Web Rule Languages to Carry Policies

Nima Kaviani
Laboratory for Ontological Research (LORe)
Simon Fraser University Surrey, Canada

nkaviani@sfu.ca
http://www.sfu.ca/~nkaviani

June 15th, 2007
Outline

- Policy-based Trust Management
  - Web services and Policies
  - Policy Languages
    - PeerTrust, KAoS, and Rei
  - The communication issues
- Interchange Frameworks
  - What is RIF?
  - What is R2ML
  - Using R2ML to exchange policies
  - The technical difficulties
  - The obtained results
- Conclusions
- Future Directions
Policy-Based Trust Management

- **Web Services and Policy-Based Trust Management**
  - Web services to facilitate collaboration
  - Trust Management to be used by web services
  - Policies to regulate Trust Management

- Dynamically regulate the behavior of the system without any need to manipulate the internal code

- Policies as *Guiding Plans* that restrict the behavior of the agents

- To protect the privacy of information by providing different levels of access

- **Policy Management Approaches and the Languages that support it**
  - Role Based (XACML, Cassandra)
  - Context Based (K AoS, Rei)
Policy Languages

- **Existing Languages for Policy-based Trust Management**

  - **PeerTrust**
  
  - **Rei**

  - **KAoS**

  1. Trust Negotiation Engine
  2. Text-based EBNF
  3. A DAML/OWL based policy language (KPO)
  4. Robust, Adaptable, Extensible
  5. Policy Specification and Management
  6. Enforcement and enforcement
  7. A GUI for policy manipulation
  8. Stanford’s JTP to perform static conflict resolution, intelligent lookup; and dynamic policy refinement
  
  - Low expandability
  - Easy to understand

- **Policy Languages**

  - Syntactically follows Description Logic (OWL-Lite)
  - Semantically follows Computational Logic (Prolog)
  - FOWL as the meta-interpreter in the backend
  - No policy enforcement
  - SpeechActs for message passing and dynamic exchange of rights
  - No policy disclosure possibility
Semantic Web Service Discovery & Composition

• The Current Proposals
  – Combination of OWL-S and Rei [Kagal, et. al, 2004]
  – Combination of WSMO and PeerTrust [Olmedilla et.al, 2004]

• Problems with the current approaches

**Rule Interchange Format (RIF)**
- RIF working group: defining a rule interlingua based on W3C standards
- *Develop a language to translate rules between rule languages and transform them between rule systems*
- Goal: enabling existing rule technologies to interoperate

**R2ML features**
- A general *rule interchange* language
- Admits to the RIF requirements
  - [http://rewerse.net/l1/](http://rewerse.net/l1/)
- Current version 0.4
Five General Rules

- Integrity Rules
- Derivation Rules
- Production Rules
- Reaction Rules
- Transformation Rules

\[ \text{if the user is a faculty} \]
\[ \text{then give him/her access to the meeting room} \]

\[ \text{if a visitor is part of a patients family} \]
\[ \text{then give him/her the allowance of visiting the patient} \]
Current Transformations to/from R2ML

- R2ML as a pivotal MetaModel
• Current Transformations to/from R2ML
  – R2ML as a pivotal MetaModel
  – URML: UML based rule language with graphical notations
Semantic Web Service Discovery

Solution

Enabling involved entities in Semantic Web Service discovery procedure to communicate

Policies can be defined in the form of R2ML rules
To get KAoS and Rei agents to communicate

- Providing transformations between KAoS and Rei [Grosof, et. al, 2003]
  - Both are Context-Based policy languages
  - Both syntactically follow Ontology Languages
  - No straightforward mapping between Rei and KAoS
  - KAoS is based on Description Logic
  - Rei follows Computational Logic (Logic Programs)

<table>
<thead>
<tr>
<th>OWL Constructor</th>
<th>DL Syntax</th>
<th>FOL Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C \subseteq D$</td>
<td>$D \leftarrow C$</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \subseteq P$</td>
<td>$\forall x, y, z (P(x,y) \land (P(y,z)) \rightarrow P(x,z)$</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$P \equiv Q^-$</td>
<td>$\forall x, y P(x,y) \leftrightarrow Q(y,x)$</td>
</tr>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>$C_1(x) \land \ldots \land C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>$C_1(x) \lor \ldots \lor C_2(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${a_1, \ldots, a_n}$</td>
<td>$x = a_1 \lor \ldots \lor x = a_n$</td>
</tr>
<tr>
<td>hasClass</td>
<td>$\exists P. C$</td>
<td>$\exists y (P(x,y) \land C(y)$</td>
</tr>
<tr>
<td>toClass</td>
<td>$\forall P. C$</td>
<td>$\forall y (P(x,y) \rightarrow C(y)$</td>
</tr>
</tbody>
</table>
# Mapping R2ML & Rei

<table>
<thead>
<tr>
<th>Rei</th>
<th>R2ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each Deontic Element</td>
<td><strong>A Derivation Rule</strong></td>
</tr>
<tr>
<td>Variable Definition</td>
<td><strong>ObjectClassificationAtoms</strong></td>
</tr>
<tr>
<td>OR</td>
<td><strong>qf.Disjunction</strong></td>
</tr>
<tr>
<td>AND</td>
<td>The <em>conclusion</em> in the rule is a conjunction of elements</td>
</tr>
<tr>
<td>NOT</td>
<td><strong>Atom is Negated</strong></td>
</tr>
<tr>
<td>SimpleConstraint</td>
<td><strong>ReferencePropertyAtoms</strong></td>
</tr>
<tr>
<td>SpeechActs</td>
<td><strong>ObjectDescriptionAtoms</strong></td>
</tr>
<tr>
<td>SubElements</td>
<td><strong>Object- or Data-Slots</strong></td>
</tr>
</tbody>
</table>

- We should get the identical Rei Policy:

  *prohibit our system from using data that is accepted by the members of a group called UserActor*
Mapping R2ML & Rei – cont’d

```xml
<entity:Variable rdf:ID="x"/>
<entity:Variable rdf:ID="y"/>
<entity:Variable rdf:ID="negAuth"/>
<constraint:SimpleConstraint rdf:ID="constraint1">
  <constraint:subject rdf:resource="#x"/>
  <constraint:predicate rdf:resource="&rdfs;type"/>
  <constraint:object rdf:resource="#AcceptData"/>
</constraint:SimpleConstraint>
<constraint:SimpleConstraint rdf:ID="constraint2">
  <constraint:subject rdf:resource="#y"/>
  <constraint:predicate rdf:resource="&rdfs;type"/>
  <constraint:object rdf:resource="#UserActors"/>
</constraint:SimpleConstraint>
<constraint:And rdf:ID="conditions">
  <constraint:first rdf:resource="#constraint1"/>
  <constraint:second rdf:resource="#constraint2"/>
</constraint:And>
<constraint:SimpleConstraint rdf:ID="actor_value">
  <constraint:subject rdf:resource="#y"/>
  <constraint:predicate rdf:resource="#performedBy"/>
  <constraint:object rdf:resource="#x"/>
</constraint:SimpleConstraint>
<constraint:SimpleConstraint rdf:ID="actio_value">
  <constraint:subject rdf:resource="#x"/>
  <constraint:predicate rdf:resource="controls"/>
  <constraint:object rdf:resource="#Plcy_Action"/>
</constraint:SimpleConstraint>
<deontic:Prohibition rdf:ID="AcpDataP">
  <deontic:actor rdf:resource="#actor_value"/>
  <deontic:action rdf:resource="#actio_value"/>
  <deontic:constraint rdf:resource="#conditions"/>
</deontic:Prohibition>
```

```xml
<r2ml:DerivationRule>
  <r2ml:conditions>
    <r2ml:ObjectClassificationAtom r2ml:classID="#AcceptData">
      <r2ml:ObjectVariable r2ml:name="x"/>
    </r2ml:ObjectClassificationAtom>
    <r2ml:ObjectClassificationAtom r2ml:classID="#UserActor">
      <r2ml:ObjectVariable r2ml:name="y"/>
    </r2ml:ObjectClassificationAtom>
  </r2ml:conditions>
  <r2ml:conclusion>
    <r2ml:ObjectDescriptionAtom r2ml:classID="Prohibition">
      <r2ml:subject>
        <r2ml:ObjectVariable r2ml:name="AcpDataP"/>
      </r2ml:subject>
      <r2ml:ObjectSlot r2ml:referencePropertyID="controls">
        <r2ml:ObjectVariable r2ml:name="x"/>
      </r2ml:ObjectSlot>
      <r2ml:ObjectSlot r2ml:referencePropertyID="performedBy">
        <r2ml:ObjectVariable r2ml:name="y"/>
      </r2ml:ObjectSlot>
    </r2ml:ObjectDescriptionAtom>
  </r2ml:conclusion>
</r2ml:DerivationRule>
```
Mapping KAoS & R2ML

• The KAoS Policy:

*prohibit* our system from using data that is
accepted by the members of a group called
UserActor
KAoS and Rei Meta-Models

**Rei Action to R2ML ObjectDescriptionAtom**
KAoS and Rei Meta-Models

Rei SimpleConstraint to R2ML ObjectDescriptionAtom
KAoS and Rei Meta-Models

KAoS Policy Rule to R2ML ObjectDescriptionAtom
<r2ml:DerivationRule>
    <r2ml:conditions>
        <r2ml:ObjectClassificationAtom
            r2ml:classID="#AcceptData">
            <r2ml:ObjectVariable r2ml:name="x"/>
        </r2ml:ObjectClassificationAtom>
        <r2ml:ObjectClassificationAtom
            r2ml:classID="#UserActor">
            <r2ml:ObjectVariable r2ml:name="y"/>
        </r2ml:ObjectClassificationAtom>
    </r2ml:conditions>
    <r2ml:conclusion>
        <r2ml:ObjectDescriptionAtom
            r2ml:classID="Prohibition">
            <r2ml:subject>
                <r2ml:ObjectVariable r2ml:name="AcpDataP"/>
            </r2ml:subject>
            <r2ml:ObjectSlot
                r2ml:referencePropertyID="controls">
                <r2ml:ObjectVariable r2ml:name="x"
                    r2ml:classID="#Plcy_Action"/>
            </r2ml:ObjectSlot>
            <r2ml:ObjectSlot
                r2ml:referencePropertyID="performedBy">
                <r2ml:ObjectVariable r2ml:name="y"/>
            </r2ml:ObjectSlot>
        </r2ml:ObjectDescriptionAtom>
    </r2ml:conclusion>
</r2ml:DerivationRule>
Evaluation of the information loss

- Reasoning on the obtained policies
  - The reasoner for Rei is not supported any more
  - No release for KAoS reasoner

- Derivation Rules or Integrity Rules

- The Difference in the underlying Logic
  - KAoS has both universal and existential quantifiers
  - Rei only has universal quantifiers

- Universal and Existential Quantifiers

- Cardinality Support for the Rules

- Language specific concepts
  - SpeechActs in Rei …. No equivalent concept in KAoS

Is it still effective when we perform the transformations?
Conclusions

- **Benefits**
  - Language Independence Policy Design
  - Architecture independent
  - Easier surfing of the web for broker agents

- **Known Issues**
  - Information loss during exchange
    - How it may affect the trust
  - Derived R2ML transformations from different languages do not exactly match
    - An internal exchange between R2ML rules might be required
Future Direction

- Towards Combining Model Driven Approaches and Policy Languages
  - Policy Modeling Language
  - Connecting various policy languages through their models
  - XACML as a widely recognized policy language

- Combining Service Oriented Architecture (SOA) with Policy Modeling

- Semantic Web and its ability to introduce context based concepts that facilitate the definition of TRUST.
Questions?

Thank you