Rei and Rules

Tim Finin, UMBC
Lalana Kagal, MIT

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Outline

- Motivation
- Rei : a policy specification language
- Rei 4.0
- Conclusions

Motivation

- **Objective:** We want to influence, constrain and control the behavior of autonomous programs, services and agents in open, heterogeneous, dynamic environments
  - E.g.: web services, pervasive computing environments, collaboration tools, Grid services, multiagent systems, …
- **Problem:** Conventional identity/authentication approaches to access control & authorization lacking
- **Approach:** Agents reason about policies expressed in a declarative language in support of decision making, trust evaluation and enforcement.

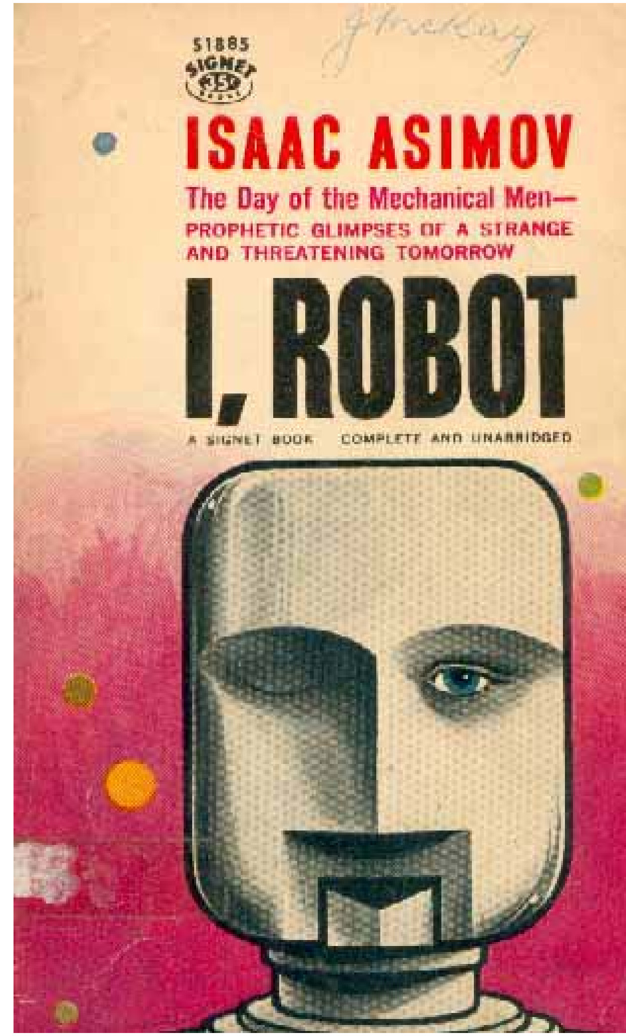
An Early Policy for Agents

1 A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

2 A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

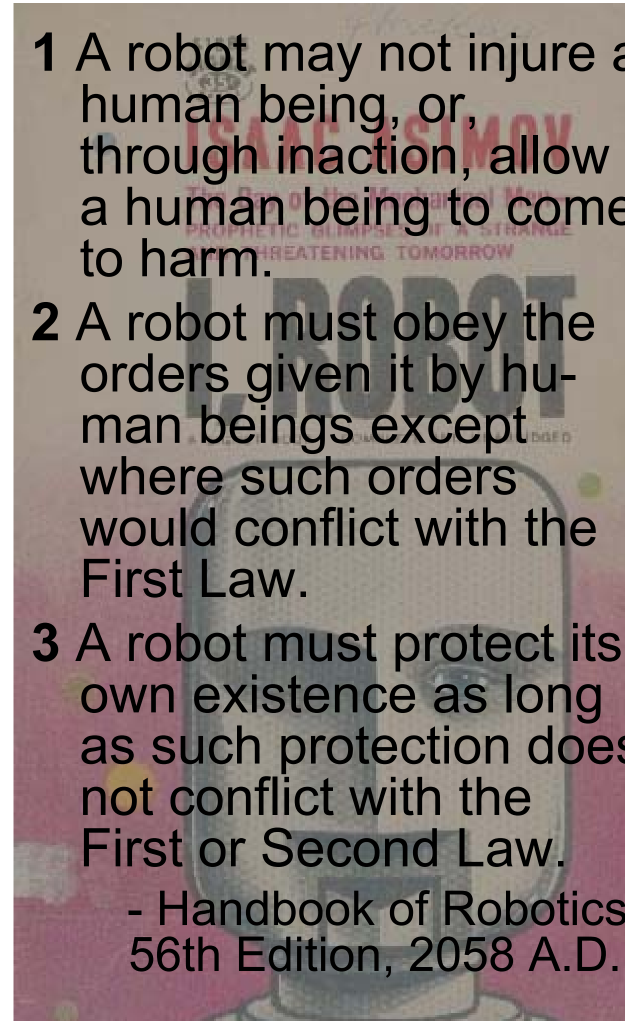
3 A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

- Handbook of Robotics, 56th Edition, 2058 A.D.



It's policies all the way down

- In Asimov's stories the robots didn't always follow the policy
  - Unlike traditional "hard coded" rules like DB access control & OS file permissions
  - Policies define "norms of behavior"
  - We use policies to govern the failure to adhere to other policies!
- So, it's natural to worry about …
  - How agents governed by multiple policies can resolve conflicts among them
  - How to deal with failure to follow policies – sanctions, reputation, trust, etc.
  - Whether policy engineering will be any easier than software engineering



1 A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

2 A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

3 A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

- Handbook of Robotics, 56th Edition, 2058 A.D.

Policies are the new black

- Machine understandable policies have been around forever; think of file permissions and DBMSs.
  - But, there are many new domains that want policies: DRM, content filtering, web services, Grid, P2P extensions, etc.
  - … and a desire for better policy languages
- Lots of work going on:
  - WS-*, SAML, XACML, EPAL, Ponder, KeyNote, etc.
- Policy languages grounded in OWL: KAoS & Rei
  - KAoS has a (pure) DL approach
  - Rei's approach uses DL + rules

JAPAN
WWW 2005

Policy Management for the Web

A WWW2005 Workshop

14th International World Wide Web Conference
Tuesday 10 May 2005, Chiba Japan

hppt://www.cs.umbc.edu/pm4w/

Home

Call for Papers

CFP flyer

Committee

Important dates

Submit paper

Register

Program

In order to realize the full potential of the World Wide Web as an open, dynamic, and distributed ``universe of network-accessible information'', it is important for web entities to behave appropriately. Policy management provides the openness, flexibility, and autonomy required to regulate this environment as entities can reason over their own policies and the policies of other entities to decide how to behave. Using policies also allows entities to specify expected behavior of entities they interact with. Entities can also adapt to increasingly complex requirements without the need for substantial changes to the structure or implementation through the use of policies.

Policy management includes policy specification, deployment, reasoning over policies, updating and maintaining policies, and enforcement. We propose that policy management is required for the web for (i) constraining different kinds of behavior including security, privacy, conversation, and collaboration, (ii) configuration management, (iii) describing business processes, and (iv) establishing trust and reputation.

Topics of Interest

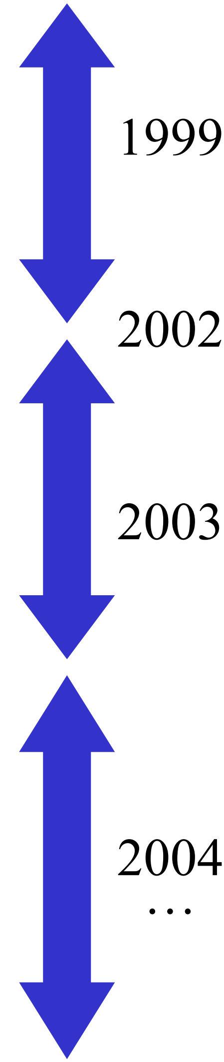
- Policy specification, implementation, and enforcement

Rei Policy Spec Language

- Rei is a product of Lalana Kagal's 2004 dissertation
- An OWL based declarative policy language
- Models deontic concepts of permissions, prohibitions, obligations and dispensations
- Uses meta policies for conflict resolution
- Uses speech acts for dynamic policy modification
- Used to model different kinds of policies
  - Security; privacy; team formation/ collaboration/maintenance; conversation constraints

Applications – past, present & future

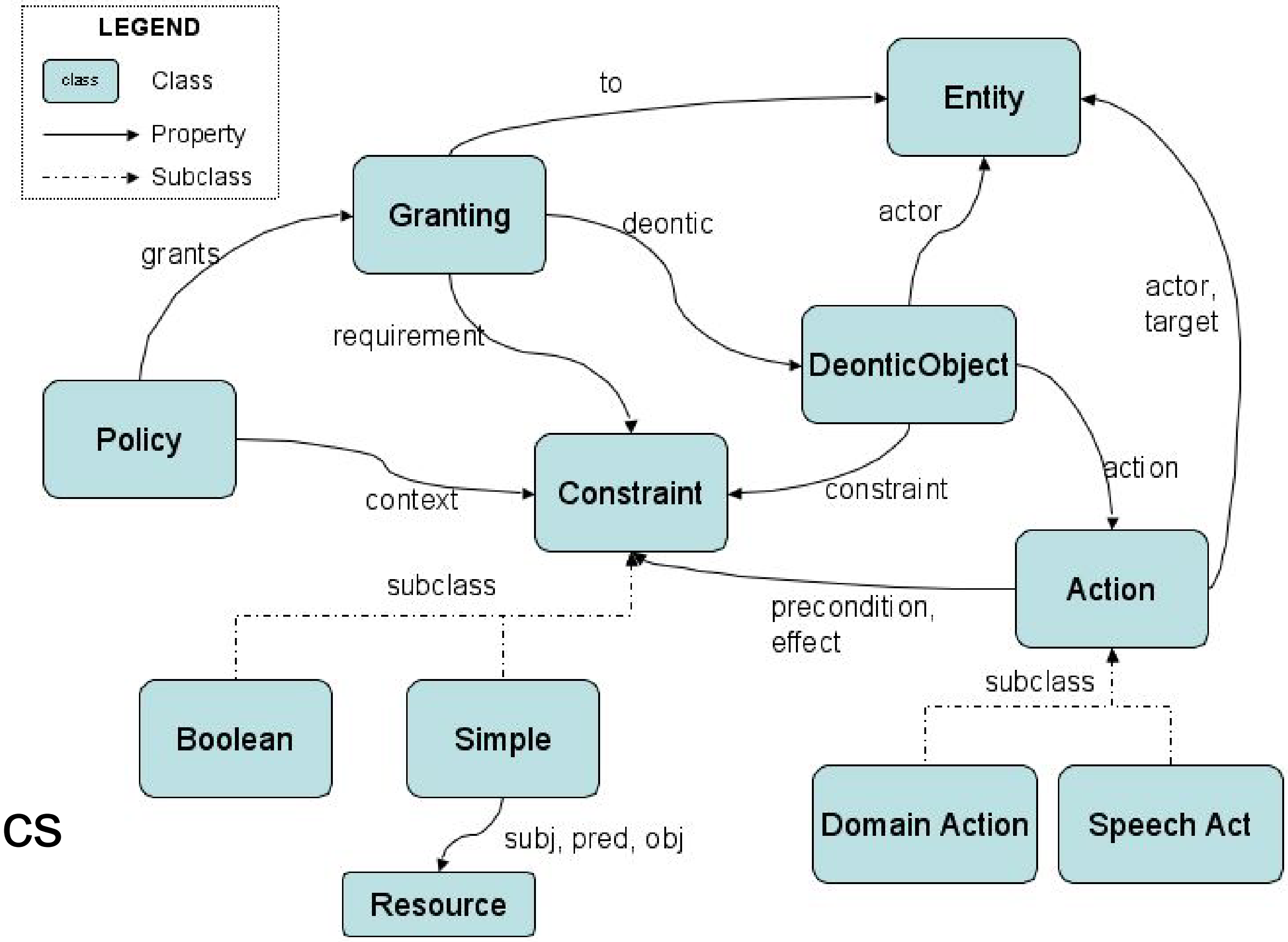
- Coordinating access in supply chain management system (EECOMS - IBM lead)
- Authorization policies in a pervasive computing environment (UMBC)
- Policies for team formation, collaboration, information flow in multi-agent systems (Genoa II (Topsail) - GITI lead)
- Security in semantic web services (UMBC, SRI, CMU)
- Privacy and trust on the Internet (UMBC)
- Enforcing domain policies on handhelds in pervasive computing environments (UMBC, NIST)
- Privacy in a pervasive computing environment (UMBC)
- Task Computing (Fujitsu)



Rei Specifications

Rei Ontologies

- Core specs
  - Policy
  - Granting
  - Deontic Object
    - …
  - Action
    - Speech Act
      - …
  - Meta Policy
  - Constraint
- Authoring aid specs
  - Analysis



Constraint

- Simple Constraints
  - Triple(Subject, Predicate, Object)
  - Example : Group of entities that are affiliated to the LAIT lab

```
<entity:Variable rdf:ID=”Var1”/>
<constraint:SimpleConstraint rdf:ID=”IsMemberOfLait">
  <constraint:subject rdf:resource="#Var1"/>
  <constraint:predicate rdf:resource="&univ;affiliation"/>
  <constraint:object rdf:resource="&univ;LAITLab"/>
</constraint:SimpleConstraint>
```

- Boolean Constraints : And, Or, and Not

Four Aspects to Meta Policy

- **Behavior**
  - ExplicitPermImplicitProh – *what's not permitted is forbidden.*
  - ImplicitPermExplicitProh – *what's not forbidden is permitted.*
  - ExplicitPermExplicitProh – *no default*
- **Priority**
  - Priority between rules in the same policy
  - Priority between policies
    - e.g., Department policy overrides University policy
- **Modality precedence**
  - e.g., Positive modality holds precedence over negative for CSDept policy
- **Meta policy default**
  - CheckModalityPrecFirst
  - CheckPriorityFirst

Modality Precedence

- Example : To state that negative modality holds for the CSDept and in case of conflict modality precedence should be checked before priorities

```
<policy:Policy rdf:ID=”CSDeptPolicy">
    <policy:context rdf:resource="#IsMemberOfCS"/>
    <policy:defaultModality
                rdf:resource="&metapolicy;NegativeModalityPrecedence"/>
      <policy:metaDefault
                rdf:resource="&metapolicy;CheckModalityPrecFirst"/>
</policy:Policy>
```

From Rules to DL and Back

- Rei 1.0 started out ~1999 with a rule-based approach implemented via a Prolog meta-interpreter
  - Subsequently translated to CommonRules XML format for interchange and interoperability
- Rei 2.0 used RDF to ground policies in sharable ontologies
- Rei 3.0 embraced a DL approach to take advantage of subsumption reasoning using F-OWL
  - Retained rule-like constraints for greater expressivity
  - Students permitted to use printers in labs with which their advisors are association
- Rei 4.0 may will revise its rule like aspects now that SWLR is available
  - Motivations: formalization, flexibility, simplicity, understandability, …

To Be Explored

- Simplify and reduce to essential form
- Develop a solid formal semantics
- Model/implement using Courteous Logic
- Compile Rei policies to SWRL or RuleML to obviate need for meta-interpreter
- Additional features
  - Support static conflict detection
  - Provide explanation facility, including explanations for "failed" expectations
  - Build on initial primitive Policy IDE
- Interoperation with or translation between {Rei, KAoS, …}

Summary

- Declarative policies are useful for constraining **autonomous** behavior in open, distributed systems
  - Important for security, privacy and trust
- These should be grounded in semantic web languages (OWL!) for semantic interoperability
- Rei and KAoS have provided a good base for exploring this approach
- SWRL and RuleML open interesting opportunities for new declarative, rule oriented policy languages
- Rei 4.0 will explore

For more information

http://rei.umbc.edu/

backup slides

Implementation Details

- XSB
  - Flora : F-logic over XSB
- F-OWL : is a reasoner for RDF, OWL
- Java wrapper

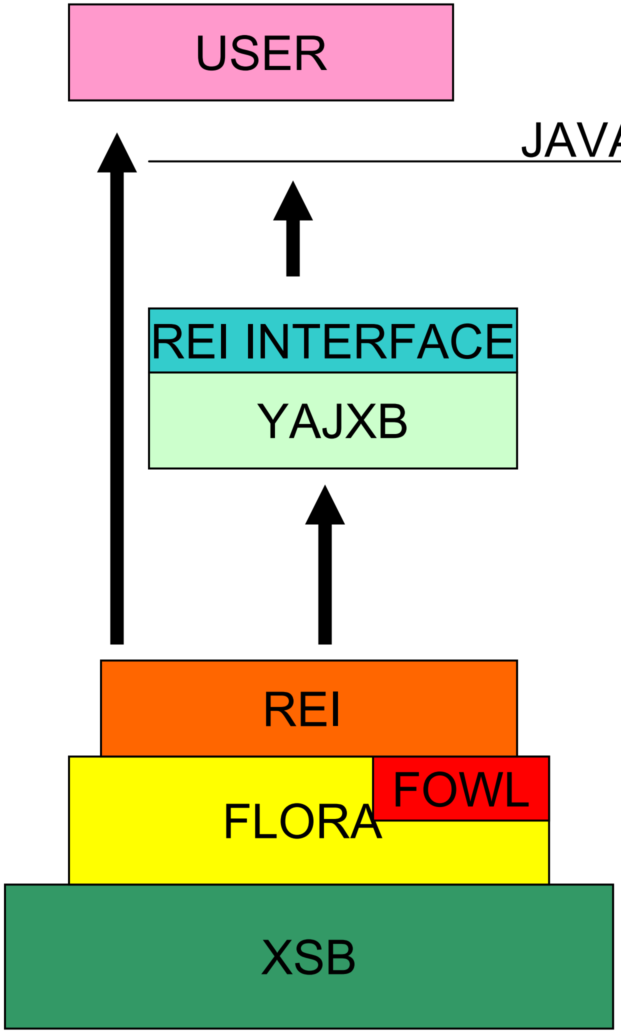


Image adapted from Mohinder Chopra

Priority

- Example : To specify that the Federal policy has higher priority that the State policy

```
<metapolicy:PolicyPriority rdf:ID="PriorityFederalState">
      <metapolicy:policyOfGreaterPriority rdf:resource="&gov;Federal"/>
      <metapolicy:policyOfLesserPriority rdf:resource="&gov;State"/>
  <metapolicy:PolicyPriority>
```

- Priorities for policies and rules must be acyclic (it is possible to check this but currently not implemented)
  - Rei does not allow
    - University policy overrides department policy
    - Department policy overrides lab policy
    - Lab policy overrides university policy

Analysis

- Use Cases (known as test cases in Software Engineering)
  - Define a set of use cases that must always be satisfied in order for the policies to be correct
  - E.g. The dean of the school must always have access to all the grad labs
- WhatIf
  - To check the effects of changes to the policy or ontology before actually committing them
  - E.g If I remove Perm_StudentPrinting from the GradStudentPolicy, will Bob still be able to print ?

Speech Acts

- Speech Acts
  - Delegation, Revocation, Request, Cancel
  - Properties : Sender, Receiver, Content (Deontic object/Action), Conditions
  - Used to dynamically modify existing policies
  - Speech acts are valid only if the entities that make them have the appropriate permissions

Policy

- Properties : Context, Grants, Default Policy, Priorities
  - A *Policy* is applicable if the *Context* is true
- Example

```
<policy:Policy rdf:ID=”CSDeptPolicy">
    <policy:context rdf:resource="#IsMemberOfCS"/>
    <policy:grants rdf:resource="#Perm_StudentPrinting"/>
    <policy:defaultBehavior
          rdf:resource="&metapolicy;ExplicitPermExplicitProh"/>
    <policy:defaultModality
          rdf:resource="&metapolicy;PositiveModalityPrecedence"/>
    <policy:metaDefault
          rdf:resource="&metapolicy;CheckModalityPrecFirst"/>
</policy:Policy>
```

Granting

- Links deontic rules to policies with additional constraints
- Allows for reuse of deontic objects with different constraints
- Encourages modularity
  - Deontic objects and constraints can be defined by technical staff
  - Policy administrator can drag and drop appropriate deontic objects and add constraints

Granting

- Example : Same permission used in Policy example with extra constraints

```
<policy:Granting rdf:ID="Granting_PhStudentLaserPrinting">
   <policy:to rdf:resource="#PersonVar"/>
   <policy:deontic rdf:resource="#Perm_StudentPrinting"/>
   <policy:requirement rdf:resource="#IsLaserPrinterAndPhStudent"/>
</policy:Granting>

<policy:Policy rdf:ID=”BioDeptPolicy">
     <policy:grants rdf:resource="# Granting_PhStudentLaserPrinting"/>
</policy:Policy>
```

Deontic Object

- Deontic objects
  - Permissions, Prohibitions, Obligations, Dispensations (waiver for obligations)
  - Common Properties : Actor, Action, Constraint {StartingConstraint, EndingConstraint}
  - StartingConstraint subproperty of Constraint

Action

- Two kinds of actions : Domain Actions and Speech Acts

- Domain Actions
  - Properties : Actor, Target, Effects, PreConditions
  - Action(Actor, Target, PreConditions, Effects)
  - *Action* can be performed on *Target* only when the *PreConditions* are true and oncce performed the *Effects* are true.

  - Example : Based on Rei

<action:*Action* rdf:ID=”EbiquityDeviceUsage">

<action:actor rdf:resource="#PersonVar"/>

<action:target rdf:resource="#ObjVar"/>

<action:location rdf:resource="&inst;EbiquityLab"/>

<action:precondition rdf:resource="#DeviceBelongsToEbiqLab"/>

<action:*Action*>

Action

- Example :

```
<owl:Class rdf:ID="CSPrinting">
    <rdfs:subClassOf rdf:resource=”&univ;Printing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
                  <owl:onProperty rdf:resource="&action;location"/>
                  <owl:allValuesFrom rdf:resource=”&inst;CSDept" />
      </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
```