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

What motivated us to do this DSL?

- Easier domain representation
- Internal v. External DSL

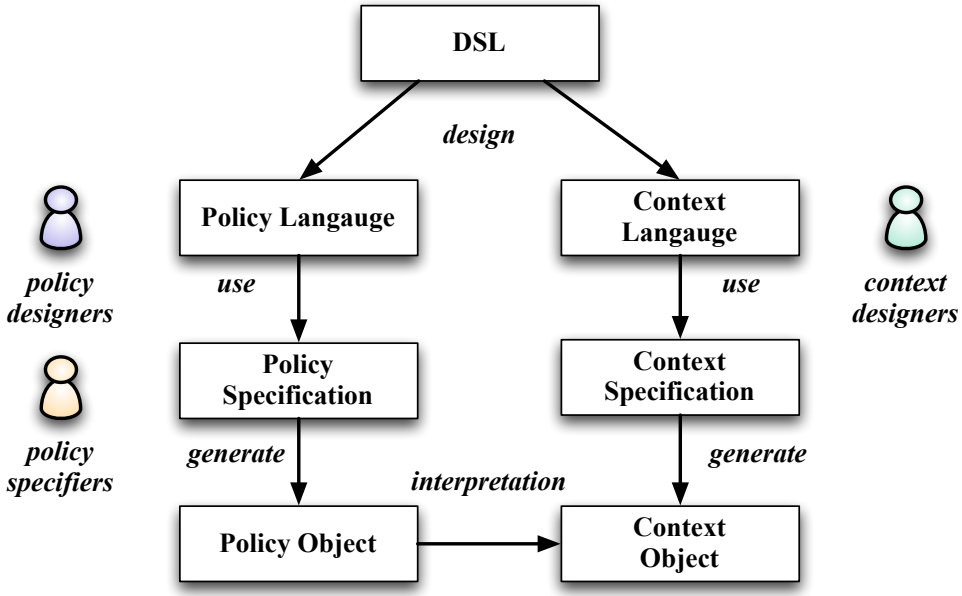
What is motivating our work?

- Applying policy-centric usage management dynamically, incorporating into network fabrics
- Providing attribution and query capabilities to policies and licensure
- Creating dynamic flexible policy environments

We think this DSL will help is in our longer term goals.

Design — Notional Use

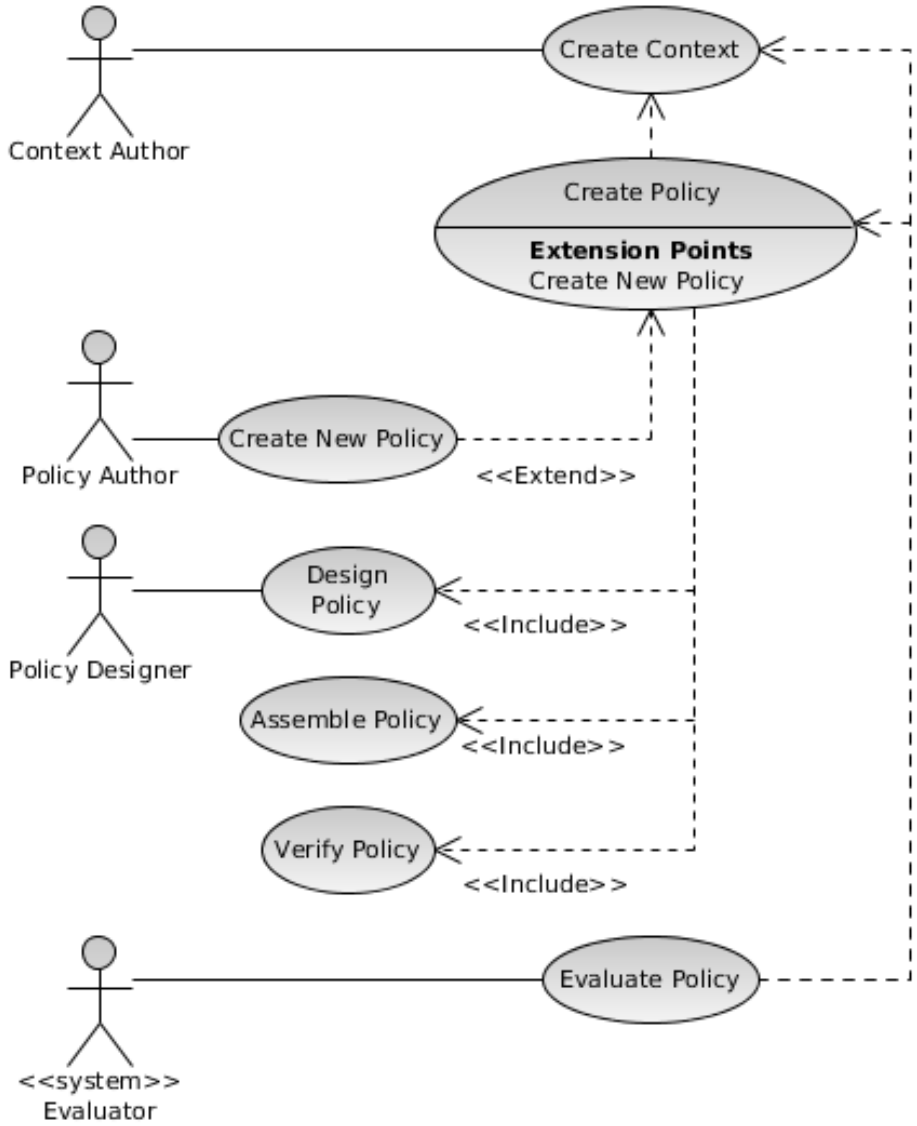
Notional Use:



- *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

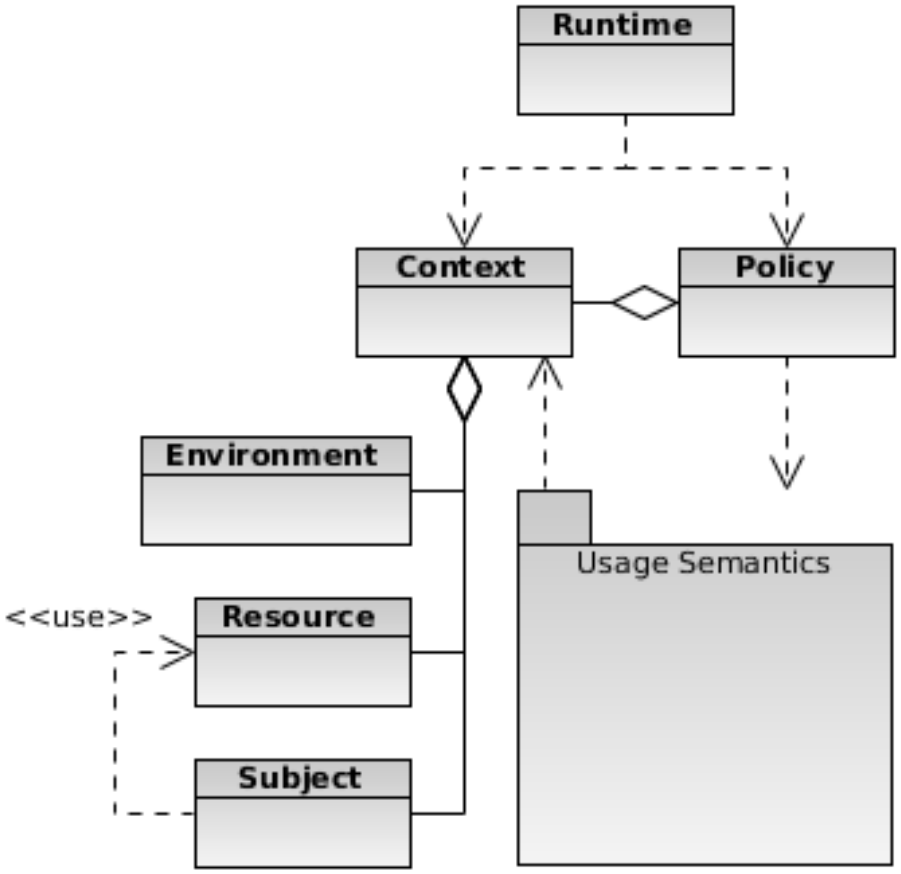
Use Cases:



- *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 — Domain Model

Domain Model:



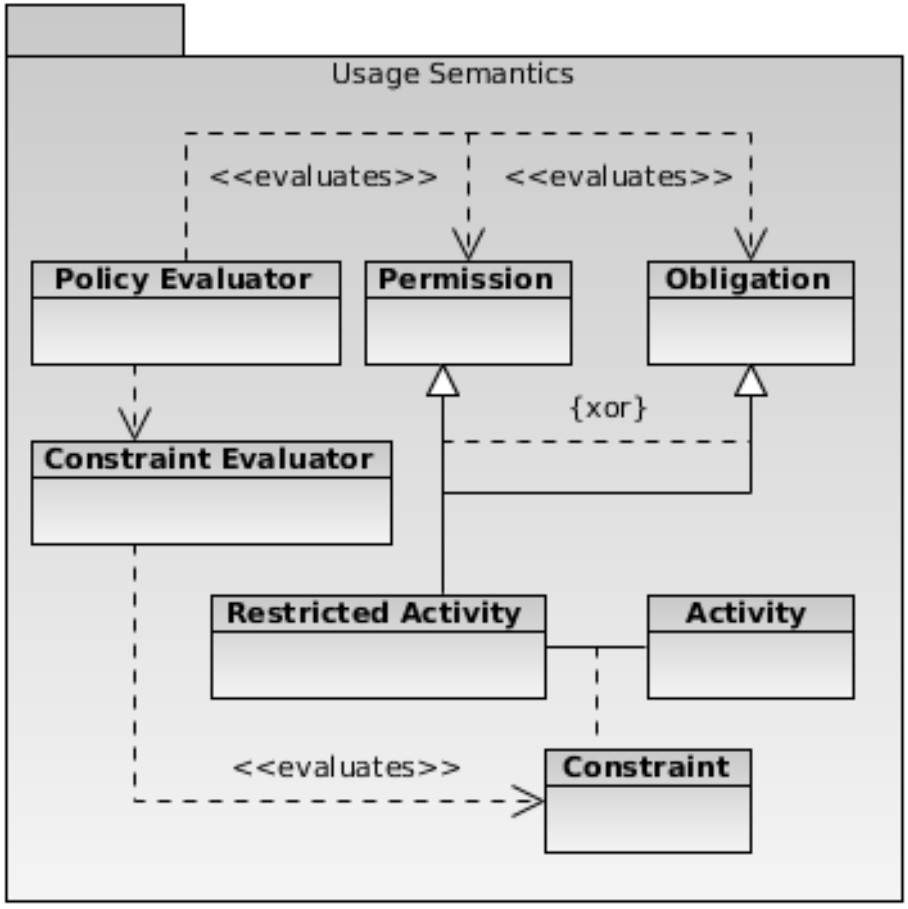
The *Runtime* accesses and activates a *policy* and manages a *context* to which the policy is given a reference.

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:



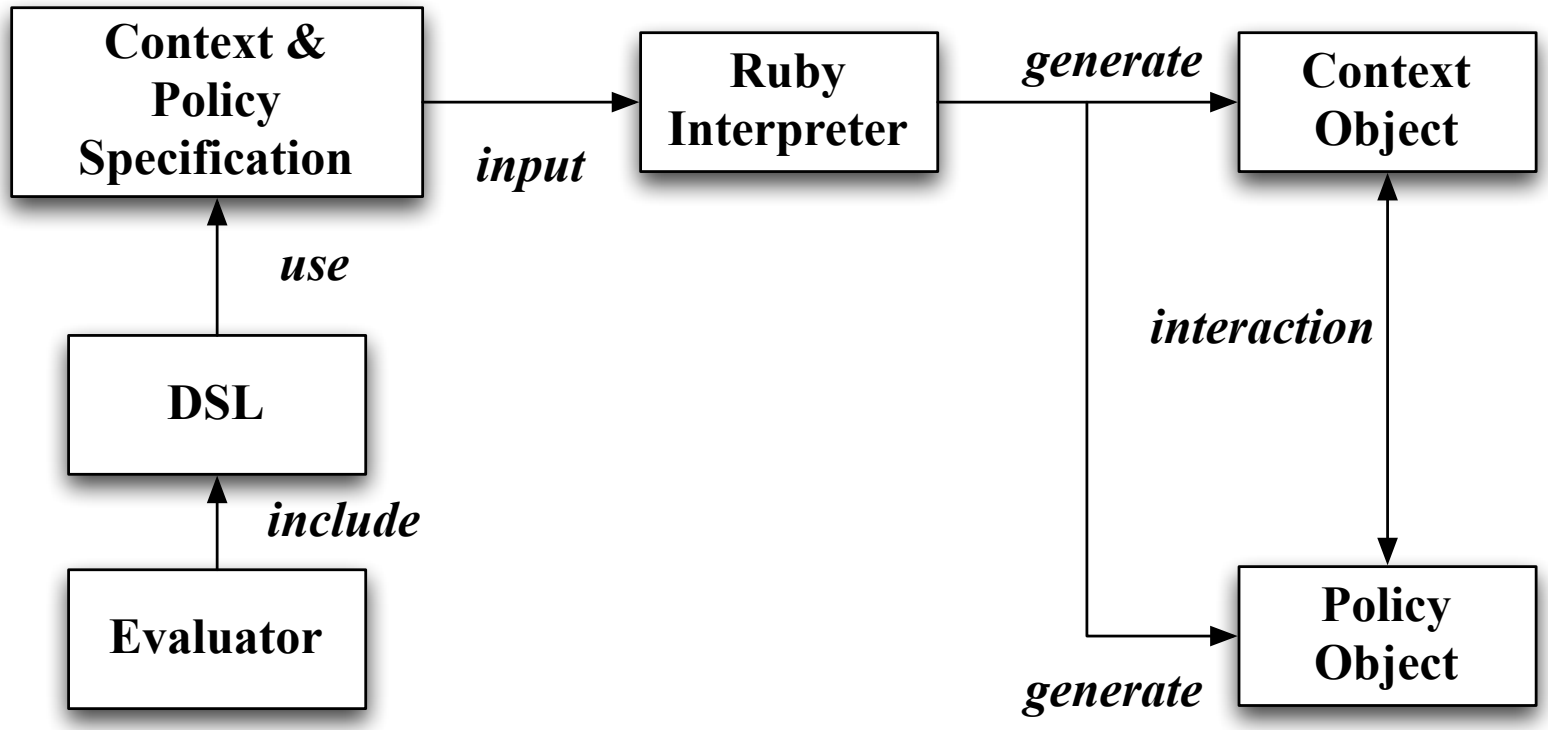
A *policy evaluator* examines and rectifies both *permissions* and *Obligations*.

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*.

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 — 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 - Interface

- **permissions?()**. Returns the set of permissions for a given policy.
- **obligations?(a)**. Returns the set of all obligations associated with a given permission.
- **remaining_obligations(a)**. Returns the set of remaining obligations for a given permission.
- **remaining_count(a)**. Returns the set of remaining count for a given permission.
- **allowed?(a, ctx)**. A boolean function that returns *true/false* whether a given activity can be carried out under a given context.
- **reset()**. Resets the policy by resetting its state.

Application - CC REL

- The Creative Commons Rights Expression Language
- RDFa (Resource Description Framework in attributes) for HTML Web pages and resources referenced therein
- XMP (Extensible Metadata Platform) for stand-alone media
- `http://wiki.creativecommons.org/CC_REL`

Application - RDFa in HTML

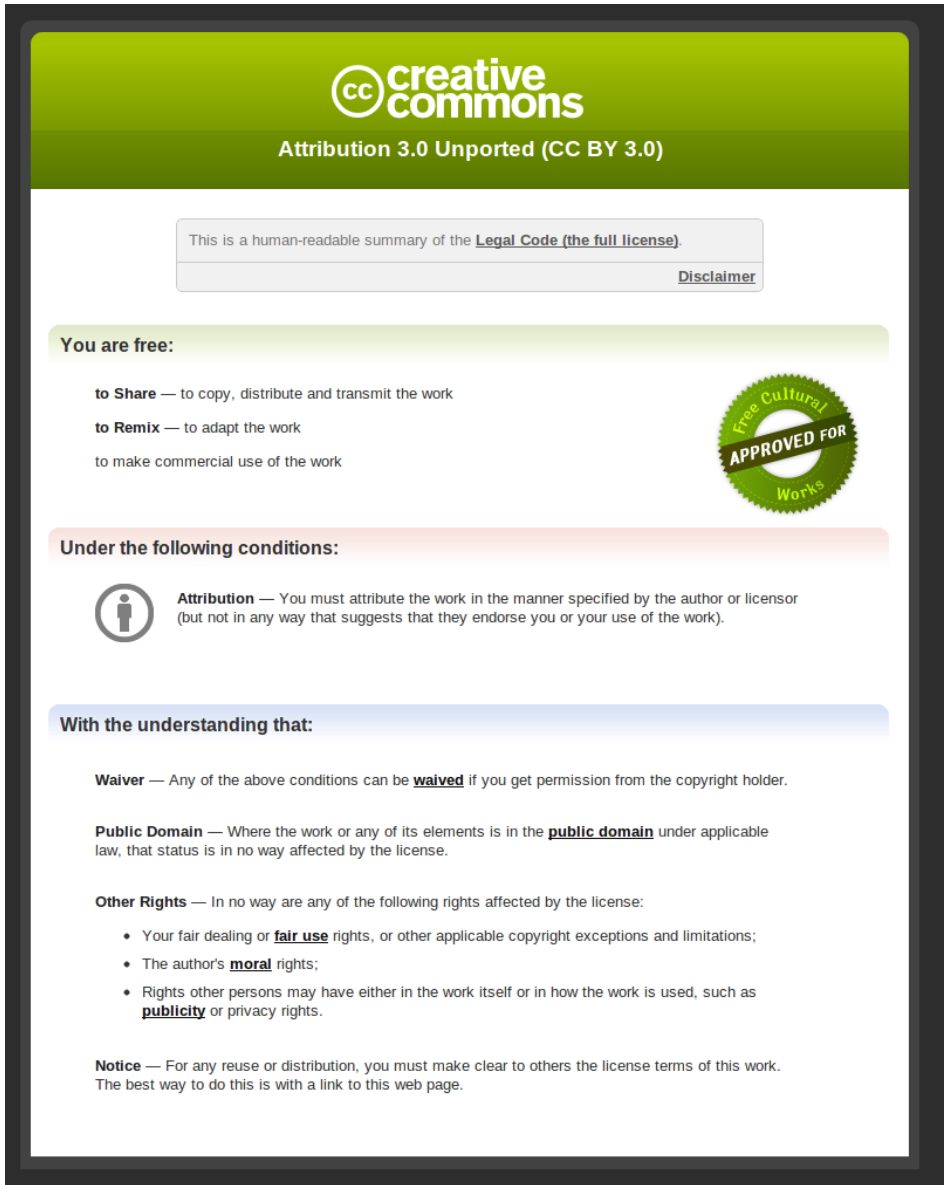
- Can simply associate web content with a CC license:

```
<div about="" instanceof="cc:Work" xmlns:cc="http://creativecommons.org/ns#"
     xmlns:dc="http://purl.org/dc/elements/1.1/" align="center">

    <a rel="license" href="http://creativecommons.org/licenses/by/3.0/">
    <img alt="Creative Commons License"
         src="http://i.creativecommons.org/l/by/3.0/us/88x31.png" /></a><br />

    <span property="dc:title">The Lessig Blog</span>,
    a <span rel="dc:type" href="http://purl.org/dc/dcmitype/Text">
    collection of texts</span>

    by <a property="cc:attributionName" rel="cc:attributionURL"
        href="http://lessig.org/"> Lawrence Lessig </a>,<br />

    is licensed under a
    <a rel="license" href="http://creativecommons.org/licenses/by/3.0/">
    Creative Commons Attribution License</a>.<br />

    There are
    <a rel="cc:morePermissions" href="http://lessig.org/blog/other-license">
    alternative licensing options</a>
</div>
```

- But what are the semantics of `http://creativecommons.org/licenses/by/3.0/`?

Application - License Deed Webpage

- Can a machine derive this page's semantics?



creative commons

Attribution 3.0 Unported (CC BY 3.0)

This is a human-readable summary of the **Legal Code (the full license)**.

Disclaimer

You are free:

to Share — to copy, distribute and transmit the work

to Remix — to adapt the work

to make commercial use of the work

Free Cultural Works — APPROVED FOR

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

With the understanding that:

Waiver — Any of the above conditions can be **waived** if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the **public domain** under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or **fair use** rights, or other applicable copyright exceptions and limitations;
- The author's **moral** rights;
- Rights other persons may have either in the work itself or in how the work is used, such as **publicity** or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

Application - RDF Embedded

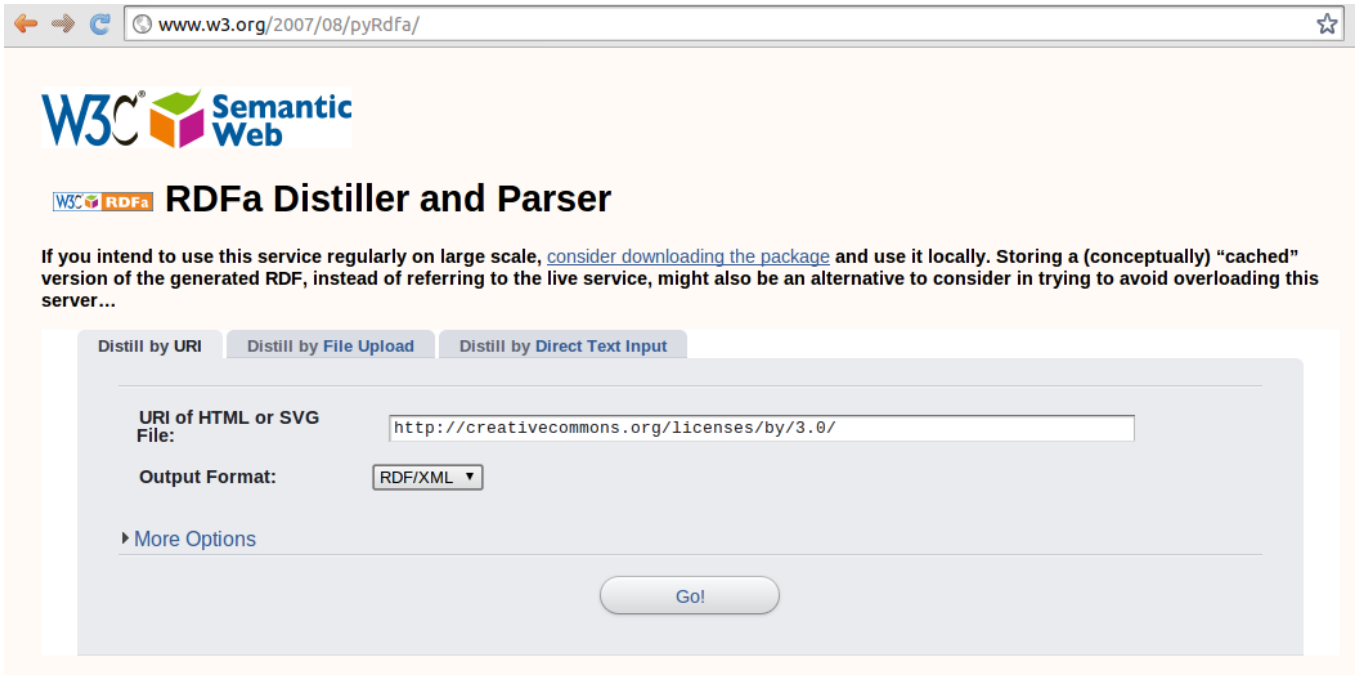
- The previous webpage contains the following embedded RDF:

```
<!-- RDF code here for backwards compatibility.  Please use the
     license's RDFa instead. -->
<!-- <rdf:RDF xmlns="http://creativecommons.org/ns#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf
      -syntax-ns#">
<License rdf:about="http://creativecommons.org/licenses/by/3.0/">
    <permits rdf:resource="http://creativecommons.org/ns#DerivativeWorks"/>
    <permits rdf:resource="http://creativecommons.org/ns#Distribution"/>
    <permits rdf:resource="http://creativecommons.org/ns#Reproduction"/>
    <requires rdf:resource="http://creativecommons.org/ns#Attribution"/>
    <requires rdf:resource="http://creativecommons.org/ns#Notice"/>
</License>
</rdf:RDF> -->
```

- From this, a machine can determine that this license:
  - Permits:
    - `#DerivativeWorks`, `#Distribution`, `#Reproduction`
  - Requires:
    - `#Attribution`, `#Notice`
- However, what do these things mean? How are they implemented?

Application - RDFa Embedded

- In addition to the RDF shown on previous slide, CC License Deeds also have embedded RDFa
- You can see that a machine can parse this data with something like the RDFa Distiller and Parser Tool:



Application - RDFa Distiller

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:cc="http://creativecommons.org/ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dct="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
>
  <rdf:Description rdf:about="http://creativecommons.org/">
    <dct:title xml:lang="en">Creative Commons</dct:title>
    <dc:title xml:lang="en">Creative Commons</dc:title>
  </rdf:Description>

  <cc:License rdf:about="http://creativecommons.org/licenses/by/3.0/">
    ...
    <dct:creator rdf:resource="http://creativecommons.org/"/>
    <dc:creator rdf:resource="http://creativecommons.org/"/>
    <cc:requires rdf:resource="http://creativecommons.org/ns#Attribution"/>
    <cc:requires rdf:resource="http://creativecommons.org/ns#Notice"/>
    <cc:permits rdf:resource="http://creativecommons.org/ns#DerivativeWorks"/>
    <cc:permits rdf:resource="http://creativecommons.org/ns#Reproduction"/>
    <cc:permits rdf:resource="http://creativecommons.org/ns#Distribution"/>
    <dct:identifier xml:lang="en">CC BY 3.0</dct:identifier>
    <dct:title xml:lang="en">Attribution 3.0 Unported</dct:title>
    <dc:identifier xml:lang="en">CC BY 3.0</dc:identifier>
  </cc:License>
</rdf:RDF>
```

Application - CC RDF Schema

- License RDF(a) references `#DerivativeWorks`, etc., in the CC namespace that's defined by a schema that's human-readable and machine-readable RDF.
- But... how immediately machine actionable is this schema?
- Partial screenshot below:

- We would like to investigate replacing or augmenting RDF(a) in the license deed with a license that's described with our DSL

Application - DSL

- By investigating replacing the contents of a license like `http://creativecommons.org/licenses/by/3.0/` with something that expresses the license in terms our DSL, we hope to:
  - Maintain equivalent license semantics
  - Express the semantics in a form that is easier for humans to read and write
  - Enable a machine to more directly execute the license and reason over it

Conclusions

- Internal DSLs are convenient, but probably not appropriate for real systems
- Overall we like the DSL but could do without some of the Ruby cruft (e.g. **do...end**, etc.)
- Application and Optimization

Questions?

Chris Lamb: cclamb@ece.unm.edu
Pramod Jamkhedkar: pramod54@ece.unm.edu
Greg Heileman: heileman@ece.unm.edu
Mathew P. Bohnsack: bohnsack@gmail.com
Viswanath Nandina: vishu@ece.unm.edu