Semantic Web Enabled Software Engineering — State of the Art and Near Term Roadmap

SWES 2010 keynote, 2010

Jeff Z. Pan
University of Aberdeen, UK
Leader of Knowledge Technology Group

Elisa Kendall
Sandpiper Software
Chief Executive Officer

Why Semantics Can Help?
Model Driven Software Development

Computation-Independent Model (CIM)
Platform-Independent Model (PIM)
Platform-Specific Model (PSM)
Code

RDF / OWL

Fragmentation of Models
Loss of Oversight

CIM2PIM
PIM2PSM
PSM2Code
Code to RDF/OWL

package org.sindice.sporc.projects.vsp;
import java.awt.Component;
import java.io.IOException;
import java.lang.String;
public class HelperThread{
    public run()
    public getDocument()
}

[Iqbal et al. 2009]

Subversion to RDF/OWL

Revision: 275
Type: committing
Modified: http://sindice.com/wp/HelperThread.java
Added: http://sindice.com/wp/Splitter.java
Author: source
Connected via RDF/OWL

- EvoOnt A Software Evolution Ontology
  - Software Analysis

- Many in MSR (Mining Software Repository) Workshop 2004 - 2007 can be done by 1~2 SPARQL query

[Tappolet et al. 2010]
Research Problems and the State of the Art

Research Questions

- RQ1: Bridging the semantics of SE models and ontologies
- RQ2: Reducing SE problems to ontology reasoning
- RQ3: Efficient tool support
TwoUse: Example

TwoUse

FreeTradeZone

Product

UML

DutyFreeOrder

OrderFromEU

Customer

getCharges()
TwoUse: Example

<<ocdlExpression>>
{context PurchaseOrder::getCharges() : Real
body: if self.isOwlInstanceOf(DutyFreeOrder)
then 0% else 60% endif}

Guidance for Physical Devices Modelling

[Ren et al. 2009]
Integrated Model

- PhysicalDevice DSL (PDDSL)
  - Structural modelling
- OWL2-DL
  - Semantic constraints on PDDSL
- Integrated PDDSL (IPDDSL)
  - Structural modelling with expressiveness of OWL2-DL

DeviceType "Cisco_7603"
  SubClassOf: pd_hasConfiguration some pd_hasSlot some pd_hasCard some Cisco_7600_SIP
  longName: "CISCO 7603 CHASSIS" description: "The Cisco® 7603 Router is a high-performance..."
  PossibleConfiguration "Cisco_7603_Configuration" (Slot 1 allowed: "Supervisor_Engine_2", "Supervisor_Engine_720" required: true
  Slot 2 allowed: "Supervisor_Engine_2", "Supervisor_Engine_720", "Catalyst_6500_Module"
  Slot 3 allowed: "Catalyst_6500_Module" required: false

Device serialNumber: "cisco_7603" hasType: "Cisco_7603"
  configuration: (Slot id: 1: Card serialNumber: "supervisor_2_5" hasType: "WS-X6K-S2U-MSFC2"
  Slot id: 2: Slot id: 3: )

[Miksa et al. 2009]

Guidance for Physical Devices Modelling
Consistency Guidance Component

- Encapsulates the burden of transformation to OWL2 and interactions with reasoner
- Guidance services:

<table>
<thead>
<tr>
<th>Component service</th>
<th>Description</th>
<th>Reasoning services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device type validation</td>
<td>Detect errors in physical device type definition</td>
<td>Satisfiability checking</td>
</tr>
<tr>
<td>Device validation</td>
<td>Find wrongly configured instances of devices and explain errors</td>
<td>Consistency checking, Explanations</td>
</tr>
<tr>
<td>Possible card category suggestions</td>
<td>Suggest card categories which are allowed in a slot</td>
<td>Subsumption reasoning</td>
</tr>
<tr>
<td>Allowed slots suggestions</td>
<td>Suggest possible slots for a card</td>
<td>Subsumption reasoning</td>
</tr>
</tbody>
</table>

Automatic Refinement Validation with Ontologies

- Transforming process refinement scenarios into ontologies
- Reducing refinement validation into ontology reasoning
- Justifying ontology inconsistency/incoherence
- Backtracking wrong refinement

[Ren et al. 2009b; Groener and Staab 2009]
Process Refinement

1. Getting the execution sets
   \{AB\} = \{a1b1b2a2, a1b1a2b2, a1a2b1b2, a1a3\}

2. Renaming
   \{AB\} = \{ABBA, ABAB, AABB, AA\}

3. Decomposition
   \{AB\} = \{ABA, ABAB, AB, A\}

4. Validation: invalid

How to represent and reason in ontology?

Process normalisation

1. Eliminating parallel gateways:
   Executions remain the same

2. Predecessor sets:
   PS(b12) = \{a11\};
   PS(a23) = \{b23\}, etc.

3. Successor sets:
   SS(a11) = \{a31, b12, a21\};
   SS(b12) = \{b23, a22\}, etc.

Exponential complexity (O(n!))

Execution sets subsumption can be reduced to PS/SS sets subsumptions
Refinement representation

1. In pre-refinement process:
   - Component_A subclassOf (to only Component_A or Component_B);
   - Component_B subclassOf from only Component_B or Component_A;

2. In post-refinement process:
   - a31 subclassOf to some End
   - b12 subclassOf (to some b23) and (to some a22);

Invalid

3. Renaming:
   - a31 subclassOf Component_A;
   - b23 subclassOf Component_B; etc.

4. Uniqueness:
   - Disjoint(Start, End, Component_A, Component_B), etc.

Tractable Reasoning for OWL 2

TrOWL: Tractable reasoning infrastructure for OWL 2 [Thomas et al. 2010]
- Quality guaranteed transformations (such as modularisation, approximations, forgetting [Wang et al. 2008, 2009, 2010])
  - OWL 2 DL -> OWL 2 QL (semantic approximation) [Pan and Thomas 2007]
  - OWL 2 DL -> OWL 2 EL (syntactic approximation) [Ren et al. 2010]
- Ontology reasoners (supporting OWL API and Protégé)
  - TrOWL Quill (OWL 2 QL / DL)
  - TrOWL REL (OWL 2 EL / DL)
- Explanation
  - Justification
  - Explanation on transformation

- URL: http://trowl.eu/
**Faithful Approximate Reasoning**

[Ren et al. 2010]

- Why bother?
  - Minor syntactic gap results in major complexity difference
  - Using approximation to bridge the gap

<table>
<thead>
<tr>
<th>DL ROQ</th>
<th>DL EL++</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid {a}$</td>
<td>$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid {a}$</td>
</tr>
<tr>
<td>$\neg C \mid \geq n R.C$</td>
<td>$\neg C \mid \geq n R.C$</td>
</tr>
<tr>
<td>$C \sqsubseteq D$</td>
<td>$r \sqsubseteq s$, $r_1 \circ \ldots \circ r_n \sqsubseteq s$</td>
</tr>
<tr>
<td>$a : C$</td>
<td>$(a, b) : r$</td>
</tr>
</tbody>
</table>

2NEXPTIME-complete  PTIME-complete
Faithful Approximate Reasoning (II)

- **TBox reasoning**
  - Directly represent non-OWL2-EL concepts with fresh named concepts
    - E.g., $\forall r.C \text{ subClassOf } D \Rightarrow A_{\forall r.C} \text{ subClassOf } D$
  - Maintain semantic relations for these named concepts
    - *complementary relations*
    - *cardinality relations*

- **ABox reasoning**
  - Internalise ABox into TBox
    - E.g., $C(a) \Rightarrow \{a\} \text{ subClassOf } C$
    - $r(a,b) \Rightarrow \{a\} \text{ subClassOf } \exists r.\{b\}, ...$

---

**Example: Physical device configuration**

- Approximation
  - Directly represent non-OWL2-EL concept with fresh names
  - Maintain complementary relations in *complementary table* (CT)
  - Maintain cardinality relations in *cardinality table* (QT)
  - Internalise ABox into TBox
  - ...

- **Supervisor2**
  - $\neg$ **Supervisor720**
  - **nSupervisor720**
  - **Supervisor720**

- **Cisco7603Config**
  - $\text{hasSlot} <= 3$ **Slot**
  - **X**
  - $\text{hasSlot} >= 4$ **Slot**
  - **nX**
  - **nX**

---
Completion Rules

- Completion Rules
  - Handling complement
    - E.g. A subClassOf B \implies \neg B subClassOf \neg A
  - Handling cardinality
    - E.g. A subClassOf \geq 3 \Rightarrow B \Rightarrow A subClassOf \geq 2 \Rightarrow B

Evaluations for the Oxford Benchmarks

Experimental evaluations
Physical Device Configuration

Demo Plan

- Physical Device Configuration Ontologies
  - 1. classification of Pellet: time consuming
  - 2. classification of REL: efficient
  - 3. inconsistency justification of REL: able to detect multiple sources of inconsistency
  - 4. inconsistency justification of Pellet: may yield incorrect results
Local Close World Reasoning

• Both open and close world reasoning are needed
• Individuals classes can be closed
  – Cat EquivalentClass: {Henry, Garfield, James}
• Two approaches
  – **DB Box** (or Dbox) [Seylan 2009]: if C in Dbox, C(a) not in Abox, then C(a) is false
    • O1: domain(teach)=Teacher, teach(Tom, CS1001)
    • if Teacher(Tom) is not in Abox, then O1 is inconsistent
  – **NAF Box** (or Nbox) [Ren et al. 2010b]: If C in Nbox, C(a) is not entailed by O, then C(a) is false
    • PhD subClassOf Student, PhD(John)
    • we should put Student in Nbox rather than Dbox

NAF / NOT in Queries

• Full Negation (NOT) and Negation as Failure (NAF)
  – A(a), A(c), A(e), B(a), B(c), B(d)
  – Q1:- A(x), NOT (B) returns an empty set
    • We don’t know whether e is an instance of B
  – Q1:- A(x), NAF (B) returns {b}
• Example:
    NAF { ?x ro:isAuthoredBy ?y .}}
Requirement Engineering

- Requirement Engineering contains many constraints
- Consistency, completeness, and quality
  - These are mapped to various reasoning services
  - **Consistency check (6 rules)**
    Eg. Excluding requirements *must not* be included in one set (Ontology Consistency Checking)
  - **Completeness check (46 rules)**
    Eg. Every FR *must* be either mandatory or optional (Query Answering with NAF)
  - **Quality check**
    Eg. A Goal *should* not be connected to more than 10% of requirements that are a negative contribution to this Goal (Query Answering)
**Ontology Definition Metamodel (ODM)**

- **Object Management Group standard for model driven ontology development**
- **Family of metamodels & profiles enabling model interchange & ontology development in UML 2**
- **Includes**
  - 5 platform independent metamodels, 4 normative
  - UML Profiles for RDF/S, OWL, & Topic Maps
  - Informative Mappings

**Motivation**

- **To provide a standard graphical notation to enhance communication of OWL to others**
- **To enable ontology-based information models to be integral parts of an information-centric system architecture that:**
  - Incorporates coherent and integrated sets of vocabularies, ontologies, and “gold standard” data models, developed & maintained independently from other aspects of a system
  - Increases platform independence as well as interoperability across services, processes, and other applications
  - Achieves limited breakage and rework as applications and services evolve, reducing maintenance costs
  - Improves software, process, and service quality (through shared information services, vocabularies, and other artifacts that are logically consistent – internally and with one another)
  - Improves opportunity for new capabilities & increasing automation in search, complex event and other transaction processing, transformation services, adaptive & predictive capabilities, etc.
The UML Profiles for RDF & OWL

- Intended to be highly intuitive for UML users
- Reuse UML constructs when they have the same semantics as OWL
- Define customized stereotypes of existing UML constructs to make them consistent with RDF and OWL semantics
- Use standard UML 2 notation
- When suitable UML constructs do not already exist, define additional combinations of stereotyped UML constructs to provide usable forms of notation for RDF and OWL semantics
- Utilize a model library to refer to defined sets of foundation elements (such as standard data types and property values)

Key Features of the RDF Profile

- `rdfs:Resource` is modeled as `UML::InstanceSpecification`
- `rdf:Property` is modeled by a combination of `UML::Property`, `UML::Association`, and `UML::AssociationClass`
Notation for OWL classes, using stereotyped UML::Class, and object properties, using stereotyped UML::AssociationClass is familiar to UML modelers.

Faithful notation for restrictions requires distinguishing necessary from necessary & sufficient membership, which is less intuitive to UMLers.

Latest thinking in the ODM Revision Task Force (RTF) for property notation includes the use of surrogates — to allow us to depict AssociationClasses in a “standalone” mode, without dragging unnecessary detail onto every diagram.

**Surrogate Property Notation**

Surrogates
- must have a base property defined via a traditional association or association class
- provide a flexible alternative for reuse in property hierarchies, complex restrictions, and property chain diagrams
OWL 2 Disjoint Union

- UML inherently supports generalization sets that are complete or incomplete, overlapping or disjoint.
- Shortcuts, such as collapsing a named class with the anonymous unionClass, when equivalence is intended, are under consideration.

Next Steps

- RTF is eliminating usability issues with the OWL 1 profile, expanding test cases, ensuring OWL 2 compatibility.
- Support for OWL 2 is in work:
  - Revision to the OWL metamodel may include two related metamodels, one that continues to support the RDF mapping approach, extended to cover all of OWL 2.
  - The other a OWL 2 standalone view, but connected via a thin veneer (i.e., common definitions for IRIs, documents, local names, namespaces, etc.).
  - Continued evolution of the UML profiles for RDF & OWL to improve usability, provide complete OWL 2 coverage.
- Publication of the ODM 1.1 revision later this year.
Near Term Roadmap

OWL API

Ontology Browser
- protégé
- OWLSight

Reasoner
- Pellet
- Racer
- FaCT++
- Hermit

Syntax
- RDF/XML
- Turtle
- OBO
- OWL/XML
- Manchester
- OWL 2

Profile validation
- OWL 2 DL
- EL
- QL
- RL

OWL Functional
Limitation of OWL API

• Not include alternative storage mechanisms (such as relational databases, triple stores)
• Not support SPARQL
• Not support Explanation
• Not an industrial standard

OWLlink : The Big Picture

• Implementation-neutral and extensible communication interface of OWL 2 components
• Status: W3C member submission (July 2010)
OWLlink

- Core Structure Specification
  - Fully Aligned with W3C Recommendation OWL2
  - Management: Identification, configuration, status, etc.
  - Telling of OWL 2 axioms
- Bindings
  - HTTP/XML (encapsulated OWL 2 XML serialization)
  - HTTP/Functional (encapsulated OWL 2 Funct. Syntax)
- Extension mechanism
  - Retraction-Extension
  - Told Data Access-Extension

Recent OMG Activities in Semantics

- Common Terminology Services (CTS2) – effort bridges HL7 & OMG work in defining services to support terminology access & transformations
  - Builds services on top of knowledge bases, registries & repositories containing various code sets, value sets, etc. for transformation given that mappings / ontologies are available for mediation
  - Two initial submissions presented this week at OMG technical meeting in Minneapolis: ii4sm/visum point & Mayo Clinic/Apelon
  - Agreement to move forward with a combined revised submission
- MOF2RDF – Effort to create a mapping from MOF to RDF/S (with a little OWL) to allow transformation of any model expressed in any MOF-based metamodel to RDF, primarily for exposing those models as Linked Data
  - First presentation of an early initial submission was presented this week, more to follow
New RFP: APIs for Knowledge Bases (API4KBs)

• Currently, there are a number of APIs for accessing RDF/S & OWL data / KBs
  — Jena
  — Sesame/Sail
  — DIG
  — OWL API
  — OWLlink
• They provide varying degrees of language coverage, varying completeness, varying levels of robustness, error handling, explanation support
• No real standard, no common way of describing IRIs, documents, local names, namespaces, or additional query services
• Organizations building tools to bridge the UML & Semantic Web standards must use multiple, often competing APIs with conflicting jar files

API4KBs Issued Today

• Calls for a single, standard set of APIs for accessing KBs, with a shared layer for accessing IRIs, documents, & other common infrastructure
• Users include parsers, ontology editors & tools, reasoners, & other applications needing common APIs & access services, including query support
• Support for
  — OWL 2 DL, profiles, OWL 2 Full, & RDFS is required
  — Common Logic, SBVR, other languages is optional
• Requests a Platform Independent Model (PIM) & 3 Platform Specific Models (PSMs) for Java, WSDL & REST, others optional
• Letters of Intent due 30 January 2011, initial submissions in March
Help wanted

• We are reaching out to folks in this community to participate –
  – OWL API developers & users
  – Jena developers
  – Others
  – Discussions initiated at SemTech, OWLED, RDF Next Steps
• OMG membership is preferred but not required
• Contact Evan Wallace (NIST) or Elisa Kendall (Sandpiper) if interested

Next Steps / Roadmap

• What other high-priority requirements does the community have for bridging software engineering & semantic technologies?
• Of those, what priorities should we set for standardization?
  – Additional APIs / Services
  – Additional standards, or extensions to ODM other than those underway
  – Mappings to support BPMN, SysML, IMM
  – Profiles to support specific extensions to ODM for metadata support (e.g., Dublin Core, SKOS, …)
  – Ontologies to describe software engineering disciplines/best practices/processes (e.g., CMMI, estimation, …)
  – Formal methodology / best practices for KR/Linked Data derived from software engineering discipline
Thank you!

Acknowledgement

• Yuting Zhao
• Edward Thomas
• Yuan Ren
• Nophadol Jekjantuk