?
  - What does the instance data look like?
  - What type of inferences are required?
- Divide application domain into reusable subsections
- Identify existing ontologies to fulfill section requirements
- Create new ontologies when necessary
Compositeness Metric Goals

□ We have developed a new compositeness metric for ontologies
□ This metric should indicate:
  ▪ How much an ontology’s creators were able to reuse existing ontologies
  ▪ How usable an ontology is in different situations with different pieces of software
  ▪ A relative idea of how long an ontology should take to develop
Compositeness Metric Definitions

- **Compositeness** as a metric
  - Degree to which an ontology is made up primarily of other ontologies, and the compositeness of those ontologies.

- **Definitions**
  - \( C(X) \): Compositeness of ontology, \( X \).
  - \( S(X) \): Size of ontology, \( X \).
  - \( I(X,Y) \): Ontology(\( X \)) imports Ontology(\( Y \)).
Properties of Compositeness Metric (1)

- If X imports Y, then X is more composite than Y.
  \[ I(X, Y) \rightarrow C(X) > C(Y) \]

- If X and Y both import Z and X is smaller than Y, then X is more composite than Y.
  \[ I(X, Z) \land I(Y, Z) \land S(X) < S(Y) \rightarrow C(X) > C(Y) \]
Properties of Compositeness Metric (2)

If X and X’ are identical except that they import 2 separate ontologies, then the one that imports the more composite ontology has a higher compositeness.

I(X, Y) \land I(X', Z) \land C(Y) > C(Z) \rightarrow C(X) > C(X')

If X and X’ are identical except that X imports an extra ontology, then X has higher compositeness.

(I(A, B) \land I(A', B) \land I(A', C) \land \exists X: (I(A,X) \&\& X = B)) \rightarrow C(A') > C(A)
Compositeness Metric

\[ C(X) = \sum_{y \in I(X)} C(y) + \frac{\sum_{y \in I(X)} S(y)}{S(X)} \]

- An ontology that imports no other ontologies will have a compositeness score of 0
We have used the ontologies developed on the GARCON-F to evaluate the metric

- **Air Defense** ontology defines the domain of air defense
- **Foundational** ontology defines spatial annotations
- **BFO** (OWL version)
- **OWL-Time** standard temporal information
- **GeoRSS** defines simple spatial locations
- **RAID** defines the domain of Marine raid missions
- **Transportation** defines vehicles, cargo, etc
- **Movement** defines simple movements
- **SAULTE** defines the contents of a simple report
Composite Domain Ontologies

C(X) = 4.95
Time to develop: 30 days

Ontology X imports Ontology Y

C(X) = 4.4

Basic Formal Ontology (BFO)
C(X) = 0

Air Defense

Foundational
C(X) = 4.4

OWL-Time
C(X) = 0

GeoRSS
C(X) = 0
Composite Domain Ontologies

SALUTE

C(X) = 31.4
Time to develop: 1 hour

Foundational

Basic Formal Ontology (BFO)
C(X) = 0

OWL-Time
C(X) = 0

GeoRSS
C(X) = 0
Going Forward

- This is only a small start...
  - How can we improve/further validate compositeness metric?
  - What is the ontology equivalent of “cohesion”?
  - What useful ontology metrics exist that do not have software analogs?
Thank You

Questions?