On the Composition of Digital Licenses in Collaborative Environments

Marco Mesiti, Paolo Perlasca, and Stefano Valtolina

DI, Department of Computer Science, University of Milano, Italy
{mesiti,perlasca,valtolina}@di.unimi.it

Abstract. In the era of Web 2.0, users are not any longer just consumers of resources but they can actively produce, share and modify content, by composing and enhancing digital resources and services. In this context, the intellectual property of the users collaborating in authoring activities should be preserved. Starting from a model for digital licences generation and management useful in collaborative environments like the Web 2.0, in this paper we propose the algorithms of a DRM component responsible for the composition and modification of digital resources and the generation of the related licenses. Then, the paper presents a compliant architecture based on a composition of web services.

1 Introduction

The complexity and expanding scale of most collaborative projects being carried out nowadays in the context of Web 2.0 require more cooperation among users in the production of digital contents and services. We are also observing the generation of communities of users (belonging to the same company or group of interest) working together for the creation of new resources (like wikies, social networks, and mashups). Users are not any longer just consumers of resources but they can actively produce, share and modify content/services eventually created by other users. In this context, the intellectual property of the users collaborating in the authoring activities should be preserved.

Different Rights Expression Languages (like ccREL [6], ODRL [8], MPEG-21 REL [10]) have been develop for the specification of licenses to preserve the intellectual property. These proposals differ from the scope and the granularity according to which it is possible to specify and manage each aspect directly or indirectly related to the license specification and management processes. Digital Rights Management (DRM) systems enable the creation, adaptation, distribution and consumption of multimedia contents and services according to the permissions and constraints specified by the content creators or rights issuers [1, 13]. MPEG-21 REL [10] and ODRL [8] natively support the specification of rights for the modification and composition of resources. Few approaches for the specification of licenses of composed resources have been proposed in the context of the creative common licences expressed by ODRL [3–5]. However, these licences have a different purpose than those created by MPEG-21 REL

H. Decker et al. (Eds.): DEXA 2013, Part I, LNCS 8055, pp. 428–442, 2013.

and current DRM systems are mainly tailored for the protection of resources rather than for supporting users in the composition process.

This paper provides the building-blocks for the realization of a tool helping an user to visualize his/her own resources (or the resources created by the communities she/he belongs to), to compose resources, and to generate related licenses. We first introduce a formal model that is compliant with the MPEG-21 REL for the representation of licenses as collection of grants. By abstracting many verbose details of MPEG-21, we create the basis for reasoning on the composition issues. Moreover, the model points out different approaches for specifying the principal of a grant (either by specifying a single user identifier, a group of user identifiers or a predicate to be evaluated by considering user certificates) that can be very useful in our collaborative environments to reduce the number of licenses to be generated and, at the same time, to make a broader usage of digital resources. Then, we provide an approach for evaluating the weak and strong compatibility of two grants that is the basis for the composition and update of resources. In the evaluation of weak compatibility, the user profile and the conditions of grants are not considered. This is useful in the early stage of composition design to quickly checking the basic conditions of composition without loosing time in the evaluation of the user profile and also when external services needed for their evaluation are unavailable. Afterward, to actually enable the user to compose resources, and to generate the final license, user profile and conditions of grants are taken into account in the evaluation of strong compatibility. We finally propose an architecture supporting the composition of resources and the generation of a new license based on the components' licenses. Key features of this architecture are the weak and strong compatibility service for the two-steps evaluation of licenses compatibility and the process of resources aggregation and generation of the corresponding license.

The paper is organized as follows. Section 2 presents the license data model and how a license is evaluated. Section 3 deals with the issue of checking whether two grants are compatible for composition and can be exercised at the current time. Section 4 provides the basic algorithms for the generation of a new license when resources are composed or updated, whereas Section 5 deals with the enabling architecture. Related work and concluding remarks are finally presented.

2 License Data Model

In this section we provide a formal model for the representation of licenses that supports the specification of communities to whom a grant is released. Finally, we discuss the mechanism for the evaluation of an access request.

Principals and Issuers. The principals to whom rights are granted can be specified through the user identifiers or relying on the possession of a given certificate. For the sake of simplicity, in our model the fact that users hold certificates are represented through predicates. Such predicates are verified on a given user, if and only if she/he holds the corresponding certificate released

$$u \downarrow p \text{ iff } \begin{cases} p = u & p \in \mathcal{U} \\ u \in p & p \text{ is a set} \\ p(u) & isPred(p) \end{cases} \quad \text{(a)}$$

$$p_1 \sqcap p_2 \text{ iff } \begin{cases} p_1 = p_2 & p_1, p_2 \in \mathcal{U} \\ p_1 \cap p_2 \neq \emptyset & p_1, p_2 \text{ are sets} \\ p_1 \in p_2 & p_1 \in \mathcal{U} \text{ and } p_2 \text{ is a set} \\ p_2 \in p_1 & p_2 \in \mathcal{U} \text{ and } p_1 \text{ is a set} \\ isPred(p_1) \vee isPred(p_2) \end{cases} \quad \text{(b)}$$

Fig. 1. Predicates on Principal specifications

by a Credential Authority. A predicate can also be evaluated on a set of users and, in this case, it is verified when each user in the set holds the corresponding certificate. The use of certificates in our collaborative environment is particularly useful to specify with a single license the community of users that can exercise a given right. In the remainder, $\mathcal{U}$ denotes the set of user identifiers.

Definition 1 (Principal). *The principal of a license can be: a unique identifier associated with a user, a set of user identifiers, or a predicate* m*. A user* i *satisfies the predicate* m*, if and only if the user i holds the corresponding certificate* m*.*

Example 1. Let $\{i_1, ..., i_{10}\} \subseteq \mathcal{U}$ be a set of user identifiers. The principal of a license can be: i_1, the set $\{i_3, i_5, i_7\}$, or `uniMi`, where `uniMi` is a predicate assessing the employment at the University of Milan. In the paper we will use the user identifiers `Alice`, `Bob`, and `Tom`. `Alice` and `Tom` are `uniMi` employees. □

The user issuing a license is named *issuer*. The issuer of a license is always a single user identifier. With $\mathcal{I}$ we denote the set of issuers, $\mathcal{I} \subseteq \mathcal{U}$. Given a principal specification p, we might need to establish whether a user u is a principal. The predicate $\downarrow$ is satisfied in the cases reported in Figure 1(a) (function $isPred$ is true when the argument is a predicate). Moreover, the predicate $\sqcap$ in Figure 1(b) is used to weakly identify when two principal specifications p_1 and p_2 can have non empty intersection. Since the evaluation of predicates requires to access the user profile, when p_1 or p_2 is a predicate, $\sqcap$ is considered verified, and its actual evaluation deferred when the strong conditions are taken into account.

Resources. A Resource can be a digital work (such as an e-book, an audio file, or an image), a service (such as an email service or a transaction service), or even a fragment of information characterizing a principal (such as a name or an email address). Resources can be the aggregation of different ones. In order to allow the specification of digital licenses at different granularities (from the entire resource to one of its component), resources can be represented through a tree. Internal nodes are labeled by resource identifiers, whereas leaves are the resources. Resource identifiers are exploited as references in digital licenses. We denote with $\mathcal{O}$ the set of resources, and with $\hat{\mathcal{O}}$ the corresponding identifiers.

Example 2. Suppose that $o_{\texttt{txt}}$ and $o_{\texttt{img}}$ are resources representing a text and a picture. The resource o_c obtained by their concatenation is denoted, by adopting a JSON notation [12], as $\{\hat{o}_c : \{\hat{o}_{\texttt{txt}} : o_{\texttt{txt}}, \hat{o}_{\texttt{img}} : o_{\texttt{img}}\}\}$. □

Rights. A principal can be granted to exercise a right against a resource. Typically, a right specifies an action (or activity) or a class of actions that a principal may perform on or using the associated resource. Rights can be classified in 3 types: *use*, through which the principal can `play`, `print`, `execute` a resource; *manage*, through which the principal can `install`, `uninstall`, `move` or `delete` a resource; *transformation*, through which the principal is authorized to manipulate the resource. The MPEG-21 transformation rights we deal with are: `reduce`, `enlarge`, `modify`, `diminish`, `enhance`, `adapt` and `embed`. These rights present a subsumption relationship existing among them: the `modify` right subsumes the `reduce` and `enlarge` rights. Another subsumption exists between the transformation rights and the `play` right because, in order to transform a resource, the user should be able to access/play the resource itself. The `modify` right allows one to apply any modification to a given resource, whereas `reduce` and `enlarge` allow a specific kind of modification. The rights `adapt`, `diminish`, and `enhance` present the same semantics and subsumption relationships of the previous three rights but their application produces a new modified resource, and leave the original ones unaltered. The `embed` right allows one to attach or include another resourse in a given resource. This right is important in order to correctly operate a composition among resources. With $\mathcal{R}$ we denote the set of rights.

Conditions. MPEG-21 provides a set of conditions for the verification of terms and obligations under which rights can be exercised by a given principal. Let NC be the set of name of MPEG-21 conditions we consider.

Example 3. `ExerciseLimit`, `ValidityInterval`, and `FeeFlat` are samples of names of conditions available in MPEG-21, representing: the number of times a given right can be exercised, temporal interval of right validity, and the obligation of the payment of a fee, respectively. □

Each condition name can be associated with a set of basic constraints representing limitations (like temporal and spatial constraints) that need to be verified to consider the condition satisfied. Let NP_c be the set of property names associated with the condition identified by the condition name $c \in NC$, a constraint is $np_c\ op\ v$, where: $np_c \in NP_c$ is the name of a property, $op \in \mathcal{O}P = \{<, >, \leq, \geq, =, \neq, \in, \notin\}$ is a comparison operator, and v is a valid value for the property name np_c (the set of legal values for a property name np_c is denoted $\mathcal{V}_{np_c}$).

Definition 2 (Condition). *Let $c \in NC$ be a condition name, $\{np_{c_1}, \ldots, np_{c_m}\} \subseteq NP_c$. A condition $\overline{c}$ is a pair $(c, \{\langle np_{c_1} op_1 v_1\rangle, \ldots \langle np_{c_m} op_m v_m\rangle\})$, where $v_i \in \mathcal{V}_{np_{c_i}}$, $op_i \in \mathcal{O}P$, $1 \leq i \leq m$*

Example 4. Consider the condition names presented in Example 3. A property, named `count`, can be specified for `ExerciseLimit`, representing the number of times a right can be exercised. Therefore, the following condition $c_1 = (\texttt{ExerciseLimit}, \{\langle \texttt{count} = 5\rangle\})$ states that the associated right can be exercised up to five times. The condition $c_2 = (\texttt{ValidityInterval}, \{\langle \texttt{time} >$ "2012-01-01T00:00:00", `time` $<$ "2013-01-01T00:00:00"$\rangle\})$ represents a validity

time interval between midnight of January 1, 2012 and, midnight of January 1 2013. The condition $c_3 = (\texttt{FeeFlat}, \{\langle \texttt{amount} = 5 \rangle\})$ is satisfied whenever the principal has paid the amount of 5. □

Grants and Licenses. A grant describes the terms of a license. In the following to identify the i^{th} component of a tuple t, we use the notation $t[i]$.

Definition 3 (Grant). *A grant g is a 4-tuple $\langle p, r, \hat{o}, C \rangle$ where p is a principal, $r \in \mathcal{R}$ is a right, $\hat{o} \in \hat{\mathcal{O}}$ is a resource identifier, and C is a set of conditions.*

Example 5. Suppose that $o_{\texttt{txt}}$, $o_{\texttt{img}}$ and $o_{\texttt{mpg}}$ are resources representing a text, a picture and a piece of music. Moreover, c_1, c_2, c_3 are the conditions specified in Example 4. The following grants specify that user `Alice` can play $o_{\texttt{img}}$ without restrictions, she can adapt, play and print $o_{\texttt{txt}}$ under the satisfaction of condition c_1 and c_2. Moreover `Tom`, `Bob` and users belonging to `uniMi` can embed $o_{\texttt{mpg}}$ under the satisfaction of condition c_3. `Tom` can reduce and print, like `Alice`, $o_{\texttt{txt}}$ under the satisfaction of condition c_1 and c_2.

$g_1 =$ < Alice, play, $\hat{o}_{\texttt{img}}$, {} >
$g_2 =$ < Alice, adapt, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ >
$g_3 =$ < Alice, play, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ >
$g_4 =$ < Alice, print, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ >
$g_5 =$ < uniMi, embed, $\hat{o}_{\texttt{mpg}}$, $\{c_3\}$ >
$g_6 =$ < {Tom, Bob}, embed, $\hat{o}_{\texttt{mpg}}$, $\{c_3\}$ >
$g_7 =$ < Tom, print, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ >
$g_8 =$ < Tom, reduce, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ > □

With $\mathcal{G}$ we denote the set of grants. A License is conceptually a container of grants expressing the rights that can be exercised on the identified resources.

Definition 4 (License). *A license l is a pair $< i, G >$, where: $l[1] = i \in \mathcal{I}$ is the party issuing the license, and $l[2] = G = \{g_1, \ldots, g_n\} \subset \mathcal{G}$ is a set of grants.*

In the following, $\mathcal{L}$ denotes the set of licenses, $\mathcal{L}_u$ is the set of licenses of user u, $G_u = \{g \in \bigcup_{l[2] \in \mathcal{L}_u} \mid u \downarrow g[1]\}$ is the set of grants issued to the user u, $G_u^o = \{g \in G_u \mid g[3] = \hat{o}\}$ denotes the set of grants of user u that refers to resource o, and $G_u^{r,o} = \{g \in G_u^o \mid g[2] = r\}$ is the subset of G_u^o referring to the right r. These sets contain all the grants referring the resource o and the resources that o aggregates.

Example 6. Consider the grants of Example 5. `Bob` can issue the following licenses to `Alice`, `Tom` and to himself respectively.

- $l_1 =$ < Bob, $\{g_1, g_2, g_3, g_4, g_5\}$ > $\in \mathcal{L}_{\texttt{Alice}}$
- $l_2 =$ < Bob, $\{g_5, g_6, g_7, g_8\}$ > $\in \mathcal{L}_{\texttt{Tom}}$
- $l_3 =$ < Bob, $\{g_6\}$ > $\in \mathcal{L}_{\texttt{Bob}}$

Relying on these licenses, we can compute the following sets.

- $\mathcal{L} = \{l_1, l_2, l_3\}$, $\mathcal{L}_{\texttt{Alice}} = \{l_1\}$, $\mathcal{L}_{\texttt{Tom}} = \{l_2\}$, $\mathcal{L}_{\texttt{Bob}} = \{l_3\}$
- $G_{\texttt{Alice}} = \{g_1, g_2, g_3, g_4, g_5\}$, $G_{\texttt{Tom}} = \{g_5, g_6, g_7, g_8\}$, $G_{\texttt{Bob}} = \{g_6\}$
- $G_{\texttt{Alice}}^{\hat{O}_{\texttt{mpg}}} = \{g_5\}$, $G_{\texttt{Tom}}^{\hat{O}_{\texttt{mpg}}} = \{g_5, g_6\}$, $G_{\texttt{Bob}}^{\hat{O}_{\texttt{mpg}}} = \{g_6\}$ □

License Evaluation. The *user profile* is used to store the information of context (for example, her/his location and time of execution), the state of given constraints, and the certificates he/she holds. This information is used when granting the access to resources The profile management will be presented when discussing the system architecture.

Definition 5 (User Profile). *Let $u \in \mathcal{U}$ be an user. The profile of user u, denoted $\mathcal{P}ro(u)$, is a set of tuples $(c, np_{c_j} w_j)$, where $c \in NC$, $np_c \in NP_c$ and $w_j \in \mathcal{V}_{np_{c_j}}$, and a set of certificates asserting the partecipation to a community whose validity are certified by a Credential Authority.*

In the following, $[\![np_{c_j}]\!]_{\mathcal{P}ro(u)} = w_j$ denotes that the current value for the property np_{c_j} in the profile of u is w_j, whereas $eval_u$ is a predicate for the evaluation of a constraint/grant for a user u.

Definition 6 (Condition and Grant Evaluation). *Consider $u \in \mathcal{U}$ and a condition $(c, \{\langle np_{c_1} op_1 v_1 \rangle, \ldots \langle np_{c_m} op_m v_m \rangle\})$. c is satisfied wrt u if and only if:*

$$eval_u(c) = \bigwedge_{i=1}^{m} eval_u([\![np_{c_i}]\!]_{\mathcal{P}ro(u)} \ op_i \ v_i) = \texttt{true}$$

A grant $g = \langle p, r, \hat{o}, C \rangle$ is satisfied w.r.t. u if and only if:

$$eval_u(g) = u \downarrow p \wedge \textstyle\bigwedge_{c \in C} eval_u(c) = \texttt{true}$$

Given an authorization request $<u, r, \hat{o}>$, representing the request of user $u \in \mathcal{U}$ to exercise the right $r \in \mathcal{R}$ on the resource $o \in \mathcal{O}$, we now present its evaluation.

Definition 7 (Authorization Request Evaluation). *Let $ar =< u, r, \hat{o} >$ be an authorization request and G_u the set of grants issued for u. ar is granted to u, denoted by $[\![ar]\!]$, if and only if $\exists\ g = \langle p, r, \hat{o}, C \rangle \in G_u$ such that $eval_u(g) = \texttt{true}$.*

3 Grant Compatibility

In this section, we first discuss whether two rights are compatible, that is, whether they can be applied at the same time either on the same resource or on a pair of resources. Then, we present the notion of grant compatibility. We differentiate between weak and strong grant compatibility because of performance issues, the latter notion is more complex to be evaluated (and it is useless to compute if the licenses are not weak compatible). Without loss of generality, we restrict ourselves to pairs of licenses.

The rights compatibility notion depends on the context (same resource or different resources) on which the rights have been specified. In order to introduce this notion, we first need to establish when two rights are in *conflict*, that is, when they cannot be exercised at the same time by the same user. The pairs of privileges (`enlarge`, `reduce`) and (`enhance`, `diminish`) are in conflict because it does not make sense to apply at the same time, or within the same transaction,

two opposite privileges on the same resource. Moreover, since these pairs of privileges are subsumed by the corresponding `modify/adapt` privilege, the grant issuer, wishing to grant a wider rights to the principal, can exploit this privilege instead of the conflicting privileges. The pairs of privileges (`embed`, `reduce`) and (`embed`, `diminish`) are in conflict when specified on different resources, because it is not possible to embed the first resource into the second one if we are only allowed to reduce/diminish the second resource.

We also need to introduce the notion of *non comparable rights*. Two rights are non comparable on different resources, when the possibility to exercise the first right does not influence the possibility to exercise the second one. The rights $\mathcal{R}_T = \{$`reduce`, `enlarge`, `modify`, `diminish`, `enhance`, `adapt`$\}$, that is, the *transformation* rights with the exception of `embed`, are non comparable among each others. Formally, each pair in $\mathcal{R}_T \times \mathcal{R}_T$ is not comparable.

Definition 8 (Rights Compatibility). *Let $(r_i, r_j) \in \mathcal{R} \times \mathcal{R}$ be a pair of rights. r_i is compatible with r_j, if and only if: when specified on the same resource, they are not in conflict, i.e., $(r_i, r_j) \notin \{$(`enlarge`, `reduce`), (`enhance`, `diminish`)$\}$; when specified on different resource, they are not in conflict and are comparable, i.e. $(r_i, r_j) \notin \{$(`embed`, `reduce`), (`embed`, `diminish`)$\} \cup \mathcal{R}_T \times \mathcal{R}_T$.*

Relying on the notion of rights compatibility, we can introduce the concept of weak grants compatibility.

Definition 9 (Weak Grants Compatibility). *Let $g_i = \langle p_i, r_i, \hat{o}_i, C_i \rangle$ and $g_j = \langle p_j, r_j, \hat{o}_j, C_j \rangle \in \mathcal{G}$ be two grants. g_i is weak compatible with g_j ($g_i \simeq g_j$) if and only if: $p_i \sqcap p_j$ and r_i is compatible with r_j.*

The fact that an user holds grants that are weak compatible does not mean that she/he can exercise them on the resources. The user profile should be considered and the grants that are weak compatible evaluated by taking the user profile into account.

Definition 10 (Strong Grants Compatibility). *Let g_i and g_j be two weak compatible grants ($g_i \simeq g_j$), $\mathcal{P}ro(u)$ the profile of user u. g_i and g_j are strong compatible w.r.t. u ($g_i \cong_u g_j$) if and only if $eval_u(g_i) = eval_u(g_j) =$ `true`*

Example 7. Consider the license l_2 of Example 6. It follows that

$g_5 \simeq g_6$, $g_5 \simeq g_7$, $g_6 \simeq g_7$, $g_7 \simeq g_8$ and $g_5 \not\simeq g_8$, $g_6 \not\simeq g_8$

Suppose now that $eval_{\texttt{Tom}}(c_1) = eval_{\texttt{Tom}}(c_3) =$ `true` whereas $eval_{\texttt{Tom}}(c_2) =$ `false`. Consequently, only $g_5 \cong_{\texttt{Tom}} g_6$ holds. □

4 License Composition

The creation of a new license for the composition of two resources or for a modified resource can be realized by exploiting the License Generation Service of our architecture (details in the next section). Therefore, a request of license

Algorithm 1. The `compose` licenses request

Input: o_a a resource, o_b a resource, $r \in \{\texttt{enlarge}, \texttt{modify}, \texttt{enhance}, \texttt{adapt}, \texttt{embed}\}$ a right, o_c the composed resourse, u a user

1: **if** $\exists g_a \in G_u^{\texttt{embed},o_a}$ and $\exists g_b \in G_u^{r,o_b}$ s.t. $g_a \cong_u g_b$ **then**
2: **if** $r \in \{\texttt{enhance}, \texttt{adapt}\}$ **then**
3: Generate the license $l_c = <u, G_u^{o_a} \cup G_u^{o_b}[^{o_c}/_{o_b}]>$
4: **else**
5: **if** $r = \texttt{embed}$ **then**
6: Generate the license $l_c = <u, G_u^{o_a} \cup G_u^{o_b} \cup \{(u, \texttt{play}, \hat{o}_c, \{\})\}>$
7: **else**
8: Generate the license $l_c = <u, G_u^{o_a} \cup G_u^{o_b}>$
9: **end if**
10: **end if**
11: **end if**

Output: The generated license (or denial to generate a license)

generation should be sent to this service. We consider the following kinds of basic license generation requests: `compose`, `update`, `add`, `remove`. The first two are used when the user wishes to automatically generate a license for the composition of two resources, or the modification of a single resource. By contrast, the last two are used to specify extra grants for new added resources, or to remove grants that do not apply any longer to the modified resource.

For the sake of understandability we present the algorithms for the evaluation of the basic licenses generation requests. However, the approach can be easily extended to consider more sophisticated sequences of combinations and modifications of resources. In these cases, the actual generation of the license can be postponed at the end of the sequence of basic modification operations. We also remark that analogous algorithms have been developed for simply checking whether there are the conditions for the generation of a new license. These checking algorithms simply exploit the weak compatibility notions instead of the strong compatibility notions discussed in the previous section. In the remainder with the notation $G[^{o'}/_{o}]$ we mean that in the licenses contained in G, all the references to the resource o are substituted with the reference o'.

Algorithm 1 generates the new license when there is a `compose` licenses request. Whenever an user u wishes to compose a resource o_a with a resource o_b, she/he should at least hold an `embed` right on o_a and a right r that allows to update (or append information to) the resource o_b. If $r \in \{\texttt{enhance}, \texttt{adapt}\}$, the resource o_b should be left unchanged and a new resource, named o_c, should be generated. Otherwise, o_c is o_b itself. Whenever user u holds the grants for performing the composition operation, a new license is generated containing the union of the grants user u holds on the original resources. In case a new resource is generated, the references to o_b occurring in the grants should be substituted with the references to o_c (line 3 in Algorithm 1). If $r = \texttt{embed}$, the resource o_c contains the concatenation of o_b and o_a. In this case we need to introduce a grant that allows the user u to access the container o_c, otherwise the access to the components would be forbidden.

Algorithm 2 is used to generate a new license when a resource is modified through a `enhance/adapt/diminish` right. For the other transformation rights

Algorithm 2. The `update` licenses request

Input: o a resource, $r \in \{\texttt{enhance}, \texttt{adapt}, \texttt{diminish}\}$ a right, o_c the modified resourse, u a user
1: **if** $\exists g \in G_u^{r,o}$ s.t. $[\![< u, r, o >]\!]$ **then**
2: Generate the license $l_c = <u, G_u^o[^{o_c}/_o]>$
3: **end if**
Output: The generated license (or denial to generate a license)

there is no need to generate a new license (the original license is still valid). Since a new resource o_c is generated, we need to substitute the references to o_b occurring in the grants with the references to o_c.

Example 8. Let $o_{txt} = \{\hat{o}_{txt} : \{\hat{o}_{\texttt{txt}_1} : o_{\texttt{txt}_1}, \hat{o}_{\texttt{txt}_2} : o_{\texttt{txt}_2}\}\}$ and $o_{txt_3} = \{\hat{o}_{txt_3} : \{\hat{o}_{\texttt{txt}_4} : o_{\texttt{txt}_4}, \hat{o}_{\texttt{txt}_5} : o_{\texttt{txt}_5}\}\}$ be two resources, l_4=< Bob, $\{g_9, g_{10}, g_{11}, g_{12}\}$ > $\in \mathcal{L}_{\texttt{Tom}}$ be another license issued from Bob to Tom consisting of the following grants:

$g_9 =$ < Tom, enlarge, $\hat{o}_{\texttt{txt}}$, $\{c_1, c_2\}$ > $\quad g_{10} =$ < Tom, embed, $\hat{o}_{\texttt{txt}_3}$, $\{c_1, c_2\}$ >
$g_{11} =$ < Tom, embed, $\hat{o}_{\texttt{txt}}$, $\{\}$ > $\quad g_{12} =$ < Tom, enhance, $\hat{o}_{\texttt{txt}}$, $\{\}$ >

Suppose that Tom wishes to compose o_{txt_3} into o_{txt} by enhancing, enlarging or embedding the latter. At the time of the request, we have that: $\mathcal{L}_{\texttt{Tom}} = \{l_2, l_4\}$, $G_{\texttt{Tom}}^{\texttt{embed},\hat{o}_{\texttt{txt}_3}} = \{g_{10}\}$, $G_{\texttt{Tom}}^{\texttt{enlarge},\hat{o}_{\texttt{txt}}} = \{g_9\}$, $G_{\texttt{Tom}}^{\texttt{modify},\hat{o}_{\texttt{txt}}} = G_{\texttt{Tom}}^{\texttt{adapt},\hat{o}_{\texttt{txt}}} = \emptyset, G_{\texttt{Tom}}^{\texttt{enhance},\hat{o}_{\texttt{txt}}} = \{g_{12}\}$, $G_{\texttt{Tom}}^{\texttt{embed},\hat{o}_{\texttt{txt}}} = \{g_{11}\}$. Finally, suppose that $eval_{\texttt{Tom}}(c_1) = eval_{\texttt{Tom}}(c_2) = \texttt{true}$. Consequently, $g_{10} \cong_{\texttt{Tom}} g_9$, $g_{10} \cong_{\texttt{Tom}} g_{11}$ and $g_{10} \cong_{\texttt{Tom}} g_{12}$ are valid and thus, in all cases, the Tom's requests can be satisfied. Referring to Algorithm 1, if Tom requires to compose o_{txt_3} into o_{txt}

by enlarging o_{txt}**.** The structure of the resulting updated resource o_{txt} is $o_{txt} = \{\hat{o}_{txt} : \{\hat{o}_{\texttt{txt}_1} : o_{\texttt{txt}_1}, \hat{o}_{\texttt{txt}_2} : o_{\texttt{txt}_2}, \hat{o}_{txt_3} : \{\hat{o}_{\texttt{txt}_4} : o_{\texttt{txt}_4}, \hat{o}_{\texttt{txt}_5} : o_{\texttt{txt}_5}\}\}\}$ whereas the generated license is l_5=< Tom, $\{g_7, g_8, g_9, g_{10}, g_{11}, g_{12}\}$ > $\in \mathcal{L}_{\texttt{Tom}}$.

by `embedding` o_{txt}**.** A new resource o_{txt_6} is generated and its structure is $o_{txt_6} = \{\hat{o}_{txt_6} : \{\hat{o}_{txt} : \{\hat{o}_{\texttt{txt}_1} : o_{\texttt{txt}_1}, \hat{o}_{\texttt{txt}_2} : o_{\texttt{txt}_2}\}, \hat{o}_{txt_3} : \{\hat{o}_{\texttt{txt}_4} : o_{\texttt{txt}_4}, \hat{o}_{\texttt{txt}_5} : o_{\texttt{txt}_5}\}\}\}$. A new license is generated for o_{txt_6}: l_6=< Tom, $\{g_7, g_8, g_9, g_{10}, g_{11}, g_{12}, g_{13}\}$ > $\in \mathcal{L}_{\texttt{Tom}}$, where $g_{13} =$ < Tom, play, $\hat{o}_{\texttt{txt}_6}$, $\{\}$ > is a new grant that allow Tom to access to the concatenated resources.

by enhancing o_{txt}**.** A new resource o_{txt_7} is generated and its structure is $o_{txt_7} = \{\hat{o}_{txt_7} : \{\hat{o}_{\texttt{txt}_1} : o_{\texttt{txt}_1}, \hat{o}_{\texttt{txt}_2} : o_{\texttt{txt}_2}, \hat{o}_{txt_3} : \{\hat{o}_{\texttt{txt}_4} : o_{\texttt{txt}_4}, \hat{o}_{\texttt{txt}_5} : o_{\texttt{txt}_5}\}\}\}$. A new license is generated for o_{txt_7}: l_7=< Tom, $\{g'_7, g'_8, g'_9, g_{10}, g'_{11}, g'_{12}\}$ > $\in \mathcal{L}_{\texttt{Tom}}$ where the grants are updates as follows in order to refer to the new resource identifier:

$g'_7 =$ < Tom, print, $\hat{o}_{\texttt{txt}_7}$, $\{c_1, c_2\}$ > $\quad g'_8 =$ < Tom, reduce, $\hat{o}_{\texttt{txt}_7}$, $\{c_1, c_2\}$ >
$g'_9 =$ < Tom, enlarge, $\hat{o}_{\texttt{txt}_7}$, $\{c_1, c_2\}$ > $\quad g'_{11} =$ < Tom, embed, $\hat{o}_{\texttt{txt}_7}$, $\{\}$ >
$g'_{12} =$ < Tom, enhance, $\hat{o}_{\texttt{txt}_7}$, $\{\}$ > □

The two presented algorithms allow the automatic generation of a new license when resources are composed and modified. However, the user might decide to introduce further grants (e.g. on new introduced resources) or to remove useless grants (e.g. grants that refer resource components that have been removed). These operations are performed through the `add` and `remove` algorithms that we do not report for space constraints. The `add` algorithm checks the consistency of

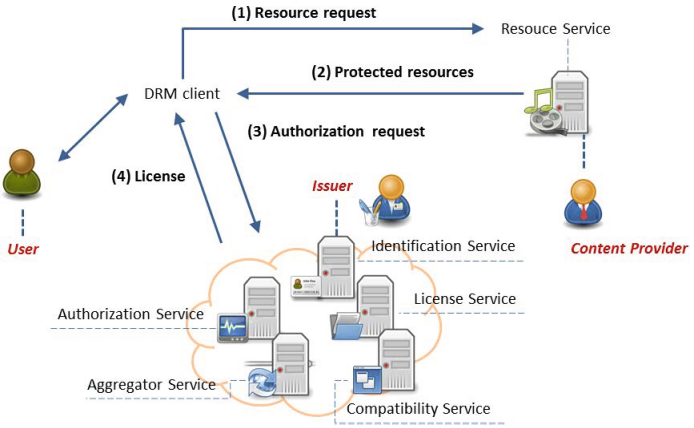

Fig. 2. A simplified DRM scenario

the new inserted grants with respect to the other grants occurring in the license, whereas the `remove` algorithm compacts the license by removing grants that cannot any longer be evaluated on the modified resource.

5 Enhanced DRM Architecture

This section presents a DRM architecture enabling a set of services and technologies to govern the authorized use of digital resources and to manage any consequences of that use throughout their entire life cycle. The architecture is approached by identifying the main involved entities: the resource provider, the users and the issuer. A DRM system is a set of DRM related services offered by the issuer to users and resource providers to enable the consumption of resources. In a typical simplified scenario, illustrated in Figure 2, a user exploits its DRM client to contact a Resource Service (1). This service enables the access to the protected resources (2). After an authorization has been requested (3) and evaluated according to the user profile (4) through a set of services, the resources can be consumed. For evaluating the user-request and for issuing the corresponding license, the architecture relies on a workflow of service-requests. In the reminder of the section, we first report a set of high level services that are needed for enabling such composition, and then we describe the correct sequence of service-requests used to compose a new resource and to generate its proper license. For the lack of space we concentrate only on the issue of composition, the update of a single resource is handled analogously.

Service-Oriented Architecture. The architecture in [14] is extended for supporting the license generation when a new resource is created by a user (henceforward the producer) through the composition of different resources. For the

sake of clarity, we avoid to present the external tracking and payment services focusing only on services devoted to check the composition compatibility.

1. *Resource Service.* This service is in charge of the management of the resources and answers to the access/composition requests posed by an user through a DRM compliant client.
2. *License Service.* This service is in charge of the management of the licenses and of issuing new licenses upon users request.
3. *Authorization Service.* The Authorization Service is responsible for the evaluation of licenses upon an access/composition request is received by the Resource Service. For its activity the Authorization Service gets in touch with the License Service for obtaining the licenses, with the Compatibility Service for checking compatibility issues in case of composition/update of resources, with the Identification Service for the evaluation of principal specification on licenses' grants, and for acquiring the user profile for the evaluation of grants, and with External Services for checking specific constraints (like for example the external Payment Service in case the principal has to pay for consuming the resources) occurring in licenses.
4. *Identification Service.* This service is responsible for the authentication of the users and for checking the users' certificates. Moreover, it gets in tough with the DRM client to obtain temporal, geographical and context of use information to be included in the user profile. Finally, the Identification Service is in charge of storing and keep updated the users profiles.
5. *Compatibility Service.* This service is responsible to check the weak and strong compatibility conditions discussed in Section 3. The Compatibility Service, like the Authorization Service, can get in tough with External Services for the strong verification of conditions of grants.
6. *License Generation Service.* This service is in charge of executing the algorithms discussed in Section 4, and of creating the new licence of the composed resource by interacting with the License Service to retrieve licenses and to store the generated ones.

DRM Services Composition in Collaborative Environment. In a collaborative environment, we should define a workflow of service-requests able to support the generation of a license of a new resource created by composing different ones. Specifically, our proposal relies on a workflow consisting of two main activities: the composition of different resources and the definition of the license on the resulting resource. To support these activities, our architecture should adopt specific composition and integration management services (see Figure 3).

For composing different resources, the DRM client forwards the composition request obtained by the producer to the Resource Service. In order to authorize the operation, the Resource Service gets in touch with the Authorization Service. This service collects the licenses of the involved resources from the License Service. By considering the grants contained in the obtained licenses, the Authorization Service evaluates the composition request by means of the Compatibility Service. The first check concerns the validity of the weak compatibility

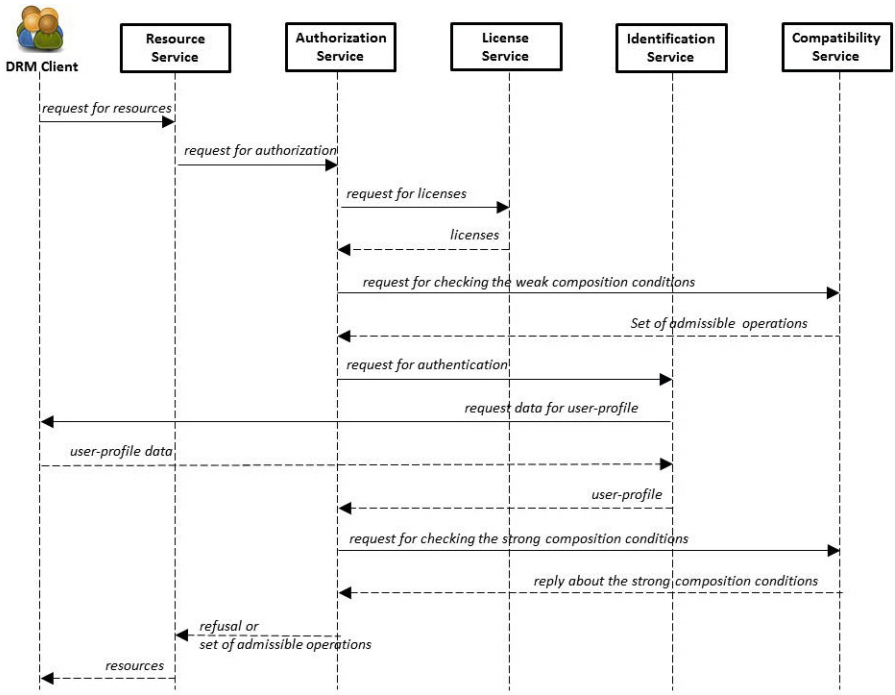

Fig. 3. Sequence of service-requests for composing a set of resources in the DRM system

conditions of the resources to compose and leads to the identification of a set of operations that the producer can carry out on the original resources according to her/his licenses. Once the Compatibility Service has checked the weak compatibility conditions, the Authorization Service has to authenticate the producer and control the grant conditions according to the producer profile. Authentication is realized in the following way. The Authorization Services asks to the Identification Service to authenticate the producer by providing the producer profile. The Identification Service, in turn, needs to query the DRM client for gathering information about the producer's context of use and for the evaluation of predicates relying on the producer certificates. The evaluation of the grant conditions is realized through the Compatibility Service by considering the producer profile and External Services. After carrying out these two activities, the Authorization Service can authorize or refuse the access to resources. In case the compatibility check comes to a satisfactory reply, it will provide the set of operations that the producer can carry out for composing the new resource.

Once the resources have been composed, the producer requests the generation of a new license. First of all, the DRM client has to contact the License Generation Service that takes care of all steps needed to create the new license and to send it to the License Service for its storage (see Figure 4). The License Generation Service requires to the License Service the producer's licenses of the

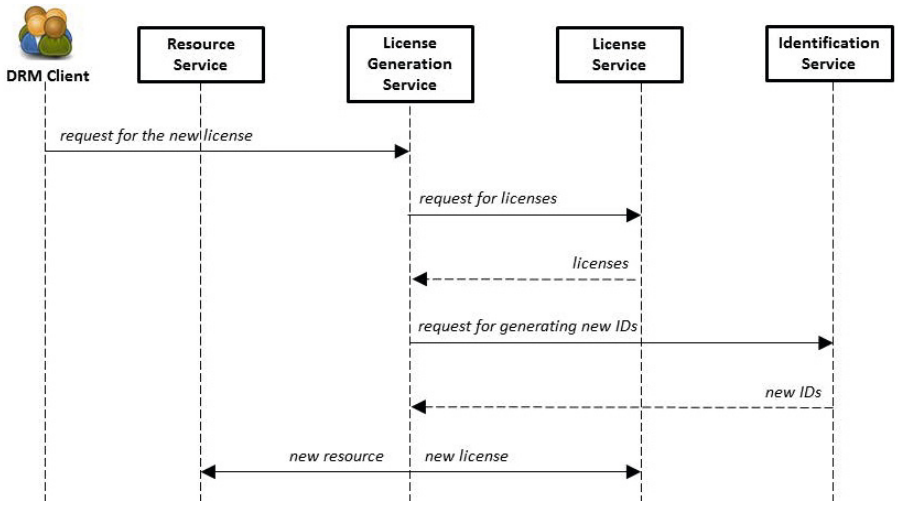

Fig. 4. Sequence of service-requests for the generation and storage of a new license

component resources and exploits the information to compose a set of grants to assign to the license of the new resource according to the operations carried out by the producer. To complete these operations, the License Generation Service has to ask to the Identification Service to generate new identifiers for resources enhanced and adapted during the composition process. By exploiting the identifiers generated by the Identification Service, the License Generation Service generates the license of the new resource, if it is possible. If the process is positively concluded, the License Generation Service sends the new resource to the Resource Service for its storage and sends the new license to the License Service.

6 Related Work

Different Rights Expression Languages (like ccREL [6], ODRL [8], MPEG-21 REL [10]) have been proposed for expressing digital licenses, terms and conditions. These proposals differ from the scope and the granularity according to which it is possible to specify and manage each aspect of the license specification and management processes. Although some of them support the necessary rights to compose resources, none of them explicitly defines how to create and assign a new license to the result of the composition.

The issue of license composition and compatibility analysis is difficult to address in a systematic way since license terms and conditions can be expressed in different ways, including the natural language, by using terms whose meaning would be ambiguous or not interpretable uniquely. This aspect makes obviously more difficult to determine which correct result should be returned from the composition process. In [3–5] the authors address the problem of service

license[1] composition and compatibility analysis by using the ODRL-s language [2], an extension of ODRL [7, 8] to implement the clauses of service licensing. A similar approach is used in [19] where the authors present a framework to associate licensing terms to web data in order to combine licenses for the data resulting from queries; the resulting composite license is still expressed as licensing terms. Finally, in [1] the authors address the issue of identifying the kind of licenses, expressed through MPEG-21 REL [10, 11], a user should hold in order to compose or transform resources and present use-cases for the main operations.

Our approach differs from them from several points of view. First, our application scenario is different from that of service and web data licensing. In their case, the focus is on the management of the business and legal contractual information expressed as CC terms, with respect to the ODRL CC profile [9] and the CC license schema [6] respectively. This is used to establish if different services are compatible and composable among them and to verify wether users can access to web data provided by different data sets and released under different licensed terms. Conversely, our formal model, compliant with MPEG-21 REL [10], is mainly focused on the issue of determining the conditions under which the transformation/composition of resources can be performed in a systematic way and to provide a license for the transformed/composed resource. Finally, the weak/strong evaluation strategy we propose allows the evaluation of the authorization conditions only when strictly required.

With the proliferation of Internet-based applications and the ready availability of powerful file sharing and distribution tools, DRM has become a critical concern in the Internet domain. The literature is rich of proposals exploiting DRM platforms based on the distribution of digital content via Internet, like: the Open and Secure Digital Rights Management Solution (OSDRM) [18]; the Microsoft Windows Media DRM platform (MSDRM) [17]; the IBM Electronic Media Management System (EMMS) [16]; and the Real Networks HelixDRM [15]. These solutions aim at creating a secure framework for delivering multimedia on the Internet, and caters for the creation of secure contents, payment collection, distribution and rendering of multimedia. Most of this literature describes a DRM system as a platform containing several functionality, including content registration and protection; it offers publication, content search, purchase and licensing, authorization and access control. Nevertheless, these systems are not able to support users in the composition process by taking into account the users' licenses on the component resources.

7 Conclusions and Future Work

Starting from a formal model for the representation of licenses, in this paper we provided algorithms for the evaluation of compatibility of license's grants and for the generation of a new license of composed/updated resources. Moreover, we discussed an architecture that allows the composition and modification of

[1] A service license describes the terms and conditions for the use and access of the service in a machine readable way that services could be able to understand.

resources. These results are the building blocks for the realization of a tool for helping a user to compose resources in collaborative environments that we are currently implementing. We remark that our licenses can be easily translated into corresponding MPEG-21 REL licenses [10].

As future work we plan to enhance our approach by allowing the composition of resources belonging to different users. In this scenario, a negotiation of users' grants should take place in order to determine the grants to be assigned to the generated resource. Moreover, we are considering the issue of distribution of licenses to the users belonging to the user's communities.

References

1. Delgado, J., et al.: Definition of mechanisms that enable the exploitation of governed content. In: AXMEDIS, pp. 136–142 (2006)
2. Gangadharan, G., et al.: ODRL service licensing profile (ODRL-s). In: Proc. 5th Int'l Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods, Germany (2007)
3. Gangadharan, G., D'Andrea, V.: Service licensing: conceptualization, formalization, and expression. Service Oriented Computing and Applications 5(1), 37–59 (2011)
4. Gangadharan, G.R., et al.: Consumer-specified service license selection and composition. In: ICCBSS, pp. 194–203 (2008)
5. Gangadharan, G.R., Weiss, M., D'Andrea, V., Iannella, R.: Service license composition and compatibility analysis. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 257–269. Springer, Heidelberg (2007)
6. Abelson, H., et al.: ccREL: The Creative Commons Rights Expression Language. Creative Commons Wiki (2008)
7. Iannella, R.: Open digital rights management. In: Workshop on Digital Rights Management for the Web, France (2001)
8. Iannella, R.: Odrl specification 1.1 (2002)
9. Iannella, R.: Odrl creative commons profile specification (2005)
10. Information Technology-Multimedia Framework. Part 5: Rights expression language, iso/iec 21000-5 (2004)
11. Information Technology-Multimedia Framework. Part 6: Rights data dictionary, iso/iec 21000-6 (2004)
12. json. Javascript object notation
13. Ku, W., Chi, C.-H.: Survey on the technological aspects of digital rights management. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 391–403. Springer, Heidelberg (2004)
14. Michiels, S., et al.: Towards a software architecture for DRM. In: DRM, pp. 65–74 (2005)
15. RealNetworks. Helixcommunity-the foundation of great multimedia applications (2013)
16. RealNetworks. The IBM electronic media management system (2013)
17. RealNetworks. Windows media digital rights management (2013)
18. Serrão, C., et al.: Open SDRM - An open and secure digital rights management solution. In: Proc. Int. Association for Development of the Information Society, Portugal (2003)
19. Villata, S., Gandon, F.: Licenses compatibility and composition in the web of data. In: COLD (2012)