Languages for the Publication and Discovery of Context-dependent Services

Naseem Ibrahim
Albany State University
Albany, GA, USA
naseem.ibrahim@asurams.edu

Ismail Al-Ani
Ittihad University
RAK, UAE
iialani@ittihad.ac.ae

ABSTRACT

FrSeC is a framework for the provision and composition of trustworthy context-dependent services. To support FrSeC, this paper introduces the *ConfiguredService Description Language (CSDL)* and the *ConfiguredService Query Language (CSQL)*. CSDL is an XML based language for the description of ConfiguredServices by service providers. This description is published through service registries to enable the discovery of these services. *ConfiguredService* is a package in which the service contract, functionality, nonfunctional properties are bundled with the associated contextual conditions. CSQL is another XML based language that can be used by service requester to query the service registry for available ConfiguredServices. A case study is also presented to illustrate the use of CSDL and CSQL.

KEYWORDS

XML, SOA, Context, Trustworthiness, ConfiguredService.

1 INTRODUCTION

In service oriented applications, the three main activities are *service publication*, *service discovery* and *service provision*. Service publication refers to defining the service contract by service providers and publishing them through available service registries. Service discovery refers to the process of finding services that have been previously published and that meet the requirements of a service requester [1]. Typically, service discovery includes *service query*, *service matching*, and *service ranking*. Service requesters define their requirements as *service queries*. Service matching refers to the process of matching the service requester requirements, as defined in the service query, with the published services. Service ranking is the process of ordering the matched services according to the degree they meet the requester requirements. The ranking will enable the service requester to select a specific or a most relevant service from the list of candidate services. Service provision refers to the process of executing a selected service. The execution may include some form of an interaction between the service requester and service provider.

A service, when published, defines the contract that it can guarantee. However, a service cannot guarantee its contract in all situations. It can only guarantee it in a predefined set of conditions related to the context of the service consumer and requester. Contextual information is any information used to characterize the situation of an entity, such as location, time and purpose [2]. In addition, legal rules will further constrain the publication and provision of services. For example, a wireless Internet provider may include in the service contract a guarantee of excellent quality, but this guarantee is not absolute. It may have a

constraining condition stating that in order to ensure excellent quality; the consumer should be located within 100 meters from the wireless station. This constraint is related to the contextual information of the service consumer.

In addition, local legal rules may black-out wireless service in secure-critical locations. Such legal rules should also be part of the contract. A distinction should be made between legal rules and nonfunctional requirements. If a nonfunctional property is 'a soft' requirement it may be ignored. However ignoring a legal rule is equivalent to a 'legal violation', which might land in legal disputes and even lead to loss of entire business. In essence, not enforcing a legal rule prevents the execution of a contract. Almost all current approaches use only functional and nonfunctional properties to enable the publication, discovery and provision of services. In the literature [3], no distinction is made between legal rules and non-functional properties. Failure to include contextual information and legal rules will only mislead the consumer to believe in excellent quality of wireless service, regardless of where the consumer is domiciled. We have introduced contextual information and legal rules in service contracts in the newly introduced service model called *ConfiguredService*.

In recent publications [4], [5] and [6], we have given a formal framework, called FrSeC, in which service publications, and service compositions are formally described. In this paper, after briefly reviewing this previous work related to, we present CSDL: the ConfiguredService Description Language and CSQL: the ConfiguredService Query Language. CSDL is concerned with service publication while CSQL is concerned with service discovery.

In Section 2 we briefly review FrSeC. In Section 3, CSDL is introduced. Section 4 introduces CSQL. Section 5 introduces a case study from the automotive domain. Section 6 discuses a prototype implementation. In Section 7 we briefly compare our work with related work. We conclude the paper in Section 8.

2 FrSeC

The introduction of FrSeC was motivated by the need for a framework that supports the *publication*, *discovery* and *provision* of services with *context-dependent contracts*. The main elements of FrSeC are shown in Figure1. A complete formal definition of FrSeC is presented in the two recent papers [4] and [6]. Below is a brief summary of their features.

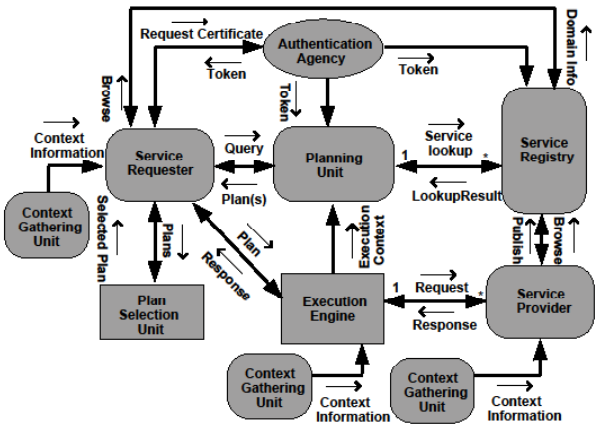

Figure 1. FrSeC Architecture.

Service Provider (SP): It is the entity that provides an implementation of a service specification. The service specification is published by the SP using SRe.

Service Registry (SRe): It is a central repository for services, in which Service Providers publish their services and PU discovers services. It includes semantic definitions for domain specific concepts.

Context Gathering Unit (CGU): FrSeC contains at least three context gathering units. One unit collects contextual information to assist SR in formulating their service queries. Another unit collects contextual information relevant to SP. The third unit collects contextual information to assist EU and PU in dynamic planning activities. A central context manager may be added to monitor and trigger the adaptive context-aware behavior of the framework.

Service Requester (SR): It is the entity requiring a certain functionality to be satisfied. It represents the client side of the interaction. It can be an application or another service. SR defines its requirements by a service query.

The Authentication Agency (AU): It is the entity responsible for ensuring trustworthy access to SRe. It provides requesters with certificates (token) that allow them to access SRe. The certificate type depends on the legal and contextual information of the requester.

Planning Unit (PU): It is responsible for managing the service discovery process by interacting with SR, SRe and AU. It also defines service composition. The composition includes defining the plans that can satisfy a query requirement. A plan defines the execution logic of a service or multiple services that collectively achieve the functional, nonfunctional and trustworthiness requirements of the requester. A complete formal composition theory is defined in [5]. This theory considers the functional, nonfunctional, legal and contextual parts of the service when defining the composition result.

Plan Selection Unit (PSU): It is responsible for helping SR to select one or more plans from the set of plans received from PU. For each plan received, it requests additional information, such as data parameters, from SR and verifies that the information in the plan is complete with respect to the request. If it finds the information incomplete the chosen plan is ignored, otherwise the plan is selected for SR.

Execution Unit (EU): It is responsible for managing the provision of services. It executes the selected plan. The execution process will include communicating with the service providers involved in the plan by sending service requests and obtaining service responses.

3. CSDL

Service providers publish service definitions through the service registry to be available for discovery. In current approaches, the service contract includes the functional and nonfunctional requirements together with any semantic information the service provider wishes to make public. But in FrSeC the service definition is much richer. It includes the service contract together with the related contextual conditions. Hence, we introduce ConfiguredService, which is a package in which service functionality, service contract, and service provision context are bundled together. SP publishes the ConfiguredService two

main elements; namely the contract and context. The contract includes function, nonfunctional properties and legal issues. The context part of the ConfiguredService includes the main parts; context info and context rules. The context info defines the contextual information of the ConfiguredService. The context rules define the contextual information related to SR that should be true for SP to guarantee its ConfiguredService contract.

CSDL is used to describe the ConfiguredServices to be published. The meta-model of CSDL is shown in Figure 2. The two main elements of a ConfiguredService are contract and context. Below is detailed discussion of each of these elements and how they are supported by CSDL.

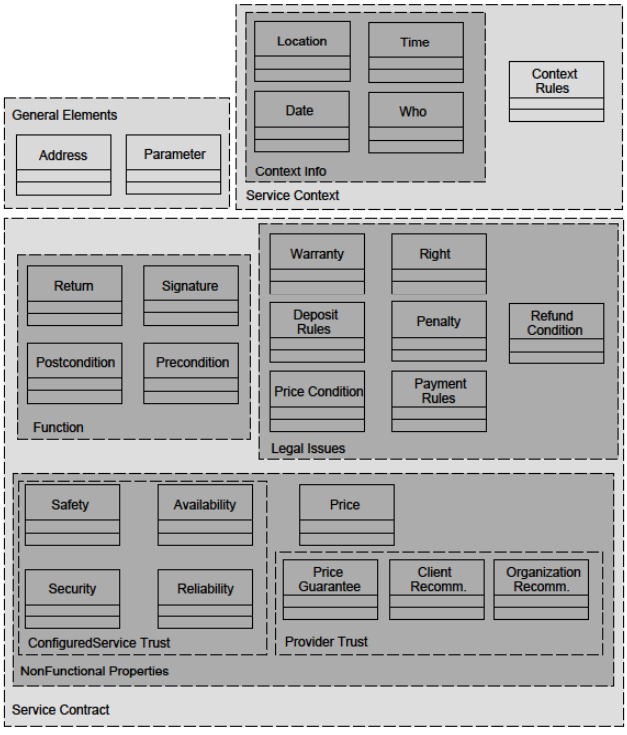

Figure 2. CSDL Meta-model.

3.1 Contract

The contract includes *function*, *nonfunctional* properties and *legal* issues. Below is a brief introduction of each element followed by the XML schema definition.

Function: ConfiguredService provides a single function. The function definition will include the function *signature*, *result*, *preconditions* and *postconditions*. The signature part defines the function *identifier*, the invocation *address*, and the *parameters* of the function. Each parameter has an *identifier* and a *type*. The result part defines the returned data of the service function. The preconditions define the conditions that should be true before the function invocation. The postconditions define the conditions that are guaranteed to be true after the function invocation. Below is the XML schema for defining function using CSDL.

```
<xs:complexType name="Precondition">
   <xs:sequence>
      <xs:element name="Condition" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Postcondition">
   <xs:sequence>
      <xs:element name="Condition" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Signature">
   <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
      <xs:element name="Address" type="xs:string"/>
      <xs:element name="Parameter" type="Parameter"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Parameter">
   <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="DataType" type="xs:string"/>
      <xs:element name="DefaultValue" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Return">
   <xs:sequence>
      <xs:element name="ID" type="xs:string"/>
      <xs:element name="Parameter" type="Parameter"
      minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="Function">
   <xs:sequence>
      <xs:element name="Signature" type="Signature"/>
```

```
    <xs:element name="Return" type="Return"/>
    <xs:element name="Precondition" type="Precondition"
    minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Postcondition" type="Postcondition"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Nonfunctional properties: A ConfiguredService definition includes *nonfunctional* properties that it can guarantee. These properties are to be chosen carefully so that they are verifiable, and encompass both quality and quantity aspects of service. *Trustworthiness* and *Price* are examples. Trust itself is further divided into *ConfiguredService trust* and *provider trust*. ConfiguredService trust defines the trustworthiness properties that are related to service provision. It includes the features *safety*, *security*, *availability*, and *reliability* [7]. Safety defines the critical conditions that are guaranteed to be true by Service Providers, such as timing conditions. Security is a composite of *data integrity* and *confidentiality*. Availability can be defined as the extent of readiness for providing correct services. Availability is specified as the maximum accepted time of repair until the service returns back to operate correctly. Reliability is the quality of continuing to provide correct services despite a failure. It is defined as the accepted mean time between failures. Below is the XML schema for defining reliability and availability using CSDL.

```
<xs:complexType name="Reliability">
  <xs:sequence>
    <xs:element name="constraint" minOccurs="0"/>
    <xs:element name="reliabilityRate" type="xs:double"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="Availability">
  <xs:sequence>
    <xs:element name="constraint" minOccurs="0"/>
    <xs:element name="availabilityRate" type="xs:double"
minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Provider trust defines the trustworthiness properties that are related to the Service Provider. It may include *recommendations* from other *clients* and *organizations*, and *lowest prices guarantees*. There is no agreed upon definition for Provider trust. The main issue here is the inclusion of verifiable information that makes a seller trusted. Below is the XML schema for defining client recommendations and price guarantees in CSDL.

```
<xs:complexType name="ClientRecommendation">
  <xs:sequence>
    <xs:element name="Client" type="xs:string"/>
    <xs:element name="Recommendation" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="OrganizationalRecommendation">
  <xs:sequence>
    <xs:element name="Organization" type="xs:string"/>
    <xs:element name="Recommendation" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="PriceGuarantee">
  <xs:sequence>
    <xs:element name="Price" type="Price"/>
    <xs:element name="Guarantee" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Legal Issues: One of the essential elements of the ConfiguredService contract is the set of legal rules that constrain the contract. *Business rules*, such as *refund conditions*, *interest* and *administrative charges*, and *payment rules*, form one part of legal issues. Another part is the set of *trade law*s enforced in the context of service request and delivery. Examples of the later kind are service requester's *rights*, *privacy laws*, and *censor rules*. Below is the XML schema for defining payment rules using CSDL.

```
<xs:complexType name="PaymentRules">
  <xs:sequence>
    <xs:element name="PaymentTime" type="PaymentTime"/>
    <xs:element name="PaymentMethod"
    type="PaymentMethod" maxOccurs="unbounded"/>
    <xs:element name="PaymentDiscount"
    type="PaymentDiscount" minOccurs="0"/>
    <xs:element name="PaymentMethodFee"
    type="PaymentMethodFee" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3.2 Context

The context part of the ConfiguredService will include the main parts; *ConfiguredService context* and *context rules*. The ConfiguredService context defines the contextual information of the ConfiguredService. Context is formally defined in [8] using *dimensions* and *tags* along the dimensions. We illustrate context specification using the three dimensions WHERE, WHEN and WHO. The dimension WHERE is associated with a location, which may be one or more of *{Point, Region, Address, Route, URI, IP}*. The dimension WHEN is associated with temporal information, such as *time* and *date*. The dimension WHO is associated with subject identities, such as the *names* of Service Providers and Service Requesters. We can also use WHO dimension to associate information from *job roles*. The context rules define the contextual information related to the Service Requester that should be true for the Service Provider to guarantee the contract associated with the ConfiguredService. Rules are defined as logical expressions within the first order predicate logic (FOPL). Below is the XML schema for defining an address using CSDL.

```
<xs:complexType name="Address">
   <xs:sequence>
      <xs:element name="StreetAddress" type="xs:string"
      minOccurs="0"/>
      <xs:element name="Unit" type="xs:string"
      minOccurs="0"/>
      <xs:element name="PostalCode" type="xs:string"
      minOccurs="0"/>
      <xs:element name="Region" type="Region" minOccurs="0"
      maxOccurs="unbounded"/>
      <xs:element name="PhoneNumber" minOccurs="0"
      maxOccurs="unbounded">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Number" type="xs:string"/>
               <xs:element name="Ext" minOccurs="0"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

4 CSQL

The *ConfiguredService Query Language CSQL* is an XML based language used for the specification of service requester requirements. The meta-model of CSQL is shown in Figure 3. The Service Query consists of the four main parts *required function*, *required legal issues*, *required nonfunctional* properties and requester and consumer *context*. Each element is specified by the service requester. Also, a service requester assigns a weight to each requirement. This weight defines the priority of each requirement and is used in ranking the set of candidate ConfiguredServices when *matching* is performed.

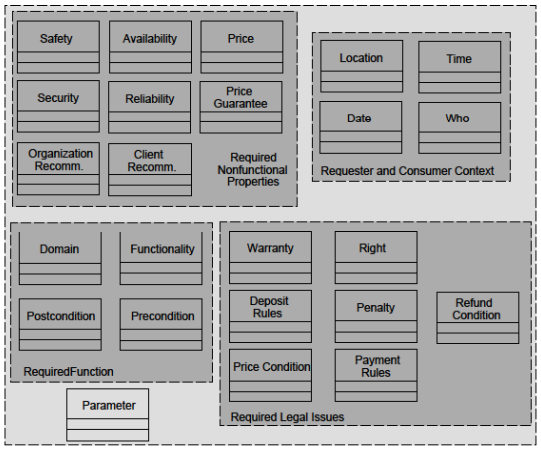

Figure 3. CSQL Meta-model.

The required functional properties defines the functionality required by the service requester and is defined in terms of the *domain*, *functionality* name, *preconditions* and *postconditions*. Below is the XML schema for specifying the required function in CSQL.

```
<xs:element name="RequiredFunction">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Precondition" minOccurs="0"
         maxOccurs="unbounded">
            <xs:complexType>
               <xs:complexContent>
                  <xs:extension base="Precondition">
                     <xs:sequence>
                     <xs:element name="weight" type="xs:int"/>
```

```
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Postcondition" minOccurs="0"
    maxOccurs="unbounded">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="Postcondition">
                    <xs:sequence>
                    <xs:element name="weight" type="xs:int"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Functionality" type="xs:string"/>
    <xs:element name="Domain" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

The required nonfunctional properties define the nonfunctional properties required by the service requester. The definition of the nonfunctional properties in CSQL is identical to the definition of the nonfunctional properties in CSDL. The only exception is the addition of the *weights*. The required legal rules are the legal rules required by the service requester. Its definition is also identical to the definition in CSDL with the addition of the *weights*. The contextual information of the service requester and consumer also use the same definition of the contextual information in CSDL.

5 AUTOMOTIVE CASE STUDY

In this section, we illustrate the use of CSDL and CSQL on a case study chosen from the *automotive industry*. This case study has been used in the literature of SOA by several researchers [9] and [10].

Vehicles today are equipped with multiple sensors and actuators that provide the driver with services that assist in driving the vehicle more safely, such as vehicle stabilization systems. Here we will focus on the road assistance scenario. This scenario deals with the case of a car failure. For example, the vehicles oil lamp indicates low oil level. This will trigger the diagnostic system to analysis the values obtained by the oil level sensor. The diagnostic system then reports, for example, the failure in one cylinder head and the car is no longer drivable. This information and the location information obtained from the GPS system is send to the road assistance center. The road assistance center will use this information to identify the appropriate: repair shop (garage), tow truck and car rental. After that, the driver makes an appointment with the repair shop, the results of the diagnosis are automatically sent to the repair shop, which will allow the garage to identify the spare parts needed to repair the car. When the driver orders a tow truck and a rental car, the GPS coordinates of the vehicle and repair shop are sent along. The driver is required to deposit a security payment before being able to order any service. Each service can be denied or canceled, causing an appropriate compensation activity.

Here, we are focusing on *single* service request and response, and not considering composition. We will assume that the requests are done sequentially and not simultaneously. First a request for repair shop is performed, and then the driver makes an appointment with this repair shop. Second, a request for a tow truck is performed, and then the driver calls the tow truck company. Finally, a request for a car rental is performed, and the driver calls the car rental company. We focus here on illustrating the specification of the ConfiguredServices providing repair shops using CSDL and the specification of the query for the repair shop using CSQL.

ConfiguredService	Function			NonFunctional	Legal	ContextRule	ContextInfo
RepairShop1	**Name**:ReserveRS **Pre**:CarBroken==T **Post**:HasAppointment==T **Address**: XXX	**InputParameters**: CarBroken:bool deposit:double CarType:string failureType:string	**ResultName**: ResultRS **OutputParameters**: HasAppointment:bool numberOfHours:int	Price = 60$/h Client Rec. = 5	deposit = 300$ Warranty= 3 **PriceCondition**: CarType= toyota	membership ==CAA	(location,montreal)
RepairShop2	**Name**:ReserveRS **Pre**:CarBroken==T **Post**:HasAppointment==T **Address**: XXX	**InputParameters**: CarBroken:bool deposit:double CarType:string failureType:string	**ResultName**: ResultRS **OutputParameters**: HasAppointment:bool numberOfHours:int	Price = 50$/h Client Rec. = 4 Recommended by CAA	deposit = 400$ Warranty= 2 **PriceCondition**: CarType= toyota	membership ==CAA	(location,montreal)
RepairShop3	**Name**:ReserveRS **Pre**:CarBroken==T **Post**:HasAppointment==T **Address**: XXX	**InputParameters**: CarBroken:bool deposit:double CarType:string failureType:string	**ResultName**: ResultRS **OutputParameters**: HasAppointment:bool numberOfHours:int	Price = 40$/h Client Rec. = 3 Recommended by CAA	deposit = 500$ Warranty= 1 **PriceCondition**: CarType= toyota	membership ==CAA	(location,montreal)
RepairShop4	**Name**:ReserveRS **Pre**:CarBroken==T **Post**:HasAppointment==T **Address**: XXX	**InputParameters**: CarBroken:bool deposit:double CarType:string failureType:string	**ResultName**: ResultRS **OutputParameters**: HasAppointment:bool numberOfHours:int	Price = 70$/h Client Rec. = 5 Recommended by CAA	deposit = 300$ Warranty= 4 **PriceCondition**: CarType= toyota	membership ==CAA	(location,montreal)
RepairShop5	**Name**:ReserveRS **Pre**:CarBroken==T **Post**:HasAppointment==T **Address**: XXX	**InputParameters**: CarBroken:bool deposit:double CarType:string failureType:string	**ResultName**: ResultRS **OutputParameters**: HasAppointment:bool numberOfHours:int	Price = 40$/h Client Rec. = 5 Recommended by CAA	deposit = 250$ Warranty= 4 **PriceCondition**: CarType= toyota	membership ==CAA	(location,montreal)

Figure 4. ConfiguredServices Details.

Repair shops Service Providers access the Registry to publish their services. They search for the appropriate domain until they find the Repair shop domain. Under this domain they search for the appropriate functionality which is in this case Reserve. Then, they will verify that their parameters are defined under the Reserve functionality. Next, Service Providers will publish the ConfiguredServices.

The service registry contains 5 ConfiguredServices that provide the repair shop functionality. The details of these ConfiguredServices are shown in Figure 4.

The 5 ConfiguredServices were specified using CSDL. Due to space limitation, we will only state sample parts of the CSDL specification of the ConfiguredService Repair-Shop1. We will first start by the CSDL specification of the contract elements function, nonfunctional and legal issues.

```
<Function>
 <Signature>
       <ID>ReserveRS</ID>
       <Address>XXX.XXX</Address>
       <Parameter>
          <Name>CarBroken</Name>
          <DataType>bool</DataType>
       </Parameter>
       <Parameter>
          <Name>deposit</Name>
          <DataType>double</DataType>
       </Parameter>
       <Parameter>
          <Name>CarType</Name>
          <DataType>string</DataType>
       </Parameter>
       <Parameter>
          <Name>failureType</Name>
          <DataType>string</DataType>
       </Parameter>
    </Signature>
    <Return>
       <ID>ResultRS</ID>
       <Parameter>
          <Name>HasAppointment</Name>
          <DataType>bool</DataType>
       </Parameter>
       <Parameter>
          <Name>numberOfHours</Name>
          <DataType>int</DataType>
       </Parameter>
    </Return>
    <Precondition>
       <Condition>CarBroken==T</Condition>
    </Precondition>
    <Postcondition>
       <Condition>HasAppointment==T</Condition>
    </Postcondition>
</Function>
<NonFunctional>
    <Price>
       <value>60</value>
       <currency>dollar</currency>
       <unit>hour</unit>
    </Price>
    <ProviderTrust>
       <ClientRecommendation>
          <Client>ClientX</Client>
          <Recommendation>4</Recommendation>
       </ClientRecommendation>
       <ClientRecommendation>
          <Client>ClientY</Client>
          <Recommendation>6</Recommendation>
       </ClientRecommendation>
    </ProviderTrust>
```

```
</NonFunctional>
<LegalIssue>
   <PriceCondition>
      <Price>
         <value>60</value>
         <currecny>dollar</currecny>
         <unit>hour</unit>
      </Price>
      <Condition>CarType==toyota</Condition>
   </PriceCondition>
   <DepositRule>
      <Amount>300</Amount>
      <Currency>dollar</Currency>
      <Rule>On Time</Rule>
      <Date>2011-07-30</Date>
      <Time>00:00:00</Time>
   </DepositRule>
<Warranty>
      <Duration>3</Duration>
      <Condition>No Condition</Condition>
   </Warranty>
</LegalIssue>
```

Below is the CSDL specification of the Repair-Shop1 ConfiguredService context part.

```
<Context>
   <ContextInfo>
      <Location>
         <Region>
            <Type>City</Type>
            <Name>Montreal</Name>
         </Region>
      /<Location>
   </ContextInfo>
   <RequesterContextRules>
      <WhoRequester>
         <Membership>CAA</Membership>
      </WhoRequester>
   </RequesterContextRules>
</Context>
```

The Service Requester, in this case the Vehicle, accesses the Service Registry to find the domain and functionality, in this case Repair shop and Reserve. The Service Requester will then access the functionality parameters and will use them in defining the Service Query. Figure 5 shows the Service Query.

Service Query	*Function*	*NonFunctional*	*Legal*	*ContextInfo*
Repair Shop Query	**RequiredPre**: CarBroken==T Weight = 6 **RequiredPost**: HasAppointment==T Weight =6 **Parameters**: CarBroken:bool deposit.double CarType:string failureType:string	Price = 45$/h Weight = 3 Client Rec. = 4 Weight = 3 Recommended by CAA Weight = 3	deposit = 280$ Weight = 4 Warranty= 3 Weight = 5 **PriceCondition**: CarType=toyota Weight = 6	membership ==CAA (location, montreal)

Figure 5. Service Query Details.

Next is the CSQL specification of this Query.

```
<Query-w>
   <RequiredFunction>
      <Precondition>
         <Condition>CarBroken==T</Condition>
         <weight>6</weight>
      </Precondition>
      <Postcondition>
         <Condition>HasAppointment==T</Condition>
         <weight>6</weight>
      </Postcondition>
      <Functionality>Reserve</Functionality>
      <Domain>RepairShop</Domain>
   </RequiredFunction>
   <RequiredNonFunctional>
      <Price>
         <value>45</value>
         <currecny>dollar</currecny>
         <unit>hour</unit>
         <weight>3</weight>
      </Price>
      <ProviderTrust>
         <ClientRecommendation>
            <weight>3</weight>
            <value>4</value>
         </ClientRecommendation>
      </ProviderTrust>
   </RequiredNonFunctional>
   <RequiredLegalIssue>
      <PriceCondition>
         <Price>
            <value>60</value>
            <currecny>dollar</currecny>
            <unit>hour</unit>
         </Price>
         <Condition>CarType==toyota</Condition>
      </PriceCondition>
      <DepositRule>
         <Amount>280</Amount>
         <Currency>dollar</Currency>
         <Rule>NoRule</Rule>
         <Date>2011-07-30</Date>
         <Time>00:00:00</Time>
      </DepositRule>
      <Warranty>
         <Duration>3</Duration>
         <Condition>NoCondition</Condition>
      </Warranty>
      <Rights>String</Rights>
      <weight>6</weight>
      <weight>4</weight>
      <weight>5</weight>
   </RequiredLegalIssue>
   <RequesterContext>
      <WhoRequester>
         <Membership>CAA</Membership>
      </WhoRequester>
   </RequesterContext>
   <AuthenticationCertificate>Certificate
    Type1</AuthenticationCertificate>
</Query-w>
```

The Service Query is send to the Planning Unit. The Planning Unit will then send service lookups to the Service Registry. The lookups result will be matched with the Service Query requirements by the Planning Unit. The Planning Unit will also rank the matching result. The matching and

ranking result of the case study is shown below:

- ConfiguredService RepairShop5 is matched by 100.0%
- ConfiguredService RepairShop4 is matched by 88.19%
- ConfiguredService RepairShop1 is matched by 83.33%
- ConfiguredService RepairShop2 is matched by 80.56%
- ConfiguredService RepairShop3 is matched by 78.12%

6 IMPLEMENTATION

A Java based application has been implemented to represent the Planning Unit. This application takes as input the set of ConfiguredService that provide a specific functionality as returned from the service registry, and the service query. The application will then match between the service query and the candidate ConfiguredServices taking into consideration the functional, nonfunctional, legal, and contextual information. Two types of matching have been implemented:

1) exact match and,
2) weighted match.

The ranking algorithm has also been implemented.

The application was tested on a standard PC using an Intel Centrino processor with 4GB of memory and running Windows 7 Professional. The testing was on multiple case studies including an extended version of the case study represented in the previous section. The average matching and ranking time was in milliseconds for each ConfiguredService which eliminate the concerns of scalability issues.

7 RELATED WORK

Related work can be divided into related provision frameworks, and related service discovery and ranking approaches. Related frameworks, such as eFlow [11], SELFSERV [12] and SWORD [13] do not provide support for including contextual information. On the other hand, frameworks such as SeGSeC [14], SHOP2 [15] and Argos [16] do provide some support to include contextual information but context is not formally represented and the relationship between the contract and context is never considered. To our knowledge, no published framework supports all the features of FrSeC presented in Section 2. They do not support the formal specification of legal rules and contextual information, and the relationship between context and contract.

Related work such as [17] and [18], on discovery and ranking use only functionality to enable the discovery and ranking of services. Approaches, such as [19], [20], [21] and [22] use nonfunctional properties to enhance service discovery and ranking**.** *However, none of the available approaches* use collectively functional, nonfunctional, legal, and contextual information in service discovery and ranking. In our work we do it, and we use formalism selectively in the different stages. This effort has led to the formal verification of the matching result and the satisfaction of the contractual and contextual obligations. No other approach does that.

8 CONCLUSIONS AND FUTURE WORK

We have presented FrSeC that supports the publication, discovery, and provision of services with context-dependent contracts. FrSeC is formally based and considers legal rules and contextual conditions during service provision. It also supports an adaptive re-discovery and re-ranking operations. We are currently working on a complete implementation of FrSeC and associated tools.

REFERENCES

1. Papazoglou, M.P.: Web Services: Principles and Technology. first edn. Prentice Hall (2008)
2. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1) (2001) 4–7
3. OSullivan, J.: Towards a Precise Understanding of Service Properties. Phd thesis, Queensland University of Technology, Brisbane, Australia (September 2007)
4. Ibrahim, N., Mohammad, M., Alagar, V.: Managing and delivering trustworthy context dependent services. In: Proceeding 8th IEEE International Conference on e-Business Engineering (accepted), IEEE Computer Society (October 2011)
5. Ibrahim, N., Alagar, V., Mohammad, M.: Specification and verification of context-dependent services. In: Proceeding of the 7th Int'l Workshop on Automated Specification and Verification of Web Systems (WWV2011), Reykjavik, Iceland (June 2011)
6. Ibrahim, N., Mohammad, M., Alagar, V.: An architecture for managing and delivering trustworthy context-dependent services. In: Proceeding of the 8th IEEE International Conference on Services Computing, Washington, DC, USA, IEEE Computer Society (July 2011)
7. Mohammad, M., Alagar, V.: A formal approach for the specification and verification of trustworthy component-based systems. J. Syst. Softw. 84(1) (2011) 77 – 104
8. Wan, K.: Lucx: Lucid Enriched with Context. Phd thesis, Concordia University, Montreal, Canada (January 2006)
9. ter Beek, M.H., Gnesi, S., Koch, N., Mazzanti, F.: Formal verification of an automotive scenario in service-oriented computing. In: Proceedings of the 30th international conference on Software engineering. ICSE '08, New York, NY, USA, ACM (2008) 613–622
10. Berndl, D., Koch, N.: Sensoria automotive scenario: Illustrating service specification. Technical report, - FAST, No. 2 (August 2007)
11. Casati, F., Ilnicki, S., Jin, L.j., Krishnamoorthy, V., Shan, M.C.: Adaptive and dynamic service composition in eflow. In: Proceedings of the 12th Int'l Conference on Advanced Info. Systems Engineering, Springer-Verlag (2000) 13–31
12. Sheng, Q.Z., Benatallah, B., Dumas, M., Mak, E.O.Y.: Self-serv: a platform for rapid composition of web services in a peer-to-peer environment. In: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment (2002) 1051–1054
13. Ponnekanti, S.R., Fox, A.: Sword: A developer toolkit for web service composition. In: Proceedings of the 11th International WWW Conference. (2002)
14. Fujii, K., Suda, T.: Semantics-based context-aware dynamic service composition. ACM Trans. on Autonomous and Adaptive Systems 4(2) (2009) 1–31Hyvarinen, A., Oja, E.: Independent Component Analysis: Algorithms and Applications. Neural Networks 13, 411--430 (2000).
15. Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D., Nau, D.: Automating daml-s web services composition using shop2. In: Proceedings of 2nd International Semantic Web Conference. (2003)
16. Ambite, J.L., Weathers, M.: Automatic composition of aggregation workflows for transportation modeling. In: Proceedings of the 2005 national conference on Digital government research, Digital Government Society of North America (2005) 41–49
17. Lu, H.: Semantic web services discovery and ranking. In: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence. WI '05, Washington,

DC, USA, IEEE Computer Society (2005) 157–160
18. Bellur, U., Vadodaria, H.: Web service ranking using semantic profile information. In: Proceedings of the 2009 IEEE International Conference on Web Services, Washington, DC, USA, IEEE Computer Society (2009) 872–879
19. Al-Masri, E., Mahmoud, Q.H.: QoS-based discovery and ranking of web services. In: Proceedings of the 16th International Conference on Computer Communications and Networks. (2007) 529–534
20. Guo, L.y., Chen, H.p., Yang, G., Fei, R.y.: A QoS evaluation algorithm for web service ranking based on artificial neural network. In: Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02, IEEE Computer Society (2008) 381–384
21. Rajendran, T., Balasubramanie, P.: An optimal agent-based architecture for dynamic web service discovery with qos. In: Proceedings of the Second IEEE International Conference on Computing Communication and Networking Technologies. (july 2010) 1 –7
22. Yan, J., Piao, J.: Towards qos-based web services discovery. In Feuerlicht, G., Lamersdorf, W., eds.: Service-Oriented Computing-ICSOC 2008 Workshops. Volume 5472 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2009) 200–210