Licensing Services: Formal Analysis and Implementation

G.R. Gangadharan and Vincenzo D'Andrea

Department of Information and Communication Technology,
University of Trento,
Via Sommarive, 14, Trento, 38050 Italy
gr@dit.unitn.it, dandrea@dit.unitn.it

Abstract. The distribution of services spanning across organizational boundaries raises problems related to intellectual value that are less explored in service oriented research. Being a way to manage the rights between service consumers and service providers, licenses are critical to be considered in services. As the nature of services differs significantly from traditional software and components, services prevent the direct adoption of software and component licenses. We propose a formalisation of licensing clauses specific to services for unambiguous definition of a license. We extend Open Digital Rights Language to implement the clauses of service licensing, making a service license compatible with all the existing service standards.

1 Introduction

Service oriented computing (SOC) is an emerging distributed systems paradigm referring to systems structured as networks of loosely coupled, communicating services [1]. While software behaves as a stand-alone application, services intend making network-accessible operations available anywhere and anytime.

In contrast to traditional software components [2], the functionality of a service resides and runs at the provider's host in a distributed way beyond organizational boundaries, and consumers are not required to download the service executable for consuming the service. While components encapsulate coarse grained functionalities, the granularity of services could range from finer to coarse. Further, services allow the applications to be constructed on-the-fly and to be reused everywhere.

As service oriented applications are rapidly penetrating the society, there arises a need for governing their access and distribution. Although services are software fragments, the distinguishing characteristics of services preclude them to be licensed under traditional software / component licenses. While we do not intend to discuss the similarities and differences between services and components in general, we explicate the significant differences of services with respect to the components from the perspective of licensing. In case of components, the provider of a component is responsible for functionality of the component. Components are downloaded and executed in the environment of clients, within

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 365–377, 2006.

an organization. Services could span across different participating organizations. Services run through provider and the responsibility for operations of a service is more complex than that of components. We have explored in [3] the dimensions of services inducing a new paradigm of licensing. Nevertheless, being services accessed and consumed in a number of ways, there is the need to carefully define a set of licenses suitable for services.

Researches focus mainly on the expression of functional as well as non-functional properties of services. There exists an obvious paucity of licensing clauses for a service and embedding a license within a service. In order to fulfill this gap, we study the strategy of implementing licenses within a service. The salient features of our approach are:

- Formal representation of licensing clauses to unambiguously describe a service license.
- Extension of Open Digital Rights Language (ODRL) to encompass the service licensing clauses.

As licenses form the basis for distribution of services, in this paper, we elucidate a formal analysis of service licenses together with an implementation scenario of expressing the licensing terms in services. We describe by presenting various examples how a service interface and realization could be exploited by other services in Section 2. Section 3 compares various languages illustrating functional and non-functional properties of services as complementary to WSDL and elucidates their lack of expressiveness in describing the clauses of licensing. The formal description of licenses are presented in Section 4. We implement some of the service licensing clauses by extending ODRL in Section 5. Finally, we illustrate licensing of a service by extended ODRL in Section 6.

2 Exploring Service Licensing Clauses

A service is represented by an interface part defining the functionality visible to the external world and an implementation part realizing the interface [4]. In this section, we will analyze some of the prominent combinations of reproduction (or not) of the service interface, relationship between services (compositional properties), and derivation (or not) from the source code.

As service interfaces (WSDL) together with bindings are publicly available, several services could be created with the same interface. These services can vary in their performance and Quality of Service (QoS) issues. However, copying and using the interface with or without modifications are twined with intellectual values.

By the following example, we show how a service could simply be reused by an other service copying the interface directly: Let S_A be a service providing a spell checking operation for words, say, $Spell(word)$. Consider S_A provides this service by wrapping a proprietary word processor (PWP) spell checker API. As the WSDL interface of this service is publicly available, any service, say S_B could copy this interface and the interface of S_A could be used by S_B with or without modifications. Thus, S_B is an another independent service, wrapping an other proprietary word processor (QWP) spell checker API, created by replicating the WSDL of the S_A. Albeit S_A and S_B are performing spell checking, S_A and S_B are two different services, executed separately.

The prominent scenarios on ***reproduction*** of interface with modifications are as follows:

1. The interface of a service could be modified by changing the name of some operations such as for translation i.e. the expression of a service in a language other than that of the original version.
2. The interface of a service could be modified by some changes in the service parameters such as for data translation or by some pre-processing and/or post-processing of the service.

Following the styles of [5] for representing figures, services are denoted by the shadowed rectangular boxes. An operation of a service interface is represented as an Unified Modeling Language package marked by a stereotype $<< desc >>$. The wrapped application for the service is shown on the left side of the service.

The reproduction by interface translation is illustrated in Figure 1. The interface of S_A is translated by S_B to provide a spell checking operation in Italian language, say $Ortografia(parole)$. In this case, S_B translates the interface of S_A and results in the Italian version of S_A as an independent service.

We refer to ***composition*** as the federation of a service with other remote services. In other words, the operations of a composite service relies on the availability of services being composed [6].

Let S_B be a service providing a spell checking operation $Spell(sentence)$ for sentences, that could compose internally operations for spelling of words with a parser. S_B could be designed in such a way (See Figure 2) that $Spell(word)$ of S_B directly invokes the operation of S_A, executing on the host of S_A. In the absence of S_A, S_B fails to perform.

A service could deny or allow to use and/or modify the service realization. A service could allow to use its realization as an executable in an other service. For example, a service S_A could allow its realization to be used as an executable by an other service S_B. However, S_A could restrict S_B not to modify the operations of S_A.

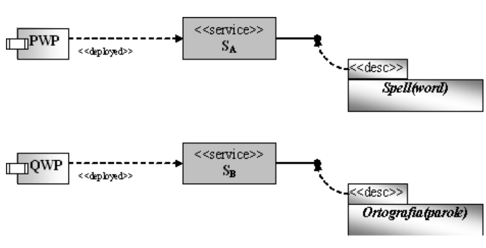

Fig. 1. Reproduction

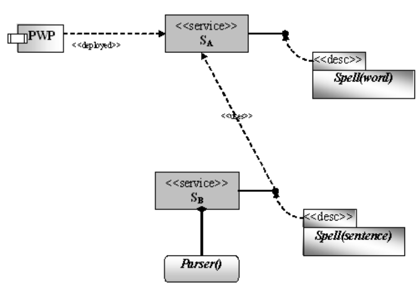

Fig. 2. Composition of Services

A service could allow to modify its realization by other service. The modification of a service realization, termed as ***derivation*** of a service, is an inspiration by Free[1] and Open Source[2] Software (FOSS) movement.

Consider a service S_A providing $Spell(word)$ operation for spell checking of a word. A new service S_B, performing spell checking for a sentence, could be derived from S_A. The derived service S_B contains an operation for parsing $Parser()$ in addition to the operation of S_A. In this case (See Figure 3), S_B significantly modifies the operation of S_A and thus S_B is a derivative service of S_A.

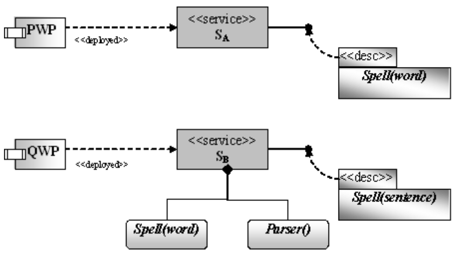

Fig. 3. Normal Derivation

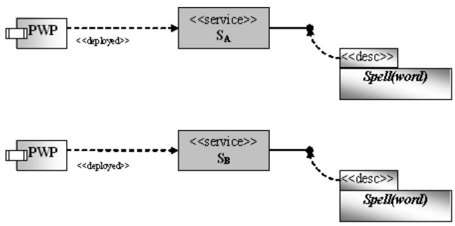

Fig. 4. Replica Derivation

Making replica of a service uses the service realization and service interface. If the WSDL interface as well as realization of a service allows copying, replica services (See Figure 4) are created. Consider S_B as an independent service created by replicating/mirroring the source code of realization and WSDL of S_A. Though S_A and S_B are performing the same operations, S_A and S_B are two different services, executed separately. Theoretically, there will be no differences (may include network delays!) in performances of both the services. Thus, derived service is a manifestation of 'Free Culture'.

Beyond these aspects, a service may expect certain moral rights [7] to be satisfied. A service, S_A, could expect the service, say S_B, being composed / derived / reproducing the interface to reflect the ***same terms and conditions*** of the S_A (Similar to 'Sharealike' of CreativeCommons [8] or Copyleft of GNU[3]).

A service may expect the ***attribution*** for its use by the other service in any of the forms. As attribution is considered a basic requirement, a service should give the proper credit for the service that it uses. In case of composition, the composite service could be required to give attribution for every level of composition as in a BSD license[4].

Further, a service could allow/deny the other service depending on the ***usage*** either for non-commercial purposes or for commercial purposes.

1 http://www.fsf.org/
2 http://www.opensource.org/
3 http://www.gnu.org/copyleft/
4 http://www.openbsd.org/policy.html

3 Licensing Clauses in Service Descriptions Languages

WSDL is the standard way to describe what a service does. Researches focusing on languages to enhance and to complete the description provided by WSDL are continually in progress. These languages being complementary to WSDL address functional/non-functional properties and business/management information of services with varying levels of details.

Web Service Level Agreement (WSLA):The WSLA framework [9] describes the complete life cycle of a Service Level Agreement (SLA) including SLA establishment by negotiation (signing of a SLA by signatory parties for a given service offering), SLA deployment (checking the validity of the SLA and distributing it), Service level measurement and reporting (configuring the run-time system to meet a set of SLAs and comparing measured SLA parameters against the thresholds defined in the SLA), Management actions (determining SLA violations and corrective management actions to be taken), and SLA termination (specifying the conditions for termination). The WSLA framework enables to specify and monitor a wide variety of SLAs for web services. Based on XML, the WSLA language defines a type system for the various SLA artifacts. A SLA in WSLA is comprised of parties (identifying all the contractual parties), service description (specifying the characteristics of service and the observable parameters like service availability, throughput, or response time), and obligations (defining various guarantees and constraints to be imposed on SLA parameters).

The WSLA language is a general purpose way to express performance characteristics of web services.WSLA encompasses the agreed performance characteristics and the way to evaluate and measure them. However, WSLA does not focus on the rights to be associated with service provider and service consumer.

SLA Notation generator (SLAng): SLAng [10] is a XML based language, for describing Service Level Specifications in the domain of distributed systems and e-business. This language has been modeled by Object Constraints Language (OCL) and Unified Modeling Language (UML) in order to define SLA precisely. SLAng formally defines SLA vocabulary in terms of the behaviour of the services and clients involved in service usage, with reference to a model of service usage. A SLA described in SLAng comprises information on parties involved (end point description of contractors), contractual statements (defining the agreement), and QoS description with the associated metrics (service level specifications). Further, SLAng supports the inter-service composition of SLAs as a description of relationship between possible service behaviors.

Although SLAng has a broader scope beyond web services enabling different types of SLAs, SLAng is silent about the intellectual rights associated with services.

Web Service Offering Language (WSOL):WSOL [11], a language for specifying constraints, management information, and service offering, provides different service levels defined by several classes of services. The same WSDL description with differing constraints (functional, non-functional, and access right) and managerial statements (price, penalty, and responsibility) is referred as 'classes of service' of a web service in WSOL. Consequently, different classes of services could vary in prices and payment models. WSOL offers several reusability elements to enable easier derivation of a new service offering from the existing offerings.

The value of WSOL lies in the simplicity of the negotiation process and the simplified management infrastructure of WSOL.However, WSOL misses the syntax of business and legal contents of contracts.

WS-Policy: WS-Policy [12] provides a general framework to specify and communicate (publish) policies for web services. It is a model for expressing the capabilities, requirements, and general characteristics of a web service as policies. WS-Policy provides a base set of constructs that can be used and extended by other web services specifications to describe a broad range of service requirements, preferences, and capabilities.

WS-Policy defines a policy as a collection of policy alternatives. In turn, each policy alternative comprises a collection of policy assertions. Each policy assertion indicates an individual requirement, capability or other property of a behaviour. WS-policy is one of the fundamental works for specifying policies for web services. However, WS-Policy does not detail the specification of functional constraints, QoS policies, and other related management information.

We have analysed the current attempts by some of the web service languages to describe functional and/or non-functional properties and managerial information of services. Every language describes certain properties of services entirely. Generally, all the standards focus on the QoS and the terms and conditions agreed by the provider and consumer. However, in our view, none of them intensively describe the distribution aspects and the ownership clauses of licensing. The business and legal contractual information are not focused in detailed level by the services research community. The issues of copyrights and moral rights [13] are unexplored by the currently available service description standards. We think, there is a need to be considered to enable a broad usage of service that preserves certain rights of the owner and presents certain rights to the consumer.

From a different perspective, few languages and models capable of expressing a range of licenses are existing in the domain of Digital Rights Management (DRM) [14] for digital contents and multimedia. In the pioneering work of [15], a mathematical model for describing payment and rendering events is described. In [16], the properties of licenses are stated and proved by using deontic logic. LicenseScript [17] based on multi-set rewriting, expresses dynamic conditions of audio/video contents. As these models and languages restrict themselves within the domain of digital contents and multimedia, they could not be adaptable for describing services. Copyrights and other related rights are also not formalised in all these models.

4 Formalising the Service Licenses

A service could allow/deny itself to be used by other services. Further, a service could allow/deny to reuse its interface with or without modification. Allowing or denying composition and derivation influences reuse of services significantly. For drafting a family of machine readable licenses, the clauses of a service license should be unambiguous. We will formalise the clauses of rights detailed in Section 2 to avoid ambiguity in describing service licenses.

Let $\{\text{op}(S_A)\}$ be the set of operations offered by a service S_A. We refer to each clause $(\mathcal{C})$ of the license for service S_A as $\mathcal{C}_{S_A}$.

We define Interface Expressive Power $(\mathcal{E})$ as the degree to which a service interface is explainable, described by the number of operations involved and the number and type of parameters of operations[5]. We define $\mathcal{E}$ as,

$$\mathcal{E} = n + \sum_{i=1}^{n} \left(\frac{\sum_{j=1}^{m} \delta_j}{m} \right)$$

where n is the number of operations of an interface and for each operation, m is the number of parameters. δ_j is the measure of the complexity of the data type. Following WSDL definitions, we consider the values for simple, derived, and complex data types as 1, 2, and 3 respectively.

Derivation (D): Derivation of a service, inspired by FOSS, is a new aspect of creating a new service from existing service, modifying the WSDL interface and implementation. We define a service as a Free/Open Service [18] if the service provides its WSDL interface as well as source code freely available for creating a new and independent service. The open service allows the new service to use a modified version of the original source code. A service S_B is said to be derived from S_A if $\{\text{op}(S_B)\} \supseteq \{\text{op}(S_A)\}$ on satisfying the following two conditions: (i) To exist S_B, S_A should be a Free/Open Service and (ii) S_A and S_B are independent in execution. Normal Derivation (See Figure 3) is represented formally as $\{\text{op}(S_B)\} \supset \{\text{op}(S_A)\}$. Replica Derivation (See Figure 4) is represented by $\{\text{op}(S_B)\} \equiv \{\text{op}(S_A)\}$. In any case of derivation, the $\mathcal{E}$ of the derived service is always higher than or equal to the $\mathcal{E}$ of the service used for derivation. Thus, $\mathcal{E}(S_B) \geq \mathcal{E}(S_A)$. However, network latency issues in delivery of S_A and S_B could exist.

Reproduction (R): Reproduction signifies making a new independent service from an existing service interface. If a service S_A is reproduced as an other independent service S_B, then $\{\text{op}(S_B)\} \neq \{\text{op}(S_A)\}$ and S_A and S_B are independent in execution.

Weyuker's property number 8 of software complexity [19] explicitly states that if a program is a straight renaming of another program, its complexity would be

[5] The interface expressive power of services could be defined based on several metrics. We have considered a few relevant metrics and we do not claim this as an optimal solution. Nevertheless, our general line of thought is not affected by the interface expressive power computation.

same as the original program. Observing this property for a reproduction that unmodifies the interface, the $\mathcal{E}$ of the reproduced service remains unchanged: $\mathcal{E}(S_B) = \mathcal{E}(S_A)$. In case of a reproduced service changing the interface, the $\mathcal{E}$ of the reproduced service could differ from the service being reproduced: $\mathcal{E}(S_B) \neq \mathcal{E}(S_A)$.

Composition (C): Composition is a form of integration of services with value addition provided a composite service could be further composable [20]. Composition of services specifies the participating services, the invocation sequence of services and the methods for handling exceptions [21]. A service S is said to be composite if $\{\text{op}(S)\} \supset \{O_f : O_f \,\epsilon\, \{\text{op}(S_i)\}\}$ and $\exists\, S \mid S_i, i = [1, .., n]$. O_f could be a single operation or a set of operations adding value addition by combining all or some of the operations of S_i.

Based on Weyuker's properties (property numbers 5 and 9) of software complexity, we propose the $\mathcal{E}$ of a composite service differing from the $\mathcal{E}$ of the composing services obviously. Thus, $\mathcal{E}(\text{S}) \neq \mathcal{E}(S_i,\ S_j)$. Though the underlying assumption of SOC is composition, a service can deny or limit other services to use itself in a composition.

Attribution (A): Attribution means to ascribe a service to the entity responsible for its creator. If a service S_B uses a service S_A, then the attribution to S_A could be formally represented as $A_{S_B} \supset A_{S_A}$. The levelled attribution as in BSD styled service licensing is represented by $A_{S_C} \supset A_{S_B} \supset A_{S_A}$ where the service S_C uses S_B and S_B uses S_A.

Similar Terms (T): A service S_B may expect another service S_A (which uses S_B) to have the same terms as of S_B. In other words, $L(S_A) = L(S_B)$ where S_B uses S_A and $L(S)$ is the service license defined as below.

Non-Commercial Use (N): A service S_B could deny its use for commercial purposes. $N_{S_B} = 1$ implies that an other service S_A could use S_B if S_A is not commercial.

Now, we define the license L of a service S as[6]

$L(S) = (D, R, C, A, T, N)$.

The combinations of these licensing clauses define a family of licenses for services ranging from the most restrictive to the most unrestrictive.

5 Implementing Licenses in Services

Instead of proposing a new language for describing the licensing aspects of services, we could draft the terms and agreements of license using existing rights expression languages. XrML [23] is a comprehensive right expression language,

[6] Further, a service license comprises the financial terms, warranties, indemnification and limitation of warranties, and other clauses [22]. These terms are integral for a legally enforceable license. In this paper, we are primarily concerned with the clauses directly associating the scope of rights of a service license.

created by the ContentGuard Inc.[7], currently the basis of MPEG-21[8]. ContentGuard has a portfolio of patented technologies, covering the distribution and use of digital works and the use of a grammar in connection with the distribution of digital works. Though the terms are not specific to XrML, XrML is restricted to be used for a context covered by the patents. Hence, to obviate any kinds of patent infringements, we avoid XrML for implementing the terms of licenses in services.

Open Digital Rights Language (ODRL) [24] is an open standard language for the expressions of terms and conditions over assets, in open and trusted environments. The models for the ODRL language and data dictionary contain the structure and core semantics for the expressions. These models provide the overall framework for the expressions into which elements can be applied. The core entities of ODRL are as follows:

- *Assets:* a resource being licensed (to be identified uniquely), for instance, a web service.
- *Rights:* rules concerning permissions (the actual usages or activities allowed over the assets), constraints (limits to these permissions), requirements (the obligations needed to exercise the permission), and conditions (the specifications of exceptions that, if become true, expire the permissions and re-negotiation may be required).
- *Parties:* information regarding the service provider, consumer, broker etc.,

With these three entities, ODRL expresses offers (proposals from rights holders for specific rights over their assets) and agreements (contracts or deals between the parties, with specific offers). These core entities together allow for a wide and flexible range of ODRL expressions to be declared.

Our motivations for ODRL as an appropriate rights expression language for describing machine readable licensing agreements for services are as follows:

- ODRL is an open standard language, for expressing rights information.
- Being defined in XML, ODRL provides syntactic and semantic interoperability.
- ODRL is extensible and capable of incorporating specific clauses related to service licenses.
- Several business scenarios across various domains are expressable in ODRL.
- Being published in the World Wide Web Consortium (W3C), ODRL has a wide acceptance
- ODRL is supported by several industries and consortia like the Dublin Core Metadata Initiative (DCMI)[9] and the Open Mobile Alliance (OMA)[10].

With this proposal, we extend ODRL to define the clauses of a service license $L(S)$, by creating a new data dictionary that imports the ODRL expression language schema (See Table 1) to describe the scope of rights of services.

[7] http://www.contentguard.com/
[8] http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm
[9] http://dublincore.org/
[10] http://www.openmobilealliance.org/

Table 1. ODRL/$L(S)$ Data Dictionary Semantics and Schema

ODRL Element	**Identifier**	**Description**
Permission	Derivation (D)	The service may be derived.
<xsd:element name="Derivation" type="o-ex:permissionType" substitutionGroup="o-ex:permissionElement"/>		
Permission	Reproduction (R)	The service may be reproduced.
<xsd:element name="Reproduction" type="o-ex:permissionType" substitutionGroup="o-ex:permissionElement"/>		
Permission	Composition (C)	The service may be composed.
<xsd:element name="Composition" type="o-ex:permissionType" substitutionGroup="o-ex:permissionElement"/>		
Requirement	Attribution (A)	The use of service must always include attribution of the service.
<xsd:element name="Attribution" type="o-ex:requirementType" substitutionGroup="o-ex:requirementElement"/>		
Constraints	SimilarTerms (T)	The license terms should be same without changed when used/reused.
<xsd:element name="SimilarTerms" type="o-ex:constraintType" substitutionGroup="o-ex:constraintElement"/>		
Constraints	NonCommercialUse (N)	The service is for non-commercial purposes.
<xsd:element name="NonCommercialUse" type="o-ex:constraintType" substitutionGroup="o-ex:constraintElement"/>		

ODRL/$L(S)$[11] Data Dictionary Semantics expresses the core $L(S)$ semantics in the ODRL.

6 A Scenario of Service Licensing

In order to illustrate our approach, we consider a simple scenario where R is a restaurant service providing the following operations (and parameters): R_0, information on location and opening hours ($address$: $complex$; $hours$: $complex$); R_1, the facility for reserving table ($seats$: $simple$; $name$: $simple$; $reservedTable$: $simple$); R_2, a catalogue of specialty cuisines ($menuType$: $simple$; $listing$: $complex$); R_3, a daily recipe for one of the specialty cuisine ($ingredients$: $complex$; $difficulty$: $simple$; $timeforPreparation$: $simple$; $preparation$: $complex$). In this scenario, the interface expressive power ($\mathcal{E}$) of

[11] Though few semantics of ODRL/$L(S)$ resembles to the ODRL Creative Commons Profile [25], the underlying clauses of a service license and the proposal of implementation within the WSDL of a service differ entirely. The meanings and motivations of ODRL/$L(S)$ data dictionary are related to the field of SOC. To the best of our knowledge, there exists no previous works on the aspects of service licenses using ODRL.

R is given by,

$$\mathcal{E} = n + \sum_{i=1}^{n}\left(\frac{\sum_{j=1}^{m}\delta_j}{m}\right) = 4 + (\frac{(3+3)}{2} + \frac{(1+1+1)}{3} + \frac{(1+3)}{2} + \frac{(3+1+1+3)}{4}) = 12$$

Consider R having the following clauses of licensing:

1. The license clauses of R may deny the provision of R_3 to other services intended for providing recipe information exclusively that means the service R denies reproduction.
2. R requires a service to be licensed same as R.
3. R allows composite works for noncommercial purposes.

The above clauses could be represented in ODRL/$L(S)$ as follows:

```
  <!-- Namespace Declarations -->
  <o-ex:offer>
    <o-ex:asset>
       <o-ex:context>
         <o-dd:uid>............</o-dd:uid>
       </o-ex:context>
    </o-ex:asset>
    <o-ex:permission>
       <ls:Composition/>
    </o-ex:permission>
    <o-ex:constraint>
       <ls:NonCommercialUse/>
       <ls:SimilarTerms/>
    </o-ex:constraint>
    <o-ex:requirement>
       <o-dd:attribution/>
    </o-ex:requirement>
  </o-ex:offer>
```

From the given licensing clauses of R, it is perceptible that R denies reproduction. A new service could not be created by directly using R. However R allows composition. Assuming R as a non-open service, R forbids derivation.

Another service, F, a restaurant finder service uses R, for the following operations: F_1, a restaurant locator giving a list of restaurants close to a given location and using R_0 (as well as similar operations for other restaurants); F_2, for intermediating table reservation, using R_1; F_3, a daily recipe randomly selected among the recipes provided by the restaurants listed using F (in the case of R, it will use operation R_3). F can use R in a composition even the reproduction is prohibited. R expects SimilarTerms license for F that is using R. In this case, the license terms of F will have to comply with R, for the request and deny provision of F_3 to other services intended to provide the recipe information exclusively (See Table 2).

Table 2. ODRL/$L(S)$ Clauses and Values for Service R

Identifier	**Value**	**Line numbers in ODRL/$L(S)$ listing**
Derivation (D)	No	(*Denied*)
Composition (C)	Yes	7 - 9
Reproduction (R)	No	(*Denied*)
Attribution (A)	Yes	14 - 16
SimilarTerms (T)	Yes	10 - 13
NonCommercialUse (N)	Yes	10 - 13

7 Concluding Remarks

Being a way to enable widespread use of services and to manage the rights between service consumers and service providers, licenses are critical to be considered in services. We have proposed a formal representation of licensing clauses to describe the licenses in machine understandable form that would be recognizable by services. We have extended ODRL to define the licensing clauses of services, as ODRL licenses are compatible with all service standards. We have focused on the aspects of copyrights and moral rights in this paper, introducing a free culture of services.

As composition federates independently developed services into a more complex service, the license proposed for the composed service should be consonant with the implemented licenses of individual services. In our future work, we intend to propose a framework to compare the service licenses, iterating over the licensing clauses of services to be composed. Based on the comparison of the rights expressed on services to be composed, the framework would also be able to suggest dynamically a license(s) for the composed service, yet legally enforceable.

Acknowledgements

We are grateful to Dr. Renato Ianella for his suggestions on enhancing ODRL for services. We acknowledge Prof. Michael Weiss and Prof. Fabio Casati for their suggestions. We thank anonymous reviewers for their helpful comments.

References

1. Foster, I.: Service Oriented Science. Science **308** (2005) 814–817
2. Szyperski, C.: Component Software: Beyond Object Oriented Programming. ACM Press, New York (1998)
3. D'Andrea, V., Gangadharan, G.R.: Licensing Services: The Rising. In: Proceedings of the IEEE Web Services Based Systems and Applications (ICIW'06), Guadeloupe, French Caribbean. (2006) 142–147

4. Papazoglou, M., Georgakopoulos, D.: Service Oriented Computing. Communications of the ACM **46**(10) (2003) 25–28
5. Heckel, R., Lohmann, M., Thone, S.: Towards a UML Profile for Service Oriented Architectures. In: Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications (MDAFA) . (2003)
6. Hamadi, R., Benatallah, B.: A Petri Net-based Model for Web Services Composition. In: Proceedings of the Fourteenth Australasian Database Conference on Database Technologies. (2003) 191–200
7. Goldstein, P.: International Copyright Principles, Law, and Practice. Oxford University Press (2001)
8. Fitzgerald, B., Oi, I.: Free Culture: Cultivating the Creative Commons. Media and Arts Law Review (2004)
9. Keller, A., Ludwig, H.: The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management **11**(1) (2003)
10. Skene, J., Lamanna, D., Emmerich, W.: Precise Service Level Agreements. In: Proc. of 26th Intl. Conference on Software Engineering (ICSE). (2004)
11. Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., Ma, W.: Management Applications of the Web Service Offerings Language. In: Proc. of the 15th CAiSE. (2003)
12. Jeffrey Schlimmer (Ed.): Web Services Policy Framework (WS-Policy). http://www-128.ibm.com/developerworks/webservices/library/ specification/ws-polfram/ (2004)
13. World Intellectual Property Organization: WIPO Copyright Treaty (WCT). http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html (1996)
14. Rosenblatt, B., Trippe, B., Mooney, S.: Digital Rights Management: Business and Technology. M & T Publishers, New York (2002)
15. Gunter, C., Weeks, S., Wright, A.: Models and Languages for Digital Rights. In: Proceedings of the HICSS-34. (2001)
16. Pucella, R., Weissman, V.: A Logic for Reasoning about Digital Rights. In: IEEE Proceedings of the Computer Security Foundations Workshop. (2002)
17. Chong, C., Corin, R., Etalle, S., Hartel, P., Law, Y.: LicenseScript: A Novel Digital Rights Language. In: Proceedings of the International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods. (2003)
18. D'Andrea, V., Gangadharan, G.R.: Licensing Services: An "Open" Perspective. In: Open Source Systems (IFIP Working Group 2.13 Foundation Conference on Open Source Software), Vol. 203, Springer Verlag. (2006) 143–154
19. Weyuker, E.: Evaluating Software Complexity Measures. IEEE Transactions on Software Engineering **14**(9) (1988) 1357–1365
20. D'Andrea, V., Fikouras, I., Aiello, M.: Interface Inheritance for Object Oriented Service Composition Based on Model Driven Configuration. In: Proceedings of ICSOC (Short Papers). (2004) 66–74
21. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services Concepts, Architectures, and Applications. Springer Verlag (2004)
22. World Intellectual Property Organization: Successful Technology Licensing. WIPO Publishers, Geneva, Switzerland (2004)
23. ContentGuard Inc.: XrML: The Digital Rights Language for Trusted Contents and Services. http://www.xrml.org/ (Accessed on May 2006)
24. Renato Iannella (Ed.): Open Digital Rights Language (ODRL) Version 1.1. http://odrl.net/1.1/ODRL-11.pdf (2002)
25. Renato Ianella (Ed.): ODRL Creative Commons Profile. http://odrl.net/Profiles/CC/SPEC.html (2005)