Mark Cornwell (GITI)
James Just (GITI)
Lalana Kagal (UMBC)
Tim Finin (UMBC, GITI)
Mike Huhns (USC, GITI)

Policies for Autonomy in Open Distributed Systems

This work was partially funded by Defense Advanced Research Project Agency under contract N66001-03-C-8001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Project Agency or the U.S. Government.

UMBC
AN HONORS UNIVERSITY IN MARYLAND

GITI
Global InfoTek, Inc.

Are policies a panacea?

Are policies a panacea?

no

IMHO

There are
no panaceas

IMHO

Except,
maybe,
...

IMHO*

the semantic web

*Disclaimer: this is MHO when I'm in one of my expansive moods.

Summary

- **Declarative policies** are useful for constraining autonomous behavior in open, distributed systems
- This enables **more autonomy**
- The **Rei** policy language and associate tools have provided a good base
- **Semantic web** languages (e.g., OWL) used, grounding descriptions in sharable, semantically rich, machine understandable ontologies
- We're evaluating and exploring the utility of policies through prototype **applications**
- In one, **Topsail**, policies guide agents to help form, operate and maintain teams of people.

Policies for Autonomy

- Policies are rules of optimal behavior
  - Optimal? Policies are normative and describe what *should* be done in an ideal world.
- Policies provide high-level control of entities in the environment
  - Entities? These can be programs, services, agents, devices and people
- Using policies reduces the need to modify code in order to change systems' behavior
  - So? We assume modifying policies will be easier than modifying Java.

Our Approach

- Declarative policies guide the behavior of entities in open, distributed environments
  - Positive and negative authorizations & obligations
  - Focused on domain actions
  - Policies are based on attributes of the action (and its actor and target) and the general context – not just on their *identity* of the actor
- Policies are applied at different levels
  - From OS and networking to applications

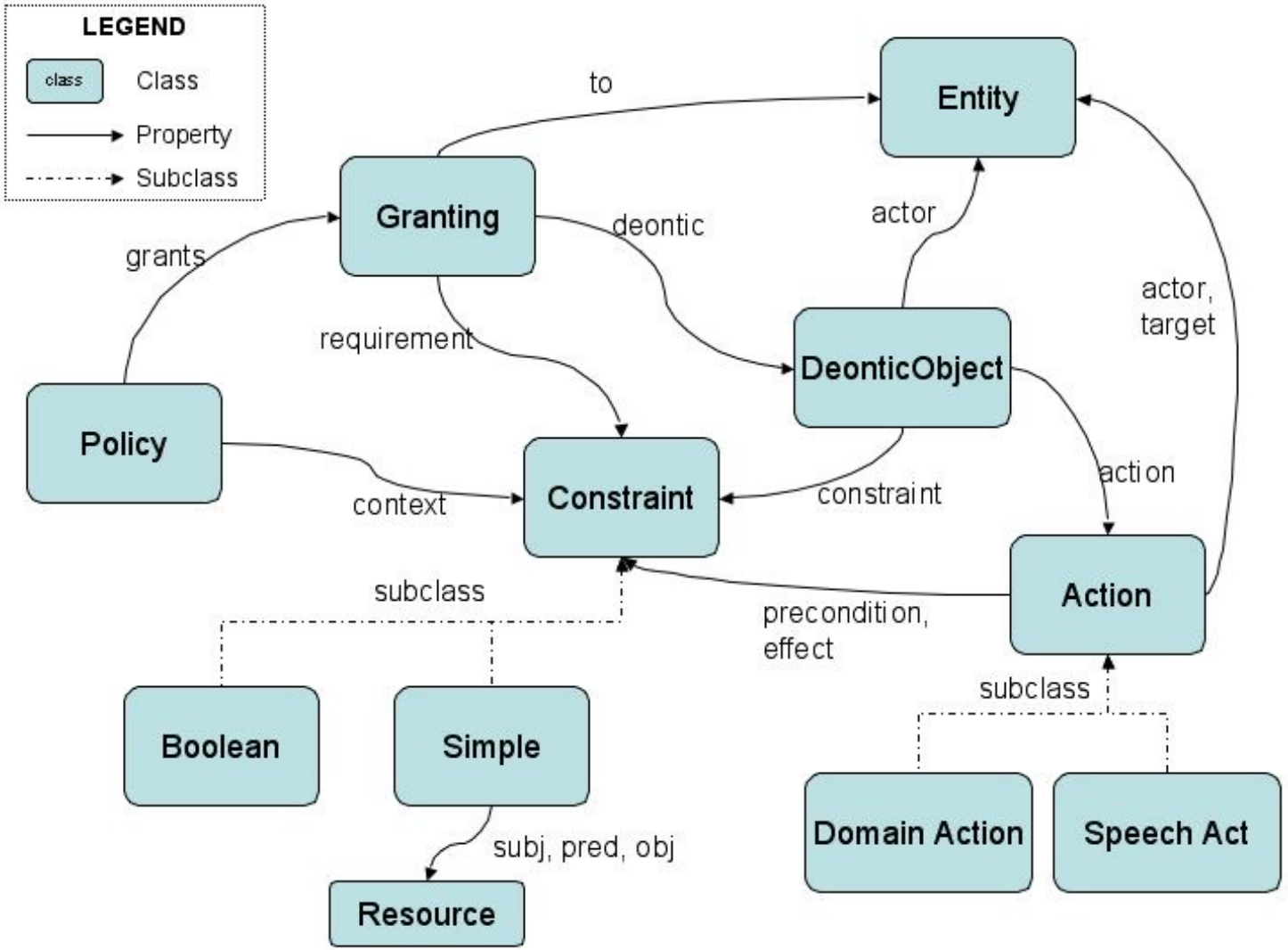
Our Approach

- Developed several versions of Rei, a policy specification language, encoded in (1) Prolog, (2) RDFS, (3) OWL
- Used to model different kinds of policies
  - Authorization for services
  - Privacy in pervasive computing and the web
  - Conversations between agents
  - Team formation, collaboration and maintenance
- Policies and attributes described in the web ontology language OWL

Rei Policy Language

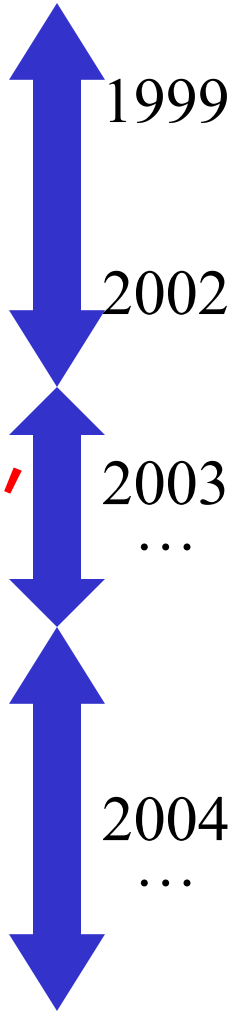
- Rei is a declarative policy language for describing policies over actions
- Reasons over domain dependent information
- Currently represented in OWL + logical variables
- Based on deontic concepts
  - Permission, Prohibition, Obligation, Dispensation
- Models speech acts
  - Delegation, Revocation, Request, Cancel
- Meta policies
  - Priority, modality preference
- Policy tools
  - Reasoner, IDE for Rei policies (Eclipse), …

Rei Specifications (partial)



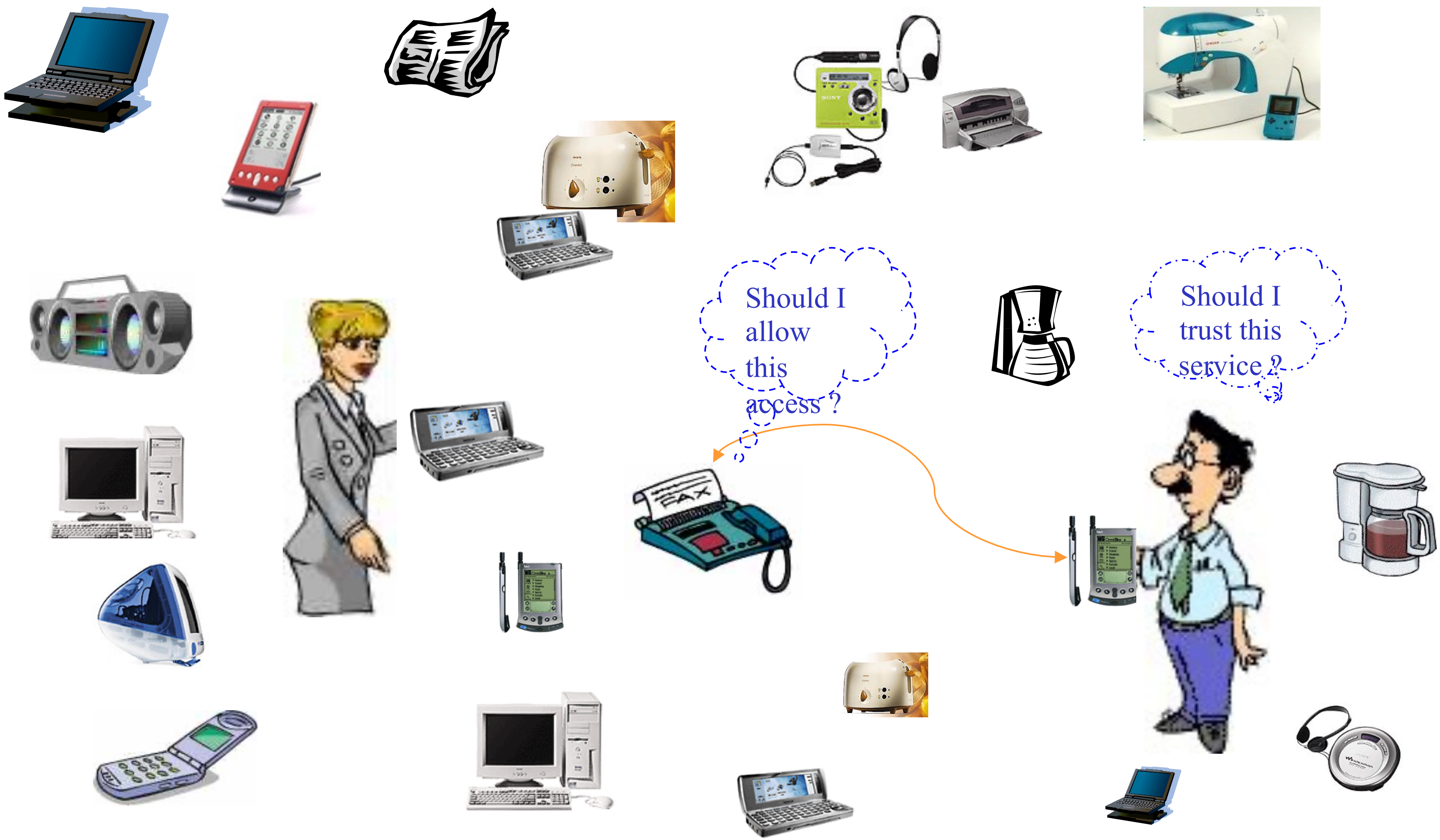
Applications – past, present & future

- Coordinating access in *supply chain management system*
- Authorization policies in a *pervasive computing environment*
- Policies for team formation, collaboration, information flow in *multi-agent systems*
- Security in *semantic web services*
- Privacy and trust on the *Internet*
- Privacy in a *pervasive computing environment*



pervasive computing

Authorization policies in pervasive computing environments



Problems

- Highly distributed, open and dynamic
- Users and resources are neither pre-determined nor permanent
- No central repository or control

Authorization policies

- Every entity describes its own authorization policy that defines security requirements for its access
  - E.g.. The grad fax machine states that only UMBC graduate students can send faxes
- No central policy or control
- Authorization is formulated as verifying that the credentials of the requesting user meet the requirements of the requested object

Meeting Room Example

Client's Meeting Room

What are your beliefs about John

John wants to use the printer in the meeting room

John, Consultant

(John Requests to use printer)
Signed with private key + credential from John's office saying he is a consultant

Printer's Security Policy

Only attendees of the meeting can use the printer
Accept the organizer's beliefs about attendees

Meeting Room Example

Client's Meeting Room

I believe John is an attendee

OK

John wants to use the printer in the meeting room

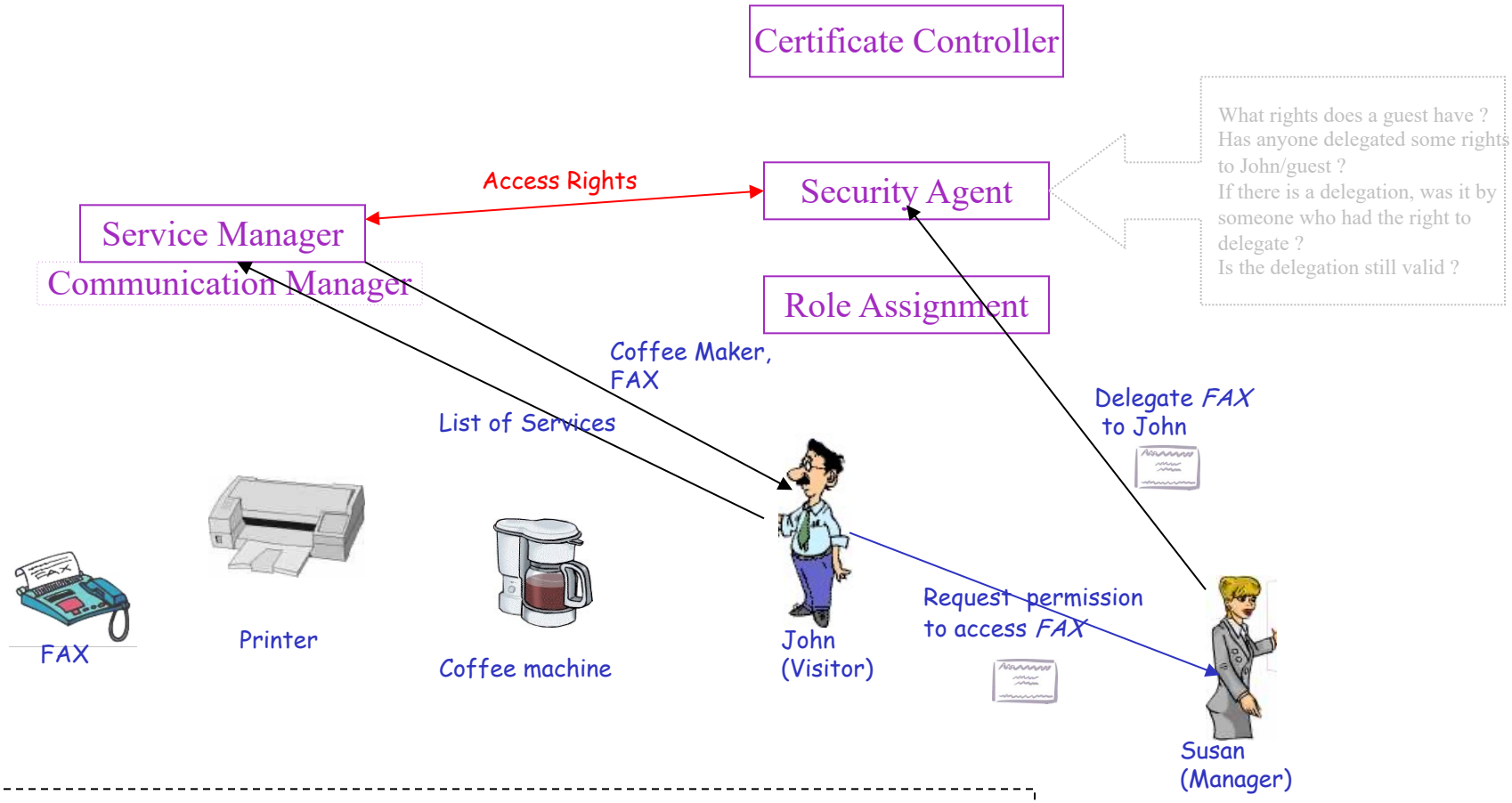
John, Consultant

(John Requests to use printer)
Signed with private key + credential from John's office saying he is a consultant

Printer's Security Policy

Only attendees of the meeting can use the printer
Accept the organizer's beliefs about attendees

Delegation Example



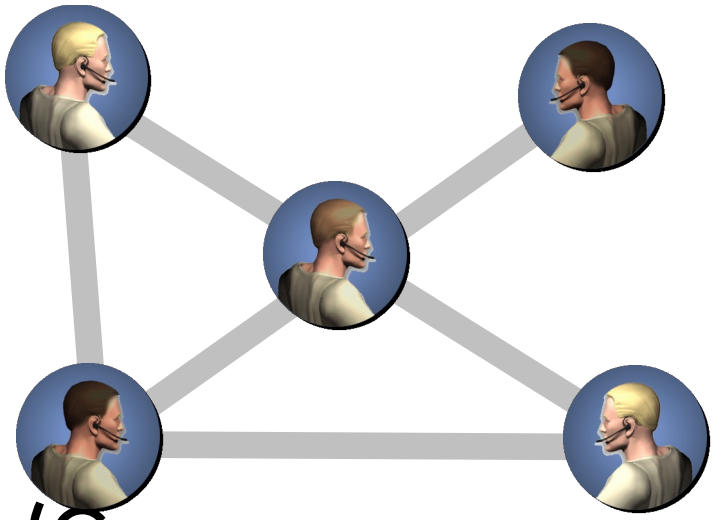
has(Person, right(delegate(right(use-fax,[])), [role(Person, abc, manager)]))

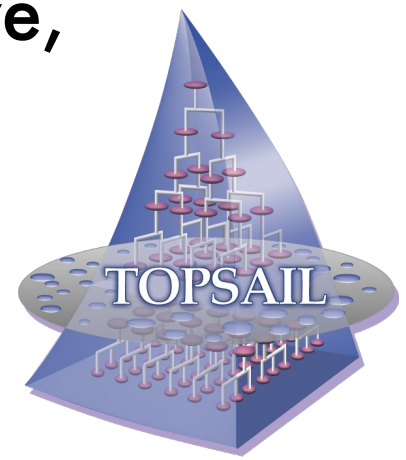
Simplified Rule for delegation :
has(Person, Right) :- delegate(From, Person, Right), has(From, right(delegate(Right))).

human collaboration

Enhancing collaboration in human teams

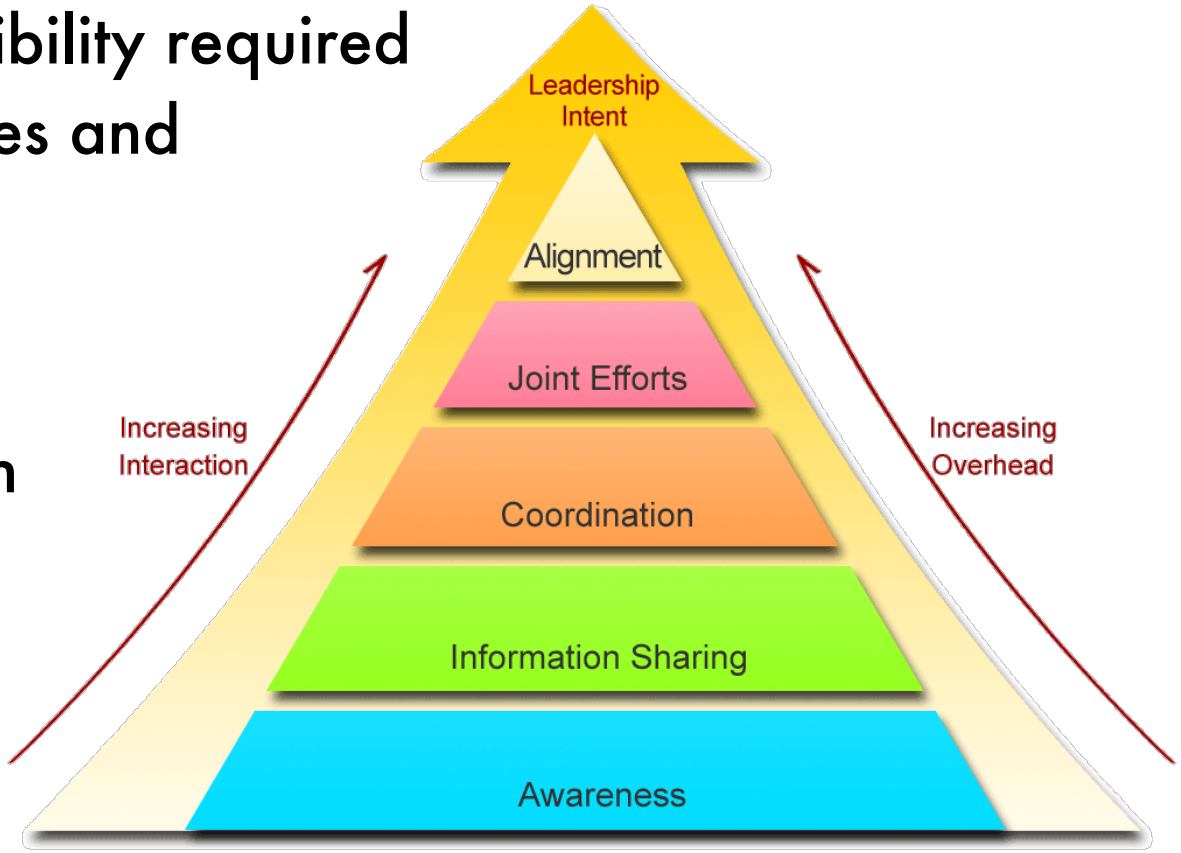
- **Objective:** facilitate collaboration in inter-agency teams
- **Scenario:** collaboration on an inter-agency team formed in face of a crisis
- **Approach:** augment conventional collaboration tools (Groove, email, workflow) with agents to assist in team formation, team maintenance, information flow, workflow, …
- **Lead:** Global Infotek Inc. for DoD



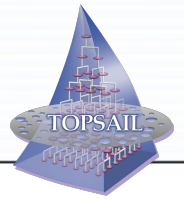


Motivation and Problems

- **Motivation:** Technology to increase collaboration effectiveness
  - Large amount of flexibility required
  - Heterogeneous entities and networks
  - Each agency has its own policy
  - Teams have their own policy and priority
  - Possibility of policy conflict is high

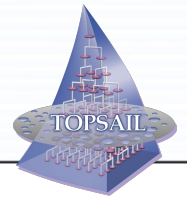


Research components



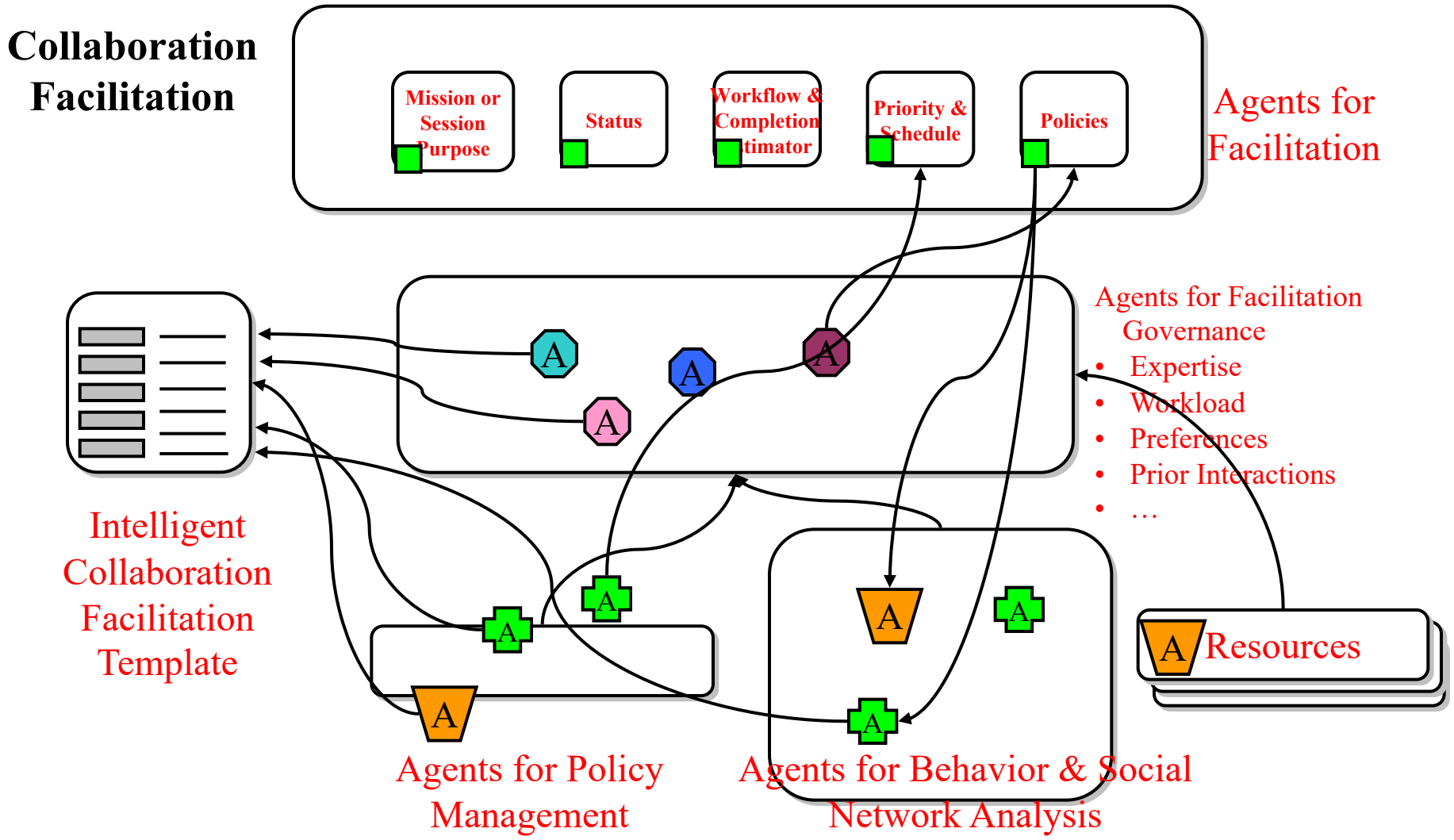
- Enhancing conventional collaboration tools with **software agents**
- Exploring the use of declarative **policies** to constrain and guide system components
- Using **social network analysis** to model and understand human team structure and roles
- Acquiring **user and team models** automatically by instrumenting coordination tools (e.g., groove, email) and employing 'smart badges'

Example: Team Support

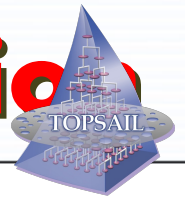


- A team is characterized by
  - Crisis type
    - Defines the skill set required for the team members, the number of team members required, etc.
  - Length of activity
  - Priority
- Actions involved
  - Team Formation
  - Collaboration support
  - Information flow monitoring and control
  - Workflow monitoring and management
  - …

Agent enhanced collaboration tools

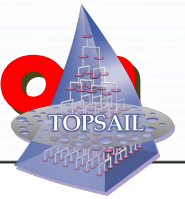


Role of policies in Team Formation



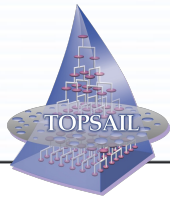
- Includes **finding leader** and **choosing members**
- Policy constrains who can be a team leader in terms of attributes of leader (experience, technical skills, …) and team (e.g., objective, size, budget, length, …)
- Modeled as negative and positive *authorization* policies
  - Eg: CIA staff with >5 years of experience and biowarfare knowledge are **permitted** to form a team of length of 6 months to deal with a crisis involving bioweapons
- Team members are specified in the same way
  - Policy constrains that sets of individuals that are a valid team via permissions and prohibitions
  - Leader queries policy management system for help in assembling a valid team and subsequent changes (e.g., replacing a member)

Role of Policies in Collaboration



- This involves several tasks that a team member must do including reporting
- These tasks are modeled as *obligations* on the team member
  - All team members of team T **must** send weekly reports to the team leader
- The workflow component of the agent reasons over these obligations while deciding what to do next

Role of Policies in Information Flow



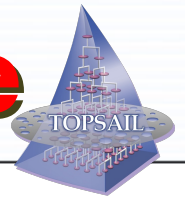
- Constrain information exchangeable by team members based on importance, team priority, agencies involved, etc.
- These are modeled as permissions and prohibitions, e.g.:
  - Members of CIA and FBI are **prohibitted** from exchanging information about X
  - Members of CIA and FBI are **permitted** to exchange information about X if they are on the same high-priority team that deals with X.
  - A **metapolicy** rule that team policy dominates agency policy resolves the conflict when a CIA member wants to send information about X to an FBI member.

Discovering social networks

- We're working on discovering a team's social network structure using several techniques
- Custom smart badges record degree of face-to-face interactions between team members.
- Instrumentation of coordination tools (e.g., Outlook) record degree and quality of computer mediated interactions

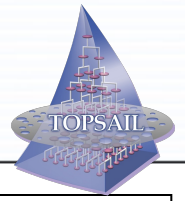
- v1 & v2 use IR to detect f2f neighbors.
- Designed to be low power and inexpensive (~$10)
- v3, prototyped on PDAs, also detect user's talking
- Correlating badge info yields conversational model

Social Network Perspective



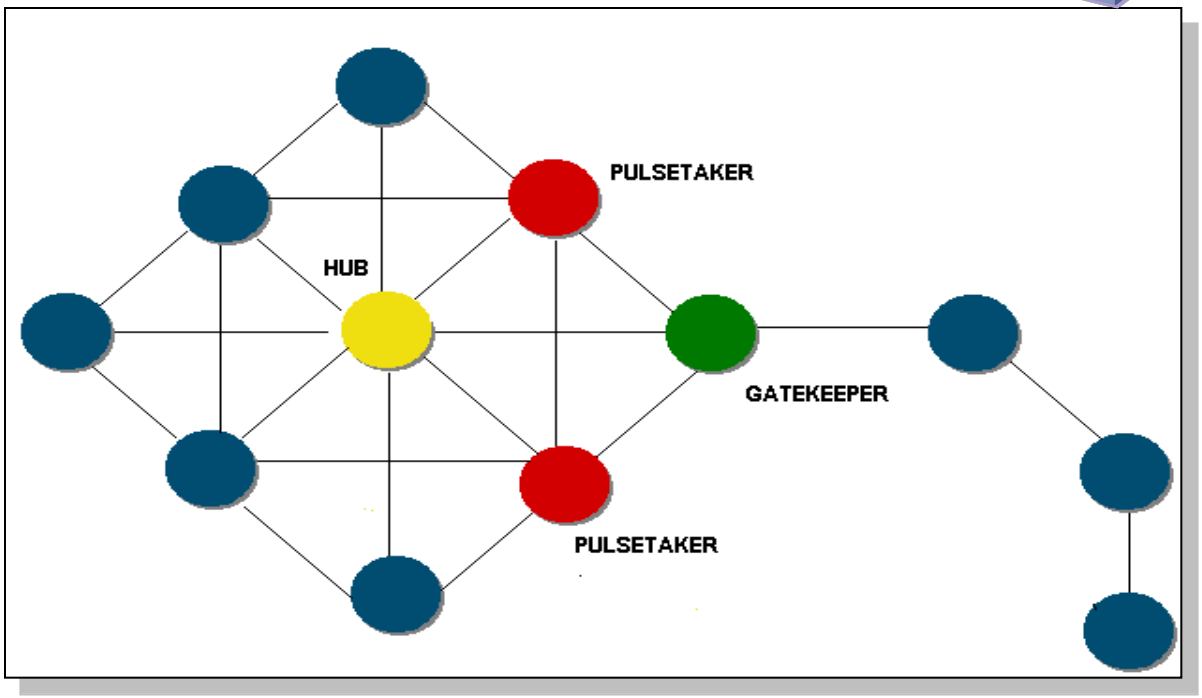
- The **reach** of an individual or team in an organization is constrained by personal influence, local networks and stovepipes.
- It constrains and limits the impact of collaborative problem solving.
- Untended or ignored networks may cut a swath through organizational silos to subsume, submerge and even *redirect* attempts to achieve innovation or change.
- Actively monitoring and nurturing network growth is critical.

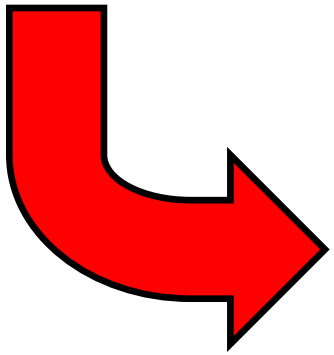
DNA of Social Networks



Analyzing the network

- Social network analysis can identify individual's roles as
  - Hubs
  - Gatekeepers
  - Pulsetakers
- and recommend the need to introduce new ones.





Tending the social network

- Add appropriate people to team based on their models (e.g., foaf, MBTI)
- or software agents can fill the gap

selecting web services

Security and Trust for Semantic Web Services

- Semantic web services are web services described using OWL-S
- Policy-based security infrastructure
- Why policies ?
  - Expressive – can be over descriptions of Requester, Service, and Context
  - Authorization
    - Rules for access control
  - Privacy
    - Rules for protecting information
  - Confidentiality
    - Cryptographic characteristics of informati exchanged

Example policies

- Authorization
  - Policy 1: Stock service not accessible after market closes
  - Policy 2: Only LAIT lab members who are Ph.D. students can use the LAIT lab laser printer
- Privacy/Confidentiality
  - Policy 3: Do not disclose my my SSN
  - Policy 4: Do not disclose my home address or facts from which it could be easily discovered
  - Policy 5: Do not use a service that doesn't encrypt all input/output
  - Policy 6: Use only those services that required an SSN if it is encrypted

Example

- Mary is looking for a reservation service
  - foaf description
  - Confidentiality policy
- BravoAir is a reservation service
  - OWL-S description
  - Authorization policy
    - Only users belonging to the same project as John can access the service

Mary

```
<!– Mary's FOAF description –>
<foaf:Person rdf:ID="mary">
<foaf:name>Mary Smith</foaf:name>
    <foaf:title>Ms</foaf:title>
    <foaf:firstName>Mary</foaf:firstName>
    <foaf:surname>Smith</foaf:surname>
    <foaf:homepage
    rdf:resource="http://www.somewebsite.com/marysmith.html"/>
    <foaf:currentProject rdf:resource=" http://www.somewebsite.com/SWS-
    Project.rdf "/>

    <sws:policyEnforced rdf:resource="&mary;ConfidentalityPolicy"/>
</foaf:Person>

</rdf:RDF>
```

Bravo Policy

```
<entity:Variable rdf:about="&bravo-policy;var1"/>
<entity:Variable rdf:about="&bravo-policy;var2"/>

<constraint:SimpleConstraint
                rdf:about="&bravo-policy;GetJohnProject"
      constraint:subject="&john;John"
      constraint:predicate="&foaf;currentProject"
      constraint:object="&bravo-policy;var2"/>

<constraint:SimpleConstraint
                rdf:about="&bravo-policy;SameProjectAsJohn"
      constraint:subject="&bravo-policy;var1"
      constraint:predicate="&foaf;currentProject"
      constraint:object="&bravo-policy;var2"/>

<!-- constraints combined -->
<constraint:And rdf:about="&bravo-policy;AndCondition1"
     constraint:first="&bravo-policy;GetJohnProject"
     constraint:second="&bravo-policy;SameProjectAsJohn"/>
```

```
<deontic:Right rdf:about="&bravo-policy;AccessRight">
    <deontic:actor rdf:resource="&bravo-policy;var1"/>
    <deontic:action rdf:resource="&bravo-
    service;BravoAir_ReservationAgent"/>
    <deontic:constraint rdf:resource="&bravo-
    policy;AndCondition1"/>
</deontic:Right>

.....

<rdf:Description rdf:about="&bravo-
    service;BravoAir_ReservationAgent">
    <sws:policyEnforced rdf:resource="&bravo-
    policy;AuthPolicy"/>
</rdf:Description>
```

How it works

Mary

URL to foaf desc
+ query request

Matchmaker
+
Reasoner

Bravo Air
Web service

Bravo Service
OWL-S Desc

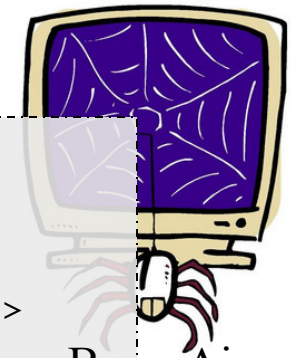
<sws:policyEnforced rdf:resource =
"&bravo-policy;AuthPolicy"/>

How it works

Mary

Mary's query = Bravo Service ? YES

Extract Bravo's policy



BravoAir

Web service

```
<deontic:Right rdf:about="&bravo-policy;AccessRight">
        <de
        <de
service;Bra
        <de
</deontic:

<policy:Gra
        <policy:to rdf:resource="&bravo-policy;var1"/>
        <po
</policy:G

<sws:Auth
        <po
</sws:Aut

<rdf:Descr
        <sw
</rdf:Desc
```

```
<constraint:SimpleConstraint
  rdf:about = "&bravo-policy;GetJohnProject"
  constraint:subject="&john;John"
  constraint:predicate="&foaf;currentProject"
  constraint:object="&bravo-policy;var2"/>

var2 = http://www.somewebsite.com/SWS-Project.rdf
```

```
<foaf:currentProject rdf:resource =
"http://www.somewebsite.com/SWS-Project.rdf"/>
```

```
<constraint:SimpleConstraint
  rdf:about="&bravo-policy;SameProjectAsJohn"
  constraint:subject="&bravo-policy;var1"
  constraint:predicate="&foaf;currentProject"
  constraint:object="&bravo-policy;var2"/>

Is  the constraint true when
var2 = http://www.somewebsite.com/SWS-Project.rdf
var1 =
http://www.cs.umbc.edu/~lkagal1/rei/examples/sws-
sec/MaryProfile.rdf
```

Status

- Policy compliance checking algorithms are implemented
- Ontologies for describing cryptographic characteristics of data
- Integration with OWL-S Matchmaker is part of our ongoing work

privacy on the web

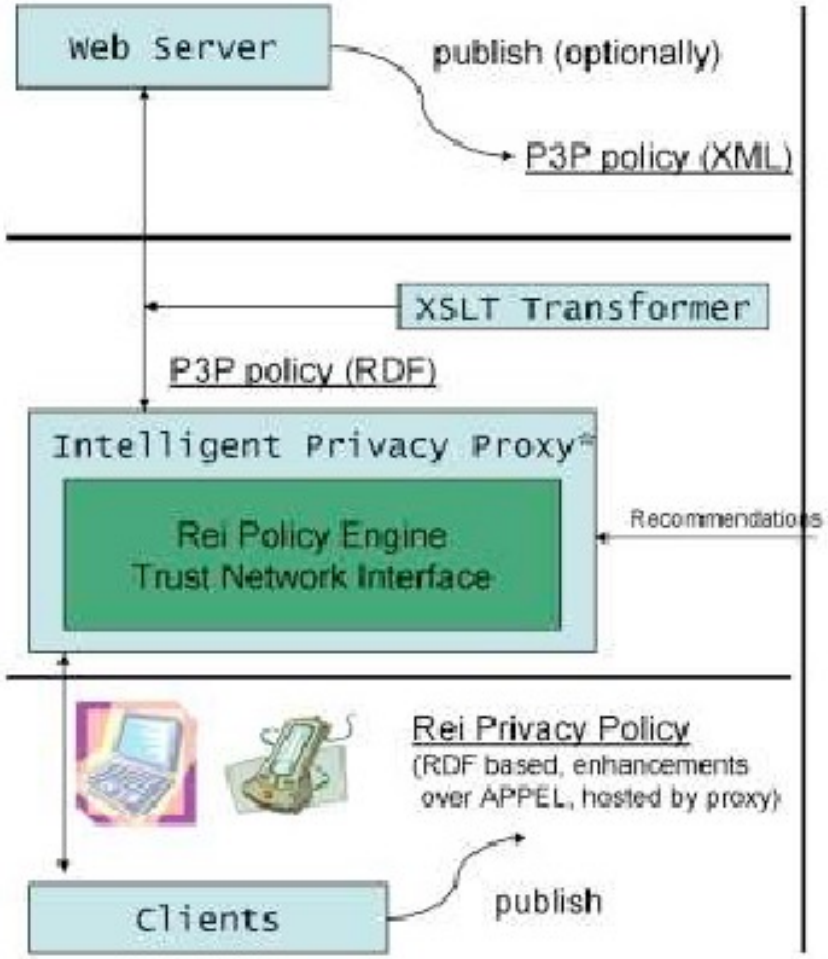
Privacy and Trust on the Internet

- Current state of the art:
  - Server's privacy practices described and published using P3P, a W3C standard
  - Clients specify a privacy policy using any of several systems, e.g., APPEL, a W3C standard used to describe client's privacy preferences
  - Browser plugin (perhaps using a proxy) alert or prevent users from visiting sites violating their privacy policy.
- Problems
  - Neither P3P nor APPEL is very expressive
  - Not extensible
  - Client side preferences only enforced when website has a P3P policy, which almost none have

Our Approach

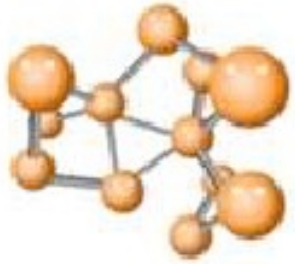
- Convert P3P into RDF (if not already in RDF)
- Model trust for website based on various attributes
- Use Rei to describe client-side privacy preferences over
  - P3P specs
  - Trust and other attributes of websites
  - Context of client

Our Approach



Website Recommender Network

Ontologies, Trust rules
Personal agents



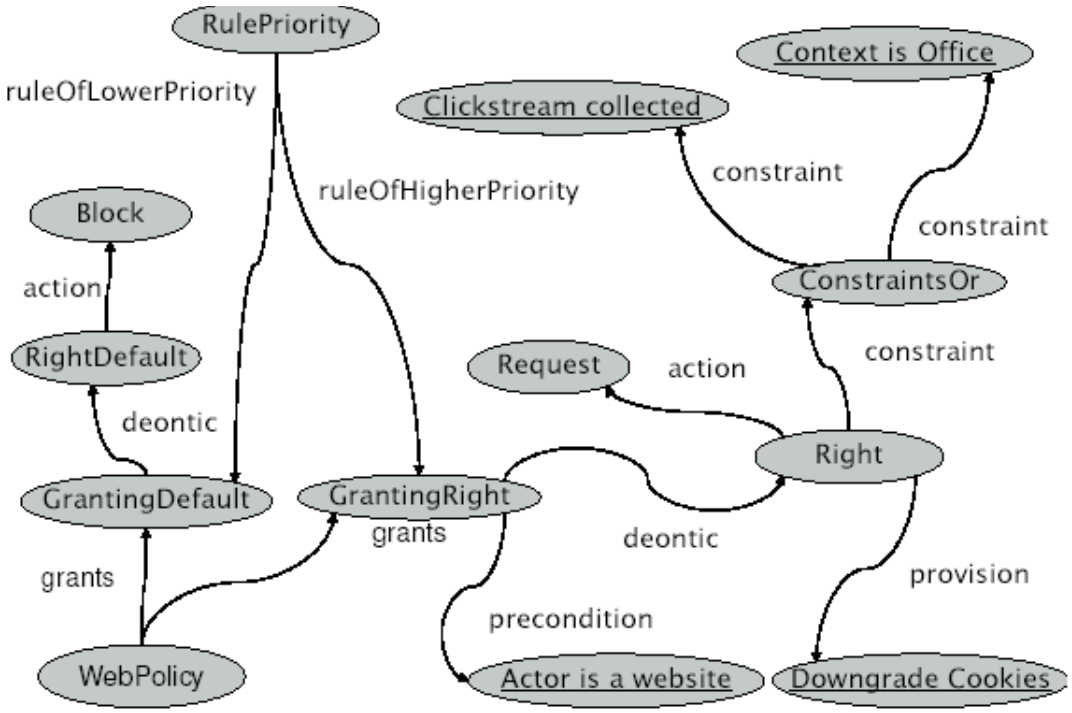
- Make assertion about sites
- Look for text policies and endorsement (e.g., by truste)
- Google ranks
- etc

Trusted Agent Network*

FOAF, Golbeck, Li ideas of Trust

Example

- Don't access a site that collects *clickstream data* unless I'm using my office workstation.
- Assume a site collects clickstream data unless there is trusted evidence that it does not.



summary

Summary redux

- **Declarative policies** are useful for constraining autonomous behavior in open, distributed systems
- This enables **more autonomy**
- The **Rei** policy language and associate tools have provided a good base
- **Semantic web** languages (e.g., OWL) used, grounding descriptions in sharable, semantically rich, machine understandable ontologies
- We're evaluating and exploring the utility of policies through prototype **applications**
- In one, **Topsail**, policies guide agents to help form, operate and maintain teams of people.

For more information

http://ebiquity.umbc.edu/

backup

Related Research

- WS-*
  - Lack of semantic expressiveness and reasoning capabilities
    - Most approaches are based on XML.
    - E.G., XML signature/encryption, WS-security, SAML.
  - Restricted extensibility
    - Possible solution is ontological approach

- Policy Languages
  - XACML : OASIS eXtensible Access Control Markup Language
  - EPAL : IBM Enterprise Privacy Authorization Language
  - Ponder
  - KeyNote
  - KAoS : Knowledgeable Agent-oriented System