# Service License Composition and Compatibility Analysis

G.R. Gangadharan[1], Michael Weiss[2], Vincenzo D'Andrea[1], and Renato Iannella[3]

[1] Department of Information and Communication Technology
University of Trento
Via Sommarive, 14, Trento, 38100 Italy
{gr,dandrea}@dit.unitn.it
[2] School of Computer Science, Carleton University
1125 Colonel By Drive, Ottawa, K1S 5B6, Canada
weiss@scs.carleton.ca
[3] National ICT Australia
Level 19, 300 Adelaide Street, Brisbane, Queensland, 4000 Australia
renato@nicta.com.au

**Abstract.** 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.

## 1 Introduction

Service oriented computing (SOC) represents the convergence of technology with an understanding of cross-organizational business processes [1]. Services enhance the World Wide Web not only for human use, but also for machine use by enabling application level interactions. Services have an important advance over stand-alone 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 [2].

B. Krämer, K.-J. Lin, and P. Narasimhan (Eds.): ICSOC 2007, LNCS 4749, pp. 257–269, 2007.

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 [3] 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 [4]. Different types of licenses exist for software. As the nature of services differs significantly from traditional software and components, services prevent 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. We have formalized the license clauses for services in [5].

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 Section 2, we briefly describe the representation of service licenses using ODRL Service Licensing Profile. Section 3 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 Section 4. Section 5 discusses related work in this field, showing the distinct contribution of this paper.

## 2 ODRL Service Licensing Profile (ODRL-S)

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. Licensing of services raises several issues, including:

1. What rights should be associated with services and how should the rights be expressed?
2. How can the composite service license be generated being compatible with the licenses in composition?

We have developed a language ODRL-S [6] by extending the Open Digital Rights Language (ODRL) [7] to implement the clauses of service licensing (see

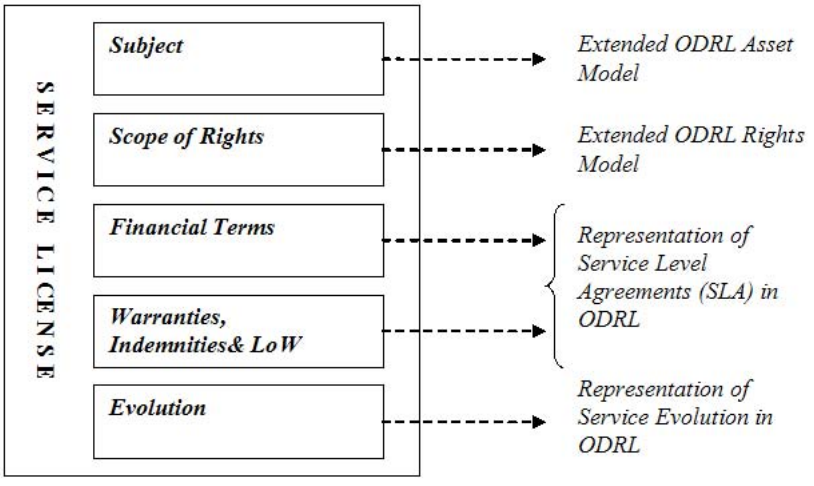


**Fig. 1.** Conceptual mapping of service license clauses

Figure 1). The complete syntax and semantics of ODRL-S can be found in [6]. 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 [7]. The subject of the license relates to the definition of the service being licensed. This defines some related information about the service and may include a unique identification code for the service, service name, service location, and other relevant information.
- The *Scope of Rights*, as explained in detail in [5], comprise the extended ODRL Permission, Requirement, and Constraint Models. Composition is the right of execution with the right of interface modification. Derivation is the right of allowing modifications to the interface as well as the implementation of a service. Furthermore, derivation requires independent execution of the service where composition is dependent on the execution of services being composed. Adaptation refers to the right of allowing the use of interface only (independent on the execution of services). ODRL-S reuses the concept of sharealike and non-commercial use from the ODRL Creative Commons profile [8]. Attribution to services is facilitated by the ODRL attribution element.
- We adopt the ODRL payment model for representing the *Financial* model of services. However, Free/Open Services [9] could be represented without a payment model.
- The *WIL* model defines warranties, indemnities and limitation of liabilities associated with services.
- The *Evolution* model specifies modifications in future releases or versions.

## 3 Service Licenses Matchmaking and Compatibility Analysis

A license $L(S)$ in ODRL-S for a service $S$ is 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 (having the element type only). Elements can have subelements (referred as subentity in ODRL). Elements can also be nested within other elements. Elements are not specified with attributes. Each subelement is specified with an attribute, a value for attribute, and a value for subelement. The structure of a license clause is modelled in ODRL-S as shown in Figure 2. An example of a service license ($L_1$) is shown in Figure 3.

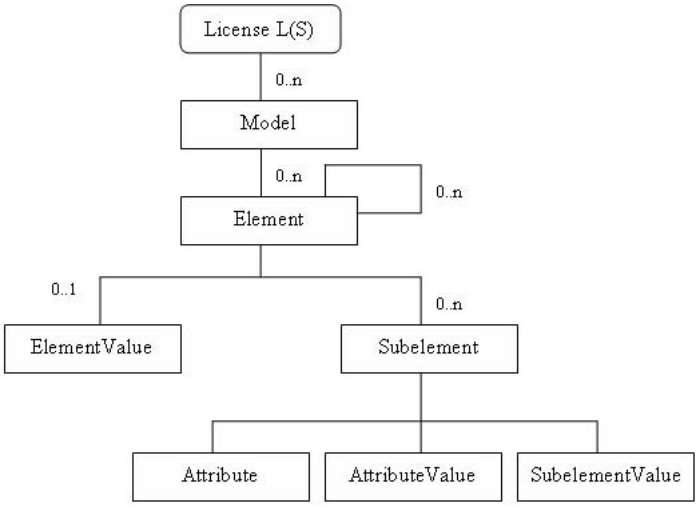


**Fig. 2.** ODRL-S license clause structure

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 [10]. 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. The subsumption rules for *Scope of Rights* are given below (see Table 1):

**Table 1.** Subsumption rules over *Scope of Rights* elements

| **Element**1 | **Element**2 | **Comparison** | **Redefinition** |
|---|---|---|---|
| Composition | Adaptation | Composition ⊃ Adaptation | Composition |
| Derivation | Adaptation<br>Composition | Derivation ⊃ Adaptation<br>Derivation ⊃ Composition | Derivation<br>Derivation |

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

**Table 2.** Compatibility with unspecified *Scope of Rights* and *Financial Terms* elements

| **Element**1 | **Element**2 | **Compatibility** | **Rationale** |
|---|---|---|---|
| Adaptation | Unspecified | *Compatible* | Adaptation is the right for interface reuse only. |
| Composition | Unspecified | *Incompatible* | A license denying composition can not 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'. |
| 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. |
| Non-CommercialUse | Unspecified | *Incompatible* | Commercial use is denied by Non-CommercialUse. |
| 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. |

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. The algorithm performs the compatibility analysis between any two given licenses[1] to decide whether they are compatible. A license is compatible with another license if all license clauses are compatible (as defined by the matchmaking algorithm). Service licenses can be combined, if they are found compatible by the matchmaking algorithm, allowing the corresponding services to be composed.

Assuming that semantics inside a license are agreed by service providers and consumers, the algorithm for matching a license $SL_\alpha$ (with subscript $\alpha$) with another license $SL_\beta$ (with subscript $\beta$) is given as follows. In the following, we use the symbol $\Leftrightarrow$ to denote compatibility. Two licenses are compatible (that is: $SL_\alpha \Leftrightarrow SL_\beta$), if all the respective models in both the licenses are compatible.

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

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

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

Now, an element $e_\alpha$ is compatible with another element $e_\beta$, if:

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

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

A subelement $s_\alpha$ is compatible with another subelement $s_\beta$, if the values of subelements are equal and if their attributes are of same type (represented by $\equiv$) and the associated values of attributes are equal.

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

$$
\begin{aligned}
& & & (Value(Se_\alpha) = Value(Se_\beta)) \\
\wedge \quad & (\forall h_\alpha : h_\alpha \epsilon Attributes(Se_\alpha) & \exists h_\beta : h_\beta \epsilon Attributes(Se_\beta) \\
& \Rightarrow & (h_\alpha \equiv h_\beta) \wedge (Value(h_\alpha) = Value(h_\beta)) \\
\wedge \quad & (\forall h_\beta : h_\beta \epsilon Attributes(Se_\beta) & \exists h_\alpha : h_\alpha \epsilon Attributes(Se_\alpha) \\
& \Rightarrow & (h_\alpha \equiv h_\beta) \wedge (Value(h_\alpha) = Value(h_\beta))
\end{aligned}
$$

Consider the 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 (say $L1$) for the service $I$ is represented in ODRL-S as follows (shown in Figure 3).

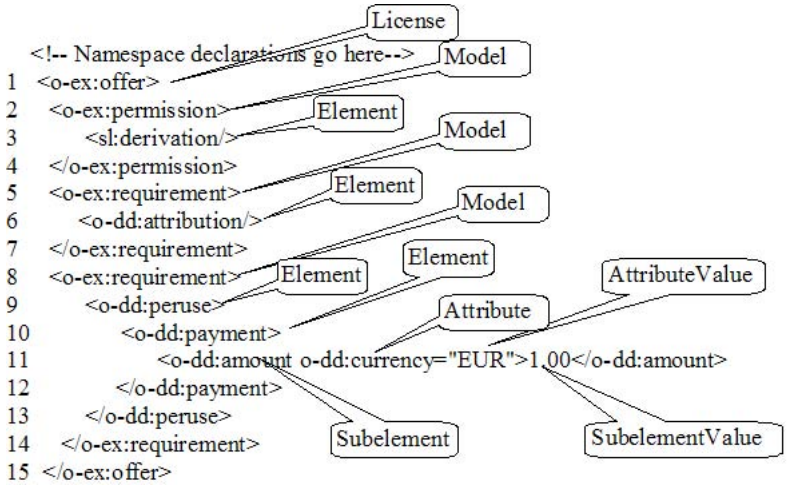


**Fig. 3.** Example license $L_1$

Assume that the map service $M$ allows composition and requires attribution. However, this service denies commercial use. The license (say $L2$) for the service $M$ is represented in ODRL-S as follows (shown in Figure 4).

Assume we now want to analyze the compatibility between license $L_1$ and another license $L_2$.

Following the matchmaking algorithm, we compare licenses at the model level. Line 2 of both licenses are `<o-ex:permission>` models. The elements in line 3 of $L_1$ (`<ls:derivation>`) and line 3 of $L_2$ (`<ls: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).

In lines 5, 6, and 7 of $L_1$ and $L_2$, 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.

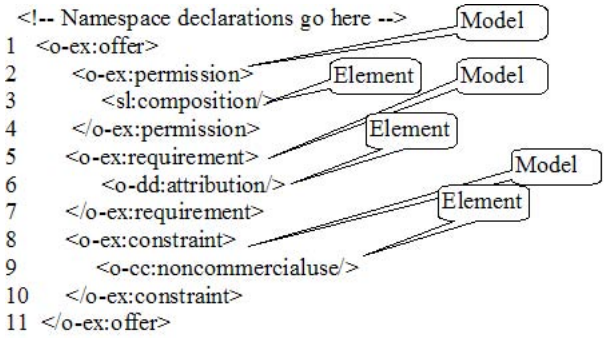


**Fig. 4.** Example License $L_2$

Then, in line 8 of $L_1$, 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 $L_2$. (The service offered by $L_2$ can be made available free of charge, without specifying the payment model.)

The `<o-ex:constraint>` model of license $L_2$ (in lines 8 and 9) specifies the element `<o-cc:noncommercialuse>`. When the algorithm looks for the element `<o-cc:noncommercialuse>` in $L_1$, the algorithm is unable to find as the element is unspecified. This indicates that the service with $L_1$ can be used for commercial purposes. From Table 2, the algorithm finds that these clauses are incompatible, and thus the licenses become incompatible.

## 4 Service License Composition

Service composition combines of independently developed services into a more complex service. 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 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 licenses for the composite service. The process of a license selection for a service is depicted in Figure 5. The service consumer/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 most permissible).

Consider our example of a restaurant service $R$, composed of a map service and a resource allocation service. The search for a map service in the service directory might return several services with the same functionality, but different licenses, say $M$, $M'$, and $M''$. The search for resource allocation results in services

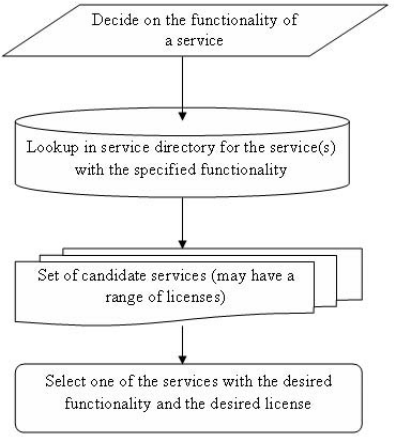


**Fig. 5.** Process of a service license selection with the service functionality

with different licenses, $I$, $I'$, and $I''$. The composite service $R$ could be licensed under a variety of licenses, but then must be compatible with the licenses of $M$ or $M'$ or $M''$ and $I$ or $I'$ or $I''$ (see Figure 6). Each combination could lead to the creation of a distinct license for $R$.

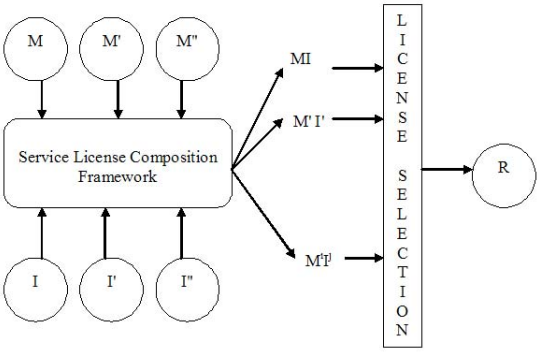


**Fig. 6.** Service license composition scenario by license generation and selection

Consider the case where $M$ allows composition and requires attribution, when $M$ is used by other services. The license of $M$ in ODRL-S may look like this:

```
  <!-- Namespace declarations go here-->
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 $I$ allows access to the source code of the service (derivation) and requires a fee of 1 Euro per use and thus license of $I$ is same as the license shown in Figure 3.

As the licenses $M$ and $I$ are compatible using the matchmaking algorithm illustrated in previous section, they can be composed. The composition of these service licenses could generate a set of licenses that $R$ may select. Assume that $R$ has the following license (one of the licenses in the set of compatible licenses), compatible with the licenses of $M$ and $I$:

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

The composition of candidate service licenses requires to be compatible among themselves. Furthermore, for composition, each of the candidate service licenses should also be compatible with the resulting composite service license. Following the matchmaking algorithm, it is possible to demonstrate the compatibility of the composite license ($R$) with each of the candidate licenses ($M$ and $I$). Space, however, does not allow us to show the details of executing the matchmaking algorithm for the example.

## 5 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 software industry, there is no active involvement

in this topic by 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 [11]. As 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 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 [12].

Current SLA and policies specifications for services (WSLA [13], SLANG [14], WSOL [15], WS-Agreement [16], WS-Policy [17]) 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 [18] 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 [6].

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 [19,20]. A selection processes for commercial-off-the-shelf components using some of the non-technical features is addressed in [21], vaguely related to our work. An interesting approach for matching non-functional properties of Web services represented using WS-Policy is described in [22]. The most comprehensive work on automated compatibility analysis of WSLA service level objectives is elaborated in [23].

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 [23] can not 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 [24]. 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.

## 6 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. Further, we have 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. Paulson, L.: Services Science: A New Field for Today's Economy. IEEE Computer 39(8), 18–21 (2006)
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services Concepts, Architectures, and Applications. Springer, Heidelberg (2004)
3. Classen, W.: Fundamentals of Software Licensing. IDEA: The Journal of Law and Technology 37(1) (1996)
4. D'Andrea, V., Gangadharan, G.R.: Licensing Services: The Rising. In: ICIW'06. Proceedings of the IEEE Web Services Based Systems and Applications, Guadeloupe, French Caribbean, pp. 142–147. IEEE Computer Society Press, Los Alamitos (2006)
5. Gangadharan, G.R., D'Andrea, V.: Licensing Services: Formal Analysis and Implementation. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 365–377. Springer, Heidelberg (2006)

6. Gangadharan, G.R., D'Andrea, V., Iannella, R., Weiss, M.: ODRL Service Licensing Profile. Technical Report DIT-07-027, University of Trento (2007)
7. Iannella, R. (ed.): Open Digital Rights Language (ODRL) Version 1.1 (2002), http://odrl.net/1.1/ODRL-11.pdf
8. Iannella, R. (ed.): ODRL Creative Commons Profile (2005), http://odrl.net/Profiles/CC/SPEC.html
9. Gangadharan, G.R., D'Andrea, V., Weiss, M.: Free/Open Services: Conceptualization, Classification, and Commercialization. In: OSS. Proceedings of the Third IFIP International Conference on Open Source Systems, Limerick, Ireland (2007)
10. Jezequel, J.M., Train, M., Mingins, C.: Design Patterns and Contracts. Addison-Wesley, Reading (1999)
11. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F., Kramer, B.: Service Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings 05462 (SOC) (2006)
12. Gangadharan, G.R., Frankova, G., D'Andrea, V.: Service License Life Cycle. In: CTS 2007. Proceedings of the International Symposium on Collaborative Technologies and Systems, pp. 150–158 (2007)
13. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. IBM Coporation (2003)
14. Skene, J., Lamanna, D., Emmerich, W.: Precise Service Level Agreements. In: ICSE. Proc. of 26th Intl. Conference on Software Engineering (2004)
15. Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., Ma, W.: Management Applications of the Web Service Offerings Language. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, Springer, Heidelberg (2003)
16. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) Version 2005/09 (2005), http://www.gridforum.org
17. Vedamuthu, A., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T., Yalcinalp, U.: Web Services Policy (WS-Policy) Framework (2007), http://www.w3.org/TR/ws-policy
18. World Intellectual Property Organization: WIPO Copyright Treaty (WCT) (1996), http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html
19. Wang, Y., Stroulia, E.: Flexible Interface Matching for Web Service Discovery. In: Proc. of the Fourth Intl. Conf. on Web Information Systems Engineering (2003)
20. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, Springer, Heidelberg (2002)
21. Vega, J.P.C., Franch, X., Quer, C.: Towards a Unified Catalogue of Non-Technical Quality Attributes to Support COTS-Based Systems Lifecycle Activities. In: ICCBSS. Proc. of the IEEE Intl. Conference on COTS Based Software Systems, IEEE Computer Society Press, Los Alamitos (2007)
22. Verma, K., Akkiraj, R., Goodwin, R.: Semantic Matching of Web Service Policies. In: Second Intl. Workshop on Semantic and Dynamic Web Processes (2005)
23. Yang, W., Ludwig, H., Dan, A.: Compatibility Analysis of WSLA Service Level Objectives. Technical Report RC22800 (W0305-082), IBM Research Division (2003)
24. Garcia, R., Gil, R., Delgado, J.: A Web Ontologies Framework for Digital Rights Management. Journal of Artificial Intelligence and Law Online First (2007), http://springerlink.metapress.com/content/03732x05200u7h27