Policy-Aware Pipes: Support Accountability in Mashup Service of Linked Data

Fuming Shih and Lalana Kagal

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
(fuming | lkagal)@csail.mit.edu

Abstract. With the proliferation of linked data on the Web, more Semantic Web applications are built in the form of mashups where data comes from multiple sources. Mashups enable easy content reuse in Web applications because of the interlinking between linked data sets, but they also complicate the governing of data usage in such an open environment. Many legal issues could arise in mashups when different sources have different legal claims and usage policies. Both publishers of linked data and mashup developers need methods to avoid inappropriate use of data in the life cycle of mashup services.

In this paper, our primary focus is to facilitate participants in a mashup environment in being policy aware and making the right decision to be held accountable for data usage. We present the requirements for policy support in a mashup environment and the methods for incorporating policies into the life cycle of mashup development. We implement a tool called Policy Aware Pipes (PAP) which provides functions that (1) validate policy compliance statement from mashup sources, (2) check for policy compliance, and (3) generate provenance and policy compliance statement of the mashup. In addition, we also provide a walkthrough of the scenario using PAP to create a mashup with existing linked data set.

1 Introduction

The Linked Open Data project [13] has successfully brought a great amount of data to the Web that serves as content for Semantic Web applications. Web developers can now create Semantic Web applications effectively with *interlinking* data sets, making use of data from many different sources. We have seen the transformation of Web services due to the change in the nature of data on the Web. With the emergence of Web 2.0, companies that once created Web services from data isolated in different *data gardens*, have begun to make these data publicly available by using APIs, connecting the data with semantic relationships as linked data [11]. The *opening up* of data encourages developers to reuse content on the Web and build applications from different sources to provide new functions, called *Mashups* [14]. With the proliferation of linked open data, we can foresee a blossoming of semantic Web applications in the form of mashups.

However, given the *open* environment for data in the mashup world, concerns arise when a user wishes to assert some controls over the use of the data. For example, government agencies that publish linked data such as census data, might prohibit the use of the data in any commercial service, directly or indirectly. Because mashups allow developers to reassemble data in different content, they also complicate the governing of data usage. To support the governing of data usage, it is clear that authorization and authentication alone are not sufficient in the *data centric* phenomenon of linked data world. Once the data is linked or retrieved in the mashup, it is impossible to ensure correct usage in different context in the data flow. Many legal issues can arise when merging data from different sources with differing legal claims and usage policies [6]. This situation places great burdens on the mashup developers, who need to synthesize these individual policies and determine the allowed uses of the merged data. It also hampers the process for data publishers, who may hesitate to open up their data due to the unpredictable ways in which it may be used [15].

To succeed in creating an environment that benefits both data publishers and mashup developers, we argue that information accountability should be introduced to mashup environments by applying policies on data usage [18]. The goal is to build necessary technology support into the mashup process to make information usage more transparent to prevent participants from misusing the data. Although policies are commonly used in claiming privacy considerations, accessibilities and restrictions of the data, they are rarely incorporated into the development process of mashup services. Different stages of the mashup life cycle require different types of policy interactions. Data publishers, for example, need ways to create and attach policies to their data, while mashup creators need to extract and understand those policies. It is important to have methods and tools to put guidance into practice, so as to minimize the efforts required of all participants in being policy-compliant when creating mashups.

We present in this paper (1) a method for incorporating policies into the process of mashup construction for different players and (2) a tool called Policy-Aware Pipes (PAP) that implements these concepts. These two contributions are presented as follows: the next section gives a motivating scenario to present the challenges and problems of building a mashup using linked open data. Section 3 presents the requirements of policy interactions in the mashup. In Section 4 we discuss the architecture overview of PAP, and in Section 5 the implementation details. In Section 6 we presents a walkthrough of using PAP for building a mashup of previous scenario. Section 7 discusses the related works issues. The conclusions and future directions are presented in Section 8.

2 Motivating Scenario

We use the scenario below to sketch out the general policy problems and challenges of building a mashup of linked data, especially when cascading data flow occurs between mashups.

2.1 Open Data from BBC Backstage

Suppose Joe is a pop-music fan who spends much time searching and reading news about albums on the market. He is also a Web expert who creates a mashup site called "music-meme.org" that aggregates the most up-to-date articles and news about pop music, and publishes RSS feeds to contribute to the community. In one section of the mashup service, called "music notes", he obtains information and reviews of each music album from linked data released by the BBC. The BBC has produced many services related to television programming and music, and recently this content has been made available as RDF data via the BBC Backstage [3]. Joe wants to make sure that his website does not infringe the copyright and other policies of the BBC. For this open data, the BBC only supports non-commercial use of the content. The following text is extracted from the license agreement and additional policies on BBC's website [2][5].

You must not:
**Charge users for accessing your work that contains or uses BBC Content.*
**Sell applications that use or incorporate BBC Content .*
...however, you own the Intellectual Property rights in and to your application and nothing prevents you being able to sell or commercially licence your application and make money from it. What you can not do is sell access to any BBC content (i.e any commercial use of your application would have to be done without using any BBC Content).

Joe feels confident that his website is compliant with the BBC's policy and publishes his site to the community.

2.2 Problems

Suppose Bob is a software engineer of website called "Alpha Music Box", an online music store that charges users for downloading songs. He implementes a function on the music store that provides professional reviews of each album. Eventually, those albums with better reviews become increasingly popular. The music reviews, though they were consumed from data feeds of music-meme.org, are actually data originally from the BBC. Bob's use of the music reviews in a commercial mashup have possibly infringed the license agreement of the data source, even the music website did not make money directly from the BBC's content. The reasons for the misuse of the BBC's content were because (1) the policies were only *human readable* and (2) the policies did not *propagate together with the data* in the mashup of Joe's website. PAP provides steps toward solving this problem by incorporating policy into the mashup workflow.

3 Requirements of the Policy Support in the Mashup

We have learned from the above use cases that a mashup of linked data needs policy support for information accountability. In this section, we list the requirements for policy interactions throughout the mashup life cycle. We also discuss how participants are assisted in the different stages of a mashup's life cycle. As shown in the right half of Fig. 1, we identify three types of participants in the mashup life cycle. The life cycle includes (1) generating the data by a data publisher, (2) using this data to define a new mashup by a mashup developer, and (3) utilizing the mashup output by a mashup consumer.

3.1 Policy for Data Usage

As owners of the data source, the data publishers need a policy language to assert the policies before publishing the data. Publishers use policy languages to specify usage restrictions and conditions for the linked data. In this paper, we suggest using Semantic Web technology as a policy language because it allows different entities to have a shared data model to describe constraints (e.g. security and privacy) within the domain [19]. This practice increases the interoperability between policies which could be separately developed by different communities.

3.2 Policy Checking and Awareness

When a mashup uses data from multiple sources, the mashup developer needs to extract policies stated in different data streams. The mashup tool needs to detect policy conflicts among the source policies and mashup contexts, then provide informative feedback to the mashup developer. Justifications of policy decisions (being compliant or not compliant) allow the mashup developer to understand how results are obtained [9] and to take the necessary actions to adjust the mashup such that it becomes policy compliant. After checking for conflicts, the mashup tool can then suggest an appropriate merged policy which is compatible with the source policies for that mashup.

3.3 Provenance Information

Throughout the construction of the mashup, a mashup developer can design the mashup in terms of various operations, including SPARQL queries. Knowing the data sources as well as the operations performed is useful to a mashup consumer in inferring the trustworthiness of the mashup. If the mashup is compliant with the policies, the mashup tool should output the policy compliance statement together with the provenance information, and publish the results together as triples with the content of the mashup. Then the data consumer could use this information as an evaluation of the trustworthiness of this mashup. The data consumer could even be another mashup developer. In such a case, this second mashup developer would validate the compliance statement and use it for future auditing to clarify responsibilities of the two parties.

4 Architecture

We have developed the Policy-Aware Pipes (PAP) as an extended operator on top of Semantic Web Pipes (SWP) [12], an RDF workflow engine that helps

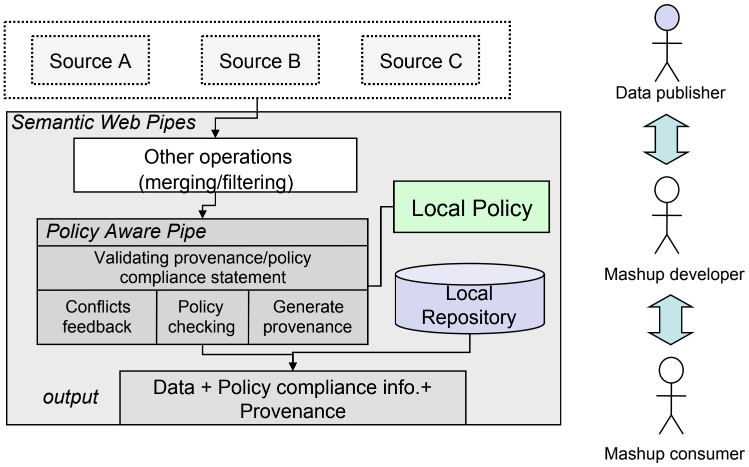


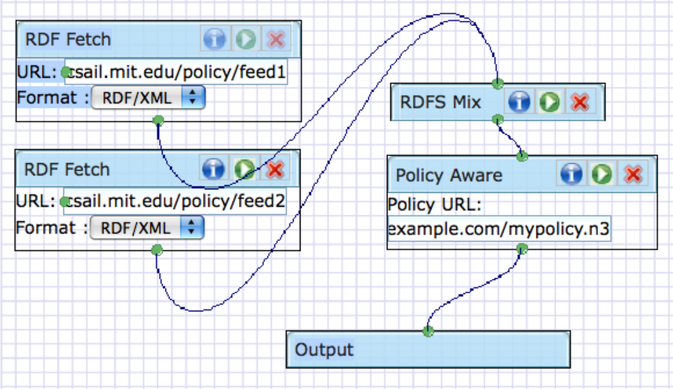
Fig. 1. Overview of Policy-Aware Pipe and mashup process

users to build customized mashup services from data sources dynamically. The intuition of PAP is to bring policy awareness to the design phase of mashup services by providing informative feedback messages to the editing interface. This policy information allows mashup creators (1) to validate that they are using data in compliance with the policies of the source providers, (2) to know what policies they can/must provide with their derived data, (3) to publish a "policy compliant" statement with their mashup. As Fig. 2 shows, the user simply feeds RDF data sources to PAP and defines the local policy of the mashup service in the editor. PAP will also display messages on the debugging pane of the pipe editor to explain the results from the policy checking.

PAP has four major functions: (1) validating source provenances and policy compliance statements, (2) checking policy compliance for the merged sources, (3) suggesting possible policies based on those of the sources, and (4) generating provenance and policy compliance statements. PAP first validates a source's provenance by checking for the existence of the triples about policy compliance from the source repository. The compliance information will be displayed in the pipe editor and bring awareness to the users about the reliabilities of the merged sources. The second step is to compute the compatibility of a merged source S, denoted as *X(S)*, by retrieving each source's policy P and running it against all the sources' profile metadata M. If the user creates any policy locally for this mashup, it will be applied as well for policy checking. The *policy conflict* C of each source is detected if the profiles of other sources are not compatible.

$$X(S) = C_1 \wedge C_2 \wedge ... \wedge C_n \qquad (1)$$

An example of a policy specified in the data source is given in Fig. 3, and an example of service profile of the data source is given in Fig. 4.

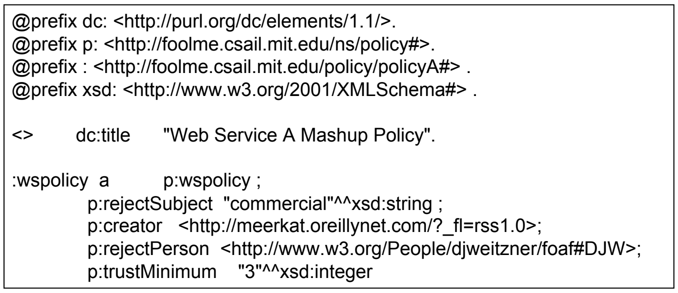


URL: csail.mit.edu/policy/feed1
Format : RDF/XML

URL: csail.mit.edu/policy/feed2
Format : RDF/XML

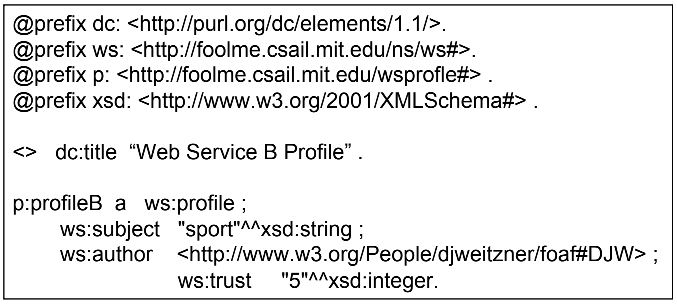
Policy URL:
example.com/mypolicy.n3

Fig. 2. Example editing interface of Policy Aware Pipe



```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix p: <http://foolme.csail.mit.edu/ns/policy#>.
@prefix : <http://foolme.csail.mit.edu/policy/policyA#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<>        dc:title      "Web Service A Mashup Policy".

:wspolicy  a          p:wspolicy ;
             p:rejectSubject  "commercial"^^xsd:string ;
             p:creator   <http://meerkat.oreillynet.com/?_fl=rss1.0>;
             p:rejectPerson  <http://www.w3.org/People/djweitzner/foaf#DJW>;
             p:trustMinimum    "3"^^xsd:integer
```

Fig. 3. Example of policy specified in the data source. It rejects mashing up with the service defined as commercial in its profile.



```
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix ws: <http://foolme.csail.mit.edu/ns/ws#>.
@prefix p: <http://foolme.csail.mit.edu/wsprofle#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<>   dc:title  “Web Service B Profile” .

p:profileB  a   ws:profile ;
       ws:subject   "sport"^^xsd:string ;
       ws:author    <http://www.w3.org/People/djweitzner/foaf#DJW> ;
                    ws:trust     "5"^^xsd:integer.
```

Fig. 4. Example of service profile of the data source. The service describes itself as having a subject of sport and trust value of 5

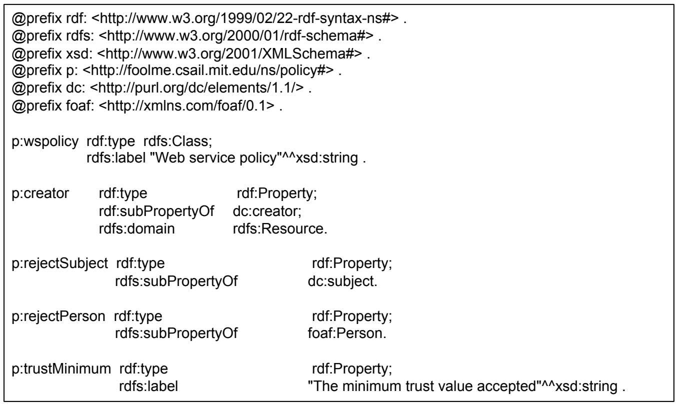
Notice that there are many ways to implement *X(S)* and compute policy conflicts. In this prototype, the overall score of a policy conflict is simply the AND logic of C_i. The policy P_i only specifies the *restrictions* of each source that will be applied against the mashup context, which in this case are the profiles M of all sources. For example, the above policy states that it refuses to mashup with any source that has described itself as "commercial" in its profile. It also rejects data sources published by the "person" identified with a specific FOAF URI. If any conflict is detected, the value of C_i will be set to 1 and a message explaining the conflict will be shown in the Pipes editor's debugging pane.

5 Implementation

Building PAP as an extension on top of Semantic Web Pipes (SWP) required us to implement two components in the SWP framework. SWP is a mashup builder for linked data with a Web GUI which shows the creation of a mashup in a workflow style. SWP is built in Java as a Web servlet operating on the Apache Tomcat Server. We first implemented the "Operator" interface that manipulates the triple store in SWP with customized functions. Then, we created a supplementary Java class called "PolicyMashup" that supports primitive policy checking, policy information logging for policy awareness, and policy compliance statement generation.

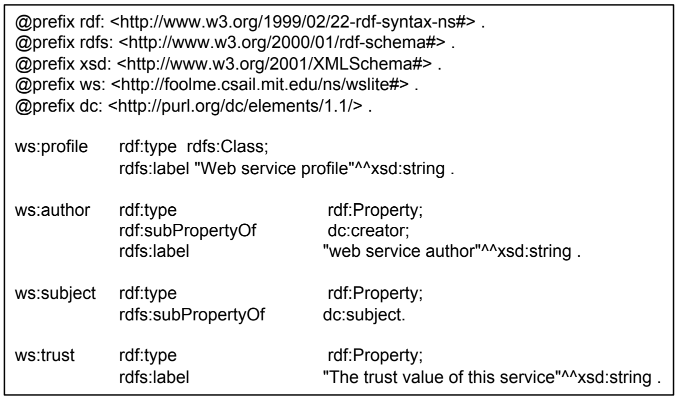
5.1 Policy and Service Ontologies

We have created lightweight ontologies for both policies and Web service profiles that can be used by data publishers to specify their services and the policies associated with them. These ontologies are placeholders in our architecture and can be easily replaced by more complicated and expressive ontologies. The policy ontology is shown in Fig. 5, and the service profile ontology is shown in Fig. 6. We have assumed that each data source has its usage policy and service profile published together with the data. Under this assumption, the policy and service profile also need a way to *attach* to the linked data, so that they can be dereferenced and queried using SPARQL queries later. To simplify this demonstration, we have assumed that the data source identifies the policy with Creative Commons' *CC:morePermissions* (CC+) tag. CC+ is suggested by Creative Commons [4] to address rights beyond those granted by a CC license. Service profiles are identified with the service ontology mentioned in Section 5.1. Fig. 7 shows how the data source can attach policy and service profiles to the published data presented in the form of RSS.1.0 [16]. Because users could use different ontologies when defining their policy statementsm PAP needs a mapping between the different ontologies used. This will allow PAP to check the policies using a singled unified ontology. We are currently developing adapters to support external policy engines for policy checking as well.



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix p: <http://foolme.csail.mit.edu/ns/policy#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix foaf: <http://xmlns.com/foaf/0.1> .

p:wspolicy  rdf:type  rdfs:Class;
            rdfs:label "Web service policy"^^xsd:string .

p:creator      rdf:type              rdf:Property;
               rdf:subPropertyOf     dc:creator;
               rdfs:domain           rdfs:Resource.

p:rejectSubject  rdf:type                  rdf:Property;
                 rdfs:subPropertyOf        dc:subject.

p:rejectPerson  rdf:type                   rdf:Property;
                rdfs:subPropertyOf         foaf:Person.

p:trustMinimum  rdf:type                   rdf:Property;
                rdfs:label                 "The minimum trust value accepted"^^xsd:string .
```

Fig. 5. Example of policy ontology that describes restrictions of source's data usage, such as subjects to reject or minimum trust value of a service to be accepted.

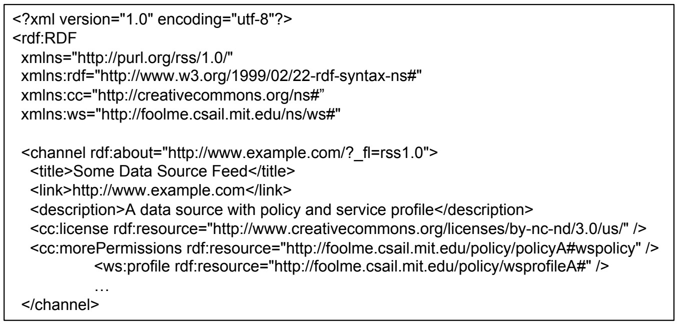


```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ws: <http://foolme.csail.mit.edu/ns/wslite#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

ws:profile     rdf:type  rdfs:Class;
               rdfs:label "Web service profile"^^xsd:string .

ws:author      rdf:type                  rdf:Property;
               rdf:subPropertyOf         dc:creator;
               rdfs:label               "web service author"^^xsd:string .

ws:subject     rdf:type                  rdf:Property;
               rdfs:subPropertyOf       dc:subject.

ws:trust       rdf:type                  rdf:Property;
               rdfs:label               "The trust value of this service"^^xsd:string .
```

Fig. 6. Example of service profile ontology that describes each source. The profile of each source or mashup itself is used later in the checking for policy compliance.

5.2 Policy Compliance

PAP helps to extract the policy of each data source in SWP. It starts by looking for policy compliance triples in the data source, since the source could be from another mashup. Then, PAP reads the content of the policy and service profile of each source as well as those of the mashup itself. To check for policy conflicts, PAP checks the context of the mashup (the union of all service profiles) against each policy. If the content of any service profile is rejected by any policy, a notification is logged to bring up policy awareness later on the interface.

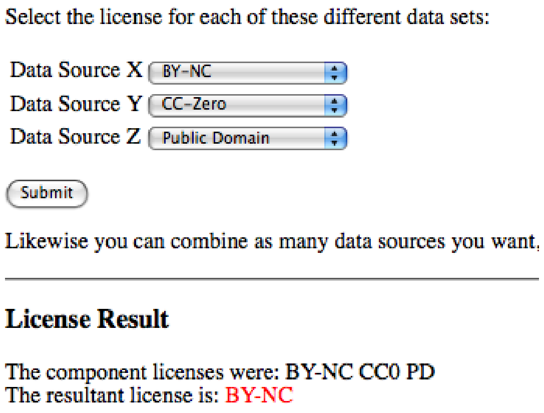


```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
 xmlns="http://purl.org/rss/1.0/"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cc="http://creativecommons.org/ns#"
 xmlns:ws="http://foolme.csail.mit.edu/ns/ws#"

 <channel rdf:about="http://www.example.com/?_fl=rss1.0">
  <title>Some Data Source Feed</title>
  <link>http://www.example.com</link>
  <description>A data source with policy and service profile</description>
  <cc:license rdf:resource="http://www.creativecommons.org/licenses/by-nc-nd/3.0/us/" />
  <cc:morePermissions rdf:resource="http://foolme.csail.mit.edu/policy/policyA#wspolicy" />
          <ws:profile rdf:resource="http://foolme.csail.mit.edu/policy/wsprofileA#" />
          …
 </channel>
```

Fig. 7. Example of data source feed

5.3 License Resolution

As discussed in Section 5.1, a data consumer can use cc:morePermissions to define an expressive policy about the data's usage, or simply use cc:license to associate a Creative Commons license with a service. In this case, PAP would make a call to a tool such as the Attribution License Violations Validator [17] to figure out whether there is a conflict between the licenses of the sources. Fig. 8 shows an example interface for CC license resolution between three different sources:



Select the license for each of these different data sets:

Data Source X BY-NC

Data Source Y CC-Zero

Data Source Z Public Domain

Likewise you can combine as many data sources you want,

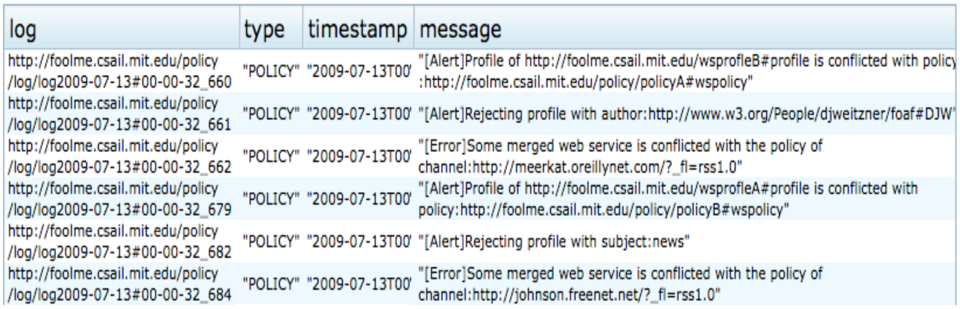
License Result

The component licenses were: BY-NC CC0 PD
The resultant license is: BY-NC

Fig. 8. Example interface of CC license resolution. The result shows the suggested output license from multiple sources

5.4 Policy Awareness with Logging

We have implemented a logging module in PAP that logs policy messages in RDF. The logging module generates a log message for every operation in the workflow of SWP (e.g. SPARQL queries) and in PAP itself. The log consists of the name of the operator in Semantic Pipes which performed the task, the time at which the task was performed, and the output of the policy checking activities. PAP treats logs as RDF *data* that can be queried everywhere using SPARQL queries, and creates feedback to bring awareness in the editor interface of SWP. It helps to integrate policy awareness seamlessly into mashup development. Fig. 9 shows a table generated by PAP, listing the details of policy aware activities and any policy conflict that occurred.



log	type	timestamp	message
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_660	"POLICY"	"2009-07-13T00	"[Alert]Profile of http://foolme.csail.mit.edu/wsprofleB#profile is conflicted with policy :http://foolme.csail.mit.edu/policy/policyA#wspolicy"
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_661	"POLICY"	"2009-07-13T00	"[Alert]Rejecting profile with author:http://www.w3.org/People/djweitzner/foaf#DJW"
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_662	"POLICY"	"2009-07-13T00	"[Error]Some merged web service is conflicted with the policy of channel:http://meerkat.oreillynet.com/?_fl=rss1.0"
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_679	"POLICY"	"2009-07-13T00	"[Alert]Profile of http://foolme.csail.mit.edu/wsprofleA#profile is conflicted with policy:http://foolme.csail.mit.edu/policy/policyB#wspolicy"
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_682	"POLICY"	"2009-07-13T00	"[Alert]Rejecting profile with subject:news"
http://foolme.csail.mit.edu/policy /log/log2009-07-13#00-00-32_684	"POLICY"	"2009-07-13T00	"[Error]Some merged web service is conflicted with the policy of channel:http://johnson.freenet.net/?_fl=rss1.0"

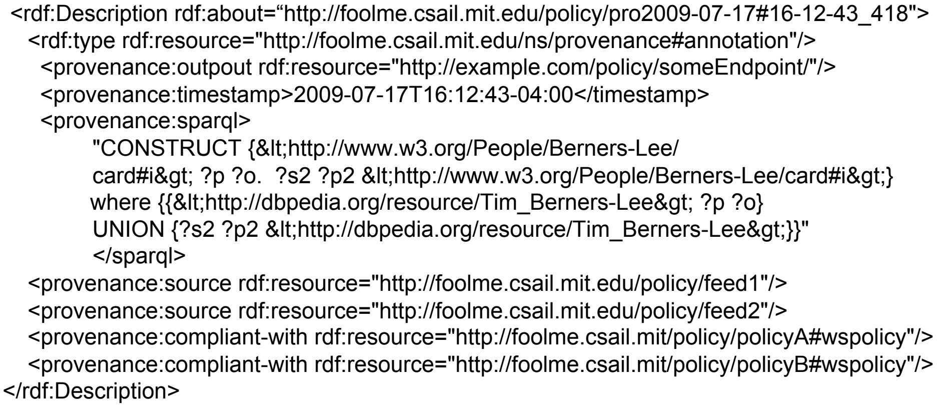
Fig. 9. Example output of the policy aware messages by logging. Explanation of policy conflicts is shown in the message.

5.5 Generating Policy Compliance Statement and Provenance

PAP will generate a policy compliance statement only if *X(S)* is 0, indicating that the output content is compliant with all the policies in the mashup context. Besides policy compliance information, PAP will also create provenance data for the mashup. The provenance data, consists of three parts: the source description, the SPARQL queries which were applied to the merged data sets from, and a timestamp. This information is encoded as triples to publish with the data in the mashup. Fig. 10 shows an example triples that describe provenance and policy compliance information.

6 Scenario Walkthrough

To demonstrate the major challenges in facilitating an accountable mashup service, and to demonstrate how PAP can help designers to ensure adequate accountability among all parties involved, we now revisit the scenario presented in

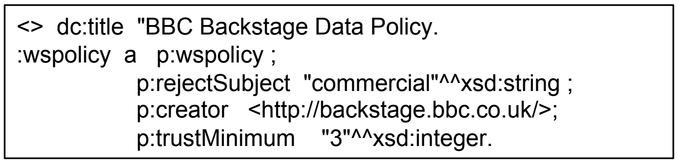


```
<rdf:Description rdf:about="“http://foolme.csail.mit.edu/policy/pro2009-07-17#16-12-43_418">
  <rdf:type rdf:resource="http://foolme.csail.mit.edu/ns/provenance#annotation"/>
    <provenance:outpout rdf:resource="http://example.com/policy/someEndpoint/"/>
    <provenance:timestamp>2009-07-17T16:12:43-04:00</timestamp>
    <provenance:sparql>
          "CONSTRUCT {&lt;http://www.w3.org/People/Berners-Lee/
          card#i&gt; ?p ?o.  ?s2 ?p2 &lt;http://www.w3.org/People/Berners-Lee/card#i&gt;}
          where {{&lt;http://dbpedia.org/resource/Tim_Berners-Lee&gt; ?p ?o}
          UNION {?s2 ?p2 &lt;http://dbpedia.org/resource/Tim_Berners-Lee&gt;}}"
          </sparql>
  <provenance:source rdf:resource="http://foolme.csail.mit.edu/policy/feed1"/>
  <provenance:source rdf:resource="http://foolme.csail.mit.edu/policy/feed2"/>
  <provenance:compliant-with rdf:resource="http://foolme.csail.mit/policy/policyA#wspolicy"/>
  <provenance:compliant-with rdf:resource="http://foolme.csail.mit/policy/policyB#wspolicy"/>
</rdf:Description>
```

Fig. 10. Example of provenance and policy compliance statement

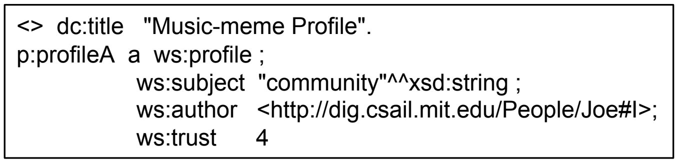
Section 2 and explained how PAP operates to accomplish the tasks. First of all, BBC Backstage could state a *machine readable* policy that describes the usage restrictions about its content as shown in Fig. 11.

Using PAP, Joe has RDF triples that describe his mashup service as shown in Fig. 12. Because Joe's service was compliant with the BBC's policy, PAP merges the policies with Joe's and generates provenance and policy compliance informa-



```
<>  dc:title  "BBC Backstage Data Policy.
:wspolicy  a   p:wspolicy ;
              p:rejectSubject  "commercial"^^xsd:string ;
              p:creator   <http://backstage.bbc.co.uk/>;
              p:trustMinimum    "3"^^xsd:integer.
```

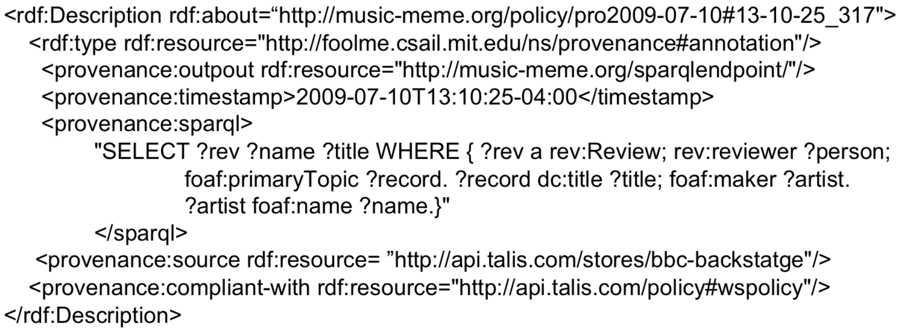
Fig. 11. Example of BBC Backstage policy for linked data



```
<>  dc:title   "Music-meme Profile".
p:profileA  a  ws:profile ;
              ws:subject  "community"^^xsd:string ;
              ws:author   <http://dig.csail.mit.edu/People/Joe#I>;
              ws:trust     4
```

Fig. 12. Example of service profile of website *music-meme.org*

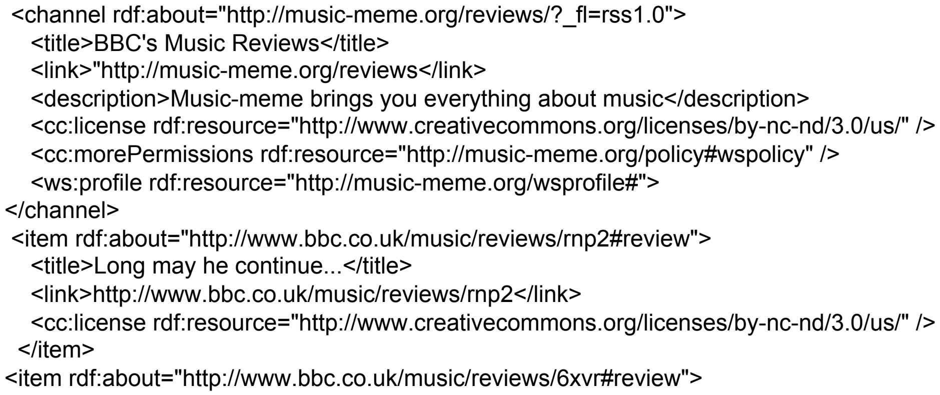
tion as shown in Fig. 13. Also, PAP embeds policy and license agreements into



```
<rdf:Description rdf:about=“http://music-meme.org/policy/pro2009-07-10#13-10-25_317">
   <rdf:type rdf:resource="http://foolme.csail.mit.edu/ns/provenance#annotation"/>
     <provenance:outpout rdf:resource="http://music-meme.org/sparqlendpoint/"/>
     <provenance:timestamp>2009-07-10T13:10:25-04:00</timestamp>
     <provenance:sparql>
            "SELECT ?rev ?name ?title WHERE { ?rev a rev:Review; rev:reviewer ?person;
                        foaf:primaryTopic ?record. ?record dc:title ?title; foaf:maker ?artist.
                        ?artist foaf:name ?name.}"
            </sparql>
    <provenance:source rdf:resource= ”http://api.talis.com/stores/bbc-backstatge"/>
   <provenance:compliant-with rdf:resource="http://api.talis.com/policy#wspolicy"/>
</rdf:Description>
```

Fig. 13. Example provenance of *music-meme.org* with policy compliance statement

the data feed of Joe's mashup output, as shown in Fig. 14. Later, when Bob's



```
 <channel rdf:about="http://music-meme.org/reviews/?_fl=rss1.0">
    <title>BBC's Music Reviews</title>
    <link>"http://music-meme.org/reviews</link>
    <description>Music-meme brings you everything about music</description>
    <cc:license rdf:resource="http://www.creativecommons.org/licenses/by-nc-nd/3.0/us/" />
    <cc:morePermissions rdf:resource="http://music-meme.org/policy#wspolicy" />
    <ws:profile rdf:resource="http://music-meme.org/wsprofile#">
</channel>
 <item rdf:about="http://www.bbc.co.uk/music/reviews/rnp2#review">
    <title>Long may he continue...</title>
    <link>http://www.bbc.co.uk/music/reviews/rnp2</link>
    <cc:license rdf:resource="http://www.creativecommons.org/licenses/by-nc-nd/3.0/us/" />
  </item>
<item rdf:about="http://www.bbc.co.uk/music/reviews/6xvr#review">
```

Fig. 14. Example of RSS Feed published by website *music-meme.org*

mashup consumes the content of reviews from Joe's mashup, it first queries the provenance information to ensure that Joe's data output was compliant with the its sources' policies. Using PAP, he found in the policy-aware logs that his service is not compliant with Joe's policy to not use the content commercially. At this juncture, Bob decides to use the reviews in another service that is not commercial.

7 Related Work

With the proliferation of linked data, it is important to have tools that help developers create Web applications rapidly through content reuse. Several tools have followed the programming paradigm of *Mashups* in Web 2.0, and developed either a language such as MashQL [8], or tools such as Semantic Web Pipes [12] and Potluck [7] that lower the barrier of programming skills needed to build mashups. Although these tools provide a way to ease the construction of the mashup, they do not address problems of inappropriate content reuse. Researchers from the law community have pointed out that legal problems arise from data reuse in mashup applications. In [6], Gerber listed the legal issues that mashup developers would face. For example, the contract law issues, copyright law issues, issues of misleading the trademark, and privacy issues all lead to legal liabilities that could occur in the life cycle of a mashup. Research showes that unclear content usage which leads to legal issues is one of the major concerns that prevents the data owner from publishing data publicly [15]. PAP introduces methods to incorporate policy interactions into the development of a mashup, and thus provides tools to encourage all participants to be policy-compliant in the mashup environment.

How to ensure information accountability in mashups still remains an open question, especially as data becomes *open* and *linked*. In the research of e-business engineering, Zou and Pavlovski [20] propose a framework to develop effective accountability solutions for mashup environments. It focuses on bringing mashup accountability to a level of quality comparable to that of enterprise Service Oriented Architecture. The paper suggests using Semantic Web technologies, such OWL-S to model the relationships and roles of entities in the mashup environment. Such a model helps to capture the traceability of service compositions and reason over the responsibilities of parties involved in the mashup. Other work [10] uses encryption technology together with extra meta Web services to support identification and traceability between layers of mashups to ensure a trusted environment. Trust establishment requires transparency of the responsibility and obligations of all participants, as well as methods and technologies to support it. Both approaches require a mechanism that globally supports traceability in the mashup environment, whereas PAP uses provenance data and policy compliance statements for evaluating the trustworthiness between two players.

8 Conclusion and Future Work

With more linked data being published on the Web, the next big challenge of Semantic Web could be the effective and appropriate use of linked data in mashup applications. We foresee the importance of the integration of policy frameworks into mashup development to facilitate both content publishing and content reuse for linked data. In this paper, we presented the requirements of policy support in mashup environments and a method for incorporating policies into the life cycle of mashup development. Then, we presented a tool called Policy Aware Pipes

(PAP) that integrates into the the existing Semantic Web Pipes mashup tool to implement policy interactions. Finally we walked through a scenario about building a mashup application for linked data and demonstrated the feasibility and benefits of using PAP.

We envision that PAP could be extended to support different policy languages for policy chekcing. We plan to build an interface to connect to external policy reasoners, and a wrapper to understand the results returned from different policy reasoners. Another challenge we plan to undertake is to further explore how to bring policy awareness into the development workflow of tools like Semantic Web Pipes. An interesting future project might be to integrate policy into the mashup development cycle by using a coding paradigm called Aspect Oriented Programming (AOP) [1]. By considering the policy as an *aspect* of the mashup system, AOP helps to separate the core concerns of a mashup, for example manipulating linked data, with crosscutting concerns such as policy conformance and enforcement. Such a system would allow a more flexible and modularized integration of policy into the content reuse life cycle.

Acknowledgements

The authors wish to thank their group members for their contributions to this project. This work was supported in part by NSF Cybertrust award NSF CT-M: 0831442 and AFOSR YIP award FA9550-09-1-0152.

References

1. A look at aspect-oriented programming. `http://www.ibm.com/developerworks/rational/library/2782.html`.
2. BBC Backstage Licence. `http://ideas.welcomebackstage.com/node/4`.
3. BBC Backstage SPARQL Endpoint. `http://welcomebackstage.com/2009/06/bbc-backstage-sparql-endpoint/`.
4. CCPlus. `http://wiki.creativecommons.org/CCPlus`.
5. Frequently Asked Questions. `http://backstage.bbc.co.uk/archives/2005/05/faq.html`.
6. R. S. Gerber. Mixing it up on the web: Legal issues arising from internet ”Mashups”. *Intellectual Property & Technology Law Journal*, 18(8):11, Aug. 2006.
7. D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data mash-up tool for casual users. *Web Semant.*, 6(4):274–282, 2008.
8. M. Jarrar and M. D. Dikaiakos. MashQL: a query-by-diagram topping SPARQL. In *Proceeding of the 2nd international workshop on Ontologies and nformation systems for the semantic web*, pages 89–96, Napa Valley, California, USA, 2008. ACM.
9. L. Kagal, C. Hanson, and D. Weitzner. Using dependency tracking to provide explanations for policy management. In *Proceedings of the 2008 IEEE Workshop on Policies for Distributed Systems and Networks - Volume 00*, pages 54–61. IEEE Computer Society, 2008.

10. A. Khalili and S. Mohammadi. Using logically hierarchical meta web services to support accountability in mashup services. In *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, pages 410–415, 2008.
11. G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee. Media meets semantic web – how the BBC uses DBpedia and linked data to make connections. In *The Semantic Web: Research and Applications*, pages 723–737. 2009.
12. D. Le-Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, and C. Morbidoni. Rapid prototyping of semantic mash-ups through semantic web pipes. In *Proceedings of the 18th international conference on World wide web*, pages 581–590, Madrid, Spain, 2009. ACM.
13. Linked Open Data Project. `http://linkeddata.org/`.
14. Mashup (web application hybrid). `http://en.wikipedia.org/wiki/Mashup`.
15. P. Miller and R. Styles. Open data commons, a license for open data. In *Linked Data on the Web (LDOW2008)*.
16. RDF Site Summary 1.0. `http://web.resource.org/rss/1.0/`.
17. O. Seneviratne, L. Kagal, and T. Berners-Lee. Policy aware content reuse on the web. In *The 8th International Semantic Web Conference*, 2009.
18. D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, 2008.
19. D. J. Weitzner, J. Hendler, T. Berners-lee, and D. Connolly. Creating a Policy-Aware web: Discretionary, rule-based access for the world wide web. *IN ELENA FERRARI AND BHAVANI THURAISINGHAM, EDITORS, WEB AND INFORMATION SECURITY. IOS*, 2004.
20. J. Zou and C. Pavlovski. Towards accountable enterprise mashup services. In *e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on*, pages 205–212, 2007.