HOW TO DO WITH OWL WHAT PEOPLE SAY YOU CAN’T

Jeffrey M. Bradshaw, Andrzej Uszok, Larry Bunch, Maggie Breedy, Matt Johnson, Hyuckchul Jung, James Lott

{jbradshaw, auszok, lbunch, mbreedy, mjohnson, hjung, jlott}@ihmc.us
TRENDS

- Increasingly demanding requirements of new applications
  - Expressivity
  - Extensibility
  - Dynamicity
  - Usability
  - Performance
- Limitations of special-purpose languages
  - Need for unified approach across application domains
  - Need for system-wide representation and reasoning (e.g., QoS)
- Significant progress in semantically-rich policy representations
  - Details of implementations have not always been well-documented or widely-available
- Explosion of interest in Web research community
- Hopes for wider adoption in policy research community
NEW FRONTIERS REQUIRING RICHER POLICY SEMANTICS

- Risk-adaptive access control
- Adjustable autonomy
- Policy learning
- “Soft” policy enforcement (e.g., dynamic QoS tradeoffs)
- Policy refinement
- Reasoning about privacy and auditing issues
OBJECTIVES

• Explore some of the advantages of OWL for policy representation and reasoning
• Dispel some of the myths and misconceptions
• Spur discussion and seek opportunities for collaboration
• Not a tutorial on OWL or KAoS
  • Will use KAoS examples as illustrations
  • See http://ontology.ihmc.us/ for examples and more information, or contact me at jbradshaw@ihmc.us
WHAT IS OWL?

- OWL stands for Web Ontology Language
- OWL is built on top of RDF and written in XML
- OWL was designed to be interpreted by computers, not people
- OWL has three sublanguages: OWL-Full, OWL-DL, and OWL-Lite
- OWL is a Web standard
- The use of OWL is not restricted to Web applications
# Semantic Web Representations for Policy Specification: why?

<table>
<thead>
<tr>
<th>Semantic web representations for policy specification</th>
<th>Traditional approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expressiveness</strong></td>
<td></td>
</tr>
<tr>
<td>Capable of representing concepts and behavior of any complex environment</td>
<td>Capable of controlling specific sorts of behavior within object-oriented systems</td>
</tr>
<tr>
<td>Multiple levels of abstraction</td>
<td>Low level of abstraction: object level</td>
</tr>
<tr>
<td>Easy to extend policy ontology at runtime with new concepts</td>
<td>Extensibility supported by object-oriented inheritance at compile-time</td>
</tr>
<tr>
<td><strong>Analyzeability</strong></td>
<td></td>
</tr>
<tr>
<td>Ontology representation simplifies and directly supports policy reasoning, conflict detection and harmonization</td>
<td>Conflict detection requires transforming policy specification into an event calculus representation</td>
</tr>
<tr>
<td>Simplified access to policy information by querying the ontology</td>
<td>Access to policy objects by API</td>
</tr>
<tr>
<td><strong>Ease-of-use</strong></td>
<td></td>
</tr>
<tr>
<td>Need of specialized GUIs to assist unskilled users with policy specification and interpretation</td>
<td>Language specifically designed for simple policy specification and direct readability</td>
</tr>
<tr>
<td><strong>Enforceability</strong></td>
<td></td>
</tr>
<tr>
<td>High-level specification requires skilled programmers or sophisticated policy automation mechanisms for enforcement</td>
<td>Detailed specifications can be directly mapped into policy enforcement mechanisms</td>
</tr>
<tr>
<td>Policy sharing among heterogeneous systems requires an agreement on a common ontology</td>
<td>Policy sharing among heterogeneous systems requires agreement on interfaces</td>
</tr>
</tbody>
</table>
POLICY REPRESENTATION

- Myth: “Policies of type X cannot be represented using OWL”

- Realities
  - OWL has proven to be a remarkably flexible and expressive representation for a wide variety of policies
  - Examples include requirements for complex policy domain scoping, RBAC, policy attachments to workflow actions, data transformations in publish-subscribe contexts, policy disclosure constraints, state, history, and dynamic context
  - Hybrid rule/ontology approaches can be avoided
  - In KAoS, only two extensions to OWL semantics have been required to date: role-value maps and XML data schemas
  - New policy representation challenges are welcome!
POLICY REPRESENTATION

- Myth: “OWL does not allow policies to be defined over attributes of classes including users, resources, and the context”

- Realities
  - KAoS allows policy restrictions for values of any attribute of existing classes representing users, resources or dynamic context
  - It also allows relating any property in the class to another property in this class or any other class through role-value-maps
POLICY REPRESENTATION

- Myth: “OWL-based obligation policies trigger decisions exclusively on access requests rather than external events, i.e., changes in context”

- Realities
  - In KAoS, the occurrence of any monitored event, change in context, or change in state can trigger an obligation policy
POLICY REPRESENTATION

- Myth: “Building OWL policies is a complicated process”

- Realities
  - Good representations should keep easy things simple and make hard things possible
  - Existing core policy and application domain ontologies can be straightforwardly used and extended
  - Developers can now rely on a variety of graphical tools instead of low-level XML syntax editors (e.g., Cmap Ontology Editor (COE), KPAT, Protégé)
  - End users can build policies through graphical editors that map natural language statements to ontology concepts
    - Interactive speech-based interfaces have even been created
  - No need for Internet connection
POLICY REASONING

- Myth: “OWL reasoning is limited and does not scale”

- Realities
  - Description logics are a decidable subset of predicate logic for which efficient reasoning support is possible
  - OWL-DL is mapped on a description logic, and a variety of reasoners are available (e.g., JTP, Pellet, FaCT++, Cerebra, and RACER)
  - Algorithms for policy conflict resolution and static policy analysis have been implemented for OWL-based policy
  - A form of incremental (non-monotonic) reasoning is supported by Pellet
  - OWL-based policy management systems can straightforwardly incorporate specialized reasoners if required (e.g., KSPARC)
  - KAoS “compiles” OWL policies for efficient monitoring and enforcement reasoning
  - OWL-DL representation and reasoning support is available in Oracle, and support for other DBs is forthcoming
**Discussion**

- What barriers currently discourage policy researchers from using OWL?
- What can be done to help encourage the wider evaluation and adoption of semantically-rich policy representations?