Ponder: Policy Specification for Large-Scale Distributed Computing

Morris Sloman

Department of Computing
Imperial College London
m.sloman@doc.ic.ac.uk,
http://www.doc.ic.ac.uk/~mss

Contributors

Naranker Dulay
Emil Lupu
Alessandra Russo
Nicodemous Damianou
Sye Loong Keoh
Arosha Bandara
Tyrone Grandison
Leonidas Lymberopoulos

Contents

- Why Policy
- Ponder authorisation & obligation policies
- Ponder composite policies - roles & management structures
- Policy analysis, validation & refinement
- Implementation issues
- Trust and security management
- Future work

Policy Based Management

Events
Monitor
Events
Policy Agent

Managed Objects
Control actions
Decisions
New functionality
Policies
Security Specification

- Who can do what to my resources.
- What resources and services group members can access in a specific environment - shopping mall, hotel, virtual organisation. Different policy for staff, visitors etc.
- Role based policies
- Actions for security violations
- Need to switch between policy profiles as users move from one environment to another, or when an attack is detected

Policy

- Rule governing choices in behaviour of the system
- Derived from trust specifications, enterprise goals and service level agreements
- Need to specify and modify policies without coding into automated agents
- Policies are persistent
- But can be dynamically modified
- Change system behaviour without modifying implementation – **not new functionality**

Ponder Policy Based Solutions

- Triggering management actions, migration etc.
- Security
- Large scale
- Multiple Organisations

Obligation Policies

Authorisation Policies

Domains/Composite policies

Ponder Policy Language

- Precise specification of subjects, targets, actions and constraints for authorisations and obligations
- Declarative
- Object-oriented
- Role-based
- Management structures (communities)
Domains ➔ Grouping

- A domain is a collection of objects which have been explicitly grouped together for management purposes e.g. to apply a common policy

- Specify policy for groups of objects
- Can change domain membership without changing policy
- Policies deployed automatically to domain members

Authorisation Policies

- Defines what a subject is permitted or not permitted (prohibited) to do to a target
- All policies can be specified as a parameterised type from which instances can be created

```plaintext
type auth+ doorcontrol (subject s, target t, string start, string end) ;
  action enter ();  when time.between (start, end); }

inst ChildSwim = doorcontrol (/hotelGuest/child, /leisure/pool “1000”, “1700”);
AdultSwim = doorcontrol (/hotelGuest/adult, /leisure/pool “1000”, “2000”);
```

Negative Authorisation

- Used for revocation of access rights

```plaintext
inst auth- revoke {
  subject /staff/JoeBloggs;
  target /leisure/ ;
  action enter ;
  when time.date > 30:4:2003 }
```

Obligation Policy

- Defines what actions a subject must do
  Assumes well behaved subjects with no freedom of choice.
- **Subject based ➔** subject interprets policy and performs actions on targets
- Event triggered obligation
- Actions can be remote invocations or local scripts
- Can specify sequencing or concurrency of actions
Obligation Example

- Enable low security profile on entry to work area and enable high security profile on exit from work area

\[
\text{Inst oblig \{ } \\
\text{on workentry (userid)} \\
\text{subject securityAgent (userid)} \\
\text{do enable.pda (userid, lowSecurity)} \\
\text{\} } \\
\text{Inst oblig \{ } \\
\text{on workexit (userid)} \\
\text{subject securityAgent (userid)} \\
\text{do enable.pda (userid, highSecurity)} \\
\text{\} }
\]

Dynamic Policy Adaptation for Network Services

- We modify network behavior by adapting network management policy for a DiffServ environment.
  - By dynamically changing the parameters of a policy to specify new attribute values for the run-time configuration of managed objects.
  - By selecting and loading or enabling a policy from a set of pre-defined network policies at run-time + select policy parameters.
- Future Challenge
  - Learn by monitoring system behaviour
  - Generate new policies

Policy Adaptation Levels

Example DiffServe Policy

\[
\text{inst oblig } /\text{Policies/DiffServPolicy } \{ \\
\text{subject } /\text{PMAs/DiffServAgent}; \\
\text{target } r = /\text{DiffServDomainA/Routers/CoreRouters}; \\
\text{on } \text{ConfigRequest(DS, max_input_rate, min_output_rate)}; \\
\text{do } /* DS: The Diffserv codepoint for EF: 101110.*/ \\
\quad r.\text{applyEFPHB(DS, max_input_rate, min_output_rate)}; \\
\text{when } \text{max_input_rate} \leq \text{min_output_rate}; \\
\quad /* Property that EF traffic must satisfy */ \} 
\]
Other types of policies

Refrain Policies – specify actions that subject should refrain from performing. Similar to auth- but subject based interpretation e.g., disclosure

Filtering Policies - transformations of parameters of positive authorisation policies, where it is not practical to provide different operations to reflect permitted parameters

Delegation Policies - Specify which actions a subject may delegate to a grantee. Must be a subset of subjects, actions and targets in an authorisation policy.

Roles

- Role groups the rights and duties related to a position in an organisation
- E.g., finance director, ward-nurse, guest, premium-guest, local user, network client
- Specify policy in terms of roles rather than persons
  ➔ do not have to re-specify policies when person assigned to new role
- Use inheritance to specialise roles

Guest Role Example

type role guest (target facilities, room) {
  inst oblig lights {
    on roomexitlock (room);
    do lightsOff (room); alarmOn (room);
    when unoccupied.room;
  }

  inst auth+ roomentry {
    target room;
    action openDoor;
  }

  // other authorisation and obligation policies
}
Create a different instance of guest for each person registering at the hotel. Premium guest inherits guest + additional rights.
Management Structures

Define configurations of roles within organisational units or ad-hoc communities

Organisational Patterns

Ad-Hoc Dynamic Communities

- Pre-defined specification of role-types in a community template -> rights and duties in the community.
- Restrictions under which community may be formed.
- Conditions under which
  - A user may join a community (authorisation policies)
  - A user should join a community (Obligation policy)
- Protocols for community instantiation

Ponder Toolkit

- Policy Implementation Overview
- Ponder Policy Toolkit
  - Domain browser
  - Compiler
  - Deployment
Policy Implementation

- Query targets
- Domain service
- Policy service
- Obligation & Refrain Policies
- Authorisation Policies
- Target Objects
- Actions
- Events
- Monitoring service
- Edit, enable disable...

Domain Browser

- Hyperbolic Tree Algorithm
- Efficient display of complex structures

Policy Editor

- Integrated development environment
  - Integrated with Domain browser and Compiler
  - Syntax Highlighting, Templates for policy specification

Policy Compiler

- Multiple back-ends for various representations and heterogeneous implementations
Policy Deployment Objects

- Policy Control Object
  Maintains information on policy state + subject and target domain membership to deploy policies to enforcement objects

  - Executable policies
    Java oblig., XML, Firewall rules etc.

  - Combined enforcement & policy decision components

Trust & Security Management

- What is trust
- Trust & recommendation specification
- Trust & policy interaction

What is Trust

- A quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context

<table>
<thead>
<tr>
<th>Trustor</th>
<th>Context: Hotel Services</th>
<th>Trustee</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trust relationship</td>
<td></td>
</tr>
</tbody>
</table>

- Distrust useful for trust revocation or in default trusted environments
- Quantification implies various degrees of trust/distrust
- Dependability implies timeliness

Trust Classification

1. Access to Trustor Resources eg MSN Messenger
   "MyMachine trusts MSNMess to save files"

2. Provision of Service by Trustee eg e-news deliveries, email, archive
   "Tom trusts news.com"

3. Certification of trustee eg VeriSign or Brit. Medical Assoc.

4. Delegation of trust eg use certification authority for trust decisions
   "May delegates all decisions concerning verification to her bank"

5. Infrastructure trust eg, network, storage
Trust Specification

- **Trust Predicate**
  trust (trustor, trustee, actions, level, ) ← constraint set
  trust (Helen, _hotel, print; processing, 50) ←
  hotelGroup ( _hotel, HolidayInn)
  - Distrust when level < 0
- **Recommend Predicate**
  recommend (recommendor, recomendee, actions, level) ← constraint set
  recommend (Morris, _attendee, verifyCredential, medium) ←
  ICstaffMember (_attendee)
  trust (Harry, GameCo, DownloadGames, medium) ←
  recommend (Tom, GameCo, DownloadGames, high)

Trust-based Authorisation Policy

type auth+ Access ( domain sub-directory, string TrustValue){
  subject Client;
  target sub-directory;
  action downloadMusic();
  when trust+(FrontEnd, ClientApp, downloadMusic(ContentDatabase), TrustValue );
}

inst auth+ AccessHigh = Access(/BMW/ContentBase, HighTrust);
inst auth+ AccessLow = Access(/BMW/ContentBase/Restricted, LowTrust);

Trust, Experience and Risk

- Trust is not static but changes with time as a result of experience
- Need for 3rd party recommendations c.f. PGP
- Reputation - anonymous recommendations
- Trust is related to risk and value
  High risk → low trust
  But high risk, low value may be medium trust
- Trust framework must monitor experience, risk and constraints in order to dynamically update trust levels and relationships.

Trust Refinement & Adaptive Security
Policy Analysis & Refinement

- Conflict Analysis
- Policy Validation
- Refinement

Modality Conflicts

- 3 types: auth+/auth-, oblig/auth-, oblig/refrain (auth+/refrain is not a conflict)
- Arise when subjects, targets and actions overlap.
- Potential conflicts identified by detecting overlaps

Semantic Conflicts

- Types of conflict:
  - separation of duty e.g., the same person is not allowed to authorise payments and initiate them
  - self-management e.g., a manager cannot authorise it’s own expenses
  - conflict for resources e.g., not more than 5 users can access the DB at the same time
- Need to specify the conditions which result in conflict
- Constraints on a set of policies (Meta-Policies). Specified using OCL
- Included in composite policies such as roles or communities

Policy Analysis and Validation

Policy Analysis

- Consider constraints when detecting conflicts
- Identify which situations lead to conflict
- Reason with partial specifications
- Ponder is not a logic based language so not amenable to this type of analysis.
- Transform to Event Calculus


Policy Validation

- Is policy compatible with device functionality?
- Resource allocation policies - are there sufficient resources?
**Static Analysis Approach**

- Need both system behavioural model and policies.
- Abduction applied to Event Calculus representation.

**Policy Refinement**

- Derive policies from SLAs, business goals & trust
- Not automatable but can apply refinement patterns
- Maintain consistency during refinement
- Ensure completeness – refined policies fully implement more abstract ones
- Use goal regression to elaborate plans of actions and identify alternatives for refinement.

**Conclusions**

- Security specification
  - Authorisation, filter, Refrain, delegation, role
- Management
  - Event-triggered Obligation, role
- Large scale
- Multiple Organisations
  - Domains + Composite policies
- Analysis
  - Declarative language

**Future Activities**

- Ubicare Centre (UK)
- Maui EU Network of Excellence
UbiCare Centre

- Ubiquitous Computing for Healthcare in the Community
- UK DTI funded
- http://www.ubicare.org
- Projects:
  - **UbiMon**: Ubiquitous Monitoring Environment for Wearable and Implantable Sensors
    Contact: Guang-Zhong Yang  gzy@doc.ic.ac.uk
  - **ANS**: Autonomic Networked System Management Tool for Ubiquitous Systems.
    Contact: Julie McCann  jamm@doc.ic.ac.uk

Collaborators

- Imperial College Computing, Medical School, Bioengineering
- Lancaster University
- University of Southampton
- Cardionetics
- Central Data Control
- Medtronic
- Orange
- Telewest
- Tyco UK
- Toumaz Technology

Objectives

- Monitoring of patients with chronic disease during normal home, work & outdoor activity
  + prediction of problems
  ➢ Fusion of information from multiple sensors for patient medical conditions and activity
- Enable widespread collection of data to aid predictive medicine
- Develop light-weight autonomic management techniques for ubiquitous computing
- Provide tools and techniques to simplify development of ubiquitous healthcare

UbiCare Architecture
Maui: Managing the Ubiquitous Internet

- EU NOE - Universities + industries
- Cater for large-scale (ISPs) as well as small-scale body and room area networks.
- Self-managed cells (SMCs) - patterns for management
- Dynamically configurable SMC management functionality

Self Managed Cell

Cell Management Interaction

- Horizontal peer-to-peer
- Vertical representing hierarchical service provision
- Composition - internal SMCs are not visible externally

Additional Information

- Links to Policy information
  - Ponder download
  - Papers
  - Workshops
    http://www-dse.doc.ic.ac.uk/policies
- Policy 2003: Workshop on Policies for Distributed Systems and Networks
  Lake Como, Italy, 4-6 June 2003
  http://www.policyworkshop.org