Usage Policies for Document Compositions

Sebastian Speiser and Rudi Studer

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe Service Research Institute (KSRI)
`firstname.lastname@kit.edu`

Abstract. The availability of contents and information as linked data or Web services, i.e. over standardized interfaces, fosters the integration and reuse of data. One common form of information integration is the creation of composed documents, e.g. in form of dynamic Web pages. Service and data providers restrict allowed usage of their resources and link it to obligations, e.g. only non-commercial usage is allowed and requires an attribution of the provider. These terms and conditions are currently typically available in natural language which makes checking, if a document composition is compliant with the policies of the used services, a tedious task. In order to make it easier for users to adhere to these usage policies, we propose to formalize them, which enables policy-aware tools that support the creation of compliant compositions. In this paper we propose an OWL model of document compositions and show how it can be used together with the policy language AIR to build a policy-aware document composition platform. We furthermore present a use case and illustrate how it can be realized with our approach.

1 Introduction

More and more data on the Internet is published with standardized interfaces, i.e. as Web services or as linked data¹. Legacy systems and data sources can be exposed with standardized interfaces by using tools, such as D2R [1] which publishes relational databases as linked data. It is also possible to integrate data from legacy systems with services, e.g. with IBM Mashup Center² which includes support for spreadsheets and SAP systems. This development fosters the reuse of information. One way to realize reuse is building dynamic document compositions. This means that a document description is created that specifies how the reused resources are composed to form the final document. A dynamic document composition system can then use this description in order to retrieve data copies from the specified resources and combine them to the composed document. Such systems are in wide use, for example in form of dynamic web pages and are conceptually similar to mashups.

¹ See e.g. <http://webservices.seekda.com> and <http://linkeddata.org>

² <http://www.ibm.com/software/info/mashup-center/>

Standardized interfaces create unlimited possibilities on a technical level to combine services and data into new applications. However in most scenarios there will be restrictions, such that not all services are freely available to everyone and not all output data belongs to the public domain. These restrictions that apply when accessing a service or using its output data are currently typically available in natural language as terms and conditions. The evaluation of these terms requires high manual effort and makes it a tedious task to check if a composed document is compliant with the policies of the used services. Usage policies are a formalization of the terms and conditions of services that can be used to support the creation of compliant compositions with policy-aware tools.

Resources may for example be only allowed to be used by specific user groups, e.g. defined by the internal structure of a company. Consider for example a resource that provides detailed financial information about a company. Access to this resource should be restricted to managers of the company. This restriction also applies to data retrieved from the resource after initial access was granted, meaning that a manager who has access to the financial information is not allowed to pass the data to third parties.

Granting someone access to data does not imply that he can use it for any purpose. Usage policies also restrict data usage after access was granted, e.g. a real-time stock quote service might prohibit to display the quotes on a public homepage. Another aspect is that a usage permission can be linked with an obligation that has to be fulfilled. An example is a weather service that requires attribution of the provider whenever a weather forecast is shown. Other typical obligations include payments or to “share-alike” (i.e. a composition of resources has to specify the same policy as its components with a share-alike obligation).

In this paper we propose that usage policies are a requirement for dynamic document compositions that are used in environments that do not only consist of resources in the public domain. Reusability of services and easy access to data sources are key features of such systems.

The contributions of this paper include (i) a formal model for usage policies for dynamic document compositions (Section 3.1), (ii) a description how a platform for document compositions can support usage policies (Section 3.3), and (iii) an analysis of shortcomings of existing policy formalisms, including a proposal how they can be overcome (Section 4.4).

The rest of the paper is organized as follows: Section 2 presents a use case including a dynamic document composition platform that illustrates the requirements for a usage policy language. In Section 3 we introduce our approach to usage policies and show in Section 4 how it can be applied to the use case. Finally Section 5 gives an overview of related work and we conclude the paper and give an outlook to future work in Section 6.

2 Use Case

We consider a company that deploys a system that allows to include data from Web services and linked data sources as contents into composed documents. The

system is used to create documents and Web pages for internal and external use. Identification of users is realized by a centralized system maintaining the following hierarchy of groups: public (all users), internal (employees of the company), and manager.

The document system includes a service repository where both internally provided services and wrappers for external services are described. Note that the term service in this context also denotes linked data sources in the sense of data services. The document manager supports the search of services in the repository and the insertion of data pieces obtained through service calls into documents. It automatically checks usage policies and in case of non-compliance gives the reasons and hints how compliance can be achieved. We consider the following services:

1. A service provided by the manager Bob that delivers a text motivating every employee to increase the shareholder value. The text was obtained externally and was published under a Creative Commons attribution license (CC-BY³).
2. A real-time stock quote service, which delivers the current stock price of the company. It is provided by the external provider Powerquote. It is only available for company internal use and requires a payment of \$ 1.00 per call.
3. A service delivering delayed stock quotes which is also provided by Powerquote. It can be used for external documents but requires attribution of the provider.
4. A service giving access to linked data about the financial status of the company, which is only available to managers.
5. A service providing a restricted view on the same financial data. The output of this service can be made available to the public.

Based on these services, we describe in the following two documents that employees of the company want to build.

First is a Web page for the intranet that shows the current stock quote and the motivational text. The builder of the document wants help of the document composition tool in order to define a policy for the Web page that is compliant with the policies of the used services. Furthermore in case that the document is classified as non compliant, he expects the tool to give him the reason for the decision.

The second document should be distributed to the public in order to convince investors to buy the shares of the company. The document's policy must allow use by the public and therefore only services should be made available that do not have a contradicting policy. The document should contain a stock quote of the company and a description of the financial status.

These tasks pose different requirements to the policy engine. The content of the first document is predetermined and the resulting policy should be modified accordingly. In the case of the second document, the desired policy is given and the content is chosen dependently.

³ <http://creativecommons.org/licenses/by/3.0/>

3 Formal Usage Policies

This section describes our approach to formal usage policies for dynamic document compositions. First we propose in Section 3.1 a model of data sources, services and compositions that is useful from a policy view. Then in Section 3.2 we present requirements for the policy formalism and introduce AIR, a general purpose policy language by Kagal et al. [2], that is used by our approach. Furthermore we define some generally applicable policies using our proposed model. Finally in Section 3.3 we describe which operations a dynamic document composition platform must implement in order to support usage policies.

Fig. 1. Dynamic Document Composition Model for Usage Policies

3.1 A Model for Dynamic Document Compositions

Our model of dynamic document compositions is visualized in Figure 1. The basic component in our model is the data item, which can be for example an image, a text or a stock quote. A copy of a data item denotes a digital manifestation of the data item with an attached usage policy. This can be realized for example by annotating an HTML document with an RDFa representation of the policy, storing it in the metadata of a picture file or referring to an external file in an XML document. In our model we abstract from the technical realization and assume that a copy of a data item has some content and a usage policy. The usage policy of a data item copy regulates the transmission of the copy to other agents. We assume for now that there are no derivations of data items and the only possible actions are viewing it and transmitting it to others. Viewing a data item is assumed to be allowed for those who have a copy of it. Therefore

only transmission is regulated which is done by the action *send* that defines the actor who is sending it, the recipient and the corresponding data item copy. Note that data item copies are sent, which means that also the attached policy is transmitted.

A service is a provisioning of a specific data item copy, by a specific agent (the service provider) over a standardized interface, e.g. as a Web service, or as linked data. An agent (the service user) can obtain the data item copy of a service by executing a *read* action with the service as its *source*. The allowed read actions are regulated by the *policy* of the service and the obtained data item copy will be associated with the *output policy* of the service. The counterpart of reading a service is sending its data item copy, an action executed by the service provider with the service user as *recipient*. This implies that the policy of the service can only allow read actions for which the provider has the right to execute the corresponding send action. The policy of the service however does not have to be the same but can be more restrictive.

A composed document is a collection of data item copies and a layout function. The layout function merges the copies into a digital artifact that is sent to the viewer of the document. We consider the layout function for now as irrelevant for the policies and therefore treat the viewing of a document as a sequence of send actions where the document provider sends each data item copy to the document viewer. Obtaining the data item copies of the document is the execution process which consists of reads of services. Viewers of the document get the results of the display process which consists of sends of the data item copies. Therefore a document consists of a number of data item copies, a document owner, an execution process, a display process and a layout function. In the following we will refer to dynamic document compositions also as compositions, as this matches our view on these documents from a policy standpoint.

The display process is defined in terms of send operations that have an agent as recipient that is defined by the display process as its user role. This agent is bound during an execution to the specific user that views the document. The possible agents that can bind the user role are restricted by the composition's read policy.

Listing 1 formalizes the model as description logic formulas that can be represented in OWL⁴. This approach was taken as it supports interoperability in a Web scenario and existing reasoners can be used to infer implicit knowledge, e.g. a condition requiring a user to be a manager can be inferred to be fulfilled by a user that is the head of a department.

3.2 Policy Formalism

A usage policy does not only regulate the possible recipients but also required and forbidden actions in the process that uses an action. So basically a policy classifies whether a process is compliant or not. Such policies can be formalized as rules. In case of non-compliance the process creator wants to find out the reason for this

⁴ OWL: Web Ontology Language, <http://www.w3.org/2004/OWL/>

Listing 1. Ontology about Dynamic Document Compositions

```

DataItemCopy  $\sqsubseteq \forall=1 \text{ content.DataItem} \sqcap \forall=1 \text{ policy.Policy}$ 
Service  $\sqsubseteq \forall=1 \text{ policy.Policy} \sqcap \forall=1 \text{ outputPolicy.Policy} \sqcap$ 
       $\forall=1 \text{ provider.Agent} \sqcap \forall=1 \text{ copy.DataItemCopy}$ 
Composition  $\equiv \forall \text{ copy.DataItemCopy} \sqcap \forall=1 \text{ owner.Agent} \sqcap$ 
       $\forall=1 \text{ execProcess.Process} \sqcap$ 
       $\forall=1 \text{ dispProcess.Process} \sqcap \forall=1 \text{ policy.Policy}$ 
Process  $\sqsubseteq \forall \text{ contains.Action}$ 
Action  $\sqsubseteq \forall=1 \text{ actor.Agent}$ 
Read  $\sqsubseteq \text{Action} \sqcap \forall=1 \text{ source.Service} \sqcap \forall=1 \text{ copy.DataItemCopy}$ 
Send  $\sqsubseteq \text{Action} \sqcap \forall=1 \text{ copy.DataItemCopy} \sqcap \forall=1 \text{ recipient.Agent}$ 

```

classification. This can be realized by a rule-system that supports dependency tracking, which means that the derived facts which led to a conclusion are saved. The owner of a non compliant document display process can then for example see that the reason for this decision is that an image is sent to the document viewer without sending a data item which attributes the image creator. As we assume that when checking a process for policy compliance we know all facts about the involved agents, actions and policies, we can evaluate the rules with a local closed world assumption. Otherwise an open world engine would not classify a process which is missing a required payment action as non compliant as long as there is the possibility that its existence can still be stated, e.g. in another file. As all knowledge about a document is maintained by the document manager in a central place, we can disregard these possibilities.

Accountability in RDF (AIR) is a policy language that comes with a reasoner that supports our requirements [2]. Policies are specified as RDF models in the N3 syntax extended with expressions to introduce quantified variables. An `air:Policy` defines via the `air:rule` relation one or more rules. Rules can use variables which are introduced by `@forSome :VAR1` or `@forAll :VAR2` statements. A rule evaluates its `air:if` clause, if it finds a binding for used variables, it further processes its `air:then` clause, which either specifies the next rule to evaluate or it `air:asserts` new axioms, e.g. that a variable is `air:compliant-with` the policy. Rules can also have `air:else` clauses which are evaluated when the if-clause does not have a binding. Besides asserting new axioms and specifying the next applicable rule, an if- or else-clause can also include an `air:description` which is a natural language explanation of the decision, which can use the bound values of variables. One or more policy files can be used to classify objects defined in one or more knowledge bases. Only explicitly stated individuals in the knowledge bases and those inferred from the policy rules are considered for the classifications.

The policies can also assert axioms other than (non-)compliance relations. For example the policy of a composition can require the actor that reads the

composition to belong to a specific class and assert that the agent playing the user role in the composition's display process therefore always belongs to this class.

The following is a general policy that applies the `process-action-rule` to all instances of `Process` (see `process-rule`). The `process-action-rule` states that if a process `:P` contains an action with a policy `:POLICY` to which `:P` is not compliant, the whole process is not compliant with the `ProcessPolicy`. If such an action does not exist, then `:P` is compliant with `ProcessPolicy`:

```
@forAll :P.
:ProcessPolicy a air:Policy ;
  air:rule :process-rule .
:process-rule a air:BeliefRule;
  air:if { :P a a:Process. };
  air:then [air:rule :process-action-rule].

:process-action-rule a air:BeliefRule;
  air:if { @forSome :ACTION, :POLICY.
    :P a:contains :A.
    :A a:policy :POLICY.
    :P air:non-compliant-with :POLICY. };
  air:then [air:assert [air:statement
    { :P air:non-compliant-with :ProcessPolicy .}]];
  air:else [air:assert [air:statement
    { :P air:compliant-with :ProcessPolicy .}]]].
```

Further general rules include propagation rules, which state:

- A read operation assigns its data item copy the same policy as the output policy of the read service.
- A send operation has to fulfill the policy of the data item copy that is sent.
- All actions in the processes of a composition have as actor the owner of the composition.

3.3 Policy-Awareness for Document Composition Platforms

In the following we describe the usage policy view of a platform for dynamic document compositions. The two main components are the document manager and the service repository.

The service repository stores descriptions of all available services. The non-functional part of a service description includes usage policies which are split in three files: (i) a knowledge base defining the service, its provider, its data item copy, (ii) a policy, and (iii) an output policy. The repository supports the retrieval of a description for a given service. Another functionality could be to search services by specifying queries on the service descriptions, including restrictions on usage policies.

The document manager maintains for each composition two usage policy relevant files. One is the usage policy of the composition itself, which regulates allowed read operations on the document. The other file is a knowledge base

which contains definitions of the document, its owner, its data item copies and its processes. If a new service call is inserted in a composition, the following actions are taken:

1. Import the service knowledge base in the knowledge base of the composition.
2. Add a new `DataItemCopy` .
3. Add a new `Read` action having the new service as **source** to the document's execution process, the **actor** is the **owner** of the document, and the **copy** is the newly created `DataItemCopy`.
4. Add a new `Send` action to the display process, which sends the data item copy to the user role.
5. Check compliance for the execution and display processes. If they are not compliant show the document owner the justifications and help him to make the right changes.

The document manager allows a composition to be in a non-compliant state, which can occur in the process of building or changing a document. However it only executes and displays compliant compositions.

4 Realization of Use Case

In this section we describe how the services and documents presented in Section 2 can be realized with our approach to usage policies. First we explain the policies for the single services and afterwards we show the steps of document creation that involve decisions by the policy engine.

4.1 Motivational Text with CC-BY License

The service itself is provided by Bob freely for every user. Therefore we do not describe the service policy but concentrate on the output policy which is a formalization of the Creative Commons attribution license. The service definition including the data item and the required attribution in N3 syntax:

```
@prefix : <http://.../bob/motivation#>.
@prefix pol: <http://.../bob/motivationPolicy#>.
```

```
:MotivationService a m:Service;
  m:provider users:Bob;
  m:policy pol:MotivatonServicePolicy;
  m:outputPolicy pol:MotivationOutputPolicy.
:AttributionData a m:DataItemCopy;
  m:content "Created by XYZ".
```

The part of the policy file, defining the output policy:

```
@prefix f: <http://.../bob/motivation#>.
@prefix : <http://.../bob/motivationPolicy#>.

@forAll :P.
:MotivationOutPolicy a air:Policy;
  air:rule :PolicyApplicable.
```


Thus far it is stated that the policy requires the evaluation of the `:PolicyApplicable` rule, which is defined in the following and specifies that all applicable processes are subject to the `:AttributionRule`. A process is considered applicable if it contains a send action referring to a data item copy with a `:MotivationOutPolicy`.

```
:PolicyApplicable a air:BeliefRule;
  air:if { :P a m:Process;
    m:contains [a m:Send;
      m:copy [m:policy :MotivationOutPolicy]]. };
  air:then [ air:rule :AttributionRule ].

:AttributionRule a air:BeliefRule;
  air:if { :P m:contains [a m:Send;
    m:copy f:AttributionData]. };
  air:then [ air:assert [ air:statement
    { :P air:compliant-with :MotivationOutPolicy.}]];
  air:else [ air:description( :P " has to attribute XYZ!." );
    air:assert [ air:statement
    { :P air:non-compliant-with :MotivationOutPolicy.}]].
```

The `:AttributionRule` checks if the process contains an action sending the attribution text. If this is the case, the process is compliant. If not, it is classified as non-compliant and the decision is justified by a natural language description requiring attribution. One future goal is to be able to represent also the other Creative Commons licenses. One example would be the non-commercial clause, which could be interpreted as granting compliance only if a process does not include payment actions and its policy does not require any payments. However it is currently not entirely clear what users consider as commercial usage⁵.

4.2 Financial Status Report Services

The public financial status report is available for everybody and can be sent to everybody, therefore its usage policy is not of interest for us, as it would just classify every process as compliant.

The internal report however is only available to managers. In the following we just show the policy fragment checking identities of senders and recipients of the document and assume that applicability was already checked by previous rules and the variables `:P` (process) and `:S` (send action) are already bound.

```
:internal-report-auth-rule a air:BeliefRule;
  air:if { :S m:actor [a users:Manager]. };
  air:then [air:rule :internal-report-rec-rule ];
  air:else [air:description("Only Managers can access the report");
    ... :P air:non-compliant-with :InternalReportPolicy. ].
```

⁵ Creative Commons has commissioned a study about the understanding of commercial vs. non-commercial use. An interim report can be found at:
http://mirrors.creativecommons.org/nc-study/NC_Use_Study_Interim_Report_20090501.pdf

```

:internal-report-rec-rule a air:BeliefRule;
  air:if { :S m:recipient [a users:Manager]. };
  air:then [... :P air:compliant-with :InternalReportPolicy.];
  air:else [air:description("Report can only be sent to Managers");
    air:assert [ ... :P air:non-compliant-with ...]].

```

Additionally to the report's policy, also the providing service has an policy allowing only managers to read from the service, which is very similar to the `:internal-report-auth-rule`, but restricting a `Read` instead of a `Send` action.

4.3 Stock Quote Services

The delayed stock quote service has a policy which is similar to that of the motivational text with the difference that `Powerquote` has to be attributed.

In the following we concentrate on the policy part of the real-time stock quote service that requires the payment, as authentication and recipient restrictions were already shown in the policy of the internal financial report. We assume that `:P` (process) is already bound and a payment action is defined in some payment ontology:

```

:real-time-pay-rule a air:BeliefRule;
  air:if { :P m:contains [a pay:Payment;
    m:recipient users:Powerquote;
    pay:amount "$ 1.00" ]. };
  air:then [ ... :P air:compliant-with :RealTimePolicy. ]];
  air:else [ air:description("Service requires payment of $1.00");
    ... :P air:non-compliant-with :RealTimePolicy ].

```

4.4 Document Creation

Making a new dynamic document composition involves creating two files, one that contains a knowledge base describing the document, its data item copies and processes, and one policy file. The following is an exemplary description that is common for both documents of the use case:

```

:Document1 a m:Composition;
  m:policy pol:Document1Policy;
  m:execProcess :ExProc;
  m:dispProcess :DispProc.
:ExProc a m:Process.
:DispProc a m:Process;
  m:userRole :UserRole.
:UserRole a m:Agent.

```

The corresponding policy file defines at least two rules. The `:document1-user-rule` propagates the user restrictions that are defined in the `:document1-read-rule` to the user role of the display process, which is the user that represents all possible recipients of the send actions.

```

@forAll :M, :P, :U, :R.
:Document1Policy a air:Policy;
  air:rule :document1-user-rule;
  air:rule :document1-read-rule.
:document1-user-rule a air:BeliefRule;
  air:if { :M a m:Document;
    m:dispProcess :P.
    :P m:userRole :U. };
  air:then [air:assert [ air:statement { :U a users:Public. }]].
:document1-read-rule a air:BeliefRule;
  air:if { :P a m:Process;
    m:contains :R.
    :R a m:Read;
    m:source [m:policy :document1-policy];
    m:actor :U. };
  air:then [air:rule :document1-read-auth].
:document1-read-auth a air:BeliefRule;
  air:if { :U a users:Public . };
  air:then [ ... :P air:compliant-with :document1-policy. ]];
  air:else [ ... :P air:non-compliant-with :document1-policy. ]].

```

In the following we discuss how these files have to be modified for both documents during their construction.

Document 1: Intranet Motivation Page. First step is the insertion of the motivational text service into the document. This is realized by importing the knowledge base describing the service and adding the following statements to the knowledge base describing the composition:

```

:Motivation a m:DataItemCopy.
:Document1 m:copy :Motivation.
:ExProc m:contains [a m:Read;
  m:source f:MotivationService;
  m:copy :Motivation ].
:DispProc m:contains [a m:Send;
  m:copy :Motivation;
  m:recipient :UserRole ].

```

Afterwards the policy engine classifies :DispProc as non-compliant and gives as justification “:DispProc has to attribute XYZ!” and refers to the :AttributionRule. The composition owner now inserts an attribution, which is reflected in the document’s knowledge base as:

```

:DispProc m:contains [a m:Send;
  m:copy f:AttributionData;
  m:recipient :UserRole ].

```

As a consequence the policy engine classifies the process as compliant. The next step is to insert the real-time stock quote service and adding the corresponding statements to the knowledge base. The consequences are that again the display process is marked as non-compliant because the stock quote is sent to the

user role which is only restricted to `users:Public`. As a consequence the document owner changes the policy by replacing the user role propagation and read actor restriction to `users:Internal`. However the display process is still non-compliant with the new reason “Real-time service requires payment of \$ 1.00”. Therefore the user adds a payment to the execution process, which is reflected by the following changes to the document description, that lead to a compliance classification and allow the execution and displaying of the document:

```
:ExProc m:contains [a pay:Payment;
                    m:recipient users:Powerquote;
                    pay:amount "$ 1.00" ].
```

Document 2: Public Investor Advertisement. For this document the delayed stock quote service and the public financial status report are inserted. The only requirement of their policies is that an attribution of Powerquote is inserted. How this works was already discussed for the previous document. The interesting part here is to select the right services that do not require a change in the document’s read policy, which disqualifies the internal financial report and the real-time stock quote service. Such a reasoning over properties of policies is currently not supported by the AIR policy engine. For different usage policy scenarios this is a required functionality and in future we plan to develop an extension that supports this.

This is also useful for increasing the expressiveness of policies. For example a data item’s policy could classify processes as non-compliant if they send out the item and have a policy that requires a payment (a possible interpretation of a non-commercial use clause).

Fig. 2. Representing a policy as a tree of nested rules

In order to check such properties we treat policies with their nested rules as trees that have nodes representing the restrictions imposed by the if clauses and as leaves the (non-)compliance classifications. Figure 2 shows the tree of a rule that requires either a payment or restricts the possible recipients of a send action. These trees can then be used to check if a restriction is required by the policy, by checking if all paths from the tree root to a compliance leaf state the restriction. With negations several other properties can be checked, e.g. checking if the negation of a restriction is not required by the policy is equivalent to checking if there is a possible path to a compliance leaf including the restriction.

The above examples could be realized in the following ways: for checking if a policy restricts the possible users, we can check if it requires a restriction that states that the user belongs to a class that is not a super-class of the desired user class. The absence of required payments can be ensured by checking if the policy has a path to a compliance leaf that does not include a payment action.

Discussion. This section showed how the use case can be realized with our approach. Based on the example services, we illustrated how different policy aspects can be formalized using the proposed model. The process of creating the second document revealed that currently not all required policy operations are supported, namely the reasoning about properties of policies. We explained how this can be fixed without a need to change our model. The use case of the first document showed how service composition tools can support the building of compliant documents using formal policies and natural language explanations.

5 Related Work

Usage policies control access and usage of resources. Traditional access control mechanisms such as the role based model build abstractions which implicitly define an access control matrix. The matrix specifies for each combination of subject and resource if the access is allowed or not [3]. Such an approach can also be used with credentials instead of subject identities [4]. Access control provides proven and useful methods which are part of usage control, but is missing two aspects which are important for dynamic document compositions and services in general: obligations and usage control after initial access was granted.

Park et al. present the UCON_{ABC} approach which is a theoretical approach that handles usage control, including obligations [5]. While the UCON_{ABC} model gives an interesting base for diverse applications such as digital rights management, privacy and usage policies, it does not provide a general purpose solution that can just be applied to specific problems. Therefore it complements our approach with a theoretical foundation.

Creative Commons⁶ defines several licenses for digital objects and an RDF formalization [6]. Such formal licenses are a form of usage policies as defined in this paper. However the language is restricted to the Creative Commons licenses, which have a limited scope. They are generally addressed to the public and therefore do not allow to specify rights for single users or user groups. Also the set of actions is restricted and does for example not include payments.

More general languages are used for digital rights management (DRM), e.g. ODRL [7]. These languages are based on XML and allow the specification of complex policies. They are mainly used for protecting digital media files and therefore contain actions from this area and cover areas such as encryption, media encodings and other domain specific aspects that are not applicable to dynamic document compositions. Another aspect is that these languages are

⁶ <http://creativecommons.org>

lacking formal semantics which can lead to ambiguities when several heterogeneous service providers and consumers exist.

Gangadharan et al. developed an ODRL profile for service licenses (ODRL-S) [8]. Such a profile adapts ODRL for a specific application area. The defined actions are rather technical, such as for example composition, which can be used to generally allow or forbid the usage of the service in combination with other services. It is for example not possible to express that a document can only be combined with other documents or a translation service. There is also no support for data usage rights in ODRL-S.

A recent work by Seneviratne et al. presents a solution for a similar scenario as ours [9]. They analyze the use of images that have a Creative Commons license in Web pages. The work also uses an approach based on AIR policies. The key difference is that Seneviratne et al. concentrate on copyright issues in static pages. In contrast our work presents an approach for dynamic documents and their creation process. Also we focus on usage policies that express more explicit restrictions based on user identification and also commercial aspects. Due to these differences our work complements their approach. Interesting overlaps do exist, as it is shown in Section 4 where we modeled a CC-BY license with our method.

6 Conclusions and Future Work

Dynamic document compositions are an powerful approach to the integration of data sources and services. The underlying model is flexible and treats widely used systems such as dynamic Web pages as a special case. In order to motivate service, data and content providers to participate in such scenarios they have to be ensured that their rights and restrictions about data and service usage are respected. We presented an approach to formal usage policies that can be used by providers to express their terms and conditions. We discussed how platforms for dynamic document compositions and service repositories can be extended to support users to adhere to usage policies.

Our approach is based on an OWL ontology, modeling the domain of dynamic document composition, which is used to describe the usage policy view of services and documents. Policies are expressed with rules that are evaluated in a local closed world environment and classify composed documents as compliant or non compliant. Evaluation is done by the AIR policy engine which not only returns the classifications but also the justifications for its decisions. This is used to give the user reasons why his document is not compliant and hints how he can change this.

For future work we plan to extend our model to include the collaborative creation of documents. For this we want to extend the simple composed document model to include more complex processes and services like for example translation and publishing services, and processes to assign the creation of new contents to providers. For a description of a powerful collaborative document creation system, we refer the reader to the work of Schuster et al. on document mashups [10].

Based on a planned implementation at hand, we want to extend the document manager's capabilities to automatically adjust a document or its policy in order to match certain policy requirements of used services. For example a user could specify that required attributions or user restrictions should be included automatically.

Another important point is to extend the capabilities of the policy engine to reason about properties of policies. For this it is necessary to inspect not only one trace through a policy's rules that leads to a compliance decision, but to have a look at the whole tree of policy rules and argue about the existence of paths with specific properties. We have argued that this is necessary in order to gain expressivity that is needed for common policy clauses, such as for example when a content creator wants to forbid that somebody is charging money for the creator's work.

Acknowledgments. The authors wish to thank Nelly Schuster, Sudhir Agarwal, Robin Fischer and Christian Zirpins for the useful discussions. This work was supported by the European project SOA4All.

References

1. Bizer, C., Cyganiak, R.: D2R Server Publishing Relational Databases on the Semantic Web. In: WWW 2003, Posters (2003)
2. Kagal, L., Hanson, C., Weitzner, D.: Using Dependency Tracking to Provide Explanations for Policy Management. In: 2008 IEEE Workshop on Policies for Distributed Systems and Networks, pp. 54–61 (2008)
3. Lampson, B.W.: Protection. In: Proc. Firth Princeton Symposium on Information Sciences and Systems, pp. 437–443. Princeton University, Princeton (1971)
4. Agarwal, S., Sprick, B.: Specification of Access Control and Certification Policies for Semantic Web Services. In: E-Commerce and Web Technologies, pp. 348–357 (2005)
5. Park, J., Sandhu, R.: The UCON_{ABC} Usage Control Model. ACM Transactions on Information and System Security 7, 128–174 (2004)
6. Abelson, H., Adida, B., Linksvayer, M., Yergler, N.: ccREL: The Creative Commons Rights Expression Language. W3C Submission (2008)
7. Iannella, R.: Open Digital Rights Language (ODRL) Version 1.1 (2002)
8. Gangadharan, G.R., D'Andrea, V., Weiss, M.: Service Licensing Composition and Compatibility Analysis. Int. J. Cooperative Inf. Syst. 17(3), 301–317 (2008)
9. Seneviratne, O., Kagal, L., Weitzner, D., Abelson, H., Berners-Lee, T., Shadbolt, N.: Detecting Creative Commons License Violations on Images on the World Wide Web. In: WWW 2009 (2009)
10. Schuster, N., Zirpins, C., Tai, S., Battle, S., Heuer, N.: A Service-Oriented Approach to Document-Centric Situational Collaboration Processes. In: 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (2009)