Evaluating Contract Compatibility for Service Composition in The SeCO$_2$ Framework*

Marco Comerio[1], Hong-Linh Truong[2], Flavio De Paoli[1], Schahram Dustdar[2]

[1] University of Milano - Bicocca, Milano, Italy
{comerio,depaoli}@disco.unimib.it
[2] Distributed Systems Group, Vienna University of Technology
{truong,dustdar}@infosys.tuwien.ac.at

Abstract. Recently, the Software-as-a-Service (SaaS) model has been increasingly supported, becoming a major part of the new emerging cloud computing paradigms. Although SaaS exists in different forms, supporting and providing SaaS developed based Web services has attracted a large effort from industries and academics because this form of SaaS allows software to be easily composed and integrated to offer new services for customers. Even though various service composition techniques, based on functional and non-functional parameters, have been proposed, the issue of service contract compatibility has been neglected. This issue is of paramount importance in the Web services-based SaaS model because services are provided by different providers, associated with different contracts which are defined by different specifications. This paper proposes techniques for supporting service composers to deal with the heterogeneity of service contracts in service composition. We describe a novel approach for modeling and mapping different service contract specifications, and a set of techniques for evaluating service contract compatibility. Our techniques consider contract terms associated with data and control flows, as well as composition patterns. Illustrating scenarios are proposed to demonstrate the efficiency of our techniques.

1 Introduction

We have recently observed the rise of cloud computing and SaaS as a part of the cloud computing paradigm [1]. In particular, many providers have provided SaaS using the Web services model. This form of SaaS has been widely supported because it enables service composition and integration.

Techniques supporting service composition and integration have been developed for a long time. It is important that when services are selected and composed from different SaaS providers, their contracts, which govern how the services should be used, have to be compatible. We need to support both, users and tools, to deal with issues related to service contracts. This support is of paramount importance because how services are used is bound to the service

* This work is partially supported by the European Union through the FP7-216256 project COIN

contract. However, there is a lack of supporting tools to deal with the evaluation of service contract compatibility, which is actually just one of many open questions about the relationship between service contracts and service composition discussed in [2]. Current techniques, such as service license compatibility [3], are not suitable because they assume that service contracts follow the same specification and they do not support service contracts for service compositions. To our best knowledge, until now, there is no work supporting service contract compatibility that takes into account the heterogeneity of service contract specifications and different aspects associated with data and control flows of the composition.

In this paper, we present an overview of SeCO$_2$, a novel framework for supporting service composers to deal with the heterogeneity of service contracts in service compositions. The framework is a part of tools and systems for supporting the life-cycle management of the ecosystem of service contracts. Within this paper we present the following contributions: (i) a novel approach for modeling and mapping different service contract specifications, and (ii) a set of techniques for service contract compatibility evaluation. Our techniques consider contract terms associated with *data* and *control flows*, as well as *composition patterns*.

The rest of this paper is structured as follows: Section 2 elaborates the context, motivation, and related work of this paper. We discuss our approach and give an overview of SeCO$_2$ in Section 3. We present techniques to achieve the modeling and mapping of service contract specifications in Section 4. We present the compatibility evaluation in Section 5. Experiments are presented in Section 6. We conclude the paper and give an outlook to the future work in Section 7.

2 Motivation and Related Work

2.1 Motivation

The main motivation of our work is, in general, how to ensure the compatibility of service contracts for service compositions. In the current service composition landscape there is the need to compose different services to provide converged services. In the SaaS model it is assumed that the service customer uses the software deployed as a service. This model allows service providers to combine different services, potentially characterizing by different service contracts specified by different languages. With the techniques developed so far, it is not so difficult for consumers to compose different services based on published service interfaces. For example, existing platforms like The Process Factory[3] and Boomi[4] provide different connectors for consumers to compose their services from various SaaS providers. However, the consumers need to ensure that the service compositions do not include conflicting service contracts. This assurance cannot be given by a single SaaS provider and currently is not available in existing composition tools.

In the SaaS and cloud computing model, no single specification would be agreed by all, making the service contract compatibility evaluation hard. Past

[3] `http://www.theprocessfactory.com`
[4] `http://www.boomi.com`

research has neglected contracts of composite services when performing service composition by considering only functional parameters (service interfaces) or assume that contracts associated with services being composed are described by a single language. Furthermore, past research has not focused on tools and algorithms dealing with contract compatibility evaluation when combining different services from different providers. Typically, they deal with only contract negotiation between consumer and service in a point-to-point manner.

Service contract compatibility is also strongly dependent on the structure of composition. This is related to not only control flows but also data flows and composition patterns. While certain works address QoS-based compatibility for control flows, currently there is a lack of a good understanding of how to check contract compatibility for data, the input/output of services, whose contract terms are not always the same to that of the service operations. We stress that contract terms associated with the use of service and the use of data are different and our objective is to address the compatibility for both data and service.

2.2 Related Work on Service Contract Compatibility

As stated in [2], the understanding between a service consumer and a service provider can be established using different approaches (e.g., policies, licenses, service level agreements). Even if some philosophical differences exist among these approaches, in this paper we identify them under the common term *service contract* which specifies conditions that a service consumer and a service provider agree. Besides functional terms, a service contract is composed of the specification of *Quality of Service (QoS)*, *Business*, *Service Context* and *License* terms of a service. *QoS* terms (e.g., response time) represent technical aspects of the service. *Business* terms (e.g., service price) describe financial terms and conditions. *Service Context* terms (e.g., service delivery location) define the characteristics of the context associated with the service. Finally, *License* terms (e.g., limitation of liabilities and usage permissions) state responsibilities among involved parties and conditions on service usage.

Currently, service contracts can be described using several specifications, such as ODRL-S [4], WSLA [5], and WSOL [6]. Even though these specifications have some common parts, there exists no reference ontology/thesaurus for describing contract properties. This means that service consumers and providers specify their service contracts as they wish, causing many issues when multiple services governed by different contracts are utilized (e.g., in a composition). Until now we are not aware of any work dealing with the definition of techniques to evaluate the compatibility among contract terms specified in different languages.

The most cited works [7–9] are related to contract-based service composition and reduce the problem to the evaluation of QoS constraints among composite services and user requirements. The AgFlow framework [7] evaluates the QoS of composite services with an extensible multidimensional quality model and considering the control flow of the service composition. Other examples of QoS-based composition are in [8] that aims at defining composition rules to evaluate global values of QoS dimensions according to specific workflow patterns. The constraint-driven Web service composition tool presented in [9] reduces the service composition problem to a constraint satisfaction problem focusing on business and process constraints. These works consider only a small set of service contract terms (i.e., QoS). The evaluation of qualitative properties (e.g., license terms) are not tackled. Moreover, they assume that property descriptions are always available and specified using a common language.

3 Overview of The SeCO$_2$ Framework

The objective of the SeCO$_2$ framework is to support service composers to deal with the heterogeneity of service contracts in service composition. In this paper we focus on techniques used by SeCO$_2$ to support the modeling and mapping of service contracts defined using different specification languages and the evaluation of the compatibility among these service contracts. Figure 1 provides an overview of actors and data involved in these activities.

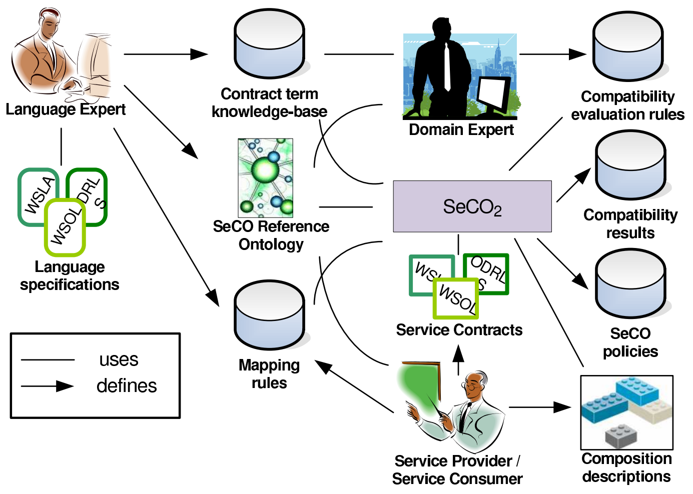


Fig. 1. The SeCO$_2$ Framework

The SeCO$_2$ framework overcomes the heterogeneity in service contract specifications using the *SeCO Reference Ontology* containing semantic descriptions of service contract properties and the *Contract term knowledge-base* specifying additional information about these properties. The *SeCO Reference Ontology* is built based on the Policy-Centered Metamodel (PCM) [10]. The PCM offers (i) the concept of *Policy* that aggregates property descriptions into a single entity with an applicability condition, and (ii) a set of constraint operators that allows for the description of both qualitative and quantitative properties.

The SeCO$_2$ framework deals with service contracts specified in different languages (e.g., ODRL-S, WSLA and WSOL). In this paper we assume that SeCO$_2$ receives these contracts from service providers and we show how it makes them comparable by wrapping them into *SeCO Policies*. In order to do this, *Language*

Experts analyze *language specifications* and create, modify, update and delete the knowledge stored in the *SeCO Reference Ontology* and in the *Contract term knowledge-base.* The mapping between ontological concepts and contract-specific terminologies is defined by *Language Experts*, *Service Providers and Consumers* with *mapping rules.* In order to define techniques for service contract compatibility evaluation, the SeCO$_2$ framework supports *Domain Experts* in the definition of *compatibility evaluation rules* by means of the *SeCO$_2$ Reference Ontology* and the *Contract term knowledge-base. Mapping rules* and *compatibility evaluation rules*, as well as the *SeCO Reference Ontology* and the *Contract term knowledge-base*, are shared information for the users. This aspect reduces the effort for their definition and improves reusability. These rules allow SeCO$_2$ to receive *Service contracts* and *Composition descriptions* as inputs, perform the wrapping to *SeCO policies* and verify the compatibility among them.

One of the most innovative characteristics of our framework is how the compatibility evaluation is performed. Currently, there is no distinction between the description of properties related to service usage (e.g., `Request Limit`) and properties related to the data produced by the service (e.g., `Data Ownership`). This produces ambiguities in service contract specifications. As an example, the property `Price` can refer to the amount of money needed for invoking a service or it can refer to the amount of money needed for receiving an amount of data from a service. This distinction is critical for the service contract compatibility evaluation. The SeCO$_2$ framework performs the compatibility evaluation considering both the *control flow* and the *data flow* of the service composition. Dependencies between each service contract property and *control* and *data flow* are identified and considered during the definition of *compatibility evaluation rules.* Furthermore, another characteristics of the service contract compatibility evaluation performed by SeCO$_2$ is that it is not limited to *QoS* but it is also extended to other types of property that can be included in a service contract like *Business*, *Service Context* and *License* terms. Table 1 shows the influences between each identified service contract property type and *control* and *data flow.*

	control flow	**data flow**	**independent**
Quality of Service (QoS)	X		
Service Context			X
Business	X	X	
License	X	X	

Table 1. Data and control flows in contract compatibility evaluation

4 Modeling and Mapping Service Contract Specifications

The first step in order to achieve the service contract compatibility is that we have to develop techniques to map different service contracts described in different specifications and terminologies. In our view, the mapping of service contract specifications are not a static, but a dynamic process because specifications and terminologies as well as knowledge about them change over the time.

4.1 Typology of Contract Specifications

Starting from the analysis of ODRL-S, WSLA and WSOL we have identified three types of languages for the specification of service contract properties:

- `Type A`: includes *languages allowing the specification of predefined properties.* In this type, e.g., ODRL-S, the properties that can be specified are known by the Language Expert.
- `Type B`: includes *languages allowing the specification of user-defined properties.* In this type, e.g., WSLA, the Language Expert knows only the structure of the specification but the properties are defined by the Service Provider.
- `Type C`: includes *languages allowing the specification of properties defined in user ontologies.* In this type, e.g., WSOL, the Language Expert knows only the structure of the specification while the properties are specified by the Service Provider using external ontologies.

We use the SeCO Reference Ontology for mapping different specifications and for allowing compatibility evaluation. This ontology is composed of: (i) a core part containing the specification of common properties (e.g., QoS) and (ii) a plug-in part that can be enriched by Language Experts with new properties.

Languages in `Type A` (e.g., ODRL-S) are characterized by profile models describing all the properties that can be included in a service contract. In this case, the Language Expert enriches the plug-in part to model all the properties not included in the ontology. Moreover, the Language Expert can define fixed mapping rules between properties and ontological concepts.

Languages in `Type B` (e.g., WSLA) allow new properties to be defined. This characteristic limits the possibility to perform the modeling and mapping of properties in these languages into SeCO_2 in advance. Thus, interactions to the Service Providers are still needed when wrapping a concrete service contract into SeCO policies. However, users of the same domain (e.g., the logistic operator domain) typically utilize common terminologies, e.g., logistic operator service providers utilize the term `Shipping Location` in their specifications which has the same meaning of the property `Service Delivery Location` available in the core part of the SeCO Reference Ontology. Common terminologies and domain-specific knowledge are used by Language Experts and Domain Experts to define customized mapping rules which will reduce the interactions needed for the wrapping of service contracts.

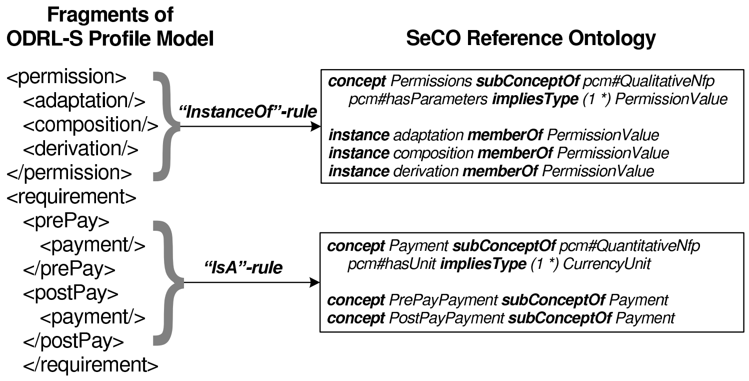
Languages in `Type C` (e.g., WSOL) are similar to the ones in `Type B` but here the properties are semantically described in external ontologies. Thus, the possibility to perform the modeling and mapping activities is limited and we need to define customized mapping rules between concepts in the most common user ontologies and concepts specified in the SeCO Reference Ontology.

Since contract specifications use different representations, ontology alignment tools [11–13], which supports mappings between concepts defined in different ontologies, cannot be used to fully automate the mapping between different specifications. Furthermore, as the interpretation of contract terms may vary from different service providers, fully automatically generation of mapping rules cannot be achieved. However, these tools can support the definition of mapping rules when we deal with ontology-based specifications. In this paper, we consider the use of these tools as external activities triggered by the user of SeCO$_2$.

4.2 Modeling and Mapping Service Contract Terminologies Into the Reference Ontology

When an XML-based profile model defining properties is available (i.e., Languages in `Type A`), a set of general rules is used to extract properties from XML-based specifications and semantically describe them into the SeCO Reference Ontology. General rules link an XML-structure to a proper PCM-based description. The same structure can be associated with several rules because also the nature of the property must be considered. Examples are: (i) the XML-structure in which an element *C1* has a set of sub-elements can be linked to the `InstanceOf`-rule that consider each sub-element as possible values assumed by *C1*; (ii) the XML-structure in which different elements (e.g., *C2* and *C3*) have the same sub-element *C1* is linked to the `IsA`-rule that considers *C2* and *C3* as specializations (i.e., sub-concepts) of *C1*.

To illustrate the above-mentioned techniques, we focus on modeling and mapping ODRL-S terminology. Figure 2 shows how general rules can be used to model ODRL-S properties into the SeCO Reference Ontology. The following ODRL-S terms [4] are considered: (i) `Permission Rights`: defines types of uses of the service, such as `Adaptation`, `Composition` and `Derivation`; (ii) `Payment`: describes the financial terms assuming values, such as `PrePay` and `PostPay`.



Fragments of ODRL-S Profile Model
SeCO Reference Ontology
<permission>
<adaptation/>
<composition/>
<derivation/>
</permission>
<requirement>
<prePay>
<payment/>
</prePay>
<postPay>
<payment/>
</postPay>
</requirement>
"InstanceOf"-rule
"IsA"-rule
concept Permissions subConceptOf pcm#QualitativeNfp
pcm#hasParameters impliesType (1 *) PermissionValue
instance adaptation memberOf PermissionValue
instance composition memberOf PermissionValue
instance derivation memberOf PermissionValue
concept Payment subConceptOf pcm#QuantitativeNfp
pcm#hasUnit impliesType (1 *) CurrencyUnit
concept PrePayPayment subConceptOf Payment
concept PostPayPayment subConceptOf Payment

Fig. 2. Modeling ODRL-S properties in the SeCO Reference Ontology

In ODRL-S, `Adaptation`, `Composition` and `Derivation` are sub-elements of `Permission`. For this property the ODRL-S Language Expert uses the `InstanceOf`-rule to define a new concept `Permissions` in the ontology which can assume a fixed set of values (i.e., `pcm#hasParameters impliesType PermissionValue`) that are `Adaptation`, `Composition` and `Derivation`. `PrePay` and `PostPay` are super-elements for `Payment`. In this case the `IsA`-rule is applied considering them as specializations of the term `Payment`. A new concept `Payment` and two sub-concepts (`PrePayPayment` and `PostPayPayment`) are added to the ontology.

After the modeling of a property in the SeCO Reference Ontology, the Language Expert defines a mapping rule between the property and the related ontological concept; the rule is used in the wrapping of service contract specifications to SeCO Policies. Moreover, the Language Expert stores information into the Contract term knowledge-base about the influences of the property on the data and control flows of the composition. This information is used by Domain Experts for the definition of the related compatibility evaluation rule.

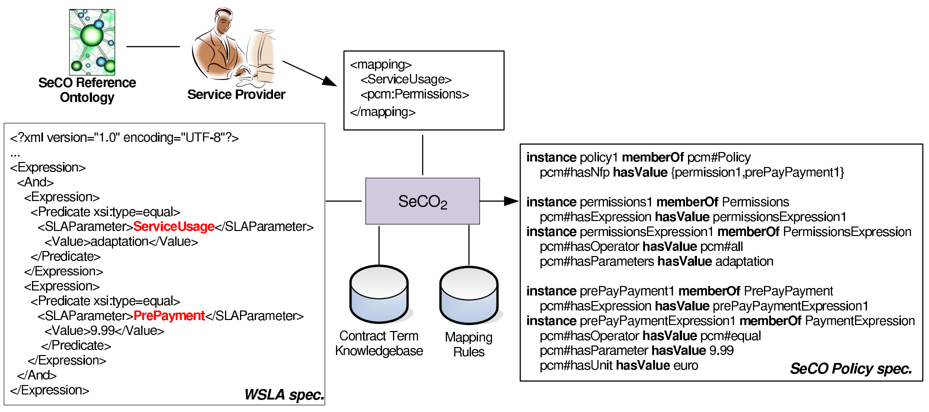
4.3 Wrapping Service Contract Specifications to SeCO Policy

A proper technique for each type of language must be defined to perform the wrapping from service contracts to SeCO Policy specifications. The wrapping of specifications in `Type A` language is directly performed by applying the mapping rules defined by Language Experts. For what concern specifications in `Type B` and `Type C` languages the wrapping activity may require interactions with the Service Providers to handle the absence of knowledge (i.e., mapping rules) on specified properties. The Service Providers must define the mapping between their properties (i.e., text labels for `Type B` and ontological concepts for `Type C`) and concepts available in the SeCO Reference Ontology.

For what concern specifications in `Type B` languages, lexical databases like WordNet support Service Providers to define mapping rules identifying synonyms between text labels and ontological concepts defined in the SeCO Reference Ontology. Different types of ontology alignment tools can be also used to support the wrapping of specifications in `Type C` languages: (i) tools for defining a mapping between concepts in two different ontologies by finding pairs of related concepts (e.g., `ANCHORPROMPT` [11]) or by evaluating semantic affinity between concepts (e.g., `H-MATCH` [12]) and (ii) tools for defining mapping rules to relate only relevant parts of the source ontologies (e.g., `ONION` [13]).

In this section, we describe a solution for the wrapping of a WSLA specification. The procedure used by SeCO_2 is the following: (i) parse the specification in order to detect properties (i.e., `SLAParameters`); (ii) search the availability of customized mapping rules related to the detected properties; (iii) if mapping rules are not identified, use WordNet to identify a possible mapping between the `SLAParameters` and concepts available in the SeCO Reference Ontology and ask confirmation about the correctness of the mapping to the Service Provider; and (iv) if the mapping is not correct or not available, ask to the Service Provider to perform the mapping manually.

Figure 3 illustrates the above-mentioned steps when wrapping a WSLA-based service contract consisting `PrePayment = 9.99` Euros and `ServiceUsage = ''adaptation''`. In this example, a customized mapping rule for `PrePayment` is identified. On the contrary, the term `ServiceUsage` is not known and no rules are available. Moreover, no synonym relations are specified in WordNet between `ServiceUsage` and terms defined in the SeCO Reference Ontology. In order to handle this absence of knowledge, the Service Provider is asked to navigate the ontology and map the SLAParameter `ServiceUsage` to any ontological concept. The result is the mapping of `ServiceUsage` with `Permissions`.



```
<mapping>
  <ServiceUsage>
  <pcm:Permissions>
</mapping>
```

```
<?xml version="1.0" encoding="UTF-8"?>
...
<Expression>
 <And>
  <Expression>
   <Predicate xsi:type=equal>
    <SLAParameter>ServiceUsage</SLAParameter>
     <Value>adaptation</Value>
   </Predicate>
  </Expression>
  <Expression>
   <Predicate xsi:type=equal>
    <SLAParameter>PrePayment</SLAParameter>
     <Value>9.99</Value>
    </Predicate>
   </Expression>
 </And>
</Expression>
```

```
instance policy1 memberOf pcm#Policy
   pcm#hasNfp hasValue {permission1,prePayPayment1}

instance permissions1 memberOf Permissions
   pcm#hasExpression hasValue permissionsExpression1
instance permissionsExpression1 memberOf PermissionsExpression
   pcm#hasOperator hasValue pcm#all
   pcm#hasParameters hasValue adaptation

instance prePayPayment1 memberOf PrePayPayment
   pcm#hasExpression hasValue prePayPaymentExpression1
instance prePayPaymentExpression1 memberOf PaymentExpression
   pcm#hasOperator hasValue pcm#equal
   pcm#hasParameter hasValue 9.99
   pcm#hasUnit hasValue euro
```

Fig. 3. Mapping between WSLA and SeCO Policy

After this preliminary step, the mapping proceeds considering the `Expressions` defined in each `Service Level Objective` of the WSLA specification. Each `Expression` follows the first order logic, including predicates and logic operators. According to the logic operators, different mapping rules can be applied. The simplest form of a logic expression is a plain predicate. The mapping to a SeCO policy includes the following steps: (i) the mapping rule is used to identify in the SeCO Reference Ontology the concept related to the `SLAParameter` specified by the Service Provider; (ii) a new instance of this concept is created. It must be characterized by an expression having constraint operator and parameter equals to `Type` and `Value` of the `Service Level Objective`; (iii) a new SeCO Policy containing the concept instance is created.

In Figure 3, the logic operator "*And*" is used to specify the aggregation of two plain predicates stating conditions on `PrePayment` and `ServiceUsage`. The mapping to a SeCO Policy consists in defining the concept instances related to all the plain predicates. The final result for the considered example is a SeCO Policy containing: (i) an instance of `Permissions` characterized by an expression stating that the value `adaptation` is assumed (i.e., `pcm#hasOperator hasValue pcm#all`; `pcm#hasParameters hasValue adaptation`) and (ii) an instance of `PrePayPayment` stating that the amount is equal to 9.99 Euros (i.e., `pcm#hasOperator hasValue pcm#equal`; `pcm#hasParameter hasValue 9.99`; `pcm#hasUnit hasValue euro`).

5 Contract Compatibility Evaluation for Service Composition

The service contract compatibility evaluation supported by the SeCO$_2$ framework accepts a full or part of a full description of service compositions, e.g., the complete structure of a composite service or a workflow region.

5.1 Contract Compatibility Evaluation Rules

The evaluation of service contract compatibility is based on rules defined for service contract properties. As described in [10], service contract properties can be classified into qualitative and quantitative properties. Moreover, as shown in Table 1, properties can differently influence control and data flows.

Qualitative properties must be evaluated considering the relations stored in the SeCO Reference Ontology. Examples of compatibility evaluation rules are:

– *"Relation-based"* rule: it is applicable to properties assuming values characterized by semantic relations among them. Examples are *partnership* (i.e., values characterized by *partOf* relations) and *subsumption* (i.e., values characterized by *isA* relations). These relations are checked to verify the compatibility among values associated to a property.
– *"Compatible value list"* rule: it is applicable to properties assuming a small set of possible values. The compatibility list among these values is stored into the reference ontology by the definition of *isCompatibleWith* relations.

Quantitative properties must be evaluated considering the constraint operators used to specified the offered values. As described in [10], constraint operators can be *binary* (e.g., $=, \leq, \geq$) or *ternary* (e.g., range of values). *"Binary operator"* and *"Ternary operator"* rules (see [14] for details) evaluate a numeric values in the range [0..1] stating the degree of compatibility between two offered values and the overlap between ranges of values respectively.

Property	**Type**	**Data Flow**	**Control Flow**	**Rule**
Service Delivery Location	Service Context			partnership
Pricing	Business	X		compatible value list
Payment (for data usage)	Business	X		binary, ternary
Payment (for service usage)	Business		X	binary, ternary
Scalability	QoS		X	binary, ternary
Request Limit	QoS		X	binary, ternary
Availability Time Range	QoS		X	ternary
Data Ownership	License	X		compatible value list
Permissions	License		X	subsumption

Table 2. Examples of common rules

Table 2 presents some common rules for the evaluation. We explain some of them in the following. The `Service Delivery Location` property is independent from data and control flows since its value must be checked in all the contracts of the services involved in the composition. The compatibility is evaluated applying a *"Relation-based"* rule focusing on partnership relations ($\sqsupseteq$). In particular, services s_1 and s_2 are compatible if $s_1.value \sqsupseteq s_2.value$ or $s_2.value \sqsupseteq s_1.value$. For example, let us assume that s_1 delivers in the `Worldwide`, s_2 in `Europe` and s_3 in the `US`. The following partnership relations are hold: `Worldwide`$\sqsupseteq$`Europe` and `Worldwide`$\sqsupseteq$`US`. Thus, services s_2 and s_3 cannot be included in the same composition since their provision is limited to different geographical area.

The compatibility on `Pricing` terms in service contracts is checked considering the data flow. The evaluation is performed using a *"Compatible value list"* rule stating the compatibility among possible pricing models. For example, `flat rate` is compatible with `pay per use with subscription` but it is incompatible with `free per use`.

The property `Scalability` is checked considering the *composition patterns* included in the control flow specification and applying a *"Binary operator"* rule. For example, assume that service s_1 and s_2 follow a *sequential execution* and that s_1 and s_2 have `Scalability` $= sc_1$ and `Scalability` $= sc_2$, respectively. Services s_1 and s_2 are compatible if $sc_1 \leq sc_2$.

5.2 An Algorithm for Contract Compatibility Evaluation

Let $S = \{s_1, s_2, \cdots, s_m\}$ denote the set of services involved in the composition. Each service is characterized by a service operation associated with one or more SeCO Policies. Let $P(s_i) = \{p_1, p_2, \cdots, p_n\}$ indicate the set of policies associated to service s_i. Each policy is composed of one or more offered properties. Let $PR(p_i) = \{pr_1, pr_2, \cdots, pr_w\}$ be the set of properties offered by policy p_i. Each property is specified by: (i) a `name` stating the related ontological concept; (ii) a `type` defining if the property is *CF-inf* (i.e., influence the *control flow*), *DF-inf* (i.e., influence the *data flow*) or *F-ind* (i.e., flow independent); (iii) an `operator`; (iv) a `value` and (v) a `unit` of measure.

Let $CF(s_i) = \{cf_1, cf_2, \cdots, cf_m\}$ denote the *control flow* where each cf_j in $CF(s_i)$ specifies the *composition pattern* between s_i and s_j. Possible values are *sequential*, *parallel* and *conditional execution.* Let $DF(s_i) = \{df_1, df_2, \cdots, df_m\}$ denote the *data flow* where each df_j in $DF(s_i)$ specifies if there is a dependency in data provisioning between s_i and s_j.

Our service contract compatibility algorithm is listed in Algorithm 1. The algorithm evaluates the compatibility among all the policies of all the couples of services available in the composition. Line 3 defines $\Omega(s_i, s_j)$ as a set of triples. Each triple will contain a policy p_w associated to s_i, a policy p_z associated to s_j, and the result of the compatibility evaluation $\lambda(p_w, p_z)$ among them. The evaluation of $\lambda(p_w, p_z)$ starts in Line 7 defining $\Upsilon(p_w, p_z)$ as a set of comparable properties $[pr_1, pr_2]$ specified in p_w and p_z. $\Upsilon(p_w, p_z)$ is populated by the `Matching` procedure (Line 8) that applies matching rules similar to the ones shown in [10]. For each identified couple $[pr_1, pr_2]$ of comparable properties, the algorithm retrieves the related evaluation rule using the procedure `Extract` and specifying the property *name* (Line 10). As stated above, at this point the evaluation proceeds considering the property `type`. If the property is *CF-inf* then procedure `EvalRuleF` is invoked specifying the retrieved rule, the two comparable properties $[pr_1, pr_2]$ and the *control flow* information about the services s_i, s_j that offer the properties (Line 12). If the property is *DF-inf* then the same procedure `EvalRuleF` is invoked but specifying the *data flow* information about the services s_i, s_j (Line 15). Finally, if the property is *F-ind* then the procedure `EvalRule` that does not consider composition flows is invoked (Line 17). The result of the evaluation is saved in $\lambda(p_w, p_z)$ that contains the evaluation of all the

Algorithm 1 Compatibility Evaluation

```
1: for all $s_i \in S$ do
2:   for all $s_j \in S(j \neq i)$ do
3:     $\Omega(s_i, s_j) = \phi$ where $\Omega(s_i, s_j)$ is a set of triples $[p_w, p_z, \lambda(p_w, p_z)]$
4:     for all $p_w \in P(s_i)$ do
5:       for all $p_z \in P(s_j)$ do
6:         $\lambda(p_w, p_z) = \phi$, where $\lambda(p_w, p_z)$ is a set of triples $[pr_i, pr_j, result]$
7:         $\Upsilon(p_w, p_z) = \phi$, where $\Upsilon(p_w, p_z)$ is a set of comparable properties $[pr_1, pr_2]$
8:         $\Upsilon(p_w, p_z) = Matching(p_w, p_z)$
9:         for all $[pr_1, pr_2] \in \Upsilon(p_w, p_z)$ do
10:          $rule = Extract(pr_1.name)$
11:          if $pr_1.type =' CF - inf'$ then
12:            $\lambda(p_w, p_z) = \lambda(p_w, p_z) \cup EvalRuleF(rule, pr_1, pr_2, cf_j \in CF(s_i))$
13:          else
14:            if $pr_1.type =' DF - inf'$ then
15:              $\lambda(p_w, p_z) = \lambda(p_w, p_z) \cup EvalRuleF(rule, pr_1, pr_2, df_j \in DF(s_i))$
16:            else
17:              $\lambda(p_w, p_z) = \lambda(p_w, p_z) \cup EvalRule(rule, pr_1, pr_2)$
18:            end if
19:          end if
20:        end for
21:        $\Omega(s_i, s_j) = \Omega(s_i, s_j) \cup [p_w, p_z, \lambda(p_w, p_z)]$
22:      end for
23:    end for
24:  end for
25: end for
```

comparable properties in p_w and p_z. Finally, the triple $[p_w, p_z, \lambda(p_w, p_z)]$ is saved in $\Omega(s_i, s_j)$ (Line 21) that contains the evaluation for all the policies offered by s_i and s_j.

6 Illustrating Scenarios

In order to demonstrate the contract compatibility evaluation techniques proposed in Section 5, we consider a process of purchase data analysis inside a supply chain management scenario. This process involves multiple services collaborating with each others: (i) a `Request Service` (RS) issuing a purchase request; (ii) a `Purchase Processing Service` (PPS) managing the standard e-commerce process; (iii) a `Merchant Validation Service` (MVS) verifying and providing data about a shopping merchant; (iv) a `Payment Verification Service` (PS) validating data related to the payment (e.g., the credit card number); (v) `Shipping Evaluation Service` (SES) calculating shipping charges and (vi) a `Purchase Validation Service` (PVS) analyzing data and validating the purchase. These services can be composed using different control and data flows. Figure 4 shows two different possible composition structures.

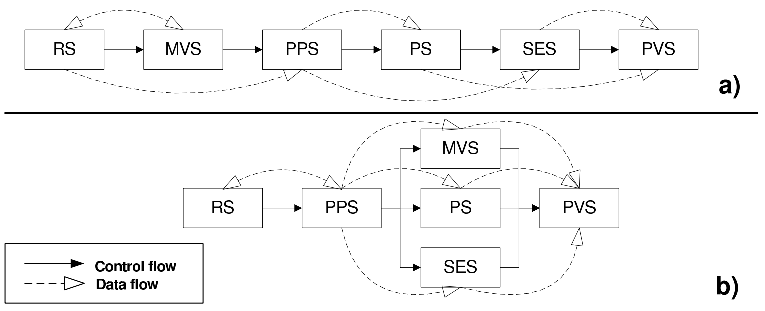


Fig. 4. Different composition structures for the *Purchase Data Analysis* service.

.

Let us assume that a service consumer wants to create a `Purchase Data Analysis (PDA)` service by composing his/her RS with the following Web services: (i) *Yahoo! Shopping Web Service*[5] as MVS; (ii) *XWebCheckOut Web Service*[6] as PPS; (iii) *Aivea Shipping Web Service*[7] as SES; (iv) *ValidateCreditCard Web Service*[8] as PS and (v) *DOTS Lead Validation Web Service*[9] as PVS. For our experiments, we focus only on service contracts. Thus, let us assume that these Web services match the functionalities required for the `PDA` service.

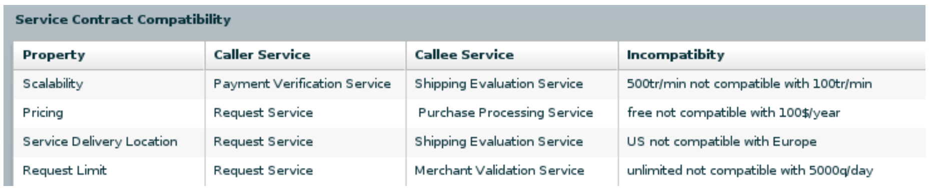
The selected Web services are characterized by service contracts available only as HTML texts in their Websites (i.e., ODRL-S, WSLA and WSOL specifications are not available). Moreover, these contracts are unclear, ambiguous and limited to few information. This forces the service consumer to manually compare them and, often, further information from the service providers are needed. Modeling and mapping techniques presented in Section 4 are not applicable due to the absence of structured specifications. In order to overcome this strong limitation, we produce SeCO Policies using information described in the HTML texts and inserting realistic properties in case of limited descriptions. These policies are summarized in Table 3. For each selected Web service, we consider the properties `Service Delivery Location`, `Pricing` and `Scalability` described in Section 5. Moreover, we consider `Data Ownership` (a license term stating how the data produced by the service are protected) and `Request Limit` (a license term defining the maximum number of requests that a user can submit to the service in a day).

Applying evaluation rules like the ones described in Section 5, the compatibility evaluation results produced by our SeCO$_2$ framework are given in Figures 5 and 6. The following results must be underlined: (i) incompatibility on `Service Delivery Location` is found in both the compositions since the property is independent from data and control flows; (ii) incompatibility on `Pricing` is found in both the compositions because both are characterized by a data flow from RS and

[5] http://developer.yahoo.com/shopping/V1/merchantSearch.html
[6] http://www.xwebservices.com/Web_Services/XWebCheckOut/
[7] http://www.aivea.com/shipping-web-service.htm
[8] http://www.webservicex.net/WCF/ServiceDetails.aspx?SID=14
[9] http://www.serviceobjects.com/products/composite/lead-validation

	Del.Loc.	**Data Own.**	**Request Limit**	**Pricing**	**Scalability**
Request Service (RS)	US	personal-use	unlimited	free	100tr/min
Yahoo! Shopping (MVS)	Worldwide	copyrighted	5000q/day	free	100tr/min
XWebCheckOut (PPS)	Worldwide	free-distrib.	unlimited	100$/year	100tr/min
Aivea Shipping (SES)	Europe	free-distrib.	unlimited	49$/month	100tr/min
ValidateCreditCard (PS)	Worldwide	free-distrib.	unlimited	free	500tr/min
DOTS Lead Valid. (PVS)	Worldwide	free-distrib.	unlimited	free	500tr/min

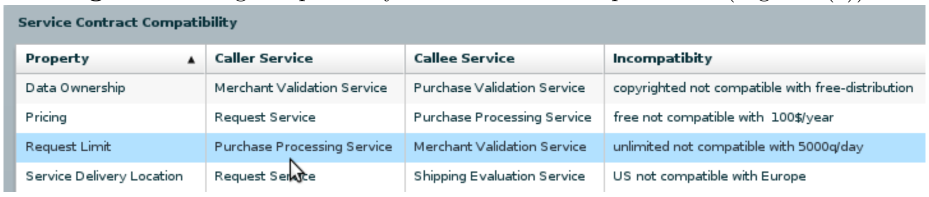
Table 3. Contracts offered by services involved in the composition



Service Contract Compatibility

Property	Caller Service	Callee Service	Incompatibity
Scalability	Payment Verification Service	Shipping Evaluation Service	500tr/min not compatible with 100tr/min
Pricing	Request Service	Purchase Processing Service	free not compatible with 100$/year
Service Delivery Location	Request Service	Shipping Evaluation Service	US not compatible with Europe
Request Limit	Request Service	Merchant Validation Service	unlimited not compatible with 5000q/day

Fig. 5. Resulting compatibility evaluation for Composition a (Figure 4(a))



Service Contract Compatibility

Property ▲	Caller Service	Callee Service	Incompatibity
Data Ownership	Merchant Validation Service	Purchase Validation Service	copyrighted not compatible with free-distribution
Pricing	Request Service	Purchase Processing Service	free not compatible with 100$/year
Request Limit	Purchase Processing Service	Merchant Validation Service	unlimited not compatible with 5000q/day
Service Delivery Location	Request Service	Shipping Evaluation Service	US not compatible with Europe

Fig. 6. Resulting compatibility evaluation for Composition b (Figure 4(b))

PPS; (iii) incompatibility on `Request Limit` is found in both the compositions but between different services. This is determined by the different data flows involving MVS; (iv) incompatibility on `Scalability` is found only in `composition a`. This result depends on the different control flows (i.e., in `composition a` SES is invoked after PS instead in `composition b` it is invoked after PPS); (v) incompatibility on `Data Ownership` is found only in `composition b`. This result depends on the different data flows (i.e., in `composition a` MVS data are managed by RS instead in `composition b` they are managed by PVS).

7 Concluding Remarks

In this paper, we have presented our approach to checking service contract compatibility for service compositions. Our SeCO$_2$ framework provides support to define, update, and share knowledge about service contracts specified by different specifications. Our work can map different service contracts and determined the compatibility based on control and data flows, as well as composition patterns.

Our approach is currently tested with ODRL-S, WSLA, and WSOL. There is no way to automatically determine the typology of a language, thus mapping rules still involve domain experts. We think that it is inevitable, unless terminologies are well-defined and agreed by all service providers. Currently, we do not consider the dynamic changes of contracts during the composition. For example, when performing the composition, the customer and service providers might negotiate the contracts, as the contract changes certain steps have to be rerun.

However, currently we consider this change can be solved only by re-running the compatibility checking. Our future work includes enhancing this dynamic interaction among actors when dealing with service contracts. Furthermore, data specific contract compatibility will be improved.

References

1. Armbrust, M., Fox, A., Grifth, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley (2009)
2. Truong, H., Gangadharan, G., Treiber, M., Dustdar, S., D'Andrea, V.: On reconciliation of contractual concerns of web services. In: In NFPSLASOC'08 (2nd Non Functional Properties and Service Level Agreements in SOC Workshop), Dublin, Ireland (2008)
3. Gangadharan, G.R., Weiss, M., D'Andrea, V., Iannella, R.: "Service License Composition and Compatibility Analysis". In: "Proceedings of the International Conference on Service Oriented Computing (ICSOC'07), Vienna, Austria". (2007)
4. Gangadharan, G.R., D'Andrea, V., Iannella, R., Weiss, M.: "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)
5. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: "Web Service Level Agreement (WSLA) Language Specification". IBM Coporation (2003)
6. Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., Ma, W.: "Management Applications of the Web Service Offerings Language (WSOL)". Information Systems **30**(7) (2005) 564–586
7. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5) (2004) 311–327
8. Jaeger, M., Rojec-Goldmann, G., Muhl, G.: Qos aggregation for web service composition using workflow patterns. In: EDOC '04: Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International, Washington, DC, USA, IEEE Computer Society (2004) 149–159
9. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven web service composition in meteor-s. In: Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004). (2004) 23–30
10. De Paoli, F., Palmonari, M., Comerio, M., Maurino, A.: A Meta-Model for Non-Functional Property Descriptions of Web Services. In: Proceedings of the IEEE International Conference on Web Services (ICWS), Beijing, China (2008)
11. Noy, N.F., Musen, M.A.: The prompt suite: Interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies **59** (2003) 2003
12. S. Castano, A. Ferrara, S.M.: H-match: an algorithm for dynamically matching ontologies in peer-based systems. In: Proc. of the 1st VLDB Int. Workshop on Semantic Web and Databases (SWDB 2003), Berlin, Germany (2003)
13. Mitra, P., Wiederhold, G., Decker, S.: A scalable framework for the interoperation of information sources. In: Stanford University. (2001) 317–329
14. Comerio, M., De Paoli, F., Maurino, A., Palmonari, M.: "NFP-aware Semantic Web Services Selection". In: "Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)". (2007)