Consumer-specified Service License Selection and Composition

G.R. Gangadharan[†], Hong-Linh Truong[§], Martin Treiber[§],
Vincenzo D'Andrea[†], Schahram Dustdar[§], Renato Iannella[¶], Michael Weiss[‡]
[†] University of Trento, Trento, Italy
{gr, dandrea}@dit.unitn.it
[§] Vienna University of Technology, Austria
{truong, treiber,dustdar}@infosys.tuwien.ac.at
[¶] National ICT Australia, Brisbane, Australia
renato@nicta.com.au
[‡] Carleton University, Ottawa, Canada
weiss@scs.carleton.ca

Abstract

Service oriented computing represents the convergence of technology with an understanding of cross-organizational business processes. A service license describes the terms and conditions for the use and access of the service in a machine interpretable way. Generally, a service provider defines individual services with corresponding service licenses which consumers have to follow. Often, service consumers are interested in selecting a service based on certain licensing terms and/or in composing individual services depending on their needs. Thus, consumer-specified licenses become pivotal in service composition as this allows consumers to make a preference on what their service licenses should be and whether they can compose certain services together in a composition satisfying their specified licensing terms. In this paper, we propose an approach allowing service consumers to specify service licensing terms and select services that match licenses and implement this approach within a semi-automated service composition framework. Furthermore, we present a directional matchmaking algorithm to compare a consumer-specified service license with provider-specified service licenses and produce a composite service license satisfying the consumer-specified license.

1 Introduction

Service Oriented Computing (SOC) currently emerges as a leading paradigm to build agile networks of collaborating distributed business applications. The seamless proliferation of SOC demands licensing related to the ownership and distribution aspects, to enable widespread use of services. 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. A service license should include all transactions between the licensor and the licensee, in which the licensor agrees to grant the licensee the right to use and access the service under predefined terms and conditions.

Generally, a service provider defines a license for the service and publishes the service with the license. Often, service consumers are interested in selecting a service with a particular type of license. For this purpose, a consumer specifies the desired license clauses which are formed as a *consumer-specified license*. By a *consumer-specified license*, in this paper, we refer to a set of license clauses specified by a consumer to be used in the selection of a license. The desired service of a consumer may be composed of several services. Currently existing composition tools integrate service discovery mechanisms in their composition algorithms and compose the candidate services based on functional specifications [4, 2]. When a consumer specifies the desired license clauses in addition to functional parameters, the composition approaches should support the proposal of a composite service license that should be compatible with the candidate service licenses being composed and with the consumer-specified license. Presently, to the best of our knowledge, there exists no works on service license composition. In this paper, we describe a novel approach to service license composition, integrated within a framework of semi-automated service composition. Service license composition supports the creation of a new license by combining service licenses which are compatible with other candidate licenses in the process of federation. The salient contributions of our paper are as follows.

- A novel approach to semi-automated service composition integrated with service licenses.
- A comprehensive algorithm towards composing service licenses together with directional compatibility analysis of service licenses.

The rest of paper is organized as follows. Section 2 discusses related work in this field and motivations to our approach. Section 3 describes a framework for semi-automated service composition integrated with service license composition approach. The process of service license composition based on the compatibility of candidate service licenses is proposed in Section 4. In Section 5, we provide details of a directional matchmaking algorithm. Section 6 presents the current prototype and experiments demonstrating service license composition followed by concluding remarks in Section 7.

2 Related Work and Motivations

Automated service composition is one of the active research areas in SOC, as a significant vision to developing dynamic composite applications integrated with service discovery mechanisms. Service discovery is the process of matching a service description of a requestor to the service descriptions of providers, with the objective of finding the most appropriate ones [8]. Service discovery involves matching of the description of a service against the description of a set of available resources and selection of services based on ranking (filtered by a set of criteria). The present researches focus on matching and finding of services through composition based on functional specifications [4, 2], but not on licenses.

A selection processes for commercial-off-the-shelf components using some of the non-technical features is addressed in [15], vaguely related to our work. An interesting approach for matching non-functional properties of Web services represented using WS-Policy is described in [16].

Service composition combines several independently developed services into a more complex service. Several researches are continually in progress addressing service composition with functional and non-functional aspects [3, 19]. The most comprehensive work on automated compatibility analysis of WSLA service level objectives is elaborated in [18]. 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 [18] cannot handle 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. However, no work considers discovery and composition of services with licenses in addition to functional specifications. Our work presented in this paper focuses on composing candidate service licenses and proposing a composite license for a composite service.

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 may include SLA terms. Thus, a service license is broader than the scope of SLA, protecting the rights of service providers and service consumers. Basically, a service license primarily focuses on the usage and provisioning terms of services. License clauses [17] are unexplored by currently available service description standards and languages. Current SLA and policies specification languages/standards for services (WSLA [9], SLANG [12], WSOL [13], WS-Agreement [1], WS-Policy [14]) define what to measure/monitor and describe payments/penalties. These specifications describe functional/nonfunctional properties and business/management information of services with varying levels of details.

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. We have formalized service license clauses and proposed a language ODRL-S for defining service licenses in machine interpretable way [5]. The anatomy of a service license includes clauses on *Subject*, *Scope of Rights*, *Financial Terms*, *Warranties, Indemnities, and Limitation of liabilities (WIL)*, and *Evolution* as detailed in [5]. The structure of a license in ODRL-S is as shown in Figure 1. A service license is a finite set of models (generally referred as licensing clauses), each of which further consists of a set of elements. Elements can be specified with value or without element value (having the element type only). Elements can have attributes (optional). Each attribute should have an attribute name and value for attribute.

Service license composition becomes important as services are composed with one another. In this case, the license of a composite service should be compatible with the licenses of services being composed. We have described a comprehensive algorithm for compatibility analysis of licenses (at license clause level) [7], as a pre-requisite for composition. The concept of service composition becomes complex when service licensing terms are specified by consumers in automated service composition. Our algorithm proposed in [7] does not consider the relationship between service consumer and service provider in the given two licenses, thus bypassing the directional issues of compatibility. The subsumption properties affect the directional compatibilities between consumer-specified licenses and

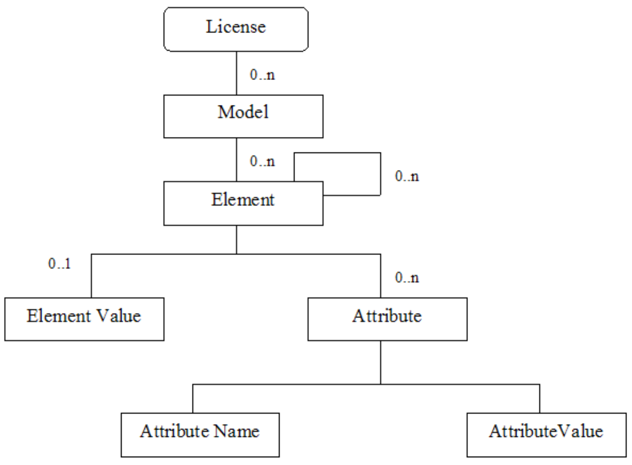

Figure 1. ODRL-S License Structure

provider-specified licenses. To handle consumer-specified licensing terms in service composition, in this paper, we propose a directional matchmaking algorithm. Furthermore, a semi-automated service composition approach is integrated with our service license composition algorithm.

3 An Approach to Semi-automated Service Composition Integrated with Licenses

Generally a service consumer looks up a service directory for services with specific functionality. The service discovery algorithm tries to find if any service advertisements given by providers matches the request of consumers and selects a service that matches with the specification of a consumer. There may be always a possibility of more than one services, offering similar functionality that differ in their licenses. Existing works do not support license-based service selection and composition.

Figure 2 depicts our framework that integrates service discovery and composition mechanism with licenses. The framework supports semi-automated service composition integrated with service licenses and consists of the following components.

The *User Interface* and *Web Services Client* supports consumers to specify functional parameters and license clauses based on which services would be selected and a license for composite services would be proposed. *Functional parameters* are inputs to the framework for selecting a service requested by consumers. The operations to be performed by a service are generally specified as functional parameters. *License clauses* are another set of inputs to the framework. Consumers specify a set of licensing clauses of their choice. For licensing clauses specified by a consumer, a service license is generated by the framework.

Functional parameters and license clauses will be handled by components of the *Service Management Framework* which manages service functionality selection, performs compatibility analysis and composition of service functionalities and licenses, and returns the result to consumers. In Figure 2, we outline the main components of the *Service Management Framework* that involves the service discovery and license composition.

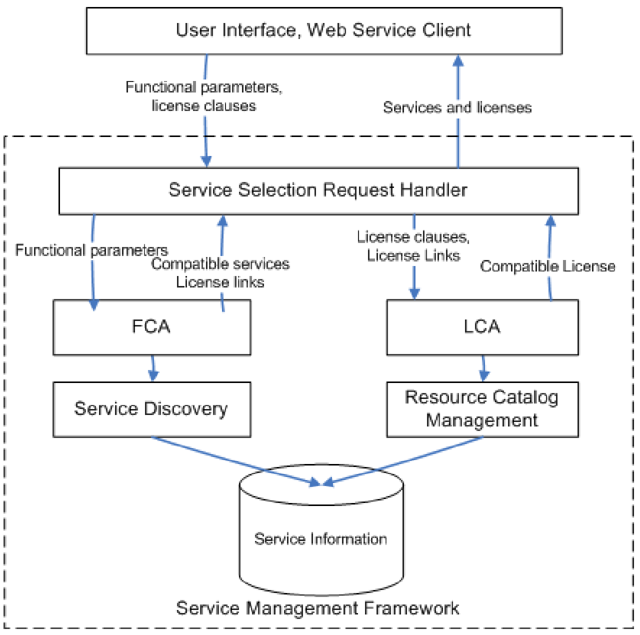
User Interface, Web Service Client
Functional parameters, license clauses
Services and licenses
Service Selection Request Handler
Functional parameters
Compatible services License links
License clauses, License Links
Compatible License
FCA
LCA
Service Discovery
Resource Catalog Management
Service Information
Service Management Framework

Figure 2. Semi-automated service composition Framework integrated with service licenses

- *Service Selection Request Handler*: receives functional parameters and license specifications from consumers and passes to FCA and LCA respectively.

- *Functional Compatibility Analyzer (FCA)*: selects services from the service directory that match with the request of consumers. Based on the inputs from consumer for the required service functionality which the consumer is searching for, FCA looks up in the *Service Directory* and returns a set of services satisfying functional specifications.

- *License Compatibility Analyzer (LCA)*: for the services returned by the *FCA*, *LCA* compares each of licenses with consumer-specified license and returns a set of compatible licenses. *LCA* is the focus of this paper.

- *Service Discovery*: discovers a set of services satisfying the required functional parameters.

- *Resource Catalog Management*: a component that manages various types of information associated with services, including service license description and rules for compatibility analysis, etc.

- *Service Information*: an XML-based repository where information associated with services are stored.

In our framework, service information is described by using Web Services Resource Catalog [10]. Service information contains description about functionalities of services and information about license. A service license can be expressed by ODRL-S [5]. The *FCA* and *Service Discovery* components are based on work presented in [11]. However, they can be replaced by any existing service discovery tools. This paper primarily focuses on service license composition approach and directional compatibility analysis of consumer-specified licenses and provider-specified licenses , which are implemented in the LCA. Note that in our framework, the execution order of LCA and FCA can be customized, depending specific composition strategy. This allows clients of the framework to combine LCA and FCA for different purposes.

4 Service Licenses Composition with Consumer-specified License Clauses

Let $F = \{f_1, f_2, \cdots, f_n\}$ denote a set of functional parameters. Functional parameters, specified by consumers, represent the requested operations performed by services. For each f_i, we assume that there exists a category of services, t_i, that offers the funtionality specified by f_i. Let $T = \{t_1, t_2, \cdots, t_n\}$ denote categories of services associated to F, where t_i provides the functionality required by f_i. Given a $t_i \in T$, there exist many services belonging to this service type, each offering the functionality f_i but with possibly different implementations and associated licenses. We denote this set of services with $S(t_i) = \{s_1, s_2, \cdots, s_m\}$. From this set, a service $s_{t_i} \in S(t_i)$ is selected for the service composition, given f_i.

Let $L(T) = \{l(s_{t_1}), l(s_{t_2}), \cdots, l(s_{t_n})\}$ be the set of licenses in which $l(s_{t_i})$ indicates the license associated with the service s_{t_i}. Our objective is to compose licenses in $L(T)$, associating functionalities f_i, so that the resulting composite license $l_{composite}$ satisfies the licensing clauses of a consumer-specified license lc[1].

Service license composition algorithm is listed in Algorithm 1. The input provided by the consumer to the *Service Selection Request Handler* are a set of functionalities F and the requested license clauses lc. Each functionality f_i is represented by a category of service t_i and each category of service can have a set of services varying in licenses, $S(t_i)$.

In lines 3-4, a service matching the functionality specified by a consumer is retrieved by *FCA* searching in the *Service Information* repository through the *Service Directory* component. Similarly, *LCA* searches in the *Service Information* for the information about licenses of each service being selected by *FCA*. The license associated with the selected service is compared with the lc as in line 7. The compatibility of these two given licenses can be analyzed by our work [7]. This compatibility analysis is performed for each of the selected service by *FCA* and grouped as $L(T)$ (in lines 8-10). The algorithm gets terminated if a compatibility is not found for a service type (in line 13). The composition cannot be performed even if any of $l(st_i)$ does not exist.

Each of these selected licenses are compared for compatibility (in lines 15-21) with other licenses [7]. The algorithm is terminated if incompatibility arises between any two licenses (in lines 17-19).

Algorithm 1 Service License Composition

```
 1: $S(t_i) = \phi$
 2: $L(T) = \phi$
 3: **for all** $t_i \in T$ **do**
 4:     ask $FCA$ for $S(t_i)$
 5:     $s_{t_i} = \phi$
 6:     **for all** $s_j \in S(t_i)$ **do**
 7:         **if** $Compatible(l(s_j), lc)$ **then**
 8:             $s_{t_i} = s_j$
 9:             $l(s_{t_i}) = l(s_j)$
10:             break
11:         **end if**
12:     **end for**
13:     Terminate if $s_{t_i} = \phi$
14: **end for**
15: **for all** $(l_x, l_y) \in L(T)$ **do**
16:     **if** $l_x \neq l_y$ **then**
17:         **if** $\neg Compatible(l_x, l_y)$ **then**
18:             Terminate
19:         **end if**
20:     **end if**
21: **end for**
22: $l_{generated} \leftarrow Compose(L(T))$
23: **if** $CLCompatible(l_{generated}, lc)$ **then**
24:     $l_{composite} \leftarrow Compose(l_{generated}, lc)$
25: **end if**
```

These licenses in $L(T)$ are composed as follows (in line 22).

- Extract elements of each license and put them together in a single license ($l_{generated}$).

[1] It is always possible that the composition of two service licenses can result in a set of new composite licenses. It is up to choice of the consumer who wants the appropriate/desired license. In case of fully automated composition process, the selection of new composite license can be facilitated by some ranking approaches. In our algorithm, we bypass the concept of multiple licenses and assume as single composite license.

- Remove redundant clauses.

$l_{generated}$ is composed with lc to generate $l_{composite}$. The purpose of this composition (in line 24) is to include a set of licensing clauses that may not be in the composite license but specified by the consumer. Then, $l_{composite}$ is compared with lc for compatibility (in line 23). If compatible, $l_{composite}$ will be the required composite license (in line 24) which is composed from the given service licenses and also compatible with the consumer-specified licensing clauses.

$Compatible(l_u, l_v)$ is an algorithm for analyzing the compatibility between any two given licenses, that considers subsumption issues and does not consider conditions as specified in [7]. $CLCompatible(l_u, l_v)$ is an algorithm for analyzing the compatibility between a consumer specified license and a provider specified service license (described in Section 5). $Compose(l_u, l_v)$ is an algorithm that extracts the elements from the given licenses and builds a new license eliminating redundant licensing clauses.

5. Directional Matchmaking Algorithm

For a service with consumer specified lc, the function ($CLCompatible(l_u, l_v)$) looks up licenses of extracted services $L(T)$ and finds a corresponding $l(s_{t_i})$, comparing with lc on one-by-one basis. This function is explicated by directional matchmaking algorithm (DMA) as follows.

There could be a scenario when analyzing the compatibility of service licenses where one of the licenses contains clauses that the other license does not[2]. In certain cases, the absence of one or several of these clauses does not affect the compatibility with the other license.

Based on the subsumption property, a set of rules is proposed in [7] to check the compatibility of two licenses in the element level[3]. For example, a provider specified license contains a clause, say *composition*. A consumer can specify *adaptation*. Based on [7], these licenses are incompatible, because the direction of interactions between the given licenses is not considered in that algorithm and we get a result of compatibility between these terms (as a result of subsumption). A consumer may not be interested in a license allowing *composition* as the consumer desires only *adaptation*. Similarly, we can see if a consumer asks for *composition*, based on subsumption, *derivation* becomes compatible. As derivation requires the service to free/open [6], the consumer need not wish this. Thus, in these cases, it is evident that the direction of interactions between the licenses in composition becomes significant.

We introduce the directional aspect of compatibility in composition of service licenses as follows. Table 1 lists rules used by the DMA to determine the compatibility between consumer-specified license clauses against unspecified *Scope of Rights* and *Financial Terms* elements in provider-specified license clauses. Table 2 lists rules used by the DMA to find the compatibility between unspecified *Scope of Rights* and *Financial Terms* elements in consumer-specified license clauses against provider-specified license clauses.

The DMA for matchmaking between a consumer-specified license (subscript c in the following definitions) with a provider-specified license (subscript p) is given in Algorithm 2.

[2]The general approach for handling unspecified elements in license compatibility analysis is through "conservative" approach i.e. unspecification equals denial of compatibility.

[3]Subsumption implies a match that should occur, if the given license element is more permissive (accepts more) than the corresponding element in the other license.

6 Experiments

6.1 Prototype Implementation

Our current prototype implementation uses the Web Services Resource Catalog (WSRC) to persist license and service related information. We have implemented a web based front end that supports consumers in defining functional parameters and licenses. We use a vector based search engine [11] for the discovery of Web services based on functional descriptions.

The current prototype supports a keyword based search for web services.Note that our approach considers the functional discovery part of our architecture as black box that can be replaced by other search engines (see Figure 2). semantic discovery techniques, etc.

In order to map licenses to web services, we use WSRC. We map interface descriptions and license information into a single WSRC element. Every WSRC element contains references to external documents that represent the interface descriptions and the corresponding licenses (see Figure 3). Lines 2 - 12 define the root of our web service entry and contain references to child entries (lines 6 - 8, 9 - 11 respectively) that represent the license (lines 13 - 24) and the interface description (lines 25 - 36). We have implemented compatibility algorithm in Java and currently we are integrating it with functional selection of services.

6.2 Scenario on Consumer-specified License Selection and Composition

In order to demonstrate our approach, we examine an illustrating scenario in which the consumer wants to create a restaurant service (by composition of various services) that offers information on restaurant location and opening hours,

Table 1. Compatibility between consumer-specified license clauses against unspecified *Scope of Rights* and *Financial Terms* elements in provider-specified license clauses

Consumer-specified license element	**Compatibility**	**Rationale**
adaptation	*Incompatible*	A consumer-specified license requiring `adaptation` cannot be compatible with a license denying `adaptation`. Even if a provider-specified license may allow broader *Scope of Rights* (composition or derivation) than the requirement by the consumer, it is considered as incompatible.
composition	*Incompatible*	A consumer-specified license requiring `composition` cannot be compatible with a license denying `composition` (as unspecified) or even allowing only `adaptation`.
derivation	*Incompatible*	Derivation requires a service to be ‘Free’/‘Open’ [6]. Even, a license allowing `adaptation` or `composition` can not be compatible.
attribution	*Compatible*	The requirement by a consumer for specification of `attribution` will not affect compatibility when unspecified in the provider’s license.
sharealike	*Compatible*	The clause `sharealike` affects the composite license requiring that the composite license should be similar to the license having `sharealike` element.
noncommercialuse	*Compatible*	Commercial use of services is denied by the clause `noncommercialuse`. However, if a consumer wishes its service license to be a non-commercial, this would not affect the compatibility with other services whose licenses may allow commercial use.
payment	*Compatible*	A consumer can charge for its service even it may compose a cost-free service.

Table 2. Compatibility between unspecified *Scope of Rights* and *Financial Terms* elements in consumer-specified license clauses against provider-specified license clauses

Provider-specified license element	**Compatibility**	**Rationale**
adaptation	*Incompatible*	As `adaptation` is the right for interface reuse, a consumer may deny. In this case, the specification of `adaptation` in a provider’s license is not compatible when it is unspecified in a consumer’s license.
composition	*Incompatible*	If a consumer denies `composition` by unspecifying `composition` clause, this cannot be considered as don’t care although `composition` supports subsumption.
derivation	*Incompatible*	Though, in general, `derivation` supports subsumption [7] over `composition` or `adaptation`, the specification by a consumer overrides the subsumption property.
attribution	*Compatible*	The specification of `attribution` in a provider’s license will not affect compatibility when unspecified in a consumer’s license.
sharealike	*Incompatible*	If a provider-specified license requires for `sharealike` clause, a consumer may not even want its service to be licensed under the same terms of the provider-specified license.
noncommercialuse	*Incompatible*	Commercial use of a service is denied by the clause `noncommercialuse`.
payment	*Compatible*	The clause `payment` does not affect compatibility directly, if unspecified by a consumer because the license elements related to payment and charging are dependent on service provisioning issues.

Algorithm 2 Directional Matchmaking Algorithm (DMA)

```
 1: for all $(m_c, m_p)$ and $(modelname(m_c) = modelname(m_p))$ do
 2:   for all $e_c \in m_p$ do
 3:     for all $e_p \in m_p$ do
 4:       bool res = $ElementCompatibility(e_c, e_p)$
 5:       if $(\neg res)$ then
 6:         Terminate
 7:       end if
 8:     end for
 9:   end for
10:   $m_c$ and $m_p$ compatible
11: end for
```

procedure boolean $ElementCompatibility(e_c, e_p)$

```
 1: if $(type(e_c) = type(e_p)) \wedge (value(e_c)$ = null$) \wedge (value(e_p)$ = null ) then
 2:   return TRUE
 3: end if
 4: if $(type(e_c) = type(e_p)) \wedge (value(e_c) = value(e_p)) \wedge (attributename(e_c)$ = null$) \wedge (attributename(e_p)$ = null$)$ then
 5:   return TRUE
 6: end if
 7: if $((type(e_c) = type(e_p)) \wedge (value(e_c) = value(e_p)) \wedge AttributeCompatibility(e_c, e_p) = TRUE)$ then
 8:   return TRUE
 9: end if
10: if $((type(e_c) = adapatation) \vee (type(e_c) = composition) \vee (type(e_c) = derivation) \wedge (type(e_p)$ = unspecified$)$ ) then
11:   return FALSE
12: end if
13: if $((type(e_c) = attribution) \vee (type(e_c) = noncommercialuse) \vee (type(e_c) = sharealike) \vee (type(e_c) = payment) \wedge (type(e_p)$ = unspecified$)$ ) then
14:   return TRUE
15: end if
16: if $((type(e_p) = adapatation) \vee (type(e_p) = composition) \vee (type(e_p) = derivation) \vee (type(e_p) = sharalike) \vee (type(e_p) = noncommercialuse) \wedge (type(e_c)$ = unspecified$)$ ) then
17:   return FALSE
18: end if
19: if $((type(e_p) = attribution) \vee (type(e_p) = payment) \wedge (type(e_c)$ = unspecified$)$ ) then
20:   return TRUE
21: end if
```

procedure boolean $AttributeCompatibility(a_c, a_p)$

```
1: if $(attributename(a_c) = attributename(a_p))$ then
2:   if $(value(a_c) = value(a_p))$ then
3:     $a_c$ and $a_p$ compatible
4:     return TRUE
5:   end if
6: end if
```

```
<Catalog>
  <Entry Id="http://vienna.vitalab.tuwien.ac.at/MenuService">
    <Resource>
      <Reference>...</Reference>
    </Resource>
    <EntryRef Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
      <EntryId>http://vitalab.tuwien.ac.at/MenuServiceLicense</EntryId>
    </EntryRef>
    <EntryRef Role="http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/roles/child">
       <EntryId>http://vitalab.tuwien.ac.at/MenuServiceInterface</EntryId>
     </EntryRef>
  </Entry>
  <Entry Id="http://paris.vitalab.tuwien.ac.at/MenuServiceLicense">
    <Classifier>http://odrl.net/1.1/ODRL-EX-11.xsd</Classifier>
    <Annotation lang="en">License</Annotation>
    <Resource>
      <ResourceRef>
        <ResourceElement LocalName="License"
         Namespace="http://vitalab.tuwien.ac.at/License.xsd"/>
        <Reference>
          <URI>http://paris.vitalab.tuwien.ac.at/MenuService.xml</URI>
        </Reference>
      </ResourceRef>
     </Resource>
  </Entry>
  <Entry Id="http://paris.vitalab.tuwien.ac.at/MenuServiceInterface">
    <Classifier>http://schemas.xmlsoap.org/wsdl/</Classifier>
    <Annotation lang="en">WSDL</Annotation>
    <Resource>
      <ResourceRef>
        <ResourceElement LocalName="ServiceInterface"
       Namespace="http://vitalab.tuwien.ac.at/MenuService.xsd"/>
          <Reference>
            <URI>http://paris.vitalab.tuwien.ac.at/MenuService.wsdl</URI>
          </Reference>
        </ResourceRef>
      </Resource>
  </Entry>
</Catalog>
```

Figure 3. Web Service Resource Catalog fragment for `MenuService`

catalog of specialty cuisines, and online table reservation. In doing so, the consumer will search for existing services that offer these individual operations, and will compose the restaurant service. Examples of possible individual services are a map service named `LocationService` displaying the location and a service facilitating intermediate table reservation named `ReservationService`. In addition to this, the consumer may also specify the licensing clauses that he/she prefers for the composite service. Our proposed framework checks the compatibility between licenses of individual services and consumer-specified license, and if compatible, proposes a new license for the composite restaurant service.

In our experiment, the license of `LocationService` allows `composition` and requires `attribution` when `LocationService` is used by other services. The license of `LocationService` in ODRL-S [5] is described in Figure 4.

The license of `ReservationService` allows the access to source code of service realization and requires `attribution` when `ReservationService` is used by other services. Furthermore, `ReservationService` requires a fee of 1 Euro per use. The license of `ReservationService` is described in Figure 5.

Based on the above-mentioned two services, we composed a restaurant service `MenuService` providing the

```
<!-- Namespace declarations go here-->
  <o-ex:offer>
    <o-ex:asset>
        <o-ex:context>
           <o-dd:uid>
           ..........
           </o-dd:uid>
        </o-ex:context>
    </o-ex:asset>
    <o-ex:permission>
       <sl:composition/>
    </o-ex:permission>
    <o-ex:requirement>
       <o-dd:attribution/>
    </o-ex:requirement>
  </o-ex:offer>
```

Figure 4. Illustrating license for `LocationService`

```
<!-- Namespace declarations go here-->
  <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>
```

Figure 5. Illustrating license for `ReservationService`

following operations (and parameters).

- `Location(Address address, OpeningHours openhrs)`: provides information about address and opening hours (where `Address` and `OpeningHours` are complex data types).
- `Reservation(int Seat, String Name, String ReservedTable)`: provides facility for reserving table.

`Location` uses `LocationService` for providing the location information and `Reservation` is derived from the service `ReservationService`. Thus, the service `MenuService` composes `LocationService` and `ReservationService`. As a result, `MenuService` has to be associated with a license compatible with the licenses of the services being composed.

The consumer-specified license clauses for `MenuService` are as follows: (i) the service should allow composition and (ii) the service is sharealiked indicating that the service expects another services being composed/ derived to reflect the same terms and conditions of this service. Based on these clauses, a license (specified as lc in Algorithm 1) is generated as shown in Figure 6.

The consumer-specified license is compared against each service licenses being composed in the composite service using [7]. In this scenario, the license of `LocationService` is compatible with the consumer-specified license. Similarly, the license of `ReservationService` is also compatible with the consumer-specified license. Furthermore, the license of `LocationService` and the license of `ReservationService` are analyzed for compatibility using [7] and found compatible. Then, licenses of services `LocationService` and `ReservationService` are composed and the newly generated license is analyzed for compatibility with the consumer-specified license. Thus, a composite license is generated from the licenses being composed and also with the specifications by consumer. A composite license for `MenuService` is presented in Figure 7.

7 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. 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. In this paper, we have analyzed the compatibility of licenses offered by service providers with consumer specified licenses. Following this, we have presented an approach to generate composite service license based on specifications by consumers.

The presented algorithm has following limitations: First, it selects a service for each service category, provided its license is compatible (3-14). Then, it verifies if the licenses of these services are compatible between themselves (15-21). The present approach may result in the incompatibility among the set of licenses for the candidate services chosen from each service type. The performance analysis of the proposed composition algorithm 1 is as follows. Let m be the maximum number of services returned for a service functionality f by FCA. Let n be the number of

```
<!-- Namespace declarations go here-->
  <o-ex:offer>
     <o-ex:permission>
        <sl:composition/>
     </o-ex:permission>
     <o-ex:requirement>
        <o-cc:sharealike>
     </o-ex:requirement>
  </o-ex:offer>
```

Figure 6. Consumer-specified license

```
<!-- Namespace declarations go here-->
  <o-ex:offer>
    <o-ex:permission>
       <sl:composition/>
    </o-ex:permission>
   <o-ex:requirement>
      <o-cc:attribution/>
   </o-ex:requirement>
   <o-ex:requirement>
      <o-cc:sharealike/>
    </o-ex:requirement>
  </o-ex:offer>
```

Figure 7. Resulting composite license

service functionality. At any point, the algorithm has to find a maximum of m^n combinations to retrieve a set of services satisfying consumer-specified functionalities. Furthermore, the concepts of SLA are not considered in the given DMA. In our future work, we are planning to fully integrate consumer-specified composite service license in the proposed service selection and management framework. Furthermore, we intend to work on monitoring and enforcing license clauses.

References

[1] 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.

[2] L. Aversano, G. Canfora, and A. Ciampi. An Algorithm for Web service Discovery through their Composition. In *Proceedings of the IEEE International Conference on Web Services*, 2004.

[3] D. Berardi, D. Calvanese, G. D. Giacomo, R. Hull, and M. Mecella. Automatic Composition of Transition Based Semantic Web Services with Messaging. In *Proc. of the 31st VLDB Conference*, 2005.

[4] A. Brogi and S. Corfini. Behaviour-aware Discovery of Web service Compositions. *International Journal of Web Services Research*, 4(3), 2007.

[5] G. R. Gangadharan, V. D'Andrea, R. Iannella, and M. Weiss. ODRL Service Licensing Profile (ODRL-S). In *Proc. of the 5th Intl. Workshop for Technical, Economic, and Legal Aspects of Business Models for Virtual Goods*, 2007.

[6] G. R. Gangadharan, V. D'Andrea, and M. Weiss. Free/Open Services: Conceptualization, Classification, and Commercialization. In *Proceedings of the Third IFIP International Conference on Open Source Systems (OSS)*, 2007.

[7] G. R. Gangadharan, M. Weiss, V. D'Andrea, and R. Iannella. Service License Composition and Compatibility Analysis. In *Proceedings of the International Conference on Service Oriented Computing (ICSOC'07)*, 2007.

[8] S. Grimm, B. Motik, and C. Preist. Variance in e-Business Service Discovery. In *Proceedings of the Semantic Web Services Workshop at ISWC'04*, 2004.

[9] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. IBM Coporation, 2003.

[10] A. Nosov, A. Hately, B. Reistad, B. Murray, D. Davis, H. Kreger, P. Niblett, R. McCollum, V. Tewari, V. Kumbalimutt, and W. Vambenepe. Web Services Resource Catalog (WS-RC). http://schemas.xmlsoap.org/ws/2007/05/resourceCatalog/, 2007.

[11] C. Platzer and S. Dustdar. A Vector Space Search Engine for Web Services. In *Proceedings of the IEEE European Conference on Web services (ECOWS)*, 2005.

[12] J. Skene, D. Lamanna, and W. Emmerich. Precise Service Level Agreements. In *Proc. of 26th Intl. Conference on Software Engineering (ICSE)*, 2004.

[13] V. Tosic, B. Pagurek, K. Patel, B. Esfandiari, and W. Ma. Management Applications of the Web Service Offerings Language. In *Proc. of the 15th Intl. Conf. on Advanced Information Systems Engineering (CAiSE)*, 2003.

[14] 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.

[15] 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)*, pages 21 – 32, 2007.

[16] K. Verma, R. Akkiraj, and R. Goodwin. Semantic Matching of Web Service Policies. In *Second Intl. Workshop on Semantic and Dynamic Web Processes*, 2005.

[17] World Intellectual Property Organization. WIPO Copyright Treaty (WCT). http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html, 1996.

[18] W. Yang, H. Ludwig, and A. Dan. Compatibility Analysis of WSLA Service Level Objectives. Technical Report RC22800 (W0305-082), IBM Research Division, 2003.

[19] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng. Quality Driven Web Services Composition. In *Proceedings of the WWW Conference*, 2003.