A Domain Specific Language for Usage Management

Christopher C. Lamb, Pramod A. Jamkhedkar, Mathew P. Bohnsack, Viswanath Nandina, Gregory L. Heileman

Department of Electrical and Computer Engineering
University of New Mexico

October 21, 2011

THE UNIVERSITY of NEW MEXICO

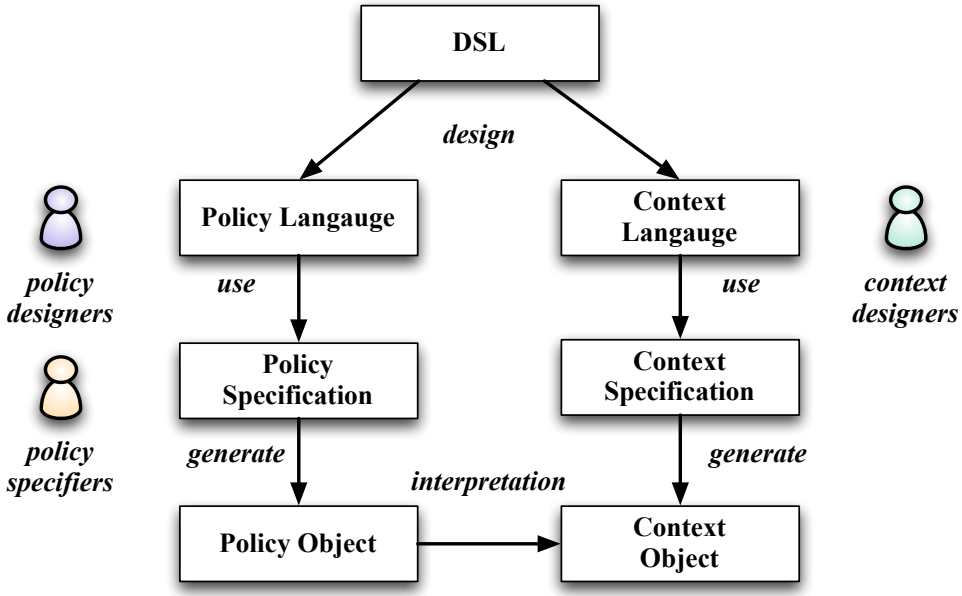
Outline

Introduction

Intro content

Design — Notional Use

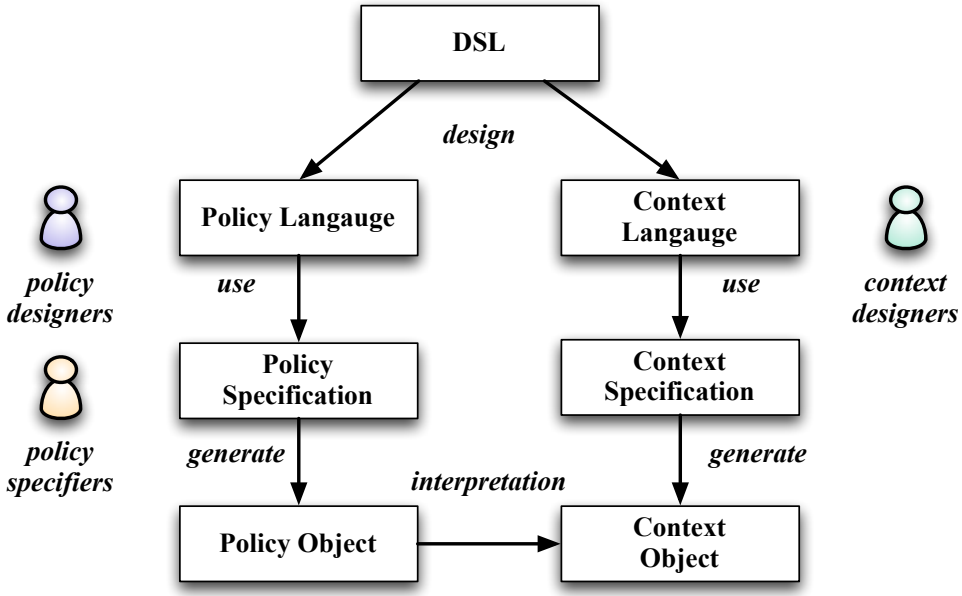
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

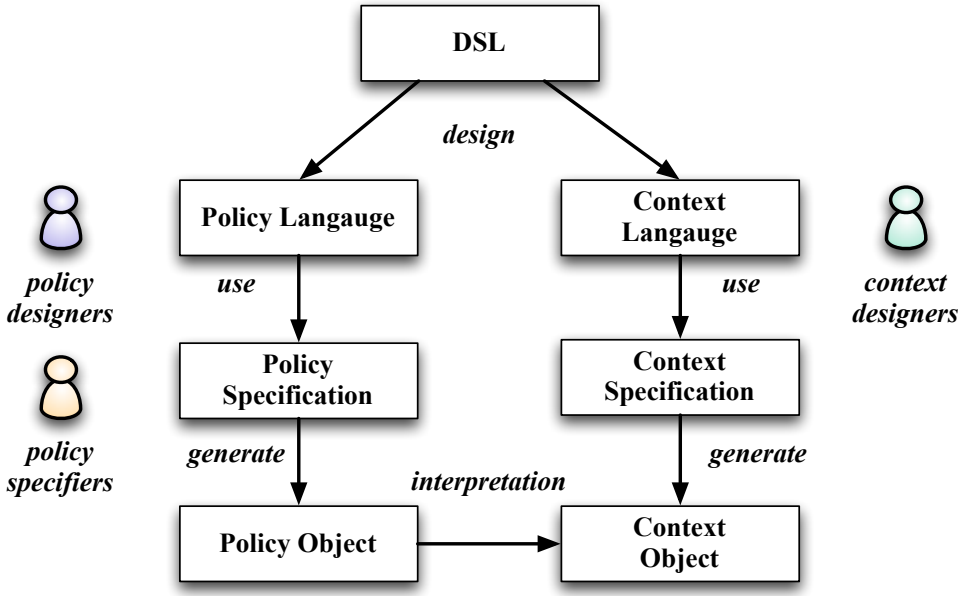
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

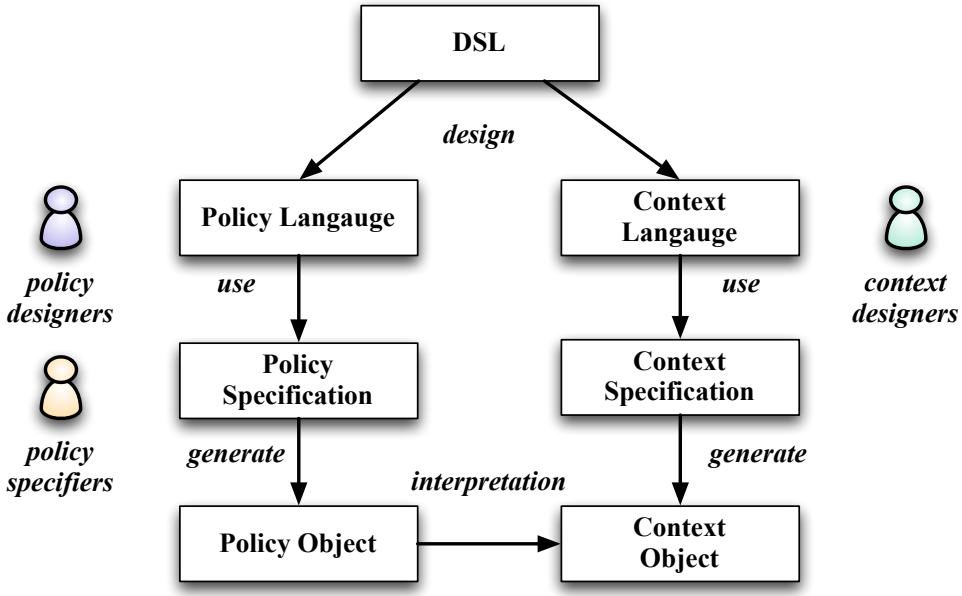
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

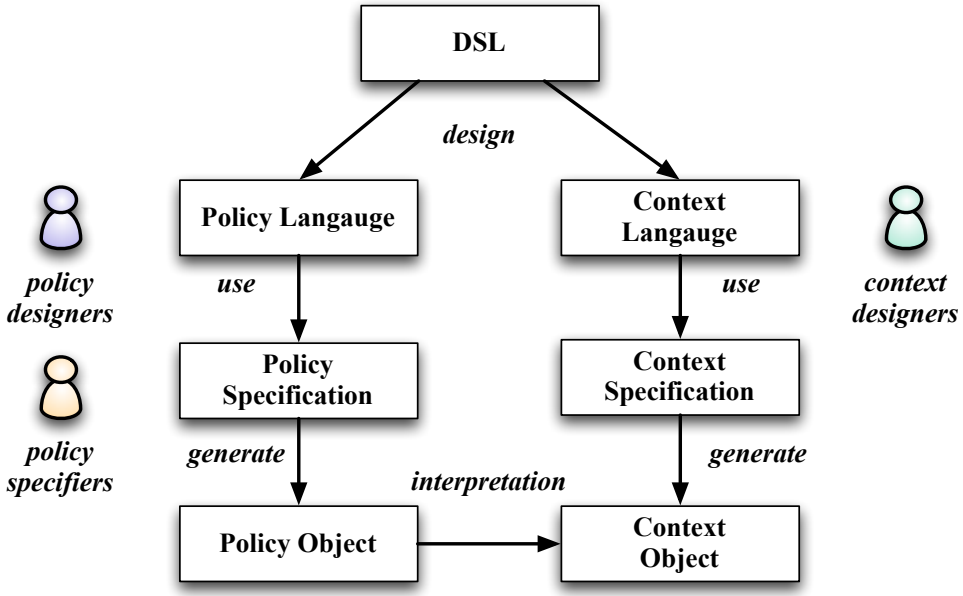
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

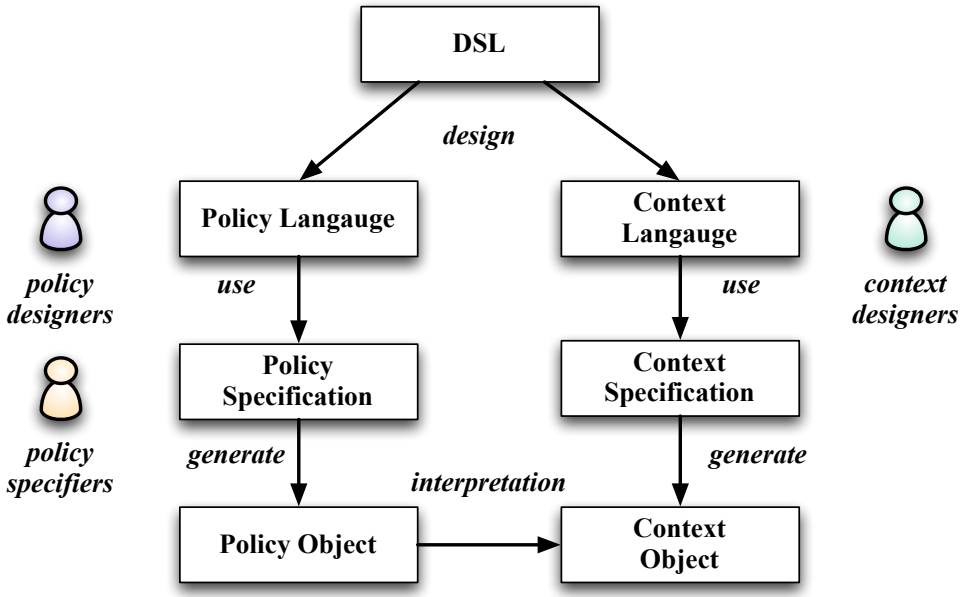
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

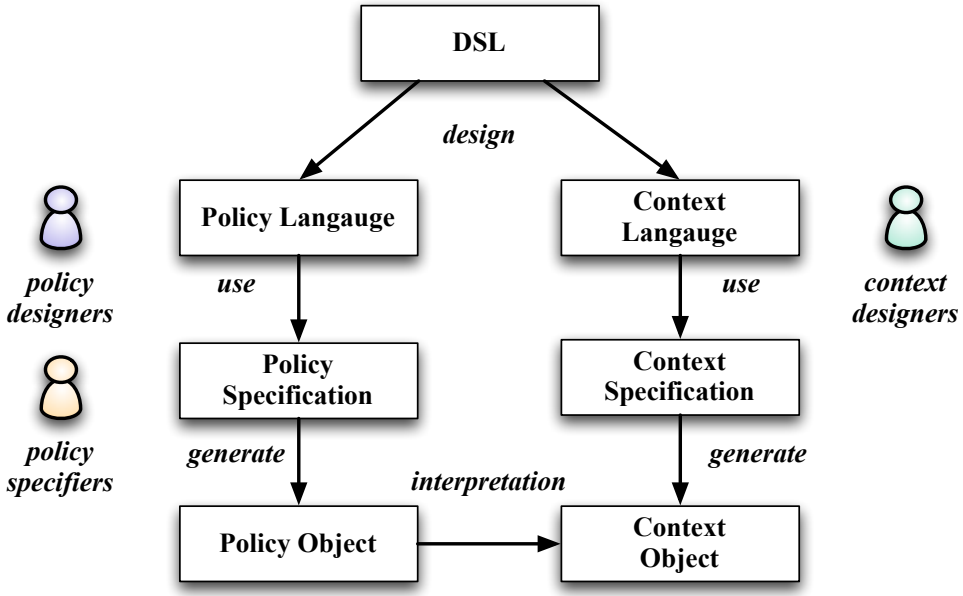
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

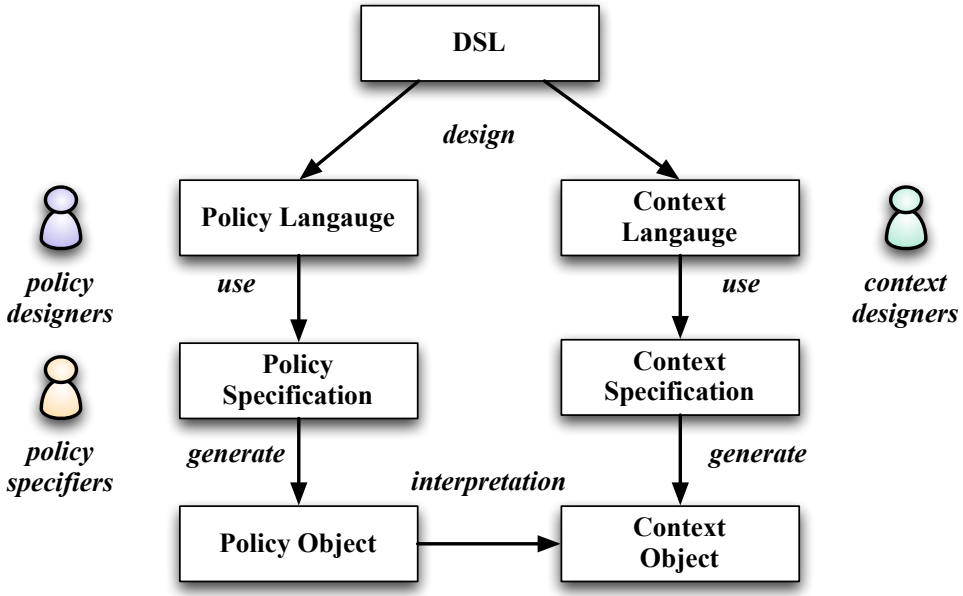
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Notional Use

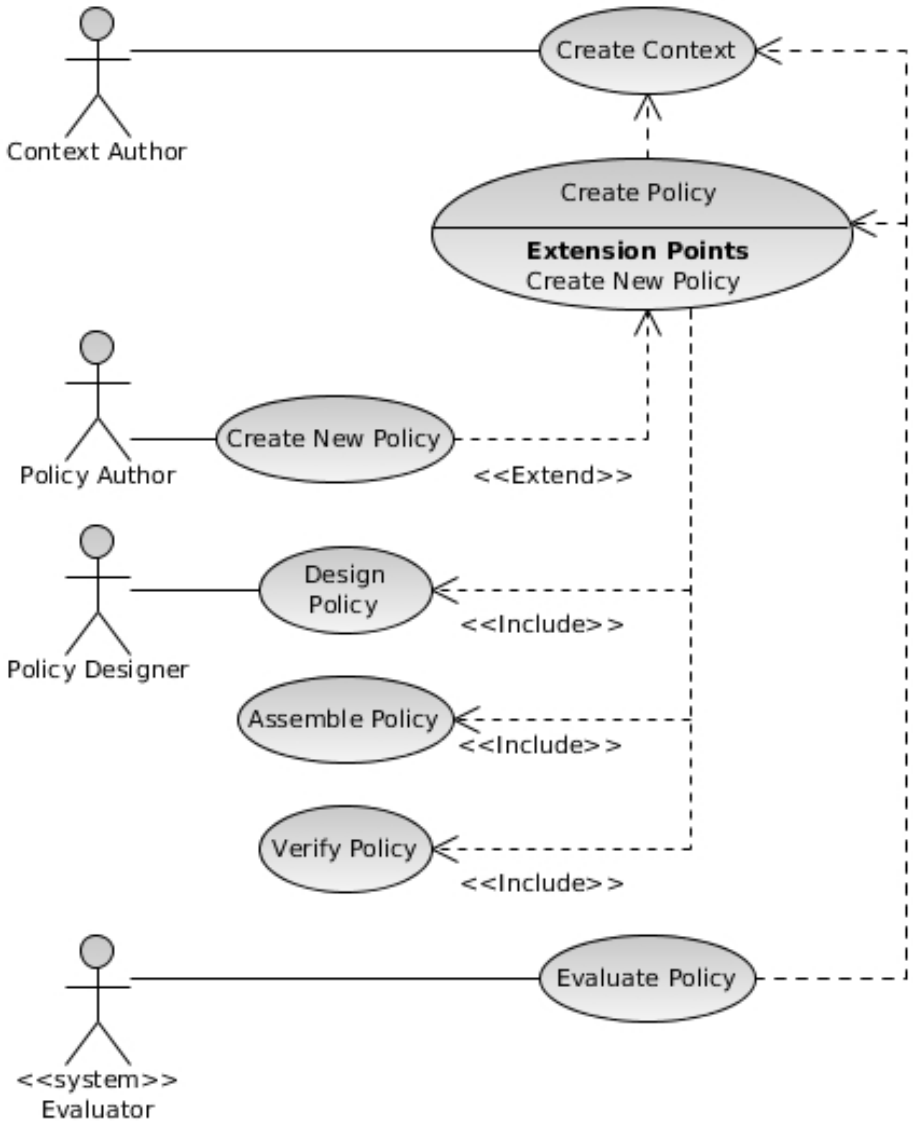
Notional Use:



- *DSL* — Domain specific language
- *Policy Language* — Language elements specific to *policy*
- *Context Language* — Language elements specific to *context*
- *Policy Specification* — Actual specification of policy
- *Context Specification* — Specification of context requirements
- *Policy Object* — An object embodying policy created from the DSL
- *Context Object* — An object containing context

Design — Use Cases

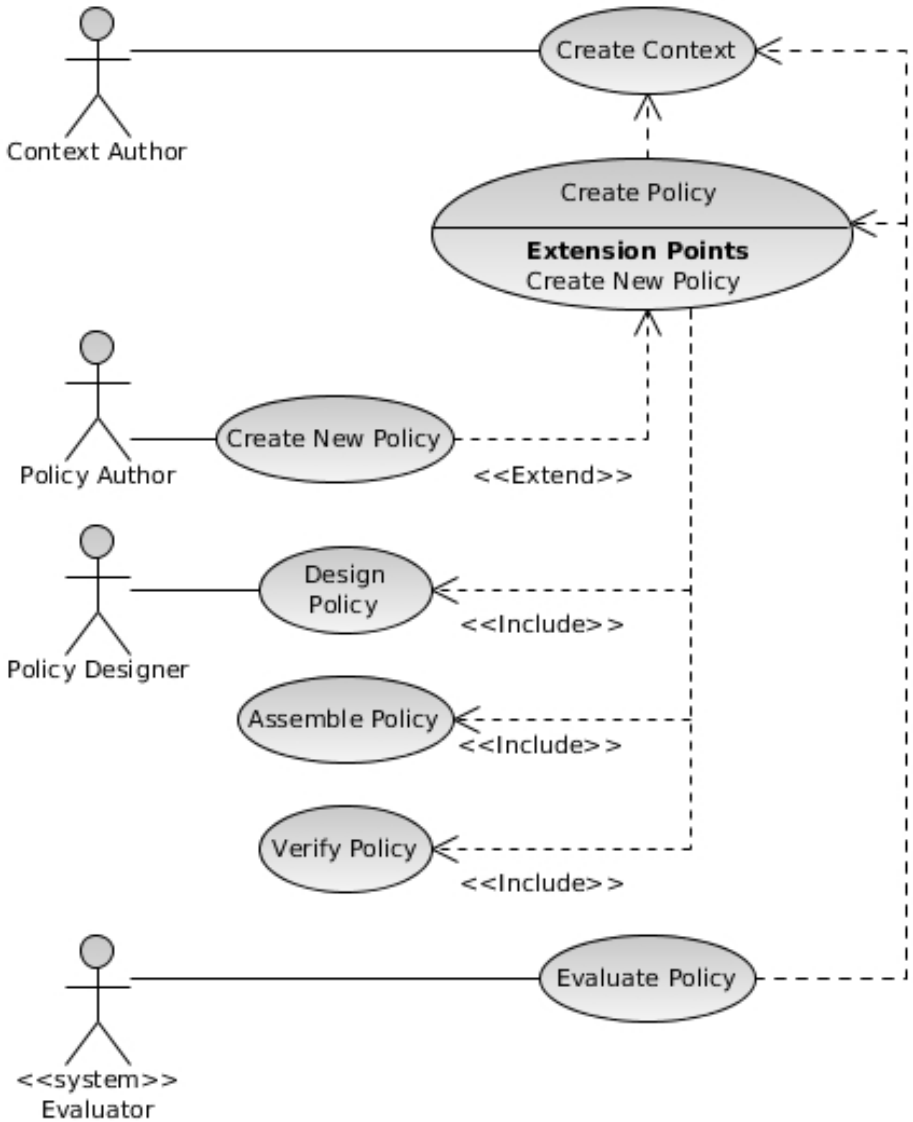
Use Cases:



- *Create Context* — Prior to creating a policy, the context in which that policy will be evaluated must be defined.
- *Create Policy* — A designer creates a new type of policy, embodied by specific extension elements or semantic constraints over existing elements. An author will use these to create an instance of a policy.
- *Evaluate Policy* — The policy is evaluated with a context.

Design — Use Cases

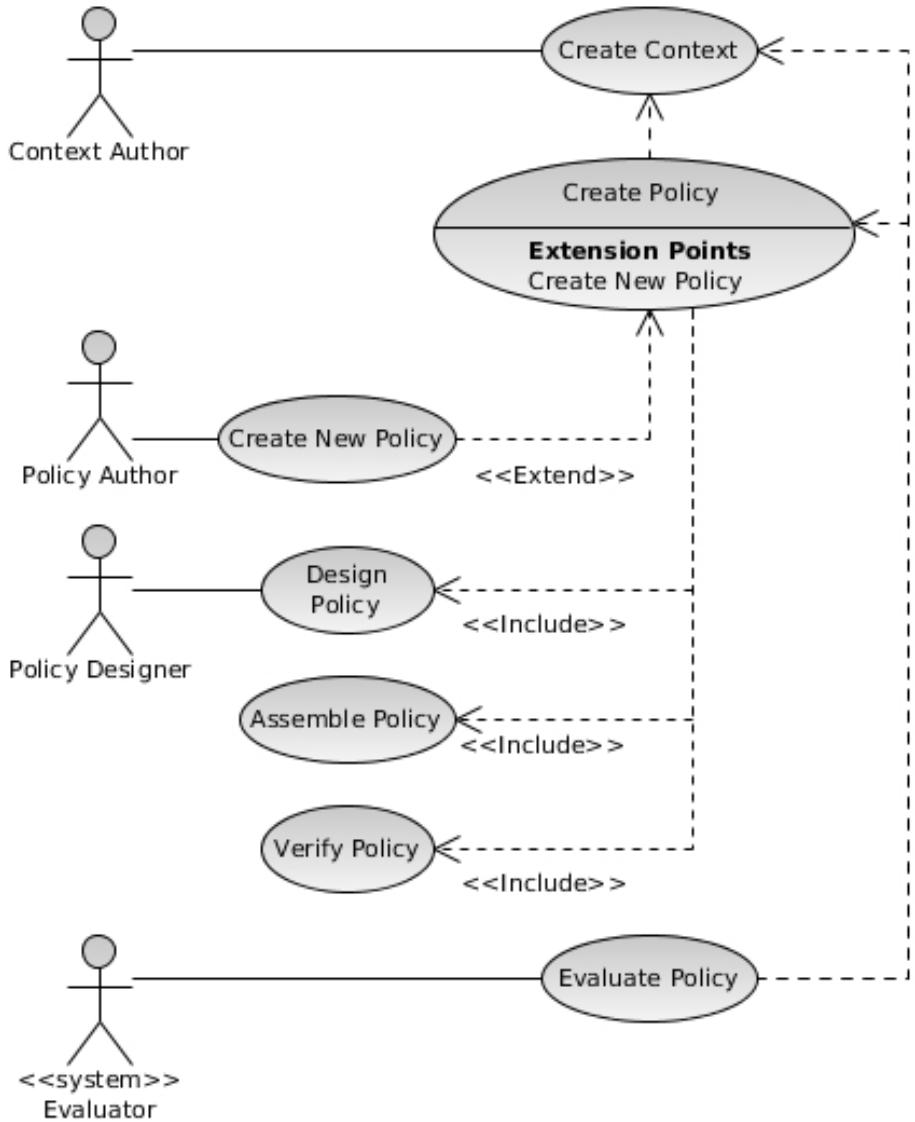
Use Cases:



- *Create Context* — Prior to creating a policy, the context in which that policy will be evaluated must be defined.
- *Create Policy* — A designer creates a new type of policy, embodied by specific extension elements or semantic constraints over existing elements. An author will use these to create an instance of a policy.
- *Evaluate Policy* — The policy is evaluated with a context.

Design — Use Cases

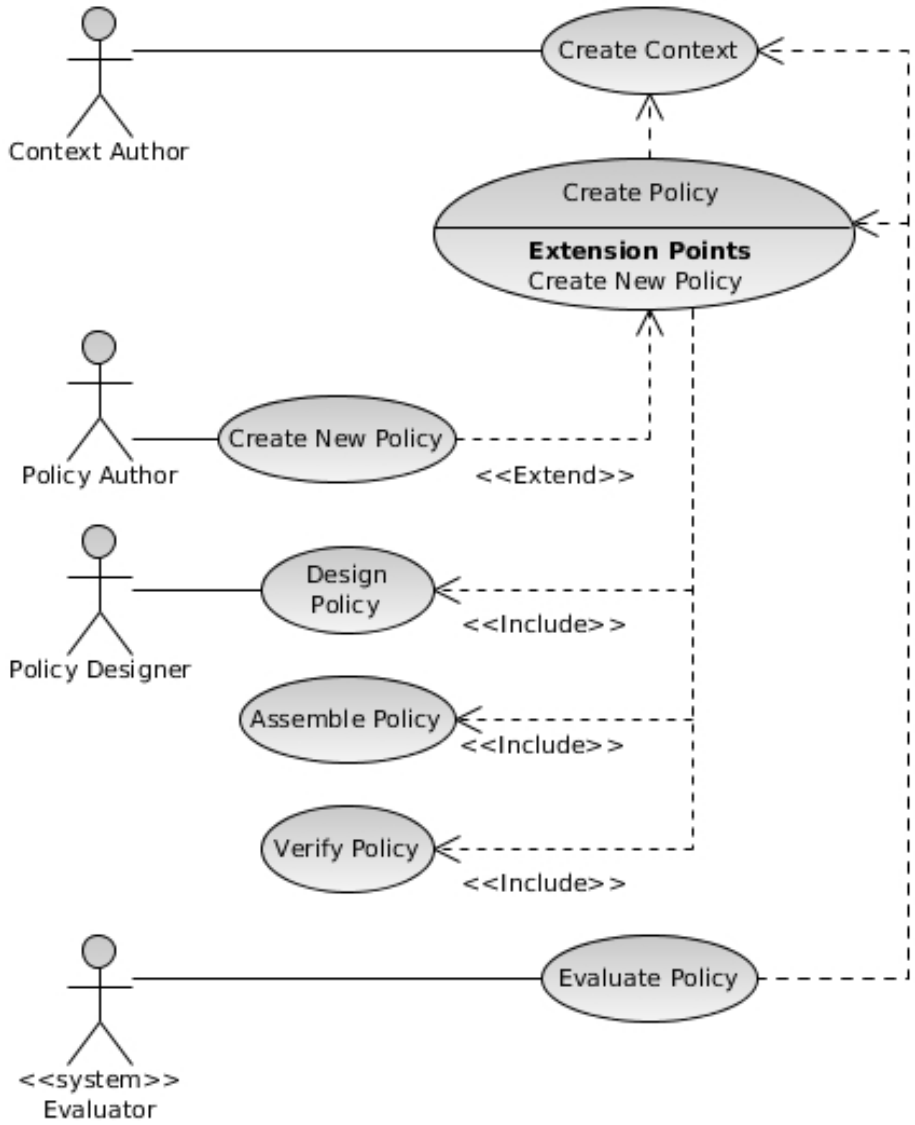
Use Cases:



- *Create Context* — Prior to creating a policy, the context in which that policy will be evaluated must be defined.
- *Create Policy* — A designer creates a new type of policy, embodied by specific extension elements or semantic constraints over existing elements. An author will use these to create an instance of a policy.
- *Evaluate Policy* — The policy is evaluated with a context.

Design — Use Cases

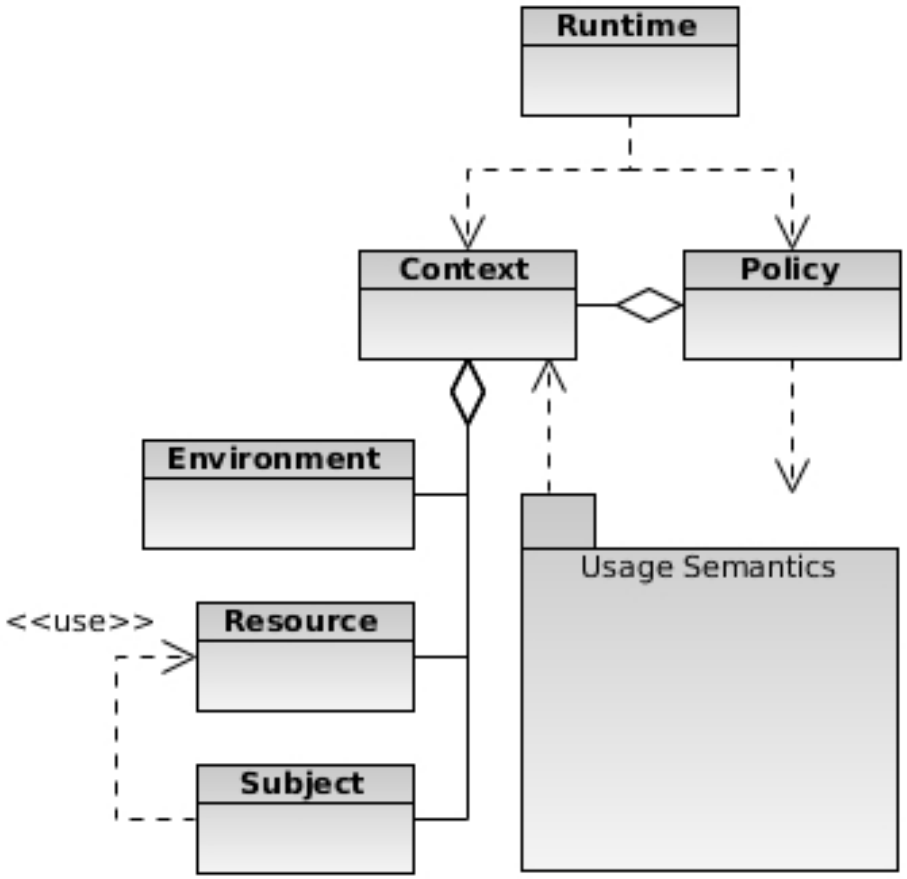
Use Cases:



- *Create Context* — Prior to creating a policy, the context in which that policy will be evaluated must be defined.
- *Create Policy* — A designer creates a new type of policy, embodied by specific extension elements or semantic constraints over existing elements. An author will use these to create an instance of a policy.
- *Evaluate Policy* — The policy is evaluated with a context.

Design — Domain Model

Domain Model:



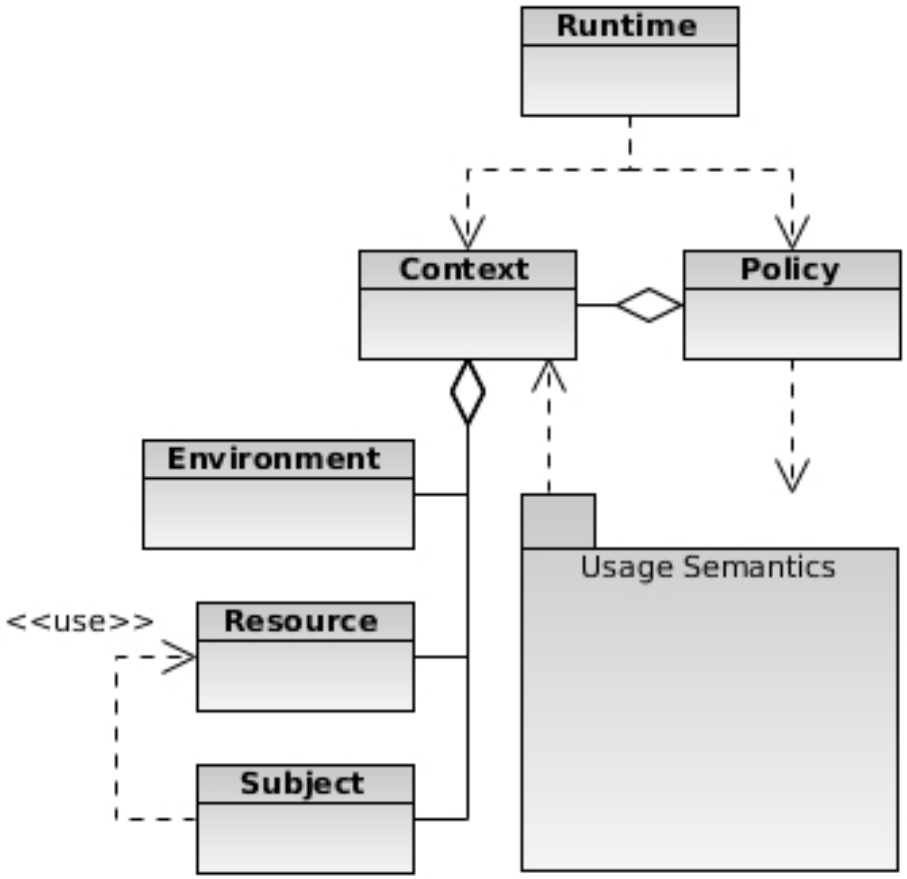
The *Runtime* accesses and activates a *policy* and manages a *context* to which the policy is given a reference.

The *context* has access to information about the *environment*, *resource* managed, and the *subject* using the *resource*.

Interactions are described by specific *usage semantics* embodied in the *policy*.

Design — Domain Model

Domain Model:



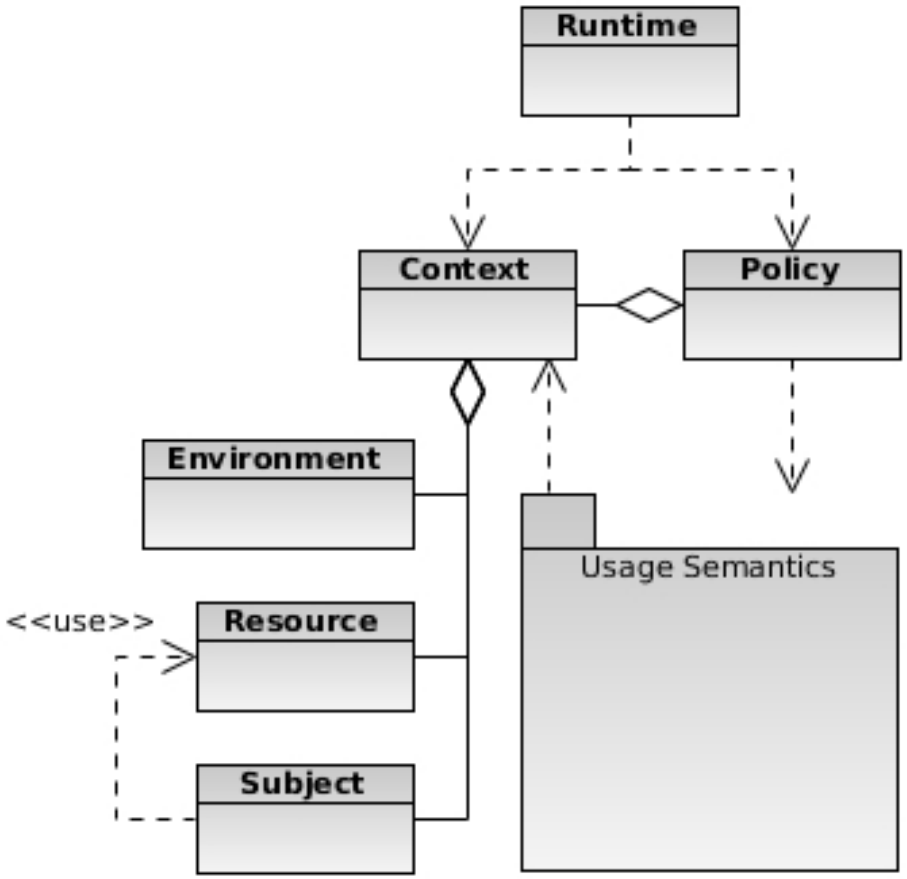
The *Runtime* accesses and activates a *policy* and manages a *context* to which the policy is given a reference.

The *context* has access to information about the *environment*, *resource* managed, and the *subject* using the *resource*.

Interactions are described by specific *usage semantics* embodied in the *policy*.

Design — Domain Model

Domain Model:



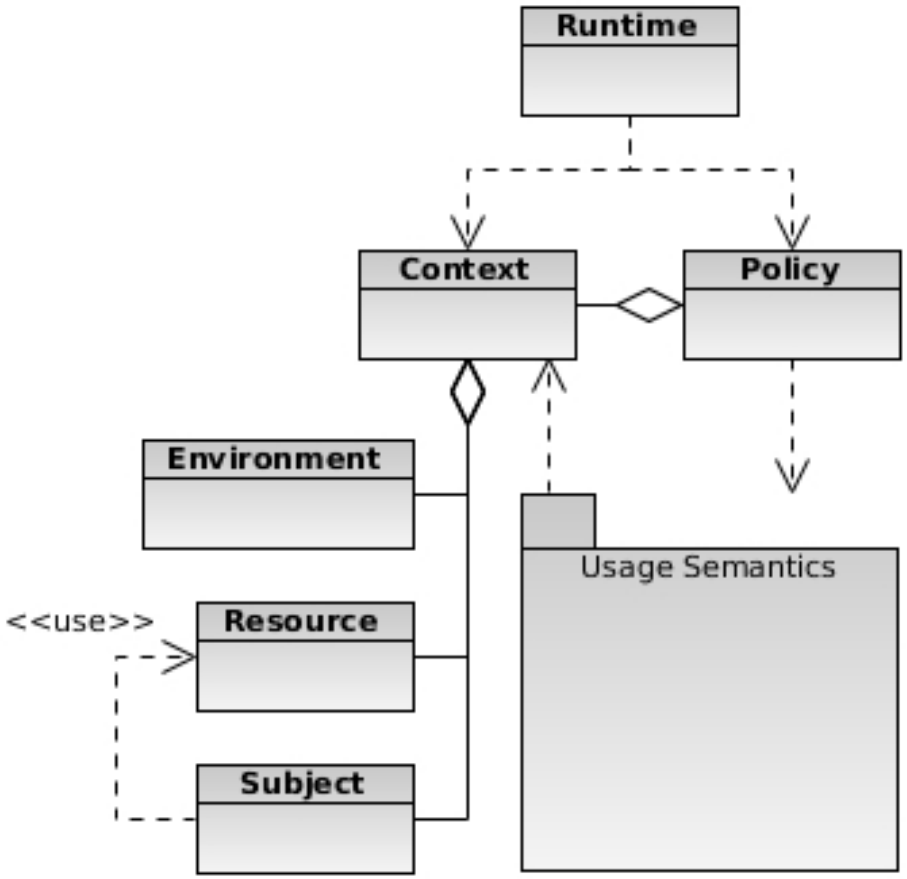
The *Runtime* accesses and activates a *policy* and manages a *context* to which the policy is given a reference.

The *context* has access to information about the *environment*, *resource* managed, and the *subject* using the *resource*.

Interactions are described by specific *usage semantics* embodied in the *policy*.

Design — Domain Model

Domain Model:



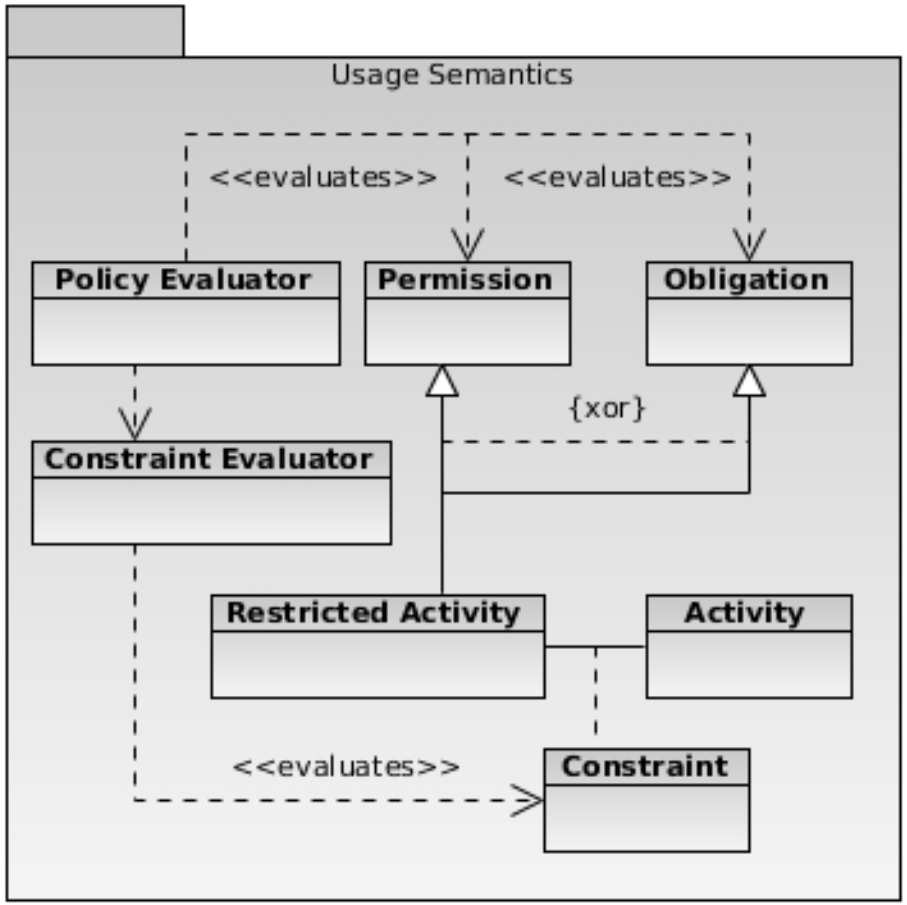
The *Runtime* accesses and activates a *policy* and manages a *context* to which the policy is given a reference.

The *context* has access to information about the *environment*, *resource* managed, and the *subject* using the *resource*.

Interactions are described by specific *usage semantics* embodied in the *policy*.

Design — Usage Semantics

Usage Semantics:



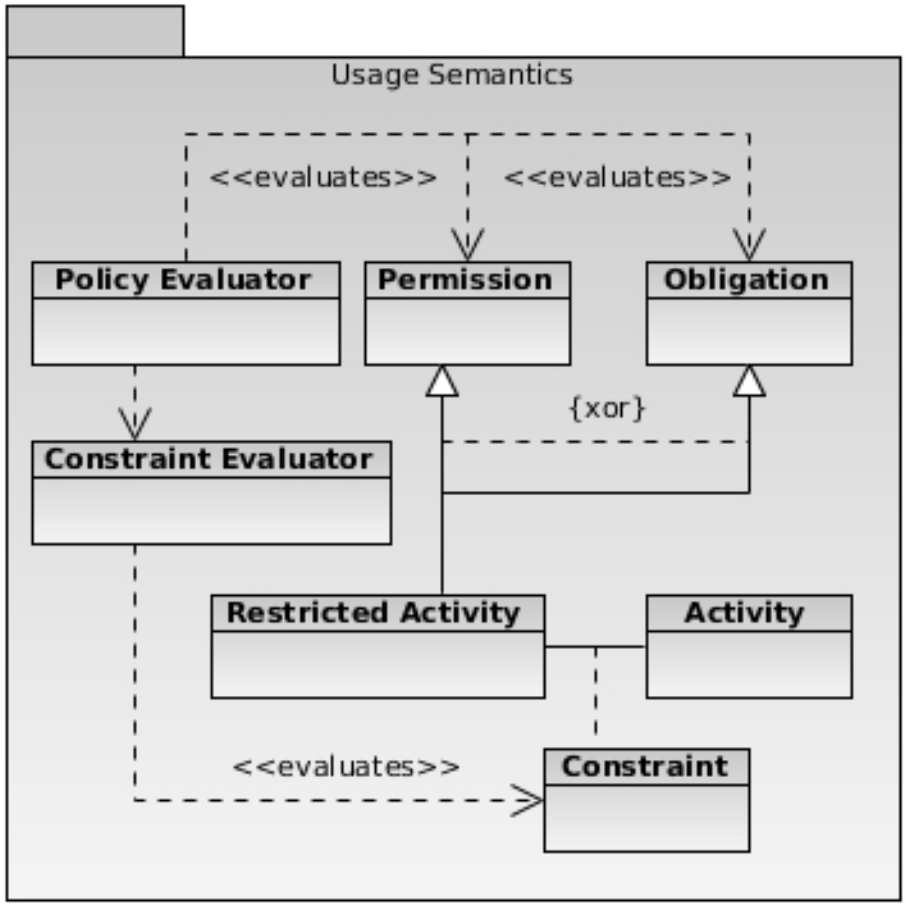
A *policy evaluator* examines and rectifies both *permissions* and *Obligations*.

A *restricted activity* is a specialization of either a *permission* or *obligation*, and is associated with a specific *activity*.

The association between an *activity* and a *restricted activity* is embodied by a *constraint*, which is evaluated by a *constraint evaluator*.

Design — Usage Semantics

Usage Semantics:



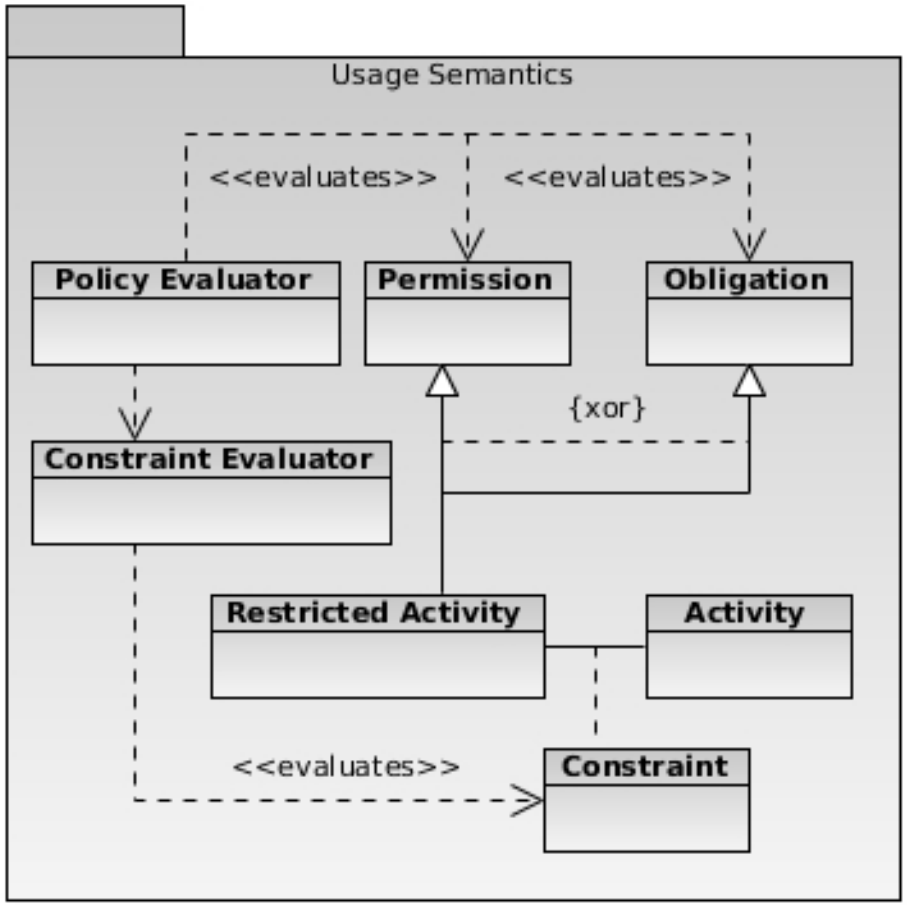
A *policy evaluator* examines and rectifies both *permissions* and *Obligations*.

A *restricted activity* is a specialization of either a *permission* or *obligation*, and is associated with a specific *activity*.

The association between an *activity* and a *restricted activity* is embodied by a *constraint*, which is evaluated by a *constraint evaluator*.

Design — Usage Semantics

Usage Semantics:



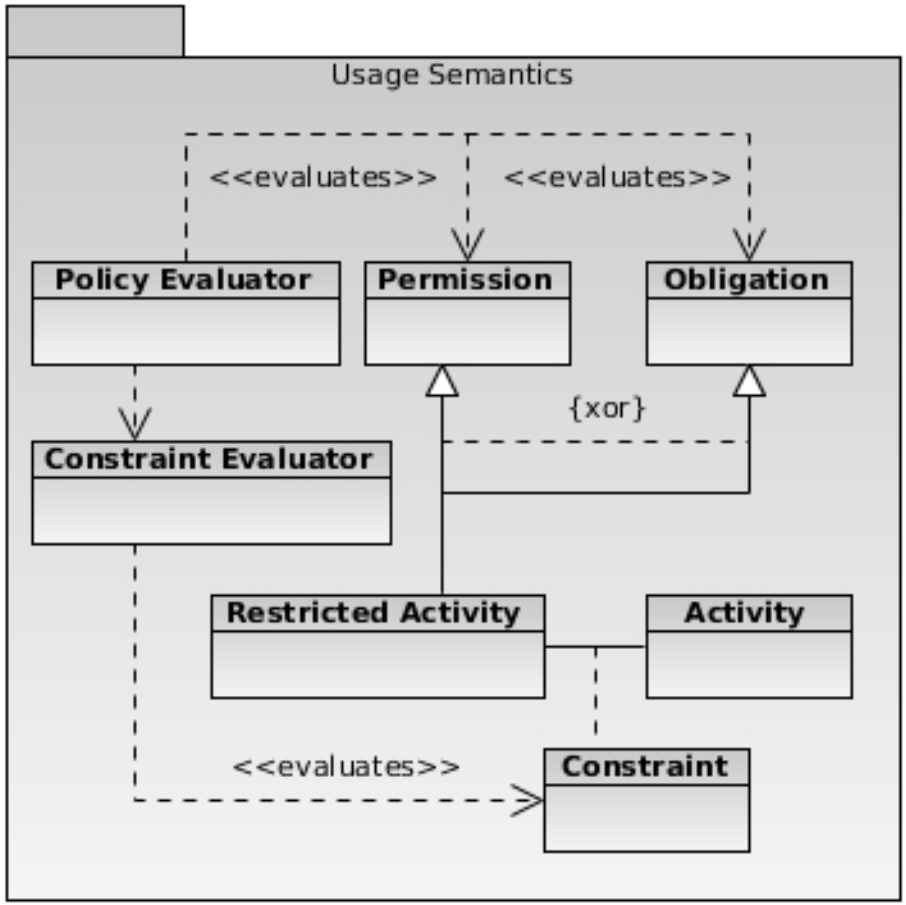
A *policy evaluator* examines and rectifies both *permissions* and *Obligations*.

A *restricted activity* is a specialization of either a *permission* or *obligation*, and is associated with a specific *activity*.

The association between an *activity* and a *restricted activity* is embodied by a *constraint*, which is evaluated by a *constraint evaluator*.

Design — Usage Semantics

Usage Semantics:



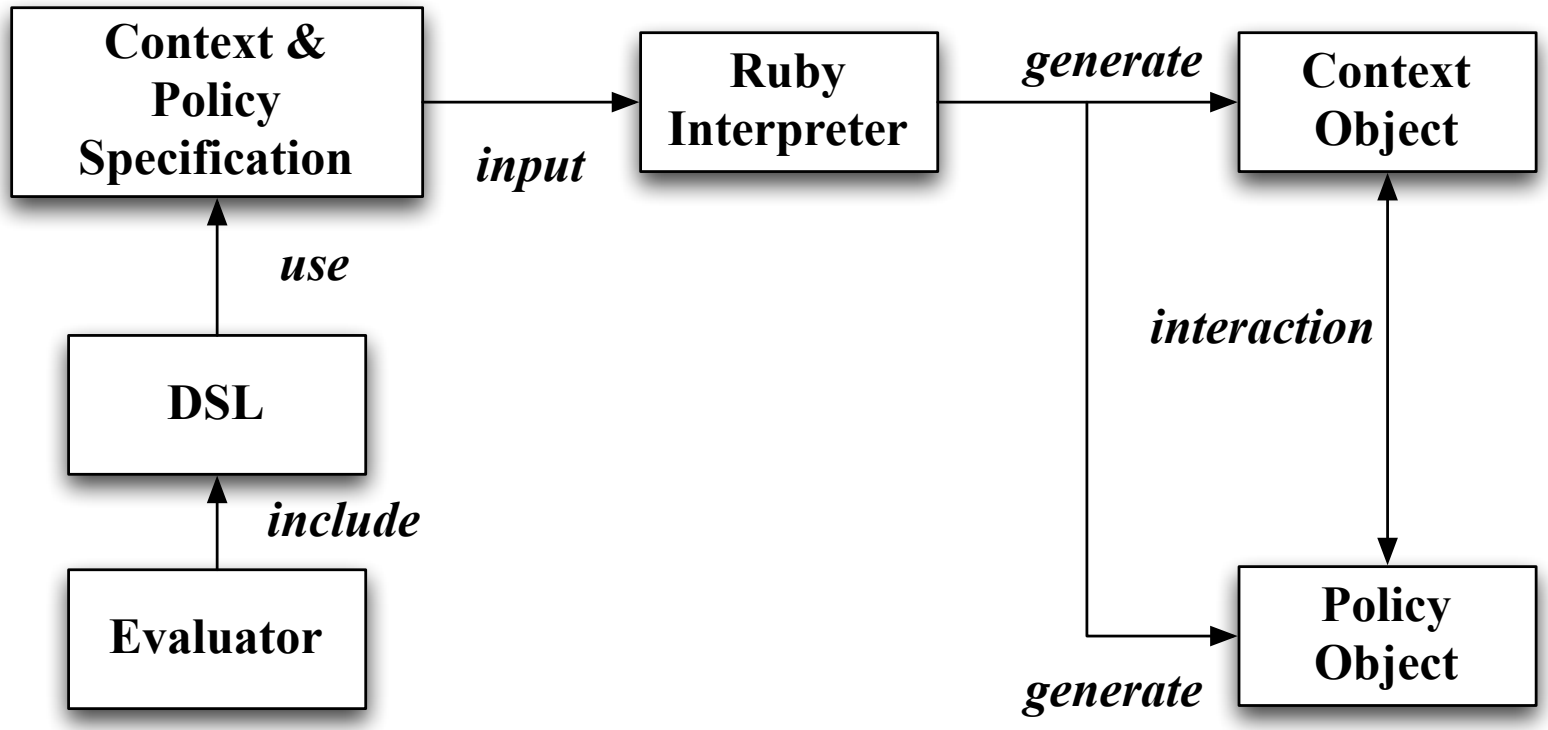
A *policy evaluator* examines and rectifies both *permissions* and *Obligations*.

A *restricted activity* is a specialization of either a *permission* or *obligation*, and is associated with a specific *activity*.

The association between an *activity* and a *restricted activity* is embodied by a *constraint*, which is evaluated by a *constraint evaluator*.

Implementation — Lifecycle

Typical DSL Lifecycle:



Implementation — Attributes

Context			
Entity	**Property (p)**	**Domain (D_p)**	**Functions (F_p)**
Environment (E)	OperatingSystem	{Windows, OSX, SELinux}	equatable
	Device	{Workstation, Handheld, Blackberry, Terminal}	equatable
	SecurityDomain	{ ABNet, SECNet, TELNet, OMNINet}	comparable
Subject (S)	SecurityClearance	{Top Secret, Secret, Confidential}	comparable
	Project	{Zebra, Yuma, Lion}	equatable
	Role	{Alpha, Beta, Delta}	equatable
Resource(R)	SecurityClassification	{ Top Secret, Secret, Confidential, Unclassified}	comparable

Environment (E):
Operating System $\rightarrow$ {Windows, OSX, SELinux} $\rightarrow$ equatable
Device $\rightarrow$ {Workstation, Handheld, Blackberry, Terminal} $\rightarrow$ equatable
Security Domain $\rightarrow$ {ABNet, SECNet, TELNet, OMNINet} $\rightarrow$ comparable
Subject(S):
SecurityClearance $\rightarrow$ {Top Secret, Secret, Confidential} $\rightarrow$ comparable
Project $\rightarrow$ {Zebra, Yuma, Lion} $\rightarrow$ equatable
Role $\rightarrow$ {Alpha, Beta, Delta} $\rightarrow$ equatable
Resource(S):
Classification $\rightarrow$ {TopSecret, Secret, Confidential, Unclassified} $\rightarrow$ comparable

Implementation — Properties

```
property :OperatingSystem do
  values :windows, :osx, :selinux
  functions :set, :get, :equatable
end

property :device do
  values :workstation, :handheld, :blackberry, :terminal
  functions :set, :get, :equatable
end

property :project do
  values :zebra, :yuma, :lion
  functions :set, :get, :equatable
end

property :role do
  values :alpha, :beta, :delta
  functions :set, :get, :equatable
end
```

Implementation — Properties

```
property :securitydomain do
  values :abnet, :secnet, :telnet, :omninet
  functions :set, :get, :comparable
  order :abnet, :secnet, :telnet, :omninet
end

property :securityclearance do
  values :topsecret, :secret, :confidential
  functions :set, :get, :comparable
  order :topsecret, :secret, :confidential
end

property :securityclassification do
  values :topsecret, :secret, :confidential,
    :unclassified
  functions :set, :get, :comparable
  order :topsecret, :secret, :confidential,
    :unclassified
end
```

Implementation — Entity, Context

```
entity :subject do
  contains :project, :role, :securityclearance
end

entity :environment do
  contains :device, :operatingsystem, :securitydomain
end

entity :resource do
  contains :securityclassification
end
```

```
context :multilevelsecurity do
  contains :subject, :resource, :environment
end
```

Implementation — Entity, Context

```
entity :subject do
  contains :project, :role, :securityclearance
end

entity :environment do
  contains :device, :operatingsystem, :securitydomain
end

entity :resource do
  contains :securityclassification
end
```

```
context :multilevelsecurity do
  contains :subject, :resource, :environment
end
```

Implementation — Activities, Constraints

```
view = activity :view do
  # Some activity to enable viewing
end

c1 = constraint do
  securityclassification >= :secret
    && project == :yuma
    && securityclearance >= :secret
    && device == :blackberry
    && securitydomain >= :secnet
end

restricted_view = restrict view do
  with c1
end
```

```
authorization = activity :project_authorization do
  is_authorized? :yuma
end
```

Implementation — Activities, Constraints

```
view = activity :view do
  # Some activity to enable viewing
end

c1 = constraint do
  securityclassification >= :secret
    && project == :yuma
    && securityclearance >= :secret
    && device == :blackberry
    && securitydomain >= :secnet
end

restricted_view = restrict view do
  with c1
end
```

```
authorization = activity :project_authorization do
  is_authorized? :yuma
end
```

Implementation — Policies

```
pol = policy do
  policy_evaluators :standard
  constraint_evauators :propositional
  permit restricted_view do
    when authorization
  end
end
```

```
pol = policy do
  policy_evaluators :standard
  constraint_evauators :propositional
  permit restricted_view do
    when authorization
    count_limit restricted_view, 5
  end
end
```

Implementation — Policies

```
pol = policy do
  policy_evaluators :standard
  constraint_evauators :propositional
  permit restricted_view do
    when authorization
  end
end
```

```
pol = policy do
  policy_evaluators :standard
  constraint_evauators :propositional
  permit restricted_view do
    when authorization
    count_limit restricted_view, 5
  end
end
```

Sample — Thing