Practical Applications of Semantic Web Technologies in Domain-specific Information Systems

MARTIN LEDVINKA
martin.ledvinka@fel.cvut.cz
Knowledge-based And Software Systems Group
FEE CTU in PRAGUE
# Table of Contents

1. Why? What’s the Catch?
2. Object-ontological Mapping
3. Java OWL Persistence API
4. Practical Applications of Semantic Web Technologies in Domain-specific Systems
5. Conclusions
KBSS @ FEE CTU in Prague

- [https://kbss.felk.cvut.cz](https://kbss.felk.cvut.cz), [https://github.com/kbss-cvut](https://github.com/kbss-cvut)

- Research areas
  - Ontologies and Semantic Web
  - Linked Open Data
  - Domain modeling and ontology engineering
  - Ontology-based information systems
  - Ontology authoring supported by text analysis
  - Reasoning and query answering in Description Logics
Glossary

• Semantic Web (SW) technologies
  • RDF(S), OWL (2), ontologies, SPARQL, Linked Data etc.

• Domain-specific Systems
  • (Information) systems supporting users in a particular domain, built upon a domain model
  • Concepts and relationships from the domain are directly reflected in the system
  • E.g., aviation, urbanism
Why? What’s the Catch?
Why Use SW Technologies?

• Formal ontologies allow precise conceptualization of the domain

• Globally identified constructs (concepts, properties) facilitate interoperability

• Model created in expressive languages allows to infer implicit knowledge from data

• Standardized data definition, manipulation and query languages

• Standardized data serialization formats
Domain Conceptualization?

Building is a construction
- both above and below ground
- spatially compact
- with walls and roof
- with heating

Building is a construction
- above ground
- with solid foundations
- spatially compact
- with walls and roof


Act 256/2013 Coll., Cadastral Law
Domain Conceptualization

- Event
- Building (process)
  - Building (406/2000)
  - Building (256/2013)
- Object
  - Building (object)
  - Foundations

Connections:
- Event has participant Building
- Building (process) has participant Building (406/2000)
- Building (process) has participant Building (256/2013)
- Building (object) has part Foundations
Unique Identifiers and Precise Definition


skos:definition “Construction both above and below ground ...” ;


<http://onto.fel.cvut.cz/ontologies/laws/256-2013/Building>

skos:definition “Construction above ground ...” ;

Query Languages and Inference

```
SELECT * WHERE {
    ufo:has-part+ ?part .
}
```
So What’s the Catch?

- Efficient access to Semantic Data storage is problematic
  - No standard defined (besides doing everything through SPARQL)
  - Number of ad hoc solutions with often imprecise semantics
  - Statement-level vs. domain object-level APIs
- Many solutions prototypical with no support and scarce documentation
- Lack of success stories
- Developers not used to SW technologies and their background
- Performance?
SW vs Programming World Mismatch

- Statement-based processing
- Open-world assumption
- Expressive inference
- Dynamic nature (e.g., attach any properties to individuals)
- Object-oriented programming
- Typically closed-world
- Language-level inference only (subclasses)
- Rigid object model (compiled languages)
Object-ontological Mapping
Why?

- RDF(S) -> 90 triples
- CRUD? REST API? Maintenance?
  - Nightmare
- Object model -> 5 classes
- CRUD? REST API?
  - Plenty of support out there
- Easier to work with domain objects in an application
Object-ontological Mapping (OOM)

- Also object-triple mapping (OTM)
- Basic mapping quite simple, but many subtle issues (multiple inheritance, inference, unmapped properties...)
- Formalization still does not exist (PhD goal)

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDFS class</td>
<td>Entity class</td>
</tr>
<tr>
<td>RDF property</td>
<td>Attribute</td>
</tr>
<tr>
<td>RDFS class instance</td>
<td>Entity (object)</td>
</tr>
</tbody>
</table>
Java OWL Persistence API
Java OWL Persistence API (JOPA)

- Java persistence library for Semantic Web-based applications
- Performs object-ontological (triple) mapping
  - Triples/Axioms to objects with attributes and vice versa
- Similarity to JPA not coincidental
  - Entities
  - Persistence context
  - Transactions
  - Queries
  - Separate storage access layer
- [https://github.com/kbss-cvut/jopa](https://github.com/kbss-cvut/jopa)
```java
@MappedSuperclass
abstract class AbstractUser implements HasIdentifier, HasTypes, Serializable {

    @Id
    URI uri;

    @ParticipationConstraints(nonEmpty = true)
    @OWLDataProperty(iri = cz.cvut.kbss.termit.util.Vocabulary.s_p_na_krestni_jmeno)
    String firstName;

    @ParticipationConstraints(nonEmpty = true)
    @OWLDataProperty(iri = cz.cvut.kbss.termit.util.Vocabulary.s_p_na_prijmeni)
    String lastName;

    @ParticipationConstraints(nonEmpty = true)
    @OWLDataProperty(iri = Vocabulary.s_p_na_uzivatelske_jmeno)
    String username;

    @Types
    Set<String> types;

    /**
     * Persists the specified instance into the repository.
     * @param instance The instance to persist
     */
    @Transactional
    public void persist(@NonNull T instance) {
        Objects.requireNonNull(instance);
        prePersist(instance);
        getPrimaryDao().persist(instance);
    }

    @Override
    public Optional<T> findById(URI id) {
        try {
            return Optional.ofNullable(em.find(type, id));
        } catch (RuntimeException e) {
            throw new PersistenceException(e);
        }
    }

    @Override
    public void persist(T entity) {
        Objects.requireNonNull(entity);
        try {
            em.persist(entity);
        } catch (RuntimeException e) {
            throw new PersistenceException(e);
        }
    }
```
Comparison of Object-triple Mapping Libraries

- Semantic Web journal
- DOI: 10.3233/SW-190345
- Designed a framework for OTM library comparison
  1. Qualitative
  2. Benchmark
     i. Performance
     ii. Memory
Comparison of OTM Libraries - Qualitative

• 12 criteria
  A. General
     • Transaction support, Storage access variability, Query result mapping, Object-level query language, Detached objects, Code/ontology generator
  B. Ontology-specific
     • Explicit inference treatment, Named graphs, Automatic provenance generation
  C. Mapping
     • Inheritance mapping, Unmapped data access, RDF collections and containers

• 12 libraries evaluated
  • ActiveRDF, AliBaba, AutoRDF, Empire, JAOB, JOPA, KOMMA, RDFBeans, RDFReactor, The Semantic Framework, Spira, SuRF
Comparison of OTM Libraries - Benchmark

- Benchmark application using aviation safety reporting tool model
- Java, RDF4J API (GraphDB used as storage)
- 6 CRUD operations
  - Create, Batch create, Retrieve, Retrieve all, Update, Delete
- 6 heap sizes
- 5 libraries evaluated
  - AliBaba, Empire, KOMMA, JOPA, RDFBeans
- By-product – demo application for each of the evaluated libraries
Comparison of OTM Libraries - Performance

Heap size – 1GB

<table>
<thead>
<tr>
<th>OP1 – Create</th>
<th>OP2 – Batch create</th>
<th>OP3 – Retrieve</th>
<th>OP4 – Retrieve all</th>
<th>OP5 – Update</th>
<th>OP6 – Delete</th>
</tr>
</thead>
</table>

07/03/2019
Comparison of OTM Libraries - Results

- OTM libraries significantly differ in supported features
- More important is that the semantics of their operation are often radically different
- Heap size has only minor effect on performance
- Some libraries support only RDF/OWL file access
  - Not suitable for application deployment
- Many libraries support only RDF4J API
  - But most triple stores support it, so not a big deal
- Languages without static type checking represent closer match for ontologies
  - But static model allows type checking and reveals typos early
Practical Applications of Semantic Web Technologies in Domain-specific Systems
Our Projects

• Reporting Tool + SISEl
  • Aviation safety

• Dataset Dashboard
  • Exploration of Linked Data datasets

• Study Manager
  • Support for clinical trials

• TermIt
  • Domain vocabulary management
Reporting Tool

• (Aviation) safety occurrence reporting, management and analysis
• For organizations reporting safety occurrences to an authority
• Built upon Aviation Safety Ontology
  • Unified Foundational Ontology (UFO)-based conceptualization of the domain
  • Consists of aviation, safety, documentation modules
• Model uses class hierarchies
• Inference for statistics done in SPARQL queries
• https://github.com/kbss-cvut/reporting-tool
SISeI

- Extension of Reporting Tool for the Czech Aviation Authority
- Manage, integrate and analyze reports
- Large taxonomies (ECCAIRS), diverse inputs, legislation
- *Towards Data-Driven Safety: An Ontology-Based Information System*
  - Journal of Aerospace Information Systems
TermIt

- Vocabulary management system
- Allows to manage terms with definitions and relationships
- Rich domain model
  - UFO-based ontology with roots in Dataset Descriptor Ontology
  - Inference – class and property hierarchy, inverse properties
- Complete Semantic Web-based stack
  - Semantic repository - currently GraphDB Free
  - JOPA for data access
  - REST API supporting JSON-LD
TermIt

- Vocabularies consisting of
  - Glossary – basic term hierarchy based on SKOS broader/narrower relationships
  - Model – detailed domain model based on glossary terms
- Vocabulary terms used to annotate resources
- Semantics-based search
  - Terms assigned to resources – more precise than annotation with keywords
  - Resources related via terms
- NLP techniques used to
  - Find existing term occurrences in file content
  - Suggest new terms based on token significance in file content
Termlt Use Case – Prague Urbanism
TermIt Use Case – Semantic Government Vocabulary

- **Basic Vocabulary**
  - *E.g. Event, Kind, Relator*
  - Based on the **Unified Foundational Ontology**

- **Vocabulary of Public Sector**
  - *E.g. Legal Subject, Organization*
  - Mapped to **ISA Core Vocabularies**

- **Vocabularies of Legislation**
  - One vocabulary per *Act*

- **Vocabularies of Agenda**
  - One vocabulary per *Agenda*

- **Vocabularies of Datasets**
  - One vocabulary per *Data Schema* (to support data series)
Conclusions
Our Experience

• Mature, easy to use and documented libraries are crucial for development

• IRIs are difficult to use in a REST API

• RDF4J has performance issues when using SPIN inference

• No matter how meticulously a domain model is designed, it never survives practical deployment in the original form

• It is possible to integrate SW libraries with popular development libraries
  • Spring, React, TypeScript
Our Tools

• JOPA
  • https://github.com/kbss-cvut/jopa

• JB4JSON-LD and JB4JSON-LD Jackson
  • https://github.com/kbss-cvut/jb4jsonld

• SPipes
  • https://kbss.felk.cvut.cz/web/kbss/s-pipes

• SForms
  • https://kbss.felk.cvut.cz/web/kbss/s-forms

• Dataset Dashboard
  • https://kbss.felk.cvut.cz/web/kbss/dataset-dashboard
Thank You