Enterprise Policies and Shrink-Wrapped Systems

Peter F. Linington
University of Kent at Canterbury
Agenda

• To look at the position of policy specification within a broader system specification context, and identify enterprise problems and requirements
  • some motivators – why do this now?
  • a little ODP Enterprise background
  • constraints on setting policy
  • constraints on changing policy

• The talk is mostly about glue.
• It’s abstract glue; maybe blue-skies glue.
The Shrink-Wrapping Problem

- In building a system, components, including collections of policies, may be locally specified, reused or purchased in a shrink-wrapped form.
- How do we ensure consistency and predictability?
- How do we link the different parts of the specification, and manage changes to them?
Model Driven Architecture

• OMG’s strategy is now focused on its Model Driven Architecture initiative.

• The drivers for the OMG’s Model Driven Architecture initiative have a great deal in common with the origins of the ODP viewpoint structure.
  • the aim is to create a strong tool chain that can be driven from high level models, and will automate much of system generation
  • Some presentations on MDA suggest that this is a “one click” process, but in reality it will involve many general policies that contribute to the system design.
Model-Driven Architecture

• OMG reacting to the facts that
  • CORBA is not the only game in town;
    • JAVA RMI, JINI, Beans
    • DCOM, .NET, SOAP
  • UML is (if they can hang on to change control);
  • the OMA is way out of date and its use is uneven.
• Discovered the need for stable models at a level above middleware.
• Start the tool-chain with set of re-usable UML fragments.
  • This is why UML hints and fragments turn up throughout this talk.
The MDA Positioning Picture

- Model Driven Architecture
- UML
- MOF
- CWM
- Pervasive Services
- Transactions
- Persistence
- Security
- Directory
- Event
- Scalable
- Real-Time
- Fault Tolerant
- Embedded

- CORBA
- XMI/XML
- WEB
- JAVA
- .NET

- Finance
- Manufacturing
- E-Commerce
- Telecom
- Space
- Transportation
- Health Care
- More...
MDA Tool Chain

Pervasive Services Model

Modelling Tool builds

Platform-Independent Application UML Model

Mapping tools performs

Mapping to CCM Server Platform

Domain Facilities Model

MOF

OMG IDL

PSDL

CIDL

C++ or Java

Config files

Compile, etc.

CCM Server
A Policy Perspective?

MDA Hype

Using policies
An ODP Perspective

- The Reference Model for Open Distributed Processing is a standard framework for organising the specification of systems. It is published as
  - ISO/IEC 10746, Parts 1-4
  - ITU-T Recommendations X.901-X.904
- The RM-ODP describes systems in terms of a set of five inter-related viewpoints, addressing different areas of concern.
- One of these viewpoints is the Enterprise Viewpoint, dealing with the way an organization uses the ODP system, and the objectives and policies involved.
  - Under final ballot as ISO/IEC 15414 | X.911
  - also PDAM in preparation
Why Specify an Enterprise Viewpoint?

• The idea of the ODP Enterprise Viewpoint is to capture constraints on the detailed design of an IT system, in terms of the enterprise structure and requirements.
  • in particular, the enterprise viewpoint can include statements of policy that will guide the design;
  • these are likely to change during the system’s lifetime, as circumstances change;
  • examples might be security policies and accountability or delegation policies;
  • federation (dynamic coalition) is a key issue;
  • the detailed form of the enterprise language depends on the form of the software development process.
Caveats

• ODP enterprise language is not yet a published standard
• The standard does not specify a specific notation
  • standard defines a conceptual framework
  • usage requires language binding
  • the detailed form of the enterprise notation depends on the form of the software development process.
• Emphasis is on declarative specification
• Examples and interpretations are the author’s, and go to levels of detail beyond that agreed in the standard.
Communities

- An Enterprise Specification is built up from a set of related communities. A community is:
  - a configuration of objects;
  - formed to meet an objective;
  - consistent with a community type;
  - instantiated (when necessary) from a template;
- A community type is defined to have a number of typed roles; in any community instance, the roles are filled by appropriately typed objects, constrained to behave as defined by the community type.
  - Note: in ODP, the concept of community role extends template roles; there are other kinds of roles, e.g. action roles.
Communities (2)

- Enterprise specification is concerned with an uncertain world:
  - Specified actions may be unavailable or in conflict
    - leads to emphasis on obligation, permission and prohibition, rather than specific behaviour
    - pseudo-economic approach to resolution
  - Convenient to use constraint based specification techniques to circumscribe behaviour;
  - In some respects, communities are like patterns.
Community Roles

- Roles are the formal parameters of the community type, and are filled subject to:
  - the object filling the role having a type that is a subtype of the role type;
  - cardinality constraints on the roles being satisfied;
  - specific population rules being satisfied
    - separation of duties, e.g. require distinct proposer and authoriser for purchase;
    - sharing requirements – same object must be involved.
- Community behaviour may allow changes in role binding by substitution of objects or change of cardinality (role creation and filling combined).
Basis of Roles

- This use of the term role is based on a theatrical metaphor:
  - the script of a play includes a character list defining a set of roles;
  - when a play is produced, actors are selected to fill the roles;
  - different productions have different sets of actors filling these roles; the mapping is quite flexible (e.g. use of understudies).
Composing Communities

- Composition can be by:
  - object refinement
  - object sharing
  - role linkage
- It forms a higher level community
Nesting of Communities

- A community has context provided by surrounding or overlapping outer communities.
  - these provide the basis for its contract;
    - when no obligation is placed, the constraints outside apply
    - new obligations must not conflict;
  - an outer community may delegate authority to an inner one
  - may have specific rules for overriding, e.g.
    - specific over general
    - new over old
Community and Structure

- Specific business logic is defined as one community, and delegation rules are defined as another.
- The two communities are linked by the designer establishing correspondences between roles.
An Example of Use of Communities

• A simple application might be specified by composing three communities, covering the business logic, security and delegation

• Consider a web-based loans system, with additional human authorisation for loans above $1000, self authorisation otherwise.
Business Logic

- Roles correspond to stakeholders in process
  - Customer making request
  - Web responding agent
  - Authorising manager
- Behaviour of community expressed in terms of the roles
- Link to observable system behaviour is via the relation between objects and the roles they fill.

```
customer->agent.ask;
([val <1000] agent->customer.agree
  []
[val >= 1000] agent<->manager.confirm;
  agent->customer.agree)
```
Delegation Model

• Roles cover potential actors and management of delegation
  • Performer
  • Candidate Executors
  • Delegation Manager

• The internal binding presents the object filling the bound Executor role as filling the Performer role

Behaviour of the community defines the rules for adding candidate executors and for selecting which is currently bound to the performer
Delegation Issues

- Depending on the delegation rules, the actual objects filling the performer role may be different subtypes.
- When is a change in the delegation allowed?
  - For new instances of the business pattern?
  - Immediately?
  - May imply significant implementation complexity if state concerned has to be transferred between executors or resynchronized.
- The change of actual object in a role may not be transparent to other constraints in effect
  - e.g. separation of duties.
- Need emergency delegation (by third party) or escalation.
Access Control Model

- Roles cover subjects, objects and administrators
- Sets of permissions associated with roles
- Rules for management of permissions, permission hierarchy, etc. form part of behaviour.
  - different models might provide the same roles and composition properties – e.g. certificates v. auth. servers

```
  subject

  object

  administrator

  Rules of the access model, covering execution and Administration aspects.
```
Access Control Issues

• The composition rules are complex, and must classify the actions and action roles in the business logic as being under control of a particular access control model instance.
  • Suitable labelling gives us RBAC roles, but other usages are possible
• Negative policies imply complimenting with respect to the controlling domain – so depends on domain definition.
• How do we support security-aware applications?
  • tests for role filler authenticated, data signed?
• Need to test for security consequences of e.g. resource allocation
  • delegation change violates Chinese Walls?
  • Denial of service attack by forcing delegation?
Linking the models

Customer -> Agent: ask;
([val < 1000] Agent -> Customer: agree
[]
[val >= 1000] Agent <-> Manager: confirm;
Agent -> Customer: agree)

Behaviour of the community defines the rules for adding candidate executors and for selecting which is currently bound to the performer.

Rules of the access model, covering execution and Administration aspects.

∀ act in BusinessLogic (∀ role in act | role ∈ AccessRoles ∧ act.role.obj ∈ sec.object.rolefillers)
select(act.role) ∪ (∀ srole in act | act.srole.obj ∈ sec.subject.rolefillers)
select(act.srole)
Policies

• Take a step back: dictionary definitions
  • courses or principles of action adopted or proposed by an organization or individual
  • prudent or expedient conduct or actions
  • in Scotland, pleasure-grounds round a mansion.

• Connotations
  • there is real choice to be exercised
  • the choice is likely to change if circumstances change
  • a policy is a structured collection of rules or preferences
  • may be applied to the set of choices rather than the set of solutions - “each department has a safety policy”.
Policies and Policy Envelopes

• The term *Policy* can be used in two ways, denoting
  • a set of rules to be obeyed in some particular situation
  • a place in a larger specification at which such a set of rules needs to be provided. Over a period of time, such a place may be associated with many different specific sets of rules – i.e. policies change.

• This is essentially a type-instance distinction, but since there are many other uses of policy type with different scopes and purposes, we will use the distinct term *policy envelope* to indicate
  • a named point at which a policy should be defined
  • the constraints on the policies that are acceptable.
Defining the Envelope

• Examples of the kind of information that might be involved in defining an envelope are:
  • core set of names – e.g. subject, target, action names – and whether this core set can be extended;
    • may be by reference to a repository? difficult to model check specification.
  • restrictions on their combination;
  • fixed rules that the defined policy must be consistent with;
  • cardinality or uniqueness constraints;
  • and more …
Linking Specifications

• The natural approach to take would be to make the policy a refinement of the policy envelope
• Policy envelope may be defined by middleware but filled by fragments of enterprise specification?
• Specification languages take the viewpoint of policy definer, not policy deployer
• Here is where the shrink-wrapped issue really bites
Enterprise Constraints on Hooks
Deployment/Configuration Models

• How do other people do it?
  • Instructive to look at UML profile for Framework Architectures – links & decoration with configuration and refinement constraints.
    • can do almost anything with tags and stereotypes
    • but need strong backing for proposals if tool vendors are to take them seriously
    • specifics of that proposal too low level – assumes you are defining implementation structure
  • EJB configuration tools have similar integration problems
    • effectively steer the process by introducing an external level of deployment specification.
Linking Specifications

• The definer of the policy envelope may need to:
  • Provide mechanisms for identifying or selecting specific policies
  • Control the working set of policies available for selection
  • Provide suitable default policies for *out of the box* running
  • Against static nature of UML class structure.
Timing, Versions and the Tool Chain

- The Enterprise Specification is maintained throughout the lifetime of the system. If policies change, then these changes need to work through to the running system, often on precise time-scales.
  - a pricing policy, or a tax structure may need to be introduced on a specific date;
  - other development and maintenance will need to be carried out in parallel;
  - recovery and back tracking need to be supported;
  - need preparatory changes and dummy running;
  - implies need for flexible build process and version control.
Policy Specification Languages

• Most of the policy languages under active discussion have a strong emphasis on a event-condition-action styles of modelling

• Can talk about policies and policy envelopes in terms of coverage and reachability of regions of state space (or, if you like, phase space, folding in event trace as part of state)
Coverage of State Space

- Consider the footprints in state space
  - the envelope has largest coverage;
  - individual policies are subsets of this;
  - They probably all share some initial state.
Changing Policy

• What happens if we want to change policy and the system is in a state that would no longer be valid?

• Transition rules; do you make the change:
  • if at rest?
  • if in overlap region?
  • when next in overlap?
    • consider concurrency – does it matter if introduction of related changes is gradual?
    • guarantee change within a bounded period?
    • need transactional change of group of policies?
  • do you define matching sections?

• Evolution and planned change – ramping down, then up?
A Model of the Development Process

• If we want to express constraints on the dynamics of policy evolution, we need a model of the development process.
  • This may constrain concurrent policy changes affecting different parts of the system
  • It may be convenient in some cases, to combine this with a configuration model that establishes policy envelopes and binding of pieces of specification.

• A similar model gives
  • a conformance testing framework
  • expresses constraints on system instantiation, e.g. to express copyright on system designs?
    • Roles for specifier, implementor, operator, owner of the system?
Unique Implementations

- The original specifier may have a unique instance in mind …
- but an implementor may have other ideas and other markets.
Policy Conflict or Absence of Solution?

- Composite behaviour that evaluates to false
  - $\text{correct} \land \text{complex} \land (\text{complex} \rightarrow \sim\text{correct})$

- Possible behavior exists, but the environment does not offer it; consider sub-goals:
  - *Maximise Returns* and
  - *Make Ethical Investments*

- Are these really distinct?
  - supplier must receive payment before delivery
  - customer must confirm delivery before payment
Is there a Policy Administrator on Board?

- There is a need to allow for, and support, heroic disregard for rules.
- Rules that aim to define good practice or normal operations should not be regarded as absolute
  - constraints on location;
  - constraints on time of day or period of duty;
- Pumping out the cellar – problem of capturing the expectations of a reasonable person.
Conclusions: Where Next?

• The RM-ODP Enterprise Language is worth looking at?
  • Well I would say that, wouldn’t I?
• Designers of policy languages should consider both
  • Expressive power
  • Ability to define and limit a policy envelope
• More work is needed on the dynamics of policies, particularly on the interaction of policy change and concurrency.
• Policies are a powerful and successful tools, but they are only one aspect of system specification, and we must not lose sight of the need to fit them into the bigger picture.
• Nothing new here, then?
A Control Problem on Campus

- **Students and Staff**
  - Student and staff use separate resources
    - Students cannot access the machines used by staff
    - All student access to internet is via caches
  - Students normally work in their study bedrooms, using resources on teaching hosts
  - A project may need access to research hosts or WLAN
  - Staff supervise student projects; there are regular meetings in staff offices, with demonstrations
  - Staff prepare exam papers
- Administrators over-ride controls, particularly to respond to threats, particularly on WLAN.
• Access which is not explicitly allowed is forbidden
  • Students are permitted access from their study bedrooms to
    • teaching machines
    • the student web cache
  • Staff are permitted access from their offices to
    • research machines
    • teaching machines
    • the staff web cache
      – create permissions to connect directly to specific external networks and services
      – they are obliged to remove such permissions when no longer needed.
Typical Policies – Ad Hoc

• Administrators respond to external threats
  • this may involve short-term over-riding of standard permissions
  • Particular need to control WLAN
• Students perform projects, and are given additional permissions to enable them to do so for duration of project
  • access to resources on research hosts – e.g. special compilers or simulators
  • access to WLAN resources
• Supervisors establish short-term permissions to allow access from their offices to student code and servers
• Staff establish short term barriers for high security activity.
Example Configuration
From Policy to Infrastructure

- The policies give a high level view of requirements
  - mapping performed by tools controlling resources
  - detailed configuration of routing controls and firewall settings need to be established
  - short term changes only feasible if automated process