International Journal of Cooperative Information Systems
Vol. 17, No. 3 (2008) 301–317

World Scientific
www.worldscientific.com

SERVICE LICENSING COMPOSITION AND COMPATIBILITY ANALYSIS

G. R. GANGADHARAN* and VINCENZO D'ANDREA†

Department of Information Engineering and Computer Science
University of Trento, Trento, 38100 Italy
**gr@disi.unitn.it*
†*vincenzo.dandrea@unitn.it*

MICHAEL WEISS

Department of Systems and Computer Engineering
Carleton University, Ottawa, K1S 5B6, Canada

RENATO IANNELLA

National ICT Australia
Brisbane, 4000 Australia

Services enable the transformation of the World Wide Web as distributed interoperable systems interacting beyond organizational boundaries. Service licensing enables broader usage of services and a means for designing business strategies and relationships. A service license describes the terms and conditions for the use and access of the service in a machine interpretable way that services could be able to understand. Service-based applications are largely grounded on composition of independent services. In that scenario, license compatibility is a complex issue, requiring careful attention before attempting to merge licenses. The permissions and the prohibitions imposed by the licenses of services would deeply impact the composition. Thus, service licensing requires a comprehensive analysis on composition of these rights and requirements conforming to the nature of operations performed and compensation of services used in composition. In this paper, we analyze the compatibility of service license by describing a matchmaking algorithm. Further, we illustrate the composability of service licenses by creating a composite service license that is compatible with the licenses being composed.

Keywords: Service licensing; rights expression; service composition.

1. Introduction

Service-oriented computing (SOC) represents the convergence of technology with an understanding of cross-organizational business processes.[18] Services enhance the World Wide Web not only for human use, but also for machine use by enabling application level interactions. Services are an important advance over standalone applications: they intend to make network-accessible operations available anywhere and at anytime. Thus, services deliver complex business processes

and transactions, allowing applications to be constructed on-the-fly and to be reused.[1]

In a dynamic market environment, the usage of services is governed by bilateral agreements that specify the terms and conditions of using and provisioning the services. A license is an agreement between parties in which one party receives benefits by giving approximately equal value to the other party in exchange. Licensing[5] includes all transactions between the licensor and the licensee, in which the licensor agrees to grant the licensee the right to use and access the asset under predefined terms and conditions.

The trend of software transforming to a service-oriented paradigm demands for a new way of licensing for services.[6] Different types of licenses exist for software, but the nature of services differs significantly from traditional software and components, thus preventing the direct adoption of software and component licenses. As services are being accessed and consumed in a number of ways, a spectrum of licenses suitable for services with differing license clauses can be definable.

As services are composed with one another, the associated service licenses are also to be composed. The license of a composite service should be compatible with the licenses of the services being composed. In this paper, we propose an environment for composing licenses and analyzing the compatibility between the licenses in case of service composition. The salient feature of our approach is a matchmaking algorithm for compatibility analysis of licenses (at license clause level). We also discuss the creation of a composite service license based on the compatibility of candidate service licenses.

The paper is organized as follows: In Sec. 2, we briefly explain service licensing clauses followed by the representation of service licenses using ODRL Service Licensing Profile (ODRL-S) in Sec. 3. Section 4 provides details of a matchmaking algorithm and analyzes the compatibility between licenses at the level of elements. The process of service license composition based on the compatibility of candidate service licenses is illustrated in Sec. 5. Section 6 presents certain licensing scenarios of real word web services. Section 7 discusses related work in this field, showing the distinct contributions of this paper.

2. Service Licensing Clauses Explained

A service license describes the terms and conditions that permit the use of and access to a service, in a machine readable way, which services can understand. Some of the key clauses of a service license are elucidated as follows.[a]

[a]Although, we have undertaken an endeavor to represent a "standard form" of a service license, we do not claim that the given anatomy of a service license is complete. It is almost impossible to generalize all the terms of a license. This article is not intended as a substitute for legal advice. We highly recommend service providers and service consumers to obtain appropriate legal counsel to make use of licenses for their services.

2.1. *Subject*

The subject of a license relates to the definition of the service being licensed. This includes an unique identification code for the service, a name for the service, location of the service and other additional relevant information.

2.2. *Scope of rights*

The Scope of Rights are the set of choices of rights that the licensor authorizes the licensee to exercise in a service. The scope of rights of a service license reflect on what could be done with the service. Following are the possible rights associated with a service.

As generally stated by copyright laws, the creator of a service retains all the so called "exclusive" rights associated with the service itself; in other words, it is possible to exclude others from any right related to the service. On the other hand, the following terms express the situation where the copyright holder is reducing her exclusive rights, providing service consumers the possibility to create other services based on the existing one.

Composition: Composition of services specifies the participating services, the invocation sequence of services and the methods for handling exceptions.[1] In accordance with widespread use of this term in SOC, we refer to the composition as the federation of a service with other remote services. The operations of a composite service relies on the availability of services being composed, at the time of service invocation.

Adaptation: We use the term "adaptation" to signify the making of a new independent service from an existing service interface without modifying the implementation. In other words, the "adapted" service can be described as a copy of the executable files deployed in a different context.

Derivation: Derivation of a service is a novel aspect of creating a new service from existing service, modifying service interface and service implementation. Furthermore, derivation requires independent execution of the service being created. The difference between Derivation and Adaptation is in the provision of the right to modify the service before deploying it in a different context. This right is inspired from similar philosophy in Free and Open Source Software.[14]

Moral rights are the connecting threads between an author and her creation. Moral rights refer to the ability of authors to control the eventual fate of their creations.[19]

Attribution: A service may require attribution (one of the moral rights) for its use by other services. Thus, attribution is ascribing a service to the entity responsible for its creation.

A service license can associate certain clauses that impose some requirements or describe a kind of use of a service. Following are these type of licensing clauses beyond exclusive rights and moral rights.

Sharealike: A service creator could impose that when a new service is created based on her service, the license of the new service retains the same terms and conditions of the source service. From the perspective of service providers and developers, this right of services could be seen as a restriction imposed to the new service, that allows value addition solely with the same conditions that the original has. However, from the perspective of a service consumer, this could be viewed as an ultimate guide for using any value-added services inheriting from a particular similar termed service.

Non-Commercial Use: A service can be used either for non-commercial purposes or for commercial purposes. By including the clause of non-commercial use, a service denies its use for commercial purposes.

2.3. *Financial terms*

Service consumers make payments either as royalties or lump sum for using services.[b] Generally royalties are based on per unit sales. In case of services, royalties can be viewed as the amount for per-use of a service (not considering a possible discount for volume sales). For a payment of p per use, a service consumer has to pay $R = n * p$ where n being the number of times the service has been used. In this case, p can be renewed annually or over the life of license.

Lump sum payments are alternative method to royalties. Sometimes lump sum payments are also used in addition to royalties. In case of services, a lump sum payment can be paid by a service consumer before using the service (prepay) or at a later stage (postpay). By paying lump sum amount, a licensee obtains rights to use the given service (irrespective of number of times that services being invoked).

2.4. *Warranties, indemnities, and limitation of liabilities*

Warranties describe functional and non-functional properties of services, provided as a way of attracting and retaining consumers. In a service license, the representation of warranties are optional. Warranties are generally similar to the notions given in WSLA[15] and in SLAng.[20]

A service license also specifies indemnification clauses,[4] a way of defense by the licensor for the licensee if a third party sues the licensee, alleging that the licensee's use of the licensed software infringes or violates the third party's intellectual rights.

Limitation of liability clauses restrict the liability of each of the parties under the license agreement.

2.5. *Evolution*

The licensing clauses of evolution define access rights to an evolved service. Over time, a service can evolve as follows:

- Modifications by the provider in functional and/or non-functional properties of a given service, represented by new releases or new versions. A service license

[b]There can be other possible models defining payment mechanisms for services.

is expected to support backward compatibility. In other words, consumers using earlier versions of a licensed service should be allowed to continue to use the service even as it evolves, at least until support for the older version is formally withdrawn.
- Termination of the current running service and substitution by a new service with different behavior.
- Replacing a service by a more generic service.

3. ODRL Service Licensing Profile (ODRL-S)

To produce a service license in machine interpretable form, service licensing clauses should be expressed in an appropriate manner. We have developed the language ODRL-S[8] by extending the Open Digital Rights Language (ODRL)[11] to implement the clauses of service licensing in machine interpretable form (see Fig. 1).

The anatomy of a service license in ODRL-S is as follows:

- The *Subject* model of a service license directly adopts the ODRL Asset Model.[11]
- The *Scope of Rights* (see Ref. 7 for details) in ODRL-S comprise the extended ODRL Permission, Requirement, and Constraint Models. ODRL-S reuses the concept of sharealike and non-commercial use from the ODRL Creative Commons profile.[12] Attribution to services is facilitated by the ODRL attribution element.
- We adopt the ODRL payment model to represent the *Financial* model of services.
- The *WIL* model defines new terms to represent warranties, indemnities and limitation of liabilities associated with services.
- The *Evolution* model specifies new terms to specify rights of access to a consumer of a given service for new releases or new versions in which the provider modifies functional and/or non-functional properties.

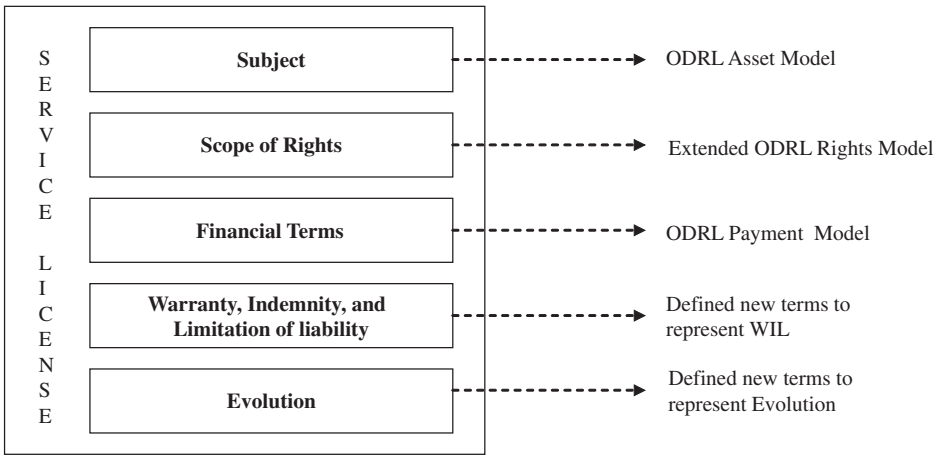

Fig. 1. Conceptual mapping of service licensing clauses in ODRL-S.

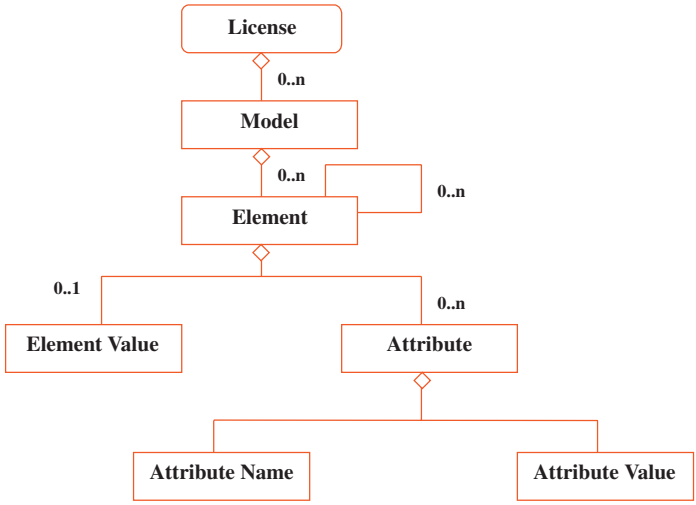

Fig. 2. ODRL-S license representation.

A license in ODRL-S for a service consists of a finite set of models (generally referred as license clauses), each of which further consists of a set of elements. Elements can be specified with value or without value (empty element having the element type only). Elements can contain other elements that can give rise to an arbitrarily deep hierarchy of elements within elements. An element contained in other element (the process of nesting) is called subentity in ODRL. Elements can have attributes which are specified generally with name value pair in the element's open tag. A service license is modeled in ODRL-S as shown in Fig. 2.

4. Service Licenses Matchmaking and Compatibility Analysis

Service license compatibility analysis is a process of matchmaking of candidate service licenses (at license clause level) in composition. The matchmaking algorithm performs the compatibility analysis between any two given licenses to decide whether they are compatible. A license is compatible with another license if all license clauses are compatible. Service licenses can be combined, if they are found compatible by the algorithm, allowing the corresponding services to be composed.

There are certain elements of licenses which are broader in scope of operation than certain other elements. Assume two services with different license elements say, composition and derivation. If a consumer is looking for a service allowing composition, a service license allowing derivation can also be used, because derivation subsumes composition. For this reason, we say that derivation and composition are compatible. For a complete compatibility analysis, the matchmaking algorithm must know about the possible subsumptions. The concept of subsumption (at the element level) is similar to the concept of redefinition of a method in a sub-class.[13] Subsumption implies a match that should occur, if the given license element is more

Table 1. Subsumption rules over Scope of Rights elements.

Element 1	Element 2	Element 1 versus Element 2	Redefinition
Composition	Adaptation	Composition $\supset$ Adaptation	Composition
Derivation	Adaptation	Derivation $\supset$ Adaptation	Derivation
Derivation	Composition	Derivation $\supset$ Composition	Derivation
Adaptation	Composition	Adaptation $\subset$ Composition	Composition
Adaptation	Derivation	Derivation $\subset$ Adaptation	Derivation
Composition	Derivation	Derivation $\subset$ Composition	Derivation

permissive (accepts more) than the corresponding element in the other license. The subsumption rules for *Scope of Rights* are given below (see Table 1).

There could also be a scenario when analyzing the compatibility of service licenses where one of the licenses contains clauses that the other license does not. In certain cases, the absence of one or several of these clauses does not affect the compatibility with the other license. Table 2 lists rules used by the matchmaking

Table 2. Compatibility analysis for *Scope of Rights* and *Financial Terms* elements.

Element α	Element β	Compatibility	Rationale
Adaptation	Unspecified	*Compatible*	A license having adaptation clause can be compatible with a license that does not specify adaptation.
Composition	Unspecified	*Incompatible*	A license denying composition cannot be compatible with a license allowing composition.
Composition	Adaptation	*Compatible*	Based on subsumption (Table 1)
Derivation	Unspecified	*Incompatible*	Derivation requires the source code of interface and implementation to be "Open" and Copyright law defaults as "closed".
Derivation	Adaptation or Composition	*Compatible*	Based on subsumption (Table 1)
Attribution	Unspecified	*Compatible*	The requirement for specification of attribution will not affect the compatibility when unspecified.
Sharealike	Unspecified	*Compatible*	Sharealike affects the composite license requiring that the composite license should be similar to the license having Sharealike element.
Noncommercial use	Unspecified	*Incompatible*	The commercial use is denied if a license specifies the use of service for non-commercial purposes. A service that does not specify noncommercial use clause cannot be compatible with a license specifying noncommercial use.
Payment	Unspecified	*Compatible*	Payment elements do not affect compatibility directly, if unspecified. The license elements related to payment and charging are dependent on service provisioning issues.

algorithm to determine the compatibility of specified against unspecified ("don't care") elements.

The matchmaking algorithm compares a license clause of a license with another license clause of another license. The algorithm analyzes the compatibility of licenses at the element level for any two given licenses[c] to decide whether they are compatible.

Assuming that semantics inside a license are agreed by service providers and consumers, the algorithm for matching a license L_α (for a service α) with another license L_β (for a service β) is given as follows.[d] In the following, we use the symbol $\rightleftharpoons$ to denote compatibility. Two licenses are compatible (referred as $L_\alpha \rightleftharpoons L_\beta$), if all the respective models in both the licenses are compatible.

$$\begin{aligned} & (\forall m_\alpha : m_\alpha \epsilon L_\alpha \quad \exists m_\beta : m_\beta \epsilon L_\beta \quad \Rightarrow \quad (m_\alpha \rightleftharpoons m_\beta)) \\ \wedge \quad & (\forall m_\beta : m_\beta \epsilon L_\beta \quad \exists m_\alpha : m_\alpha \epsilon L_\alpha \quad \Rightarrow \quad (m_\alpha \rightleftharpoons m_\beta)) \end{aligned}$$

A model m_α is compatible with another model m_β, if the model types are same (represented by $\equiv$) and their elements are compatible.

$$\begin{aligned} & & & & & (m_\alpha \equiv m_\beta) \\ \wedge \quad & (\forall e_\alpha : e_\alpha \epsilon Elements(m_\alpha) & \exists e_\beta : e_\beta \epsilon Elements(m_\beta) & \Rightarrow & (e_\alpha \rightleftharpoons e_\beta)) \\ \wedge \quad & (\forall e_\beta : e_\beta \epsilon Elements(m_\beta) & \exists e_\alpha : e_\alpha \epsilon Elements(m_\alpha) & \Rightarrow & (e_\alpha \rightleftharpoons e_\beta)) \end{aligned}$$

Now, an element e_α is compatible with another element e_β, if:

- e_α and e_β have same type (represented by $\equiv$) or e_α can be redefined as e_β using Table 1 ($Redefinition(e_\alpha, e_\beta)$) or in case of unspecification of either e_α or e_β, use Table 2 ($Unspecification(e_\alpha, e_\beta)$) for looking the compatible element.
- e_α and e_β have equal value.
- for all nested elements, corresponding elements are compatible.
- all attributes of e_α and e_β are compatible.

$$\begin{aligned} & ((e_\alpha \equiv e_\beta) \quad \vee \quad Redefinition(e_\alpha, e_\beta) \quad \vee \quad Unspecification(e_\alpha, e_\beta)) \\ & & & & \wedge \quad (Value(e_\alpha) = Value(e_\beta)) \\ \wedge \quad & (\forall e_\alpha : e_\alpha \epsilon Elements(e_\alpha) & \exists e_\beta : e_\beta \epsilon Elements(e_\beta) & \Rightarrow & (e_\alpha \rightleftharpoons e_\beta)) \\ \wedge \quad & (\forall e_\beta : e_\beta \epsilon Elements(e_\beta) & \exists e_\alpha : e_\alpha \epsilon Elements(e_\alpha) & \Rightarrow & (e_\alpha \rightleftharpoons e_\beta)) \\ \wedge \quad & (\forall a_\alpha : a_\alpha \epsilon Attributes(e_\alpha) & \exists a_\beta : a_\beta \epsilon Attributes(e_\beta) & \Rightarrow & (a_\alpha \rightleftharpoons a_\beta)) \\ \wedge \quad & (\forall a_\beta : a_\beta \epsilon Attributes(e_\beta) & \exists a_\alpha : a_\alpha \epsilon Attributes(e_\alpha) & \Rightarrow & (a_\alpha \rightleftharpoons a_\beta)) \end{aligned}$$

[c]The described algorithm does not support service consumer and service provider relationship between the given licenses, thus bypassing the directional issues of compatibility.

[d]We denote a license of a service S by L_S. Subsequently, we refer to the elements of a model in a service S as $Elements(m_S)$ and the elements nested inside an element as $Elements(e_S)$. We refer to the attributes of an element in a service S as $Attributes(e_S)$ and the value of an attribute by $Value(a_S)$.

Algorithm 4.1 Matchmaking Algorithm

```
for all (m_α, m_β) and (modelname(m_α) = modelname(m_β)) do
  for all e_α ∈ m_α do
    for all e_β ∈ m_β do
      bool res = ElementCompatibility(e_α, e_β)
      if (¬res) then
        Terminate
      end if
    end for
  end for
  m_α and m_β compatible
end for
```

An attribute a_α is compatible with another attribute a_β, if the attributes are of same type (represented by $\equiv$) and the associated values of attributes are equal.

$$(a_\alpha \equiv a_\beta) \wedge (Value(a_\alpha) = Value(a_\beta))$$

The described matchmaking algorithm is represented as pseudocode in Algorithm 4.1.

Consider an example of a restaurant service R, composed of a resource allocation service (I) and a map service (M). Assume that I allows derivation and costs 1 Euro per use of the service. Furthermore, the service requires attribution. The license for the service I (say L_I) is represented in ODRL-S as follows.

```
<o-ex:offer>
<o-ex:permission>
   <sl:derivation/>
</o-ex:permission>
<o-ex:requirement>
   <o-dd:attribution/>
</o-ex:requirement>
<o-ex:requirement>
   <o-dd:peruse>
      <o-dd:payment>
         <o-dd:amount o-dd:currency ="EUR"> 1.00 </o-dd:amount>
      </o-dd:payment>
   </o-dd:peruse>
</o-ex:requirement>
</o-ex:offer>
```

Assume that the map service M allows composition and requires attribution. However, this service denies commercial use. The license (say L_M) for the service

M is represented in ODRL-S as follows.

```
<o-ex:offer>
<o-ex:permission>
   <sl:composition/>
</o-ex:permission>
<o-ex:requirement>
   <o-dd:attribution/>
</o-ex:requirement>
<o-ex:constraint>
   <cc:NonCommercialUse/>
</o-ex:constraint>
</o-ex:offer>
```

procedure boolean $ElementCompatibility(e_\alpha, e_\beta)$

if $(type(e_\alpha) = type(e_\beta)) \wedge (value(e_\alpha) = null) \wedge (value(e_\beta) = null)$ **then**
  return TRUE
end if
if $(type(e_\alpha) = type(e_\beta)) \wedge (value(e_\alpha) = value(e_\beta)) \wedge (attributename(e_\alpha) = null) \wedge (attributename(e_\beta) = null)$ **then**
  return TRUE
end if
if $((type(e_\alpha) = type(e_\beta)) \wedge (value(e_\alpha) = value(e_\beta)) \wedge AttributeCompatibility(e_\alpha, e_\beta) = TRUE)$ **then**
  return TRUE
end if
if $((type(e_\beta) = composition) \vee (type(e_\beta) = derivation) \vee (type(e_\beta) = noncommercialuse) \wedge (type(e_\alpha) = unspecified))$ **then**
  return FALSE
end if
if $((type(e_\beta) = adaptation) \vee (type(e_\beta) = attribution) \vee (type(e_\beta) = sharealike) \vee (type(e_\beta) = payment) \wedge (type(e_\alpha) = unspecified))$ **then**
  return TRUE
end if
if $((type(e_\alpha) = composition) \vee (type(e_\alpha) = derivation) \vee (type(e_\alpha) = noncommercialuse) \wedge (type(e_\beta) = unspecified))$ **then**
  return FALSE
end if
if $((type(e_\alpha) = adaptation) \vee (type(e_\alpha) = attribution) \vee (type(e_\alpha) = sharealike) \vee (type(e_\alpha) = payment) \wedge (type(e_\beta) = unspecified))$ **then**
  return TRUE
end if

procedure boolean *AttributeCompatibility*(a_α, a_β)
1: **if** ($attributename(a_\alpha) = attributename(a_\beta)$) **then**
2: **if** ($value(a_\alpha) = value(a_\beta)$) **then**
3: a_α and a_β compatible
4: return TRUE
5: **end if**
6: **end if**

Assume that we now want to analyze the compatibility between license L_I and another license L_M.

Following the matchmaking algorithm, we compare licenses at the model level. Line 2 of both licenses are `<o-ex:permission>` models. The element in line 3 of L_I (`<sl:derivation>`) is not of the same type as in line 3 of L_M (`<sl:composition>`), but we can redefine one (derivation) as the other (composition) by applying a rule from Table 1 (derivation subsumes composition).

We compare the model `<o-ex:requirement>` containing the element `<o-ex:attribution>` in lines 5, 6, and 7 of L_I and L_M. As the models are of the same type and the elements are of the same type, the model is compatible.

Then, in line 8 of L_I, the `<o-ex:requirement>` model contains the element `<o-dd:peruse>`, which contains `<o-dd:payment>` element, and in turn, contains `<o-dd:amount>`. The corresponding payment term specifications are not specified in L_M. (The service offered by L_M can be made available free of charge, without specifying the payment model.)

The `<o-ex:constraint>` model of license L_M (in lines 8 and 9) specifies the element `<cc:NonCommercialUse>`. When the algorithm looks for the element `<cc:NonCommercialUse>` in L_I, the algorithm is unable to find as the element is unspecified. This indicates that the service with L_I can be used for commercial purposes. From Table 2, the algorithm finds that these clauses are incompatible, and thus the licenses become incompatible.

5. Service License Composition

Service composition combines independently developed services into a more complex service.[3] The license of the composite service should be consistent with the licenses of the individual services. Composability of licenses refers to the generation of the composite service license from the given service licenses for the services being composed. A pre-requisite for the composability of licenses is that the licenses are to be compatible.

A lookup in a service directory for services with a given functionality may result in multiple candidate services. Each candidate service may be provided under a different license. When the services are composed, there can be several

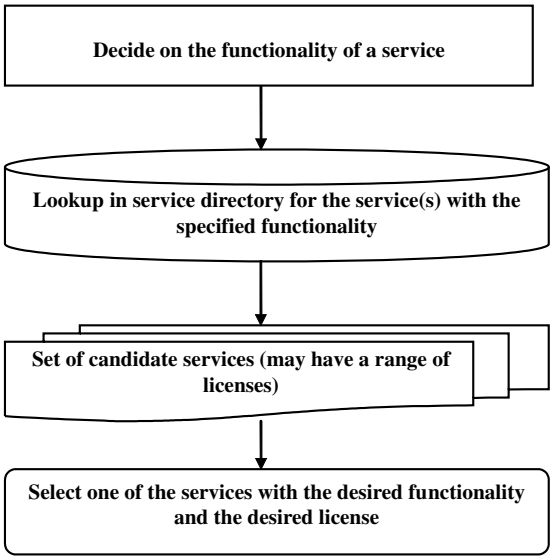

Fig. 3. Process of a service license selection with the service functionality.

licenses for the composite service. The process of a license selection for a service is depicted in Fig. 3. The service consumer or service aggregator could manually select one of the services with the desired functionality and the desired license. Otherwise, the process assigns a license to the composite service (may be the most permissible).

Consider the case where a map service M allows composition and requires attribution, when M is used by other services. The license of M (L_M) in ODRL-S is as follows.

```
1   <o-ex:offer>
2      <o-ex:permission>
3         <sl:composition/>
4      </o-ex:permission>
5      <o-ex:requirement>
6         <o-dd:attribution/>
7      </o-ex:requirement>
8   </o-ex:offer>
```

Assume that a resource allocation service I allows access to the source code of the service (derivation) and requires a fee of 1 Euro per use and thus license of I (L_I) is same as the license of I shown in the previous section.

L_M and L_I can be composed as they are compatible according to the matchmaking algorithm illustrated in the previous section. The composition of these service licenses could generate a set of licenses that R may select. R can have the following license (one of the licenses in the set of compatible licenses), compatible

with L_M and L_I.

```
1   <o-ex:offer>
2      <o-ex:permission>
3         <sl:derivation/>
4      </o-ex:permission>
5      <o-ex:requirement>
6         <o-dd:attribution/>
7      </o-ex:requirement>
8      <o-ex:requirement>
9         <o-dd:peruse>
10           <o-dd:payment>
11              <o-dd:amount o-dd:currency="EUR">1.00</o-dd:amount>
12           </o-dd:payment>
13        </o-dd:peruse>
14     </o-ex:requirement>
15  </o-ex:offer>
```

We compare the candidate service licenses (L_I and L_M) at the model level. Line 2 of both licenses are `<o-ex:permission>` models. The elements in line 3 of L_I (`<sl:derivation>`) and line 3 of L_M (`<sl:composition>`) are not of the same type, but we can redefine one (composition) as the other (derivation) by applying a rule from Table 1 (derivation subsumes composition). We compare the model `<o-ex:requirement>` with the element `<o-ex:attribution>` in lines 5, 6, and 7 of L_I and L_M. As the models are of the same type and the elements are of the same type, the model is compatible. Then, in line 8 of L_I, the `<o-ex:requirement>` model contains the element `<o-dd:peruse>`, which contains `<o-dd:payment>` element, and in turn, contains `<o-dd:amount>`. The corresponding payment term specifications are not specified in L_M. The service offered by L_M can also be made available free of charge, without specifying the payment model. Thus, the given candidate service licenses are compatible.

Now, we compare the proposed composite license (L_R) with each of the candidate service licenses L_I and L_M. Comparing L_I with L_R, line 2 of both licenses are models of same type (`<o-ex:permission>`). The elements in line 3 of L_I (`<sl:derivation>`) and line 3 of L_R (`<sl:derivation>`) are the same, thus the models become compatible. In lines 5, 6, and 7 of L_I and L_R, we compare the model `<o-ex:requirement>` with the element `<o-ex:attribution>`. As the models are of the same type and the elements are of the same type, the model is compatible. In lines 8–11 of L_I and L_M, we compare the model `<o-ex:requirement>` with the element `<o-ex:attribution>`. As the models are of the same type and the elements are of the same type, the model is compatible.

Comparing L_M with L_R, line 2 of both licenses are `<o-ex:permission>` models. The elements in line 3 of L_M (`<sl:composition>`) and line 3 of L_R (`<sl:derivation>`) are not of the same type, but by redefining using Table 1 (derivation subsumes composition), they become compatible. In lines 5, 6, and 7 of L_M and L_R, we compare the model `<o-ex:requirement>` with the element

<o-ex:attribution>. As the models are of the same type and the elements are of the same type, the model is compatible. Lines 8–14 of L_R describe <o-dd:payment> terms whereas payment terms are unspecified in L_M. Referring to Table 1, this model becomes compatible.

6. Towards Real World Service Licensing Scenarios

In real world, though service licenses are not much familiar, all proprietary web services have terms and conditions for service use. These terms are written in a natural language like English. However, we can attempt a representation of these terms in ODRL-S.[e]

The clause 12 of Yahoo! Terms of Service[f] specifies as follows.

> You agree not to reproduce, duplicate, copy, sell, trade, resell or exploit for any commercial purposes, any portion of the Service (including your Yahoo! ID), use of the Service, or access to the Service.

This clause can be represented in ODRL-S as <o-cc:NonCommercialUse>.

The clause 4.2.3 of Amazon Web Services Customer Agreement[g] specifies as follows.

> You may not remove, obscure, or alter any notice of any Mark, or other intellectual property or proprietary right appearing on or contained within the Services or on any Amazon Properties.

We can represent this clause equivalent to <cc:Notices> in ODRL-S.

When service consumers or service aggregators use Google, Amazon, Yahoo! or other similar services with their services, it is required for consumers and aggregators to understand these terms. The terms of these services should be expected to be compatible. Representing licensing clauses in a machine interpretable way would facilitate the automated compatibility analysis.

7. Related Work and Discussion

Though there are examples of service licenses in practical use (by Amazon, Google, Yahoo!), to the best of our knowledge, there appears to be no conceptualization of service licensing in general. The business and legal contractual information are not described at a detailed level by the services research community, either in industry or academia. Though the design of service licenses seems to be an initiative of the

[e]The given licensing clauses in ODRL-S does not necessarily represent the meaning of terms of service as given by the corresponding service providers. Furthermore, the machine interpretable form does not represent the views of the given service provider and/or associated third party sources.

[f]http://info.yahoo.com/legal/us/yahoo/utos/utos-173.html (accessed on March, 2008).

[g]http://www.amazon.com/AWS-License-home-page-Money/b?ie=UTF8&node=3440661 (accessed on March, 2008).

software industry, there is no active involvement in this topic by the industry. One of the primary causes for this could be fear still faced by industries over the lack of standardization of technologies surrounding service-oriented computing. The need for a language defining both the internal business needs of an organization and its requirements on external services, and for a systematic way of linking them to business processes is proposed in Ref. 17. Because of the mechanism of technology transfer, licensing addresses how a process is related to and affects business requirements and needs, describing the legal requirements. Licenses affect the design of business strategies and relationships, linking the business processes across boundaries.

In the business domain, consumer confidence is established through a contract with the service provider. In SOC, service level agreements (SLA) and policies support these contractual terms. A service license primarily focuses on the usage and provisioning terms of services. A service license may include the SLA terms. Thus, a service license is broader than the scope of SLA, protecting the rights of service providers and service consumers. In general, an agreement is negotiated between the service provider and the service consumer and agree upon a SLA that covers a service (or a group of services). The agreement is terminated when either of the party terminates or violates the agreement. If one of the partners violates the agreement, the agreement might be renegotiated (in case of recoverable violation). In the case of a service license, there is a service provider that plays the main role of the licensor. There could be many service consumers (the licensees) binded by the service license. The agreement between the service provider and a consumer is bound to comply with license clauses, but the license itself is generally not part of the negotiation. If a license is modified, it leads to the creation of a new version of the license. A new invocation of a service might use the modified version of the license. However, the unmodified version of the license, if it is implemented and executed by a service, will remain active and will not be overridden by the new version.[9]

Current SLA and policies specifications for services (WSLA,[15] SLAng,[20] WSOL,[21] WS-Agreement,[2] WS-Policy[22]) define what to measure/monitor and describe payments/penalties. Generally, all the specifications focus on the QoS and the terms and conditions agreed by the provider and consumer. License clauses[26] are unexplored by current service description standards and languages (as mentioned above). We have proposed ODRL-S as a language to represent a service license concretely in a machine interpretable form so that any services can automatically interpret license clauses. Using ODRL-S, a service license can be described in service level and feature level.[8]

Compatibility between services is one of the active research areas in service-oriented computing. The present researches on the compatibility of services have been focused on the matching of functional properties of services.[16,25] A selection processes for commercial-off-the-shelf components using some of the non-technical features is addressed in Ref. 23, vaguely related to our work. An interesting approach for matching non-functional properties of Web services represented using WS-Policy is described in Ref. 24. The most comprehensive work on automated compatibility

analysis of WSLA service level objectives is elaborated in Ref. 27. However, license clauses are not simple as in the case of service level objectives of WSLA or policies of WS-Policy and the algorithm presented in Ref. 27 cannot be parse service license clauses. The problem of licensing compatibility is difficult to resolve automatically as license clauses are generally written in a natural language (like English) and contains highly legalized terms, sometimes even difficult for the end users to understand. A comprehensive semantic approach for digital rights management based on ontologies is proposed in Ref. 10. However, the framework does not describe the rights expression for services and their composition. To the best of our knowledge, there is no research on a framework for composing licenses (at least semi-automatically) for services. Not only have we developed an algorithm for matchmaking of service licenses, but we have also proposed the way of composing candidate service licenses. The illustrated compatibility analysis of service licenses in the element level can be applicable to analyze the compatibility of licenses in any digital assets. We position our work as a complementary approach in service license composition.

8. Concluding Remarks

The full potential of services as a means of developing dynamic business solutions will only be realized when cross organizational business processes can federate in a scale-free manner. Today, services offer programmatic interfaces to applications. However, many available services are not even considered to provide relevant business value. As a way of managing the rights between service consumers and service providers, licenses are of critical importance to services. In this paper, we have analyzed the compatibility between licenses by describing a comprehensive service license matchmaking algorithm and described the composition of service licenses. In our ongoing work, we are describing license conflicts during service composition and resolving them by feature interactions.

References

1. G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web Services Concepts, Architectures, and Applications* (Springer Verlag, 2004).
2. A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke and M. Xu, Web Services Agreement Specification (WS-Agreement) Version 2005/09, www.gridforum.org, 2005.
3. F. Casati and M.-C. Shan, Dynamic and adaptive composition of e-services, *Information Systems* **6**(3) (2001).
4. A. Chavez, C. Tornabene and G. Wiederhold, Software component licensing: A primer, *IEEE Software* **15**(5) (1998) 47–53.
5. W. Classen, Fundamentals of software licensing, *IDEA: The Journal of Law and Technology* **37**(1) (1996).
6. V. D'Andrea and G. R. Gangadharan, Licensing services: The rising, in *Proceedings of the IEEE International Conference on Internet and Web Applications and Services (ICIW'06)*, Guadeloupe, French Caribbean (2006), pp. 142–147.

7. G. R. Gangadharan and V. D'Andrea, Licensing services: Formal analysis and implementation, in *Proceedings of the Fourth International Conference on Service Oriented Computing (ICSOC'06)*, Chicago, USA (2006), pp. 365–377.
8. G. R. Gangadharan, V. D'Andrea, R. Iannella and M. Weiss, ODRL service licensing profile (ODRL-S), in *Proceedings of the 5th International Workshop for Technical, Economic, and Legal Aspects of Business Models for Virtual Goods* (2007).
9. G. R. Gangadharan, G. Frankova and V. D'Andrea, Service license life cycle, in *Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS'07)* (2007).
10. R. Garcia, R. Gil and J. Delgado, A web ontologies framework for digital rights management, *Journal of Artificial Intelligence and Law Online First* (2007), http://springerlink.metapress.com/content/03732x05200u7h27.
11. R. Iannella, ed. *Open Digital Rights Language (ODRL) Version 1.1* (2002), http://odrl.net/1.1/ODRL-11.pdf.
12. R. Iannella, ed. *ODRL Creative Commons Profile* (2005), http://odrl.net/Profiles/CC/SPEC.html.
13. J. Jezequel, M. Train and C. Mingins, *Design Patterns and Contracts* (Addison-Wesley, 1999).
14. A. Laurent, *Understanding Open Source and Free Software Licensing* (O'Reilly, 2004).
15. H. Ludwig, A. Keller, A. Dan, R. King and R. Franck, Web Service Level Agreement (WSLA) Language Specification, IBM Corporation, 2003.
16. M. Paolucci, T. Kawamura, T. Payne and K. Sycara, Semantic matching of web services capabilities, in *Proceedings of the First International Semantic Web Conference (ISWC)* (2002).
17. M. Papazoglou, P. Traverso, S. Dustdar, F. Leymann and B. Kramer, Service oriented computing research roadmap, in *Dagstuhl Seminar Proceedings 05462 (SOC)* (2006).
18. L. Paulson, Services science: A new field for today's economy, *IEEE Computer* **39**(8) (2006) 18–21.
19. B. Rosenblatt, Moral Rights Basics (1998), http://cyber.law.harvard.edu/property/.
20. J. Skene, D. Lamanna and W. Emmerich, Precise service level agreements, in *Proceedings of 26th International Conference on Software Engineering (ICSE)* (2004).
21. V. Tosic, B. Pagurek, K. Patel, B. Esfandiari and W. Ma, Management applications of the web service offerings language, in *Proceedings of the 15th Conference on Advanced Information Systems Engineering* (2003).
22. A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and U. Yalcinalp, Web Services Policy Framework (2007), http://www.w3.org/TR/ws-policy.
23. J. P. C. Vega, X. Franch and C. Quer, Towards a unified catalogue of non-technical quality attributes to support COTS-based systems lifecycle activities, in *Proceedings of the IEEE International Conference on COTS Based Software Systems (ICCBSS)* (2007), pp. 21–32.
24. K. Verma, R. Akkiraj and R. Goodwin, Semantic matching of web service policies, in *Second International Workshop on Semantic and Dynamic Web Processes* (2005).
25. Y. Wang and E. Stroulia, Flexible interface matching for web service discovery, in *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE)* (2003).
26. World Intellectual Property Organization, WIPO Copyright Treaty (WCT) (1996), http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html.
27. W. Yang, H. Ludwig and A. Dan, Compatibility analysis of WSLA service level objectives, Technical Report RC22800 (W0305-082), IBM Research Division (2003).