Usage Management In Multi-level Security Environments

Gregory Heileman and Christopher C. Lamb

Department of Electrical and Computer Engineering
University of New Mexico

April 4, 2012

Introduction

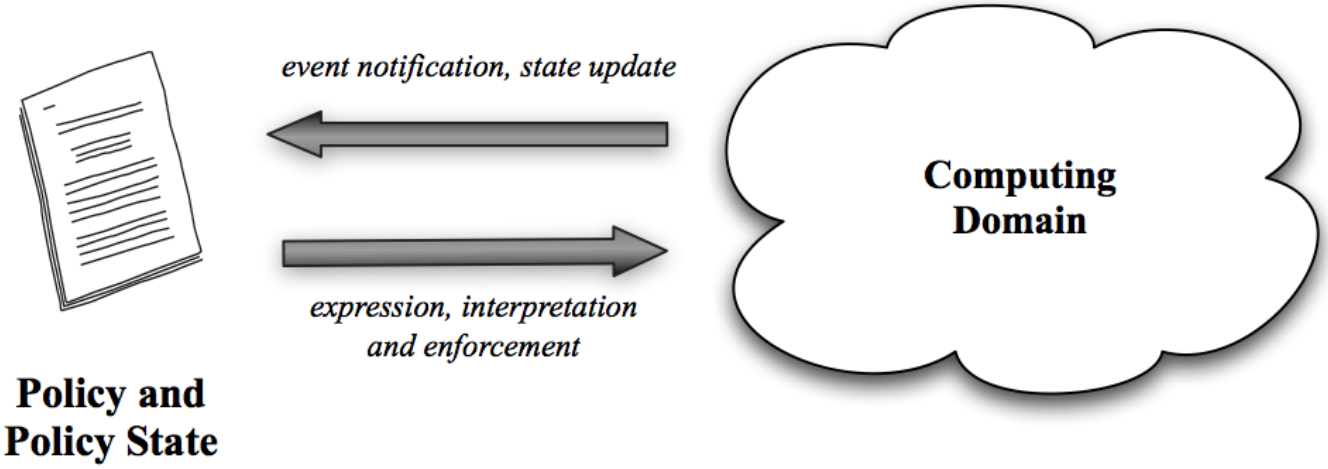
Today we will cover our work in usage management, new architectures we are developing to enable usage management, and how they apply to the cross domain information management problem.

Usage Management

- **Usage management** refers to the ability to control how resources (data and services) are used across and within computing domains.
- Controlling how information is used becomes increasingly difficult as computing infrastructure becomes more distributed. However, the ability to share information between domains provides for powerful capabilities, as well as increased security risks.
- Usage management incorporates aspects of access control and digital rights management (DRM):
  - Access control – access rules are defined in terms of relationships between a set of resources and a set of users.
  - DRM – access is implicit, the focus is on specifying and enforcing rules related to how the resource can be used.
- We seek the development of open frameworks for usage management, based upon fundamental system design principles, that support interoperability within distributed computing environments.

Usage Management Meta-Model

Fundamental design principle: **Policy and computing domain separation** –

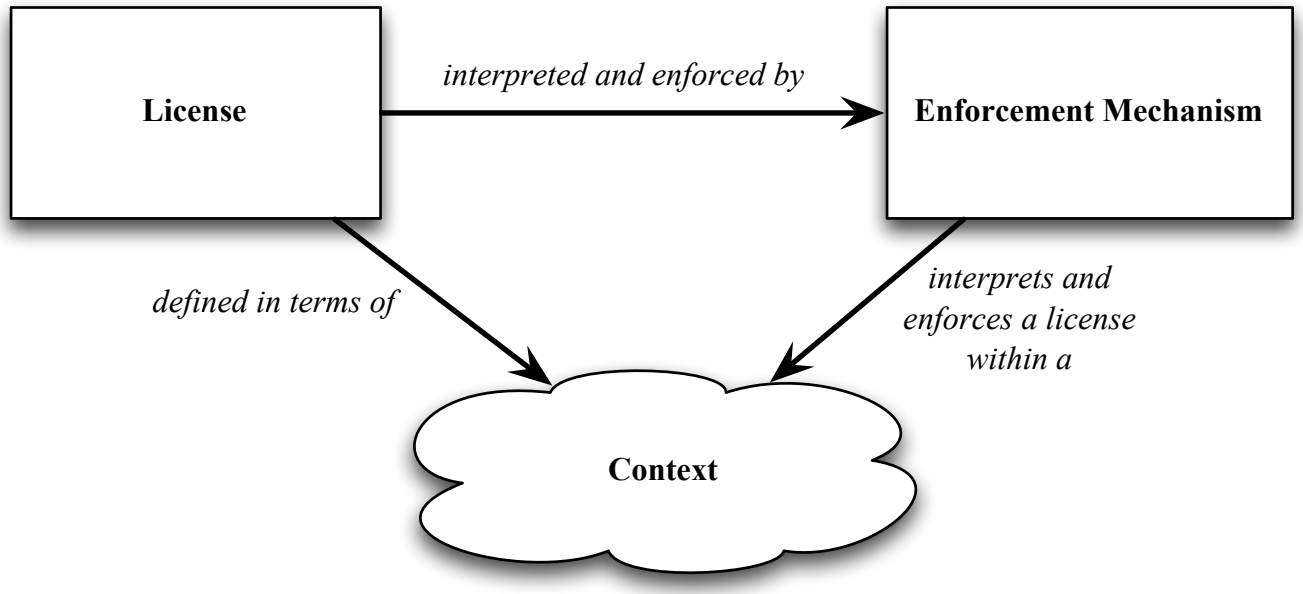


- A policy describes usage rules that govern how content may be used within computing domains.
- The primary application of a policy is for reasoning whether or not a given action may be carried out within a given computing domain.
- The current state of the policy (previous actions can affect this state), as well as the current conditions of the computing domain may be used in specifying usage rules.

Usage Management Meta-Model

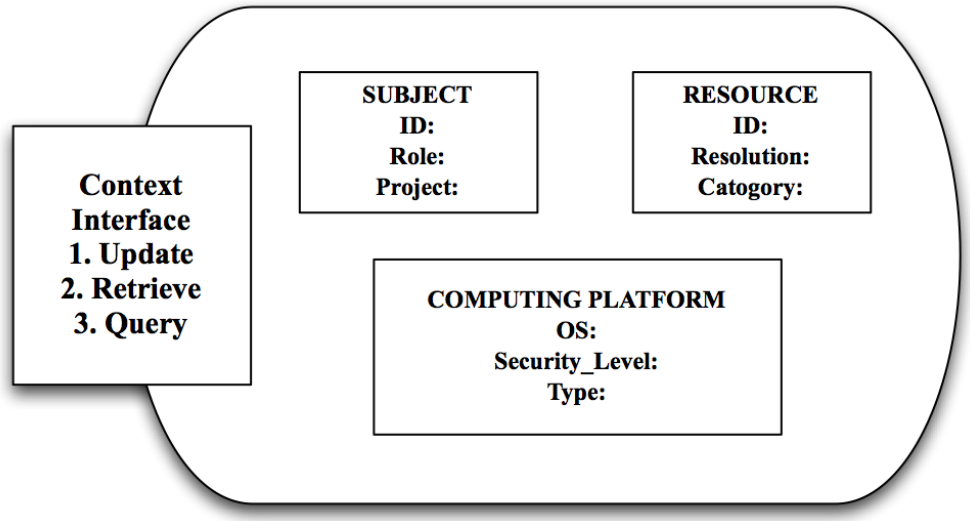
- Adherence to this design principle directly supports interoperability, and more specifically, the ability to operate across multiple computing domains.
- Most of the "policy" languages that have been proposed in the areas of access control and DRM can be mapped to this meta-model.
- Based upon this meta-model we can identify the three major functionalities that a usage management framework must support:
  1. Policy Expression – the usage rules.
  2. Policy Interpretation – can an action be performed in a given domain?
  3. Policy Enforcement – mechanisms for enforcing the allowed actions.

A High-level UM Framework



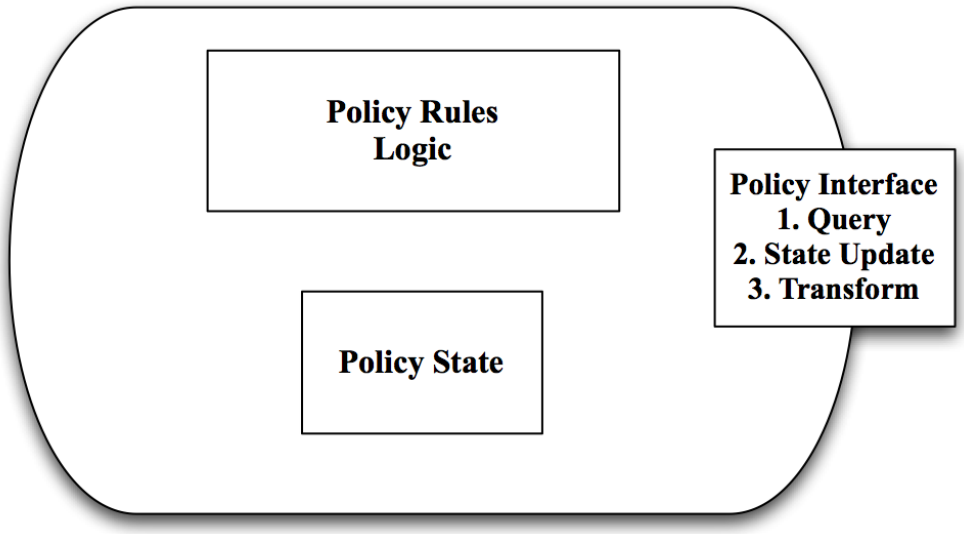
- We have created a usage management model based upon the meta-model that satisfies the previous separation principle through the interaction of two objects:
  - **Context Object** – An instantiation of the "context", i.e., the entities that exist within the computing domain, along with their relationships, and current state.
  - **License Object** – Usage policies expressed in some program logic, along with the current policy state.

UM Framework – Context Object



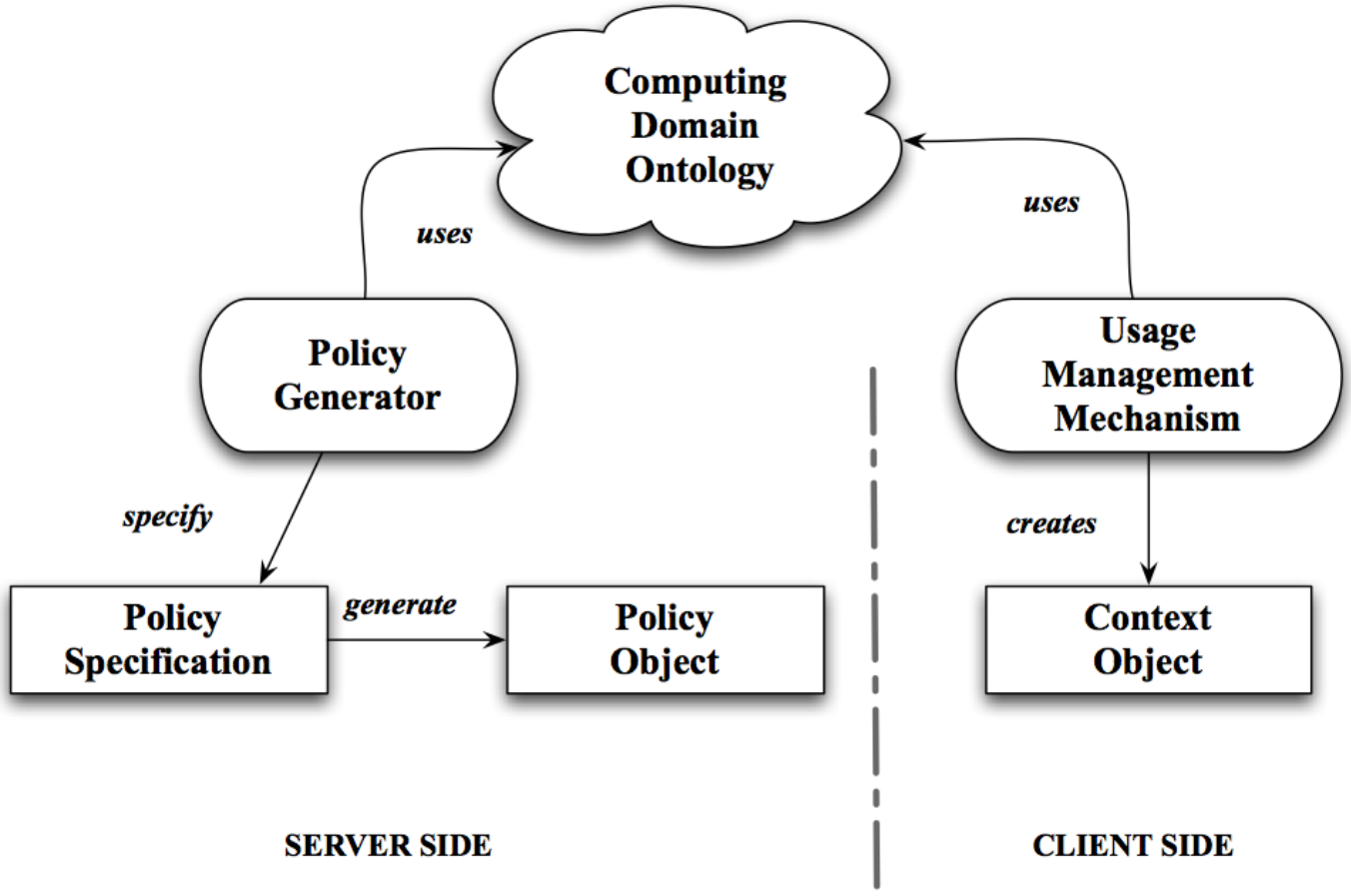
- Primary purpose: to capture the current conditions in the computation environment, and to provide them to the policy interpretation mechanism.
- An update interface allows updating, retrieving and querying of the attribute values associated with the entities in the context.
- A context object is updated by the system with appropriate values using the context interface.

UM Framework – License Object



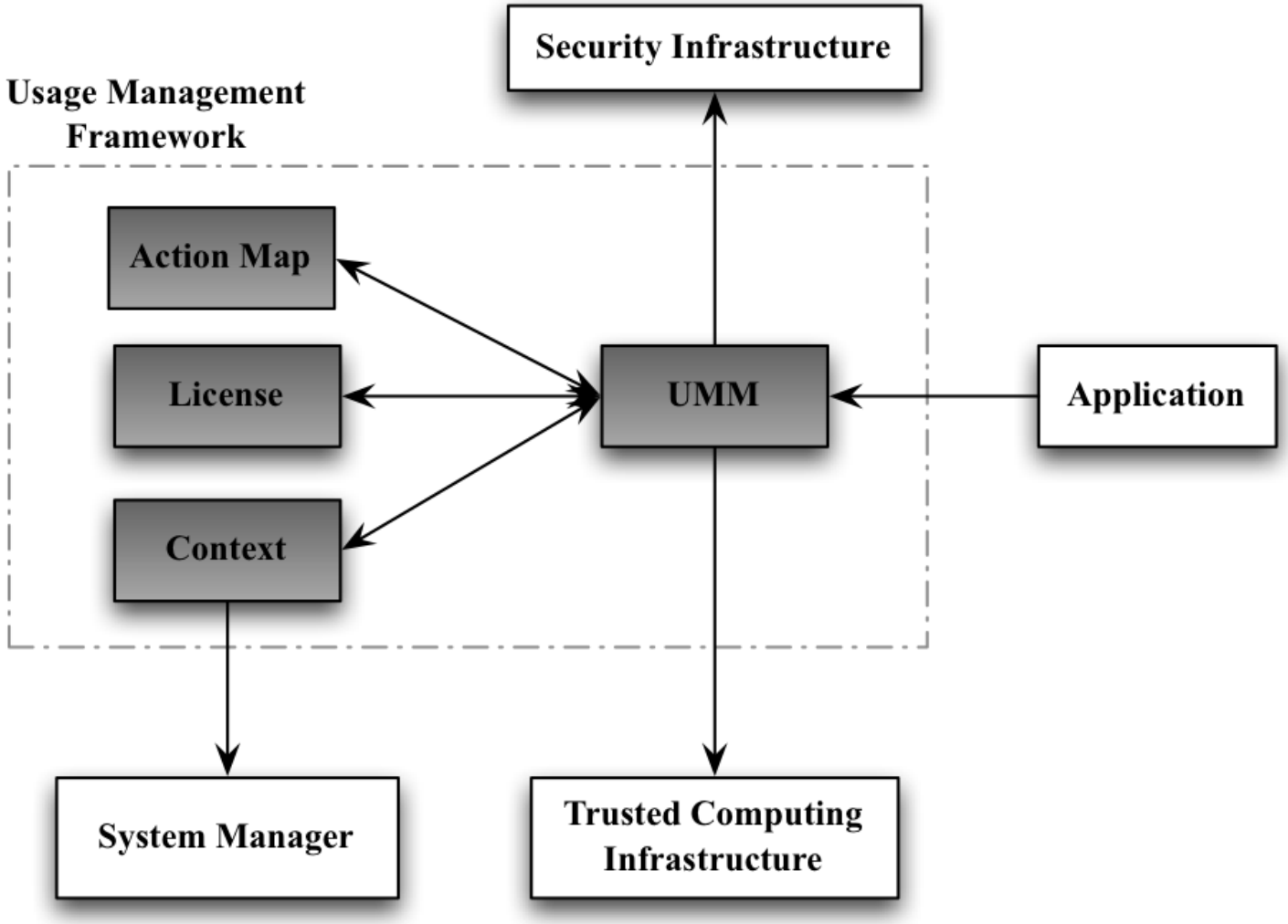
- Primary purpose: put a "wrapper" around usage policies, and provide an interface to access them.
- The interface supports:
  - License Querying – the policy interpretation mechanism will use this.
  - State Update – update, retrieve or reset license state
  - Transform – check compatibility with another license, or merge licenses.

UM Framework – Setup

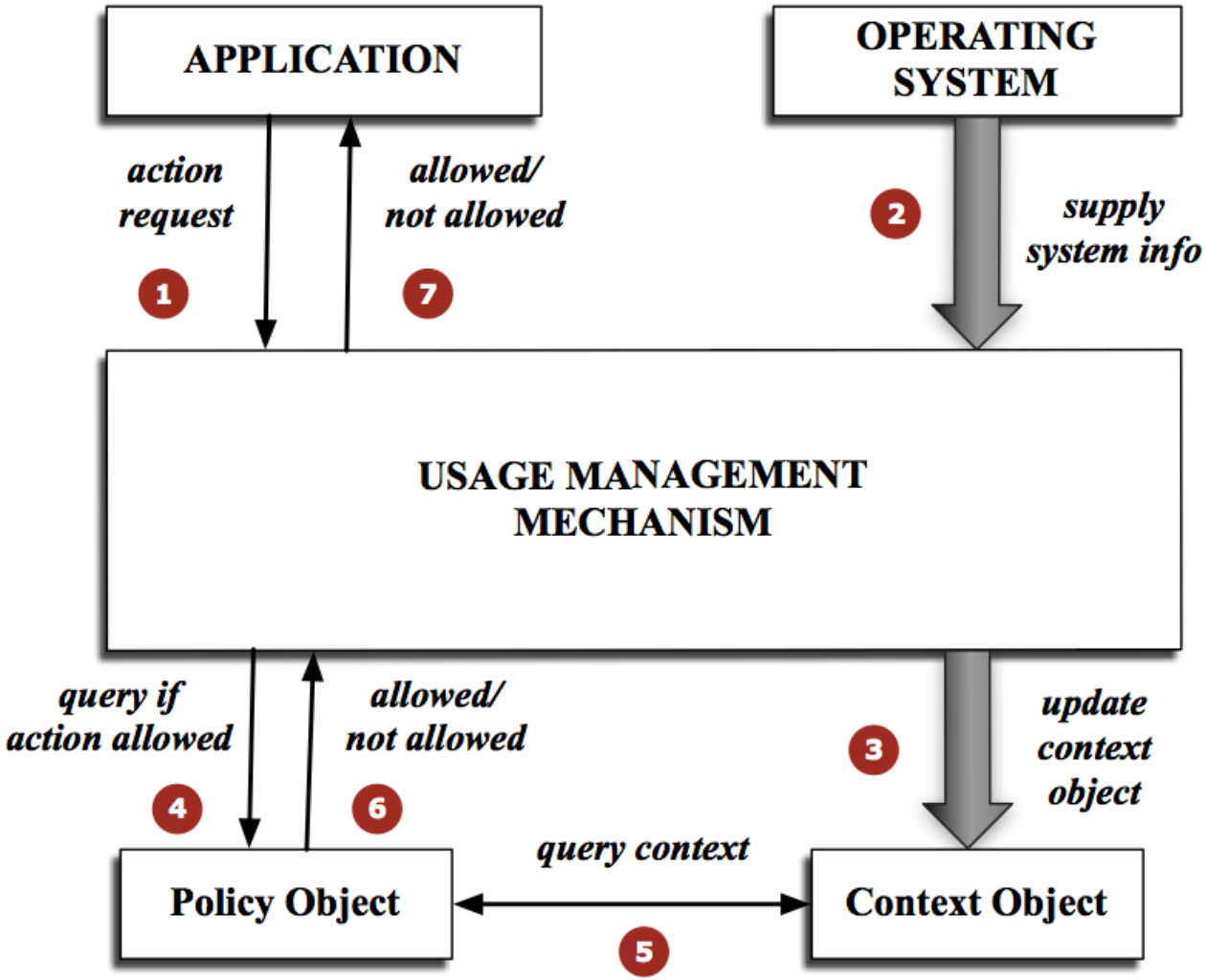


Note: the policy interpretation and enforcement mechanisms have been rolled up into the usage management mechanism.

UM Framework – Operation

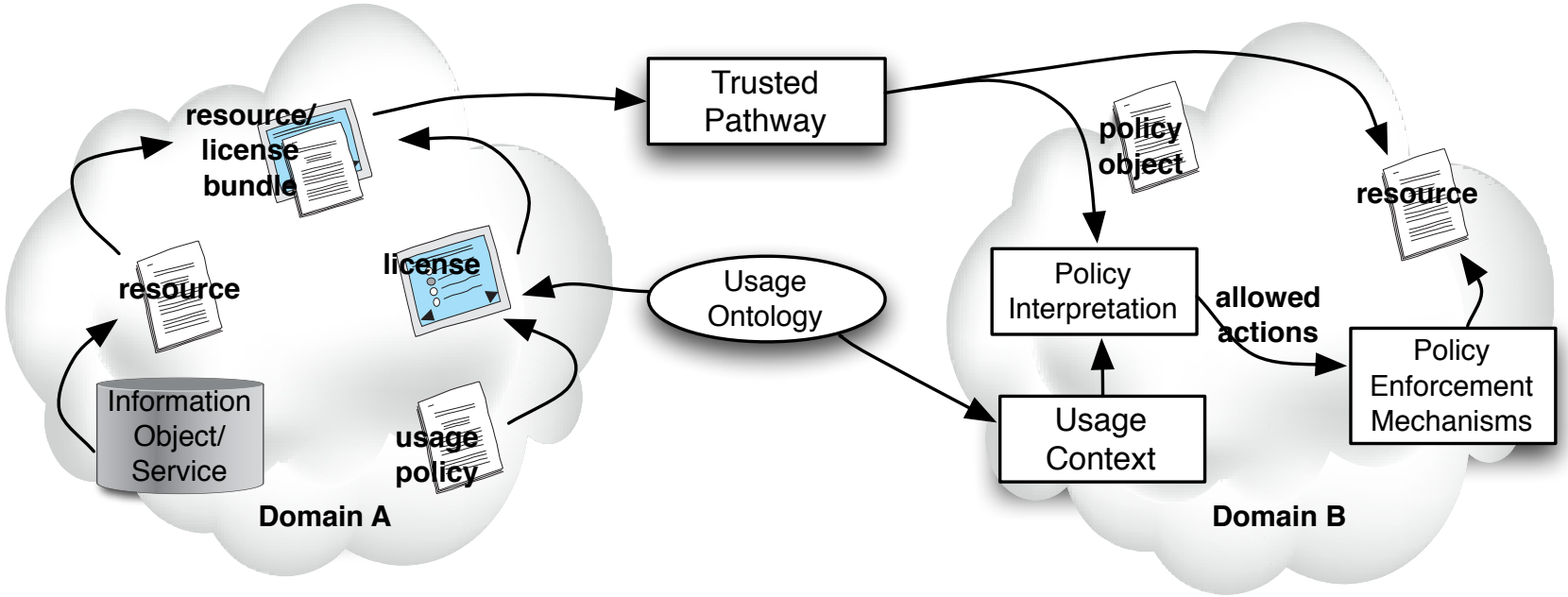


UM Framework – Operational Semantics



Note: the operational semantics of the framework are independent of syntax and semantics of any particular type of policy languages or logics.

UM Framework – Cross-domain



Note: this is one possible high-level architectural realization of the UM model that is meant to depict the pieces necessary in each domain, along with their dependencies. These are NOT data flows.

Applicable Scenarios

In order to demonstrate the flexibility of our approach, we will work through a series of possible scenarios with regard to sensitive information dissemination and use.

- **Go back to class!**
- **Changes roll downhill**
- **Operational Chaos**

Scenario 1: Go Back to Class!

Imagine a new cabinet secretary is appointed. Shortly thereafter, the new secretary unclassifies swathes of information - documents, presentations, data sets, you name it. Well, no secretary is forever, so eventually that secretary moves on and a new one comes in. That new secretary is appalled at the open flow of information in his agency and immediately clamps down, reclassifying oceans of documents.

Results of this change include:

- **Lost Time** — Agency personnel and contractors spend significant portions of their lives remarking documents...
- **Inaccuracy** — ...and despite their best efforts, humanity rears it's ugly head in the form of reams of mismarked information that takes years to sort out.

Scenario 1: Core Problem

So why did so many people need to sacrifice so much time to reclassify previously released information?

Hierarchical Mismatch.

Here, we have one one side a clear organizational hierarchy, spanning from the secretaries down to those who actually needed to clean up the messes those hypothetical secretaries made. On the other, we have a mass of essentially undifferentiated documents with embedded security metadata.

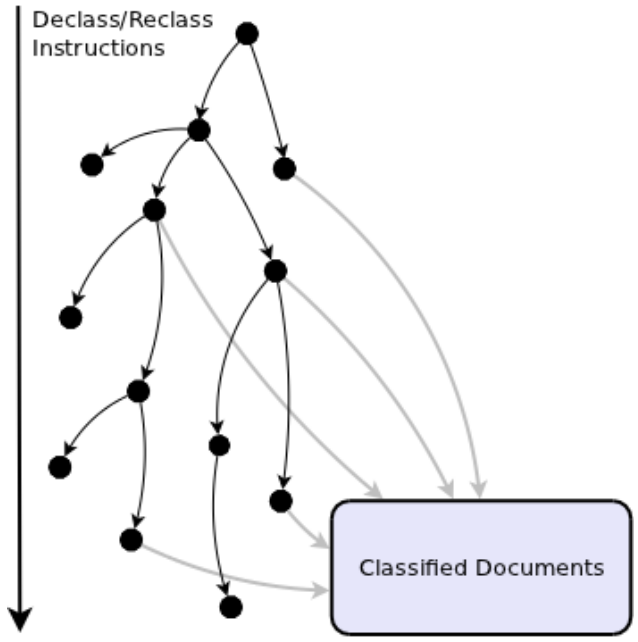
Scenario 1: Policy/Content Separation

One way to address this problem is by separating the **content** we're protecting from the **policies** describing how that content can be used.

This gives us **indirection** between policy and content, allowing us to be more flexible with how policies are managed.

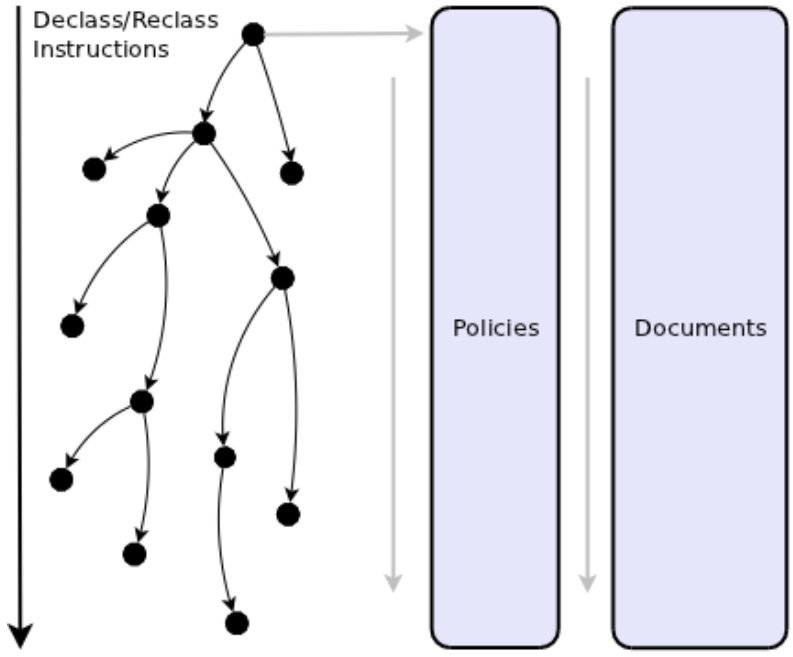
We could organize policies in a clear hierarchy, where content classifications can be defined one once, in policies within that hierarchy, rather than innumerable times within classified documents.

Hierarchical Policy Solution



Classification Propagation without Policies:

- **Everyone involved**
- **Lost time**
- **Wasted Funds**



Policy-based management:

- **Less change required**
- **Policy hierarchy does the work**

Scenario 2: Changes Roll Downhill

Scenario 1 shows the use of top-down control of a given group of policies. If top-down is effective, can we have similar advantages from a bottom-up use of a policy hierarchy?

Yes, we can.

Imagine a case where an organization:

- **... is working with an untrusted partner** — Say, for example, some coalition partner that is not entirely trusted with all the intelligence a given organization may have about a situation.
- **...must limit information access** — Not only by content, but perhaps by other factors like time.
- **...must be able to retract access** — Access to information needs to be retracted after a given time window.

Scenario 2: Core Problem

The problem here is somewhat different from that presented in the first Scenario.

Here we have:

- **Unequal information sharing** — Partners do not have the same access as organizational members.
- **Decisions made on the basis of content and context** — Decisions are made based on dynamic conditions, including the current reputation of the partner and the type of information accessed.
- **Decisions made close to the sharing scenario** — These decisions need to be made frequently by those most familiar with immediate use of the information and the context of use.

Scenario 2: Bottom-up Locality

This gives us a certain degree of bottom up locality.

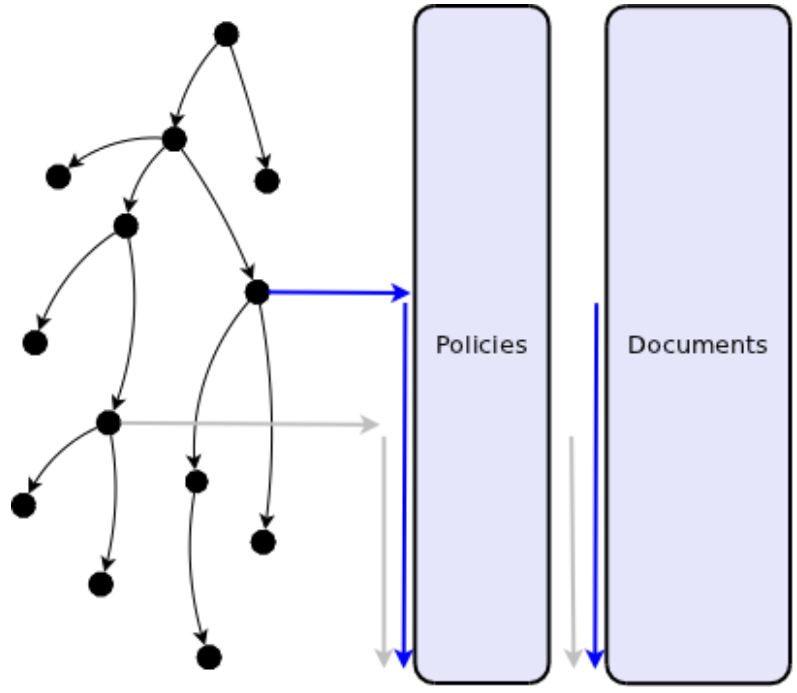
In this case, we have:

- **Decisions made close to operational environment** — Access decisions must be made frequently with knowledge of local context.
- **Locality of leadership vital** — Frequency of decisions and dynamic nature of context preclude decision-from-a-distance.

Scenario 2: Hierarchical Mirroring

Separation of **policy** from **content** at this level allows local decisions.

Decisions can be made at multiple levels based on local contexts, propagating to vital parties quickly.



Scenario 3: Operational Chaos

Under common conditions, technology and communications equipment is not very effective. These kinds of situations occur regularly under the stress of catastrophic conditions, for example.

These conditions generally share certain characteristics:

- **Dynamic operations** — The communication and computer operational networks have very dynamic topologies where groups are unexpectedly isolated and then reconnected.
- **System breakdown** — Systems tend to breakdown unexpectedly and catastrophically.
- **Centralized systems unavailable** — As a result of communication breakdowns, centralized systems become unavailable or unreliable.

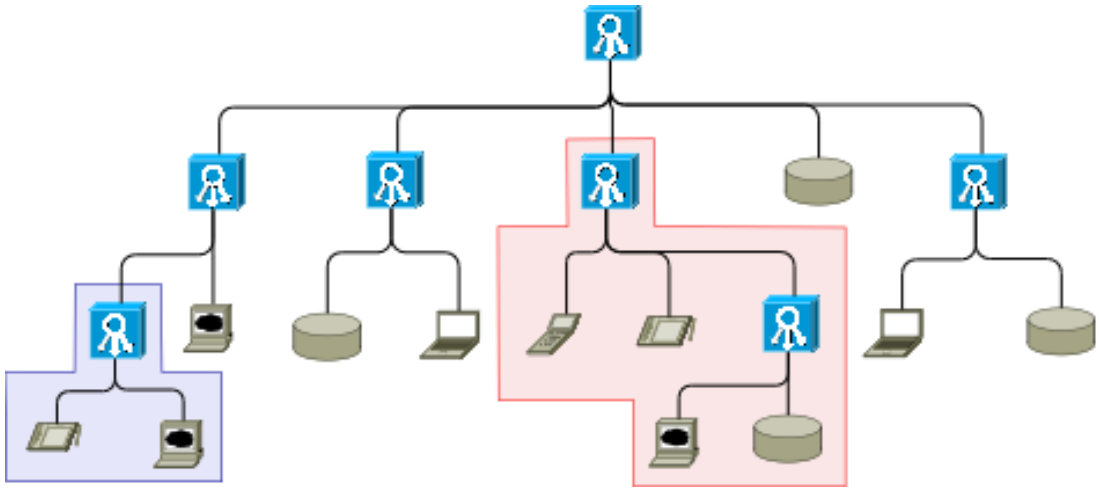
Scenario 3: Core Problem

As communications becomes less reliable, information becomes more valuable and highly demanded.

- **Need information in high stress environments** — Information demand increases under stress as those involved demand more situational awareness.
- **Communication capabilities highly variable** — Communications capabilities decrease as systems are destroyed or otherwise go offline leading to unreliable infrastructure.
- **Must operate under attack** — Even though the conditions are hostile, systems **must** provide some level of service and degrade gradually if at all in extreme environments.

Scenario 3: Information Management Cells

Under these conditions, end-to-end principles in system design can provide the appropriate scalability and reliability to support continued operations under stress.



End-to-end design provides:

- **Stalability, Reliability, Perfomance** — Complex nodes, simple core
- **Exploits Locality** — Centralized queries offline, local still available

Underlying Characteristics

What are the underlying principles that enable this flexibility?

- **Separation of Concerns** — Specifically, we separate **policy** from **content**.
- **End-to-End principle application** — As we move away from centralized cross-domain guards we push functionality closer and closer to content. We also begin to create distinct usage management cells, enabling *collaboration over centralization*.
- **Dynamic Context** — Context change radically in the context of artifact use. Environmental conditions that forbid access to information content can change gradually or quickly, leading to conditions under which that content must be widely accessed.

Separation of Concerns

How does separation of concerns help us?

Why does it help in other domains? Say Javascript/CSS/HTML?

- **Rates of change** — Some things change at different rates. In web development, content (HTML) may stay the same, but the presentation (CSS) of the content will change. Likewise, the presentation may be static, but the behavior (Javascript) may need to evolve.
- **Roles** — Different roles may be suitable for different types of work. Here, content may be edited near constantly, while the presentation of that content may be static.

In usage management scenarios, we have similar issues.

End-to-End Principles

How do end-to-end principles help us?

End-to-End principles were originally outlined to help design forerunner networks to what we now know as the internet.

Essentially, they encourage simplicity in the core switching fabric of a network with application complexity pushed out to nodes in a network graph.

Although recent approaches have abandoned these ideas, they have done so at the price of loss of scalability, reliability, and performance.

Dynamic Context

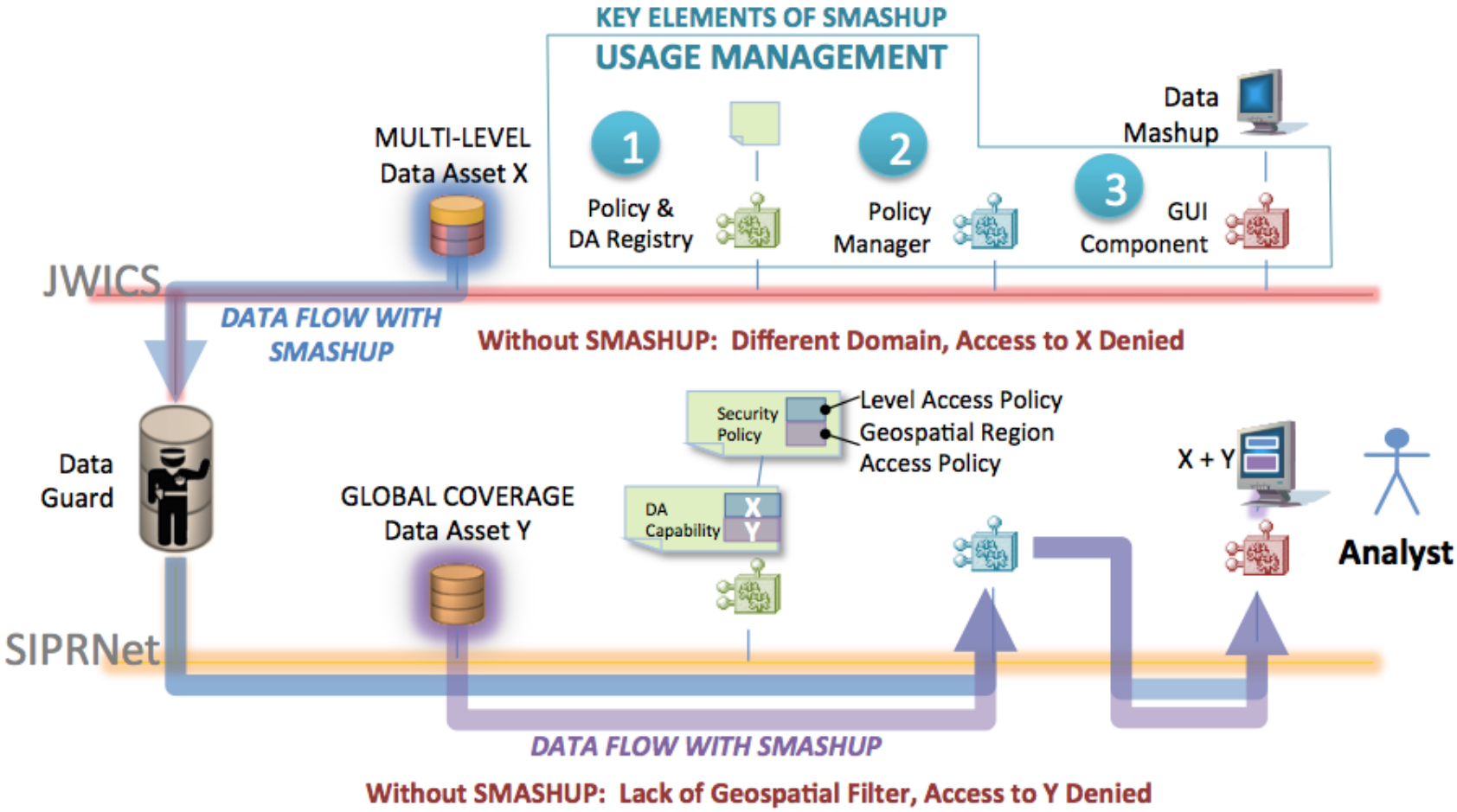
How does recognition of dynamic contexts help us?

We know that environments and situations change, and change rapidly. Not recognizing this leads to brittle systems that are prohibitively expensive to maintain, if they work at all.

With dynamic context support we can:

- **Retract access based on conditions** — Simply recognizing a larger context enables the system to automatically restrict access to sensitive material based on changing conditions.
- **Map activities to actions** — Reading a document may in fact not be a single activity, but rather a sequence of actions (lookup, retrieve, save, open, display) that are managed separately in certain situations.

Prototype Demonstration



Note: the guard functionality in this demonstration was not a separate module, and was provided as part of usage management framework.

Prototype Demonstration – Capabilities

The Phase I prototype demo was used to exhibit the following capabilities:

- Licenses that contained policies expressing both access and usage control rules were constructed to enable persistent information security over data elements.
- The concept of a license being interpreted in real-time according to the current context was exhibited.
- The capability of real-time aggregation of licenses, and reasoning over the resulting combined policies was demonstrated. This included situations were polices precluded certain data elements from being mixed together, and the reasoning about this occurred in real-time, as the mashup was being constructed.
- Information sources belonging to different realistic DoD security domains, and the sharing of objects between them, were simulated (there was no communication through any type of trusted pathway though), taking into account the roles of the personnel involved, the devices they were using, etc.

Prototype Demonstration – Limitations

The Phase I prototype demo was limited in the following ways:

- The demo did not provide a formal ontology for the domain. Rather this was "hard- coded" into the demo.
- User authentication was simulated.
- Although access to data sources was dynamically computed according to policy, the data sources themselves were static. I.e., map data was not generated by a database query, but rather was stored as fixed images.
- It did not provide a mechanism for generating licenses using an underlying ontology; XML licenses were constructed "by hand."
- The logic associated with reasoning over license aggregation was very simplistic, essentially a logical ANDing over the license terms.
- The semantics of the licenses were also rudimentary, e.g., only permissions were considered, not obligations, usage history, etc.

Prototype Demonstration – Limitations

- Reasoning over the interoperability of licenses between different domains was not considered. With a formal notion of interoperability, one context (domain) can query another, in order to determine if the other domain is able to interpret a particular policy.
- The demo did not exhibit the capabilities of dynamic interpretation, where one activity specified in a license could lead to different actions depending upon the environment in which it is being interpreted.
- The security of the system was not rigorously evaluated against any type of threat model. If we can formally capture a threat model, then we can better understand how usage management plays in the architecture. (AFRL help would be appreciated here.)
- The trusted pathway was not a stand-alone entity. We implicitly supplied one through our usage management mechanism, thereby providing a distributed cross-domain sharing solution.
- Query capabilities over usage scenarios (i.e., what led to a particular rule in a combined license) were not provided.

Conclusions and Future Work

We have developed initial systems embodying these concepts.

Currently, additional distributed systems are in development that will apply these ideas to XML and HTML/RDFa content using HTTP-centric protocols.

Additional future areas of study include migration of these ideas into the core of cloud systems (e.g. Eucalyptus), application to addtional domains (e.g. medicine)