



WordNet-based Approach Towards a Better Comprehension between Providers and Clients

Kaouthar FAKHFAKH^{1,2,3}, Tarak CHAARI³, Said TAZI^{1,2}, Khalil DRIRA^{1,2}, Mohamed JMAIEL³

¹CNRS; LAAS; 7 avenue du colonel Roche, F31077 Toulouse, France

²Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F31077 Toulouse, France

³National Engineering School of Sfax. ENIS

Informatics and applied Mathematics Department; ReDCAD,

Route de la Soukra, B.P. W, 3038 Sfax, Tunisia.

{kchaari,tazi,khalil}@laas.fr

tarak.chaari@redcad.org

mohamed.jmaiel@enis.rnu.tn

Abstract - SLAs (Service Level Agreements) are commonly signed agreements forming the contract between a service provider and a service consumer defining the obligations of the two parties. The client has often predefined choices by the provider. Consequently, he may not understand these choices as he can be a non specialist and he cannot express freely his requirements using his own knowledge and language. The goals of this work are (1) offering opportunities to the client to express his needs freely and (2) facilitating the analysis of these needs and the contract generation to the provider. This paper proposes the necessary models and algorithms (1) to capture the client intention and the provider offers, (2) to perform a complete matching process between them to test if they are compatible and (3) to automatically generate a draft version of an SLA to the service provider for its verification and proposal to the client.

I. INTRODUCTION

In Service Oriented Computing paradigm, an SLA (Service Level Agreements) is defined as a binding contract which formally specifies user expectations about the required services. SLA is a collection of service level requirements that have been negotiated and mutually agreed upon by the providers and the clients. Usually, providers define some service levels as a fixed combination of their specific capabilities on a set of quality dimensions, and users must choose one of these levels. Besides, contracts have become increasingly complex and detailed with terms and conditions, service level requirements, multi-tiered pricing, and similar critical variables. These complex parameters make their comprehension difficult to the client. They don't have tools to freely express their service level requirements by their simple knowledge and language. Moreover, providers need comprehensive tools to understand the

client intentions and to automatically generate the corresponding service level agreement.

Our objective is to bridge the gap between the client's intention and the provider offers to facilitate the contract establishment. In this paper, we present high level semantic enabled models to facilitate the expression of client intentions and provider offers. Then, we propose matching algorithms between these models in order to automatically generate a service level agreement in case of compatibility between client intentions and provider offers. Our models are based on ontologies offering rich semantics to the given terms by the clients and the providers. This helped us developing our matching algorithms that are fully based on inference rules on the given knowledge in the ontologies. These latter provide a formal, syntactic, and semantic description model of concepts, properties and relationships between concepts. Moreover, they provide comprehensive means to infer new knowledge and take decisions according to a given asserted data.

In this work, we present a Digital Video Library example. We assume that a client (who is not an expert in IT field), wants to download films with a download time less than ten minutes and a maximum price equal to three euros per film. This client wants to freely express his requirements and to send it to the provider. Consequently, the problems to be tackled are: how to check and satisfy the client constraints although the client requirements and the provider offers can be expressed differently. For example, the provider can define offers based on throughputs for downloading films and the client has expressed its intention in download time.

II. RELATED WORK

Automating the contract negotiation belongs to research issues pursued by multiple European projects. The OntoGrid project takes an agent-based approach, bringing Grid and agent technologies together in order to enable efficient provisioning and management of services [1]. In OntoGrid framework, Rubinstein's alternating offers protocol is adapted for web services negotiation XML schemas expressing offers and actions (e.g. agree, reject) have been developed, together with a set of negotiation strategies. NextGrid project publications (e.g. [2]) point out the importance of Service Level Agreement negotiation for e-business and foresee the adoption of Semantic Web and agent technologies in this area. Currently, they propose a brokered approach, where the negotiation service is outsourced to an intermediary third party, without any semantic support. Finally, BREIN (Business objective driven Reliable and Intelligent grids for real business, [3], also aims to develop a business-oriented which also covers the contracts' negotiation issue. To recapitulate, all mentioned projects acknowledge the importance of Service Level Agreements and view semantic technologies as a mean of automating SLA development and negotiation [4]. The semantics of a contract and its negotiation should be machine-understandable and this is where ontologies take their real interests. Enhancements to WS-Agreement [5], which makes extensive use of ontology engineering and reasoning to automate the SLA establishment, are presented in [6]. Also, several Quality of Service ontologies have been proposed lately, e.g. [7], to facilitate precise specification of a contract between negotiating parties. There is also an initiative of creating a unified QoS ontology (under the name of OWL-QoS, formerly DAML-QoS) [8].

However, while these approaches are still evolving, they present some limitations. Generally, frameworks for SLA management only specify predefined protocol by the provider for the negotiation based on "take it or leave it" principle. In major cases, the clients are not specialist and don't understand the provider offers. Our objective is to bridge the gap between the client's intention and the provider offer to give the opportunity to the client to freely express himself and to facilitate the contract establishment between the parties. In the next section, we present our solution in this domain.

III. SEMANTIC ENABLED APPROACH FOR MATCHING CLIENT INTENTIONS WITH PROVIDER OFFERS

In this section, we present the models that we developed to perform the matching process between the client intention and the provider offers. We defined the ClientOnto ontology to describe the high level model of

the client intention and the ProviderOnto ontology to describe the provider offer. We used the protégé editor [9] to create these ontologies.

A. ClientOnto: High Level Model of Client Intention

The intention corresponds to a mental state of any actor who executes an action. Several works attempted to account the relations between an action undertaken by a human being and the mental state which guides this action. To be able to use the notion of the intention, it is necessary to describe it using a formal language. In [10], they represent an intention as:

$I(A, G^*, M^*, R^*)$ Where:

- 'I' represents the intention carried out by an action A;
- A is the action that expresses what the author of the intention wants to do;
- G represents the goal to achieve by performing the action;
- M represents the means to express how the action is accomplished;
- R represents the reason to express why the author chooses this action and for which reasons,

*indicate that the number of the acts composing the intention can be 0 or N.

We adapted and enriched this intentional model by adding service level agreements negotiation elements. Figure 1 illustrates our extended intentional model for client requirements. A client has a goal. He wants to perform an action and to obtain an intended result. An action is defined by a subject to describe the searched product by the client according to constraints in order to express requirements and conditions. Each constraint is defined by a property, a comparison threshold and an operator. For example, the client property can correspond to the price. The threshold corresponds to two euros and the operator corresponds to the "inferior" operator.

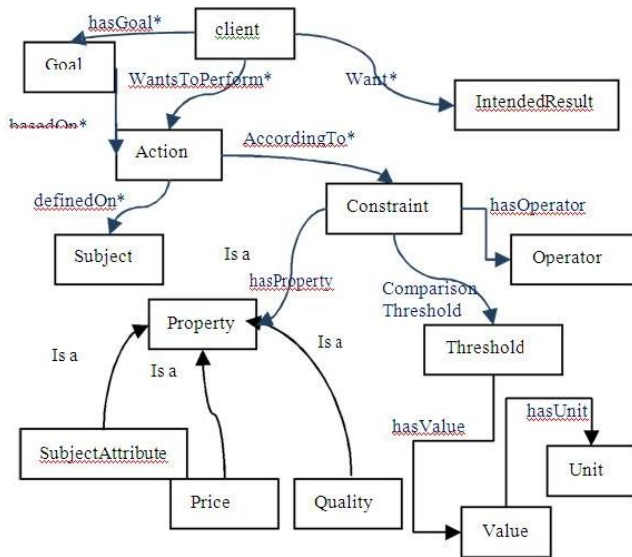


Fig 1. ClientOnto model

based language which allows specifying service levels by being attached to WSDL service descriptions. WSOL is suitable for the definition of quality dimensions, their metrics and quality constraints. However, the extraction of information is difficult to achieve given its complexity. WSOL does not help analyzing constraints which are expressed as strings. Consequently, WSOL does not help in evaluating provider offers according to client constraints. For these reasons, we have developed a new semantically richer model as shown in Figure 2 to allow providers expressing their offers in order to compare them with client intentions. The root concept of this model is the Provider concept which identifies any provider. Each provider has offers which are defined by constraints, guarantees and products. Each constraint is defined by a property (which can be a quality or a product feature or a price), a threshold and an operator. The property and the threshold are defined as variables. Each variable has a name, a type, a measurement unit and a value. The constraints defined by the provider can be functional constraints, non functional constraints, price constraints or access right constraints.

B. ProviderOnto: Business and Commercial Provider Model

In the literature, we noticed that there is a description language for provider offers named WSOL (Web Services Offerings Language) [11]. WSOL is an XML

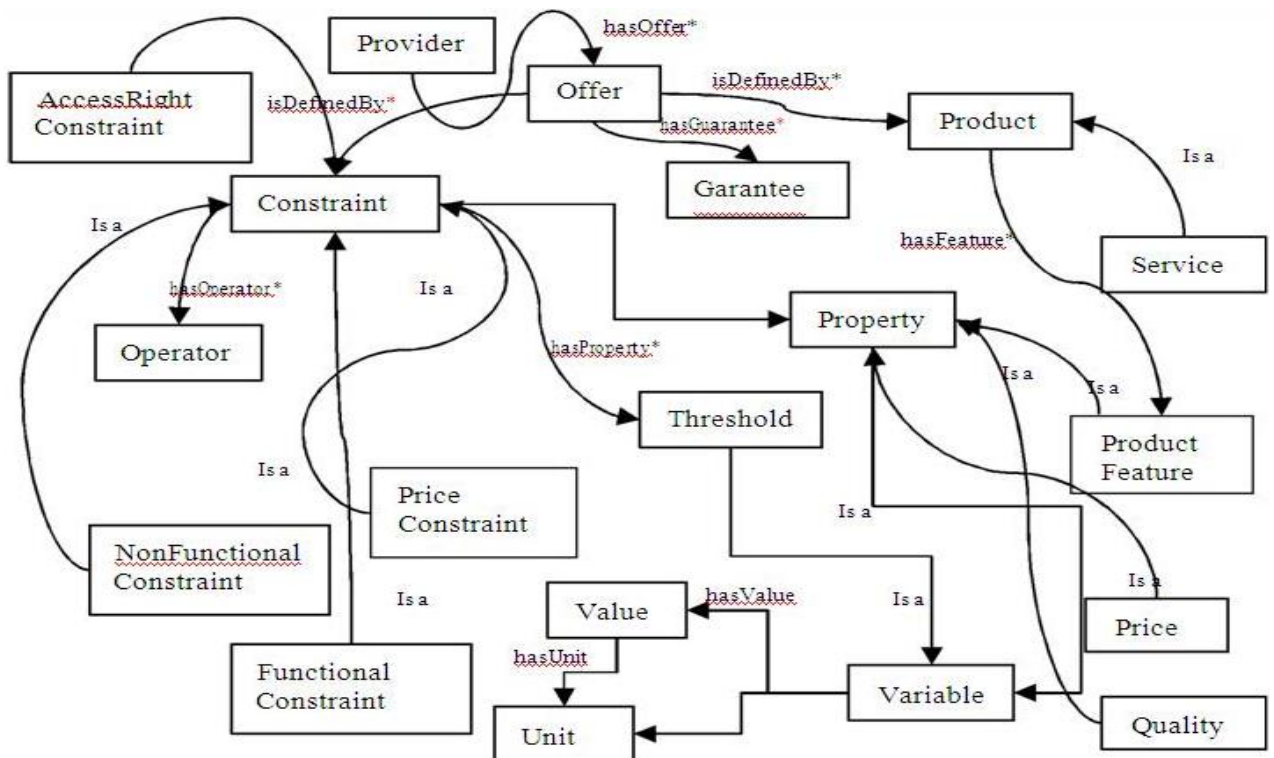


Fig 2. ProviderOnto: our provider offer model

C. General Overview of our Matching Approach

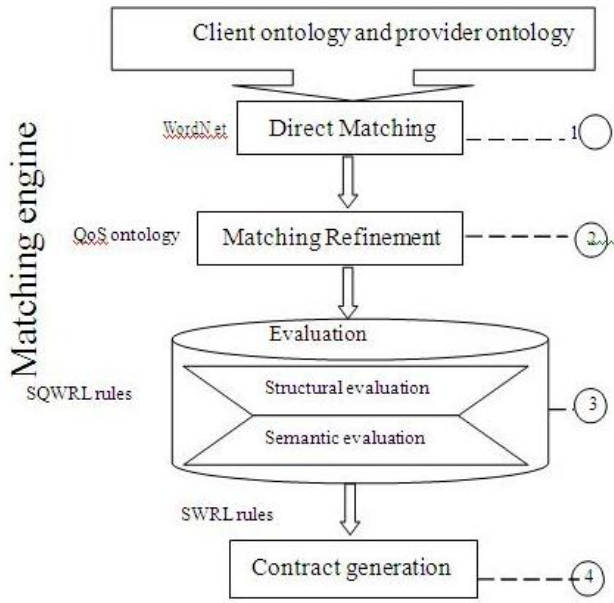


Fig 3. The matching process of contract generation

After developing the client intentional model and the provider offer model, we present the overview of our matching approach between these two models as shown in Figure 3. Our matching approach is composed of four main steps. The first step consists in searching the possible correspondences between the two ontologies using WordNet relations. The second step consists in refining these correspondences using another ontology (that we called QoS ontology) to calculate the QoS properties values required by the client. We detail this ontology in section 3.5. The third step is a verification process that ensures the correspondence evaluation of our matching process. This step is composed of two types of evaluation: (1) a structural evaluation to test if a sufficient amount of the terms of the client intention have correspondences with the provider offers terms and (2) a semantic evaluation to test if the client constraints could be satisfied by the provider offers. In the last step, we finish by generating a first version of a contract in case of compatibility. In the next section, we give more details about each step.

D. Matching the Client's Intention with the Business Provider Model

In this section, we present a matching algorithm (as shown in Listing 1) between the client ontology terms and the provider ontology terms using Wordnet1 to search the possible semantic relations between them. WordNet is an English large lexical database developed under the direction of George A. Miller. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive

synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual, semantic and lexical relations. WordNet also includes a set of relationships such as *wn: similarTo*, *wn: hypernym*, *wn: hyponym*. We used these relations to create semantic correspondences between the terms of the two ontologies.

The principle of our algorithm is presented in Listing 1. Once we detect a WordNet relationship between a client instance and a provider instance, we create a *correspondsTo* association between these two terms. A client instance can have several semantic relations with provider instances. These instances can be expressions composed of several terms. These expressions can be nominal phrases or verbal phrases. For verbal phrases we assume that the most significant term is the verb. For the nominal phrases it is the last term in the expression. These assumptions should be more explored and analyzed in future works.

Listing 1. Matching algorithm between client terms and provider terms

```

 $R_{wn} = \{wn.similarTo, wn.hyponym, wn.hypernym, wn.glossary\}$ 
 $C_{matching} := C_{Client} \cup C_{Provider}$ 
 $R_{matching} := R_{Client} \cup R_{Provider}$ 
 $instanceCount := 0$ 
 $wnMatchingCount := 0$ 
For each  $C_{Client}^{ij} \in C_{Client}$  do
  For each  $C_{Client}^{ij}$  instanceOf  $C_{Client}^i$  do
     $instanceCount++$ 
    If  $\exists R \in R_{wn} \wedge C_{Provider}^{lk} \in C_{Provider} \setminus R =$ 
       $(C_{Client}^{ij}, C_{Provider}^{lk})$  OR  $R = C_{Provider}^{lk}, C_{Client}^{ij}$ 
    then
       $R_{matching} = R_{matching} \cup R;$ 
       $wnMatchingCount++;$ 
       $isMarked(C_{Client}^{ij}, "wn");$ 
       $isMarked(C_{Provider}^{lk}, "wn");$ 
    end if
  end if

```

The established semantic relations are the WordNet relations. We proceed to this process until we complete all the client instances. We mark each term involved in a matching relation by the 'wn' label to facilitate the matching analysis in the next steps.

E. Matching Refinement Algorithm with QoS Ontology

After developing the matching algorithm, we have noted that matching ontologies using WordNet is not sufficient in the case where the client constraints are linked by mathematical relations and not linguistic ones. For example, the provider has a throughput offer for downloading films and the client has expressed his intention in download time. To address this problem, we propose to find indirect relations between the client

constraints and the provider offers through the QoS ontology. In the literature, we find some existing QoS ontologies like OWL-QoS [12], QoSOnt [13], SL-ontology [14], WS-QoS [15], FIPA QoS [16], MOQ [17]. However, these models do not express the semantic dependency between their metrics where they are composite. Consequently, we can't determine the value of the composite metric. For this reason, we propose a QoS ontology as shown in Figure 4. It defines a set of metrics modeled by the QoSMetric concept. Each metric is sub concept of a variable having a name, a type and a measurement unit. A variable can have a value which also has an instance and a unit. Each metric can depend on other parameters that can be other metrics or variables. These parameters constitute the input of a function to calculate the aggregate value of the composite metric. A QoSMetric has an attribute called BetterQualityDirection to define the better quality direction of the metric. It can be, downward, upward, or constant. This attribute is useful for further monitoring analysis to detect QoS degradations. Each parameter required to compute a composite metric has a range that identifies its position in the inputs of the function. The function can also have a result represented by a variable.

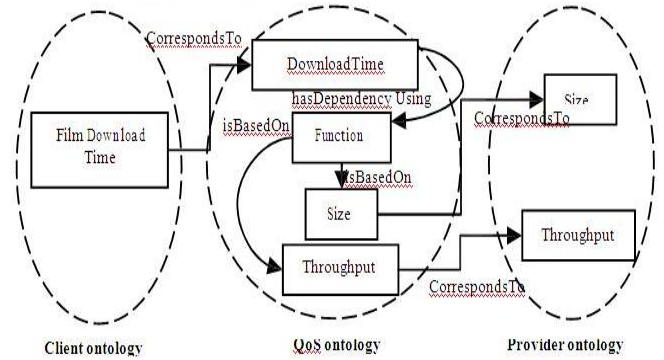


Fig 4. Indirect correspondence example with QoS ontology

We used our QoS ontology to develop a refinement algorithm to detect further indirect relations between the client ontology and the provider ontology. This algorithm (presented in Listing 2) checks if the client terms have WordNet relations with the quality of service ontology terms. Then we verify that these terms have WordNet relations with the provider instances. We keep only the terms that can be computed using elementary metrics given by the provider. Figure 5 shows an example of indirect correspondence using the QoS ontology. The film download time term has a correspondence with a download time term in QoS ontology. The corresponding value of this term depends on the film size and the throughput which both have correspondences with terms in the provider ontology.

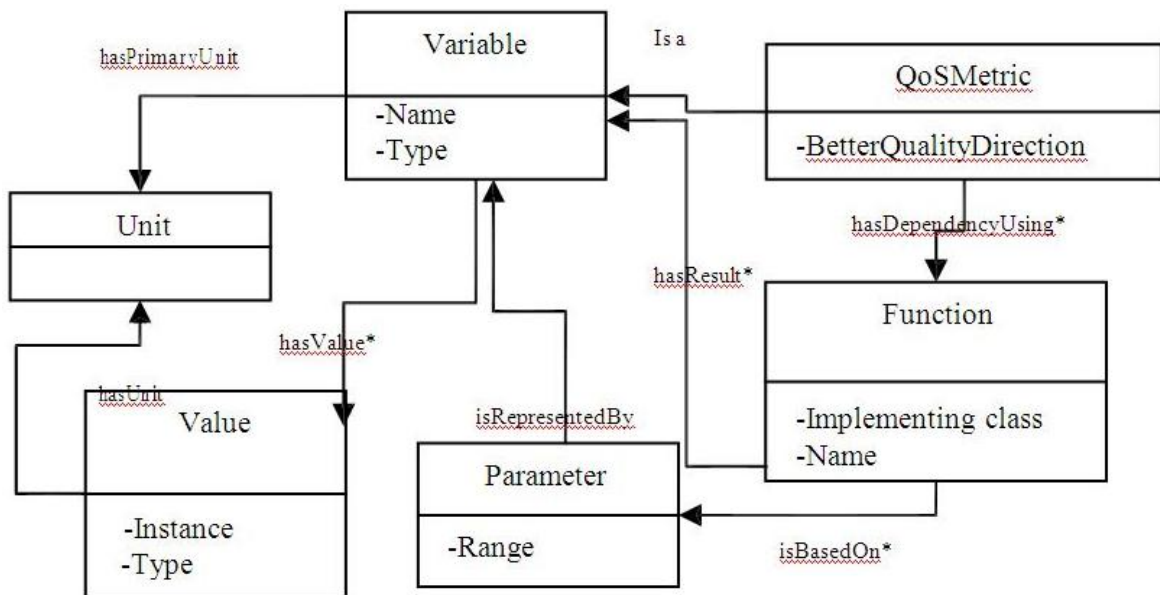


Fig 5. QoSOnto: our QoS ontology

Listing 2. Refinement algorithm

```

QoSMatchingCount := 0
For each  $C_{Client}^i \in C_{Client}$  do
For each  $C_{Client}^{ij}$  instanceOf  $C_{Client}^i$  do
If  $\exists R \in R_{wn} \wedge C_{QoS}^{u,v} \in C_{QoS} \setminus C_{QoS}^{u,v}$  instanceOf  $QoSMetric$  or  $C_{QoS}^{u,v}$ 
instanceOf  $Variable$   $\wedge R = (C_{Client}^{ij}, C_{QoS}^{u,v})$  OR  $R = (C_{QoS}^{u,v}, C_{Client}^{ij})$  then
     $R_{matching} = R_{matching} \cup R$ 
     $C_{matching} = C_{matching} \cup \{ C_{QoS}^{u,v}, C_{Client}^{ij} \}$ 
If ( $C_{QoS}^{u,v}$  instanceOf  $QoSMetric$ ) then
    getOperandsRule := "hasDependencyUsing (" +  $C_{QoS}^{u,v}$  + " , ?function) ^
isBasedOn(?function,?operand) → sqwrl:select(?operand)";
else
    getOperandsRule := "isBasedOn(?function, "+  $C_{QoS}^{u,v}$  + ") ^
isBasedOn(?function,?operand) → sqwrl:select(?operand)";
    end if
    operands := SWREngine.infer(getOperandsRule) ;
    linkedToProvider := true
    While  $\exists$  operand in operands and linkedToProvider
If ( $\exists R \in R_{wn} \wedge C_{Provider}^{l,k} \in C_{Provider} \setminus R = (C_{Provider}^{l,k}, operand)$  OR  $R =$ 
( $operand, C_{Provider}^{l,k}$ )) and ( $\exists R \in R_{wn} \wedge R_{Client}^{ij} \in C_{Client} \setminus R = (R_{Client}^{ij}$ 
operand) OR  $R = (operand, R_{Client}^{ij})$ ) then
    linkedToProvider := false
end if
    operands := operands / {operand}
end While
if (linkedToProvider == true)
isMarked( $C_{Client}^{ij}$ , "QoSToProvider");
QoSMatchingCount ++
    end if

```

F. Ontology Matching Evaluation

To ensure the evaluation of the direct and indirect matching results, we perform two types of evaluation: (1) the structural evaluation in which we check that we have a minimal number of correspondences between the client and the provider ontologies and (2) the semantic evaluation in which we check the semantic correspondence between these two ontologies according to the client's constraints.

- Structural Evaluation

The goal of the structural evaluation is to verify that the matching has produced a minimal number of correspondences between the two ontologies. For example, we consider that two ontologies are not compatible if the number of wordNet correspondences between their terms is inferior to a given threshold. This threshold is fixed by an expert depending on the selection criteria of the provider. The structural evaluation is not a real verification of compatibility between the client intention and the provider offers. It is just an incompatibility verification. Indeed, it is useless to drive more complex semantic evaluations if we are sure that the two ontologies are incompatible. In our testing example, we fixed this threshold to the half of the number of instances in the client ontology. We consider that if we

are below this threshold, the client is looking for something that the provider can't offer. Listing 3 shows an example of a structural evaluation. The goal of this evaluation is to look for the constraints correspondence between the two ontologies. The constraint concept is characterized by its operator, its property and its threshold as shown in Figure 6. For each client constraint, we consider its Property and Threshold and we try to get its corresponding value (or interval value) from the provider. If there are client constraint property or threshold without direct or indirect relation with the provider ontology, or if the client constraint is not verified by any provider constraint, then we affirm that there is a constraint semantic incompatibility between the two ontologies.

Listing 3. Structural evaluation example

```

if (wnMatchingCount + QoSMatchingCount < matchingCount/2) then
    prepareMatchingResult(Incompatibility)
end if /

```

- Constraints Semantic Evaluation

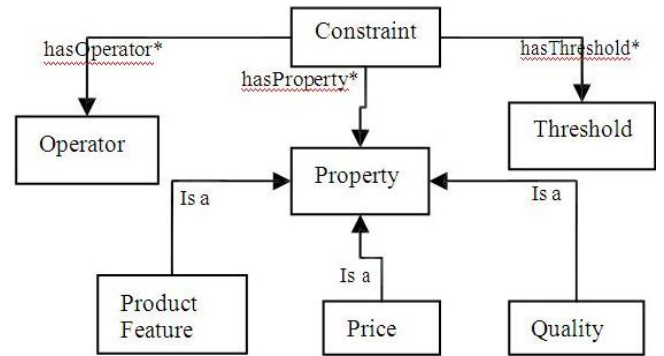


Fig 6. The constraint representation

To implement this constraint semantic evaluation, we have developed SQWRL (Semantic Query-Enhanced Web Rule Language named SQWRL) [18] rules which allow querying the client ontology constraints (see Listing 4). The rule named GetClientConstraints is considered to find all constraints and to put them in a vector named *ConstraintsVector*.

Listing 4. GetClientConstraints SQWRL rule

```

GetClientConstraints := "client:Constraint(?Client:constraint) → sqwrl:select
(?Client:constraint)"
ConstraintsVector := SWREngine.infer(GetClientConstraints) ;

```

After that, for each constraint, we defined a rule that looks for its property, its threshold and its operator to retrieve all its values (see Listing 5).

Listing 5. GetClientConstraintDetails SQWRL rule

```

For each constraint in ConstraintsVector do
  GetClientConstraintDetails:=hasProperty("+constraint+",?client:prop
erty)^hasOperator("+constraint+",?client:operator)^hasThreshold("+constr
aint+",?client:threshold)→sqwrl:select("?client:property, ?client:operator,
?client:threshold");
  (client:property, client:operator,
client:threshold):=SWRLEngine.infer(GetClientConstraintDetails)
end For

```

Then, we check if the property is marked 'wn' (i.e. there is a direct relation with WordNet). In this case, we retrieve its property values from the provider with a SQWRL request to compare it with the client operator and threshold values as shown in Listing 6. The retrieved values are added as instances of *propertyProviderValues* which is dynamically created for each client constraint property.

Listing 6. Extraction of property values in case of direct relation

```

checkConstraintsRule:= ""
If(∃R∈Rwn^provider:property∈Cprovider/
R=(client:property, provider:property))
directProviderValues:= "hasValue("+provider:property+",?provider:val
ue)→"+client:property+ "ProviderValues(?provider:value)"
SWRLEngine.infer(directProviderValues);
checkConstraintsRule+=client:property+"ProviderValues(?value)^chec
kConstraint(?value, "+client:operator +", "+client:threshold+ ")")
end if

```

If the property is marked "QoStoProvider" (i.e. there is an indirect relation with the provider), we extract its function and its operands like shown in Listing 7. For each operand, we retrieve the list of possible values either from the client or the provider or the QoS ontology and we store them in a data structure to calculate the SLA parameter value (Listing 8). Finally, we generate a SWRL rule to test all the client constraints simultaneously as shown in Listing 9. The result is compatibility or incompatibility between the client intention and the provider offers.

Listing 7. Extraction of operands in case of indirect relation

```

If (isMarked(client:property, "QoStoProvider")) then
  R4:= "Rwn(?client:property, ?QoS:Metric)^hasDependencyUsing(?QoS:Me
tric, ?Function) →sqwrl:select(?function)"
  function:= SWRLEngine.infer(R4)
  R5:=isBasedOn("+function+",?Operand)^hasRange(?operand,?operandRa
nge)^ →sqwrl:select(?Operand)"
  Operands:=SWRLEngine.infer (R5);
end if

```

Listing 8. Extraction of property values in case of indirect relation

```

calculateFunctionRule:= " calculateFunction("+function+", "
For each operand in Operands do
  calculateFunctionRule+= "?"+operand+"Value,"
  calculateFunctionRule:=operand+"Values(?"+operand+"value)^"+calc
ulateFunctionRule
  OperandValueRule:= ""
  If (∃ R ∈ Rwn^ CClientij ∈ CClienti \ R=(operand, CClienti)) then
    OperandValueRule:= "Rwn(?Operand, ? CClientij)^hasValue(?CClientij
, ?client:value) → "+Operand+"values(?client:value) "
  Else If (∃ R ∈ Rwn \ R=(operand, CProviderlk)) then
    OperandValueRule:= "Rwn(?Operand, ?CProviderlk)^hasValue(?CProviderlk, ?
provider:value) → "+Operand+"values(?provider:value) "
  else
    OperandValueRule:= "hasValue(?Operand, ?QoS:value) → "+Operand+
"values(?QoS:value) "
  end if
  SWRLEngine.infer(OperandValueRule)
end For
calculateFunctionRule+= "?result) → "+client:property+"ProviderValues(?
result)"
SWRLEngine.infer(calculateFunctionRule)

```

Listing 9. Check constraints algorithm

```

checkConstraintsRule+=client:property+"ProviderValues(?value)^checkCo
nstraint(?value, "+ client:operator +", "+client:threshold+ ")")
if (Not isLast(client:property, client:propertyVector)) then
  checkConstraintsRule+= ""^
else
  checkConstraintsRule+= " →Compatibility"
end For
Compatibility:= SWRLEngine.infer(checkConstraintsRule)
If (Compatibility) then
  GenerateSLA(OMatching, SLAOnt)
End if

```

G. SLA Draft Generation in Case of Compatibility

To generate the SLA that describes the agreement contract between the client and the provider we used the SLAOnt structure [20]. SLAOnt defines an ontology describing various concepts and properties needed in a quality of service contract. It's a semantically rich SLA model that we have developed in a previous work to automate SLA monitoring and to take concrete actions in case of violations. The SLA generation principle is based on the execution of SWRL rules to retrieve the necessary information from the client and the provider ontologies. The measurement directive instances, their associated protocols and their operations in the provider ontology will constitute respectively the measurement directive, the protocols and the operation values in SLA ontology. In case of direct correspondence between the client and the provider ontologies, the metrics definition will be extracted from the provider properties which have a valid correspondence (i.e. its certainty is above 0.8), with the client property (see Listing 10).

Listing 10. SLACreateSimpleMetric rule

```

:lient:Property(?property)  $\wedge$  Correspondance(?c)  $\wedge$  hasSourceTerm(?c,
?property)  $\wedge$  hasCertainty(?c, ?m)  $\wedge$  swrlb:greaterThan(?m, 0.6)  $\wedge$ 
hasDestinationTerm(?c, ?destination)  $\wedge$ 
provider:hasMeasurementDirective(?destination, ?measurement)  $\wedge$ 
provider:hasValue(?destination, ?value)  $\wedge$ 
provider:hasMeasurementFrequency(?destination, ?freq)  $\wedge$ 
provider:hasPrimaryUnit(?destination, ?unit)  $\wedge$ 
provider:hasTypeValue(?destination, ?type)  $\wedge$ 
provider:hasName(?destination, ?name)  $\wedge$  provider:hasUnit(?value,
?unitValue)  $\wedge$  provider:hasCorrespondValue(?value, ?corresValue)  $\wedge$ 
provider:hasTypeValue(?value, ?typeValue)  $\rightarrow$  query:select(?destination

```

property instance of the client ontology will constitute an SLA parameter in the SLA ontology.

In case of indirect correspondence, the metric definition will be extracted from the provider ontology which has a valid correspondence with a variable of the QoS ontology (see Listing 11). This variable must have a valid correspondence with a client property. Each

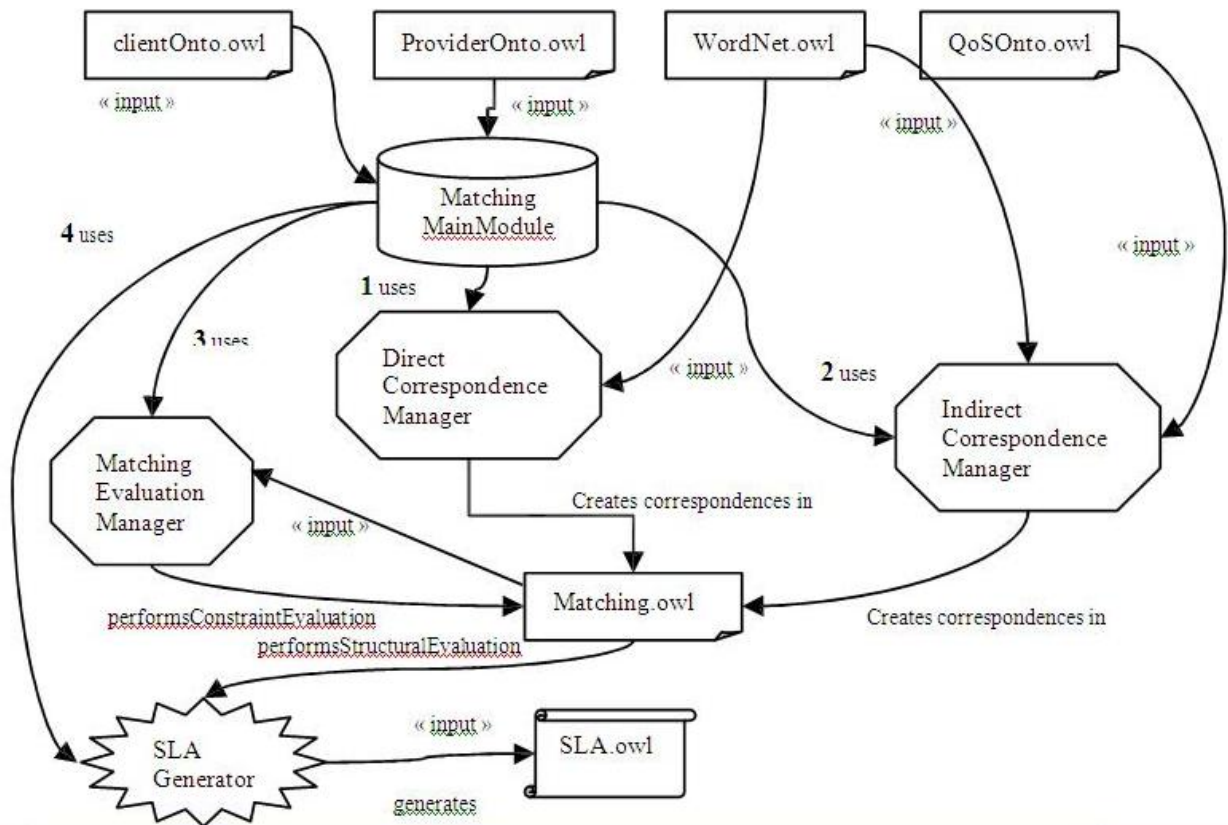


Fig. 7. Simplified architecture of matching client intentions with provider offers

Listing 11. SLACreateComposedMetric rule

```

client:Property(?property) ∧ Correspondance(?c) ∧ hasSourceTerm(?c,
?property) ∧ hasCertainty(?c, ?m) ∧ swrlb:greaterThan(?m, 0.6) ∧
hasDestinationTerm(?c, ?qos) ∧ qos:QoSterm(?qos) ∧
qos:hasDependencyUsing(?qos, ?function) ∧ qos:isBasedOn(?function,
?operand) ∧ qos:isRepresentedBy(?operand, ?var) ∧
qos:hasTypeValue(?var, ?type) ∧ qos:hasPrimaryUnit(?var, ?unit) ∧
qos:hasName(?var, ?name) ∧ Correspondance(?c1) ∧ hasSourceTerm(?c1,
?var) ∧
hasCertainty(?c1, ?m1) ∧ swrlb:greaterThan(?m1, 0.6) ∧
hasDestinationTerm(?c1, ?destination) ∧
provider:ProviderTerm(?destination) ∧
provider:hasMeasurementDirective(?destination, ?measurement) ∧
provider:hasValue(?destination, ?value) ∧
provider:hasMeasurementFrequency(?destination, ?freq) ∧
provider:hasUnit(?value, ?unitValue) ∧

```

We defined also a generatePredicate SWRL rule as shown in Listing 12 to get the property, the threshold value and the operator of each client constraint. For each constraint, this rule will create a corresponding predicate (contract clause) that should be respected by the provider.

Listing 12. generatePredicate rule

```

client:Constraints(?c) ∧ client:hasProperty(?c, ?property) → query:select(?c,
?property) ∧

```

When the predicate is not satisfied, the associated guarantee (in the provider ontology) is automatically triggered. The predicates are expressed as SWRL rules too. They are represented in negative clauses asserting that if the client constraint is not satisfied than trigger the

guarantee action. Listing 13 gives an example of a generated predicate.

Listing 13. downloadTimeParameterPredicateEvaluation rule

```

slaont:hasEvaluation(downloadTimeParameter, ?x) ∧
swrlb:greaterThanOrEqual(?x, 1.0)
→ slaActions:disseminateViolation("downloadTimeConstraint", "false",

```

IV. IMPLEMENTING AND TESTING OUR APPROACH

In this section, we present the implementation of our matching approach. Figure 7 shows the simplified architecture that performs the matching process between the client intentions and the provider offers. For each step defined in our matching approach (see Figure 3), we developed a modules that handles the different actions to be taken within the step. We have also developed a matching main module that starts by loading the client ontology and the provider ontology. This module uses first, the direct correspondence manager and second, the indirect correspondence manager to create correspondences in a new ontology (matching.owl in Figure 7). After that, it uses the matching evaluation manager to perform structural and constraint evaluations in the same matching.owl ontology. Last, the matching main module uses the SLA generator to generate an SLA draft in case of compatibility. This SLA is instance of the SLAOnt ontology [20].

Our matching approach is mainly based on SWRL ules and SQWRL queries. We used the protégé API to execute these rules and queries.

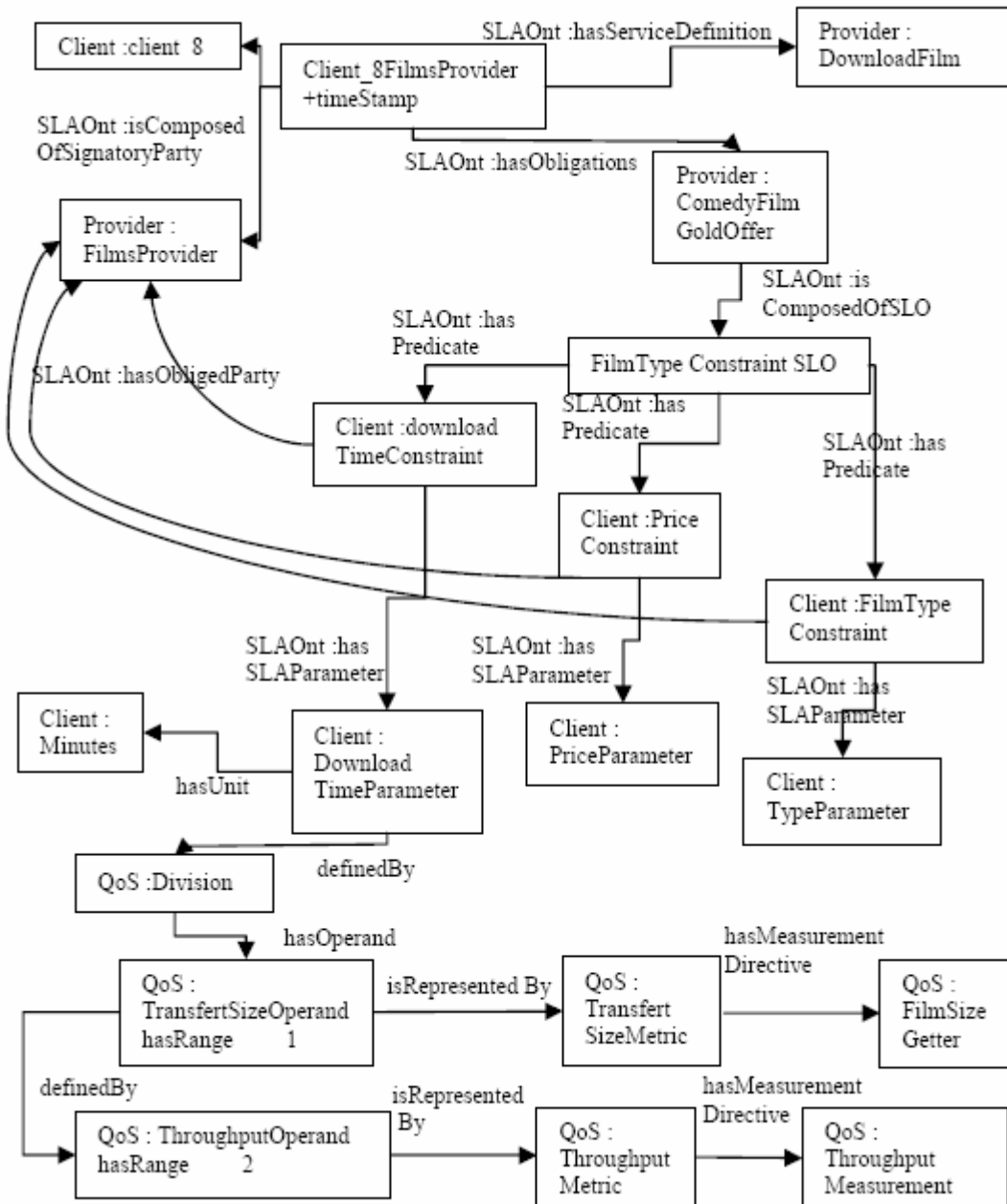


Fig. 8. A simplified generated SLA model

This API has several core software components, including (1) an editor that supports interactive creating, editing, reading, and writing of SWRL rules; (2) a rule engine bridge that provides the infrastructure necessary to interoperate with third-party rule engines and OWL reasoners; (3) a bridge that provides a mechanism for defining libraries of built-ins; and (4) an extensive set of built-in libraries. Built-ins are analogous to functions in production rule systems.

A number of core built-ins are defined in the SWRL specification. This core set includes basic mathematical operators and built-ins for string and date manipulations. These built-ins can be used directly in SWRL rules. To

perform all the steps of our matching approach, we extended the predefined libraries by personalized built-ins in our SWRL rules. We used the Jess2 rule engine to evaluate these rules and to produce the inferred knowledge.

To give a concrete idea of our matching approach, we present a case study that used to test our prototype. We take back the motivation example that we presented in the introduction of this paper where the client wants to perform a download action according to a download time constraint that must be inferior to ten minutes and a price constraint that must be inferior to two euros. On the other side, the provider proposes a comedy film gold offer

among several other offers. This offer is defined by (1) a film type comedy constraint in which the property is equal to a comedy, (2) a comedy film gold price which equal to two euros and (3) a gold throughput constraint (throughput equal to 100 Mbits per second). We notice that this provider can satisfy the client constraint price which is equal two euros. Concerning the download time constraint, we use the QoS ontology to calculate the client download time which is equal to the division of the transferred size value by the throughput value.

Figure 8 presents the main elements of the generated contract according to the given case study in this section. This generated contract is a draft version represented by an OWL [21] file that should be validated by an expert at the provider side. The service definition in this contract concerns the download film service which is imported from the provider ontology. In this SLA, we find a generated service level objective represented by three predicates corresponding to the client constraints: `DownloadTimeConstraint`, `PriceConstraint`, and `FilmTypeConstraint`. Each constraint is defined by an `SLAParameter` corresponding to the properties of the client constraints: `DownloadTimeParameter`, `PriceParameter` and `FilmTypeParameter`. Each parameter is attached to a set of metrics in two possible ways: directly from the provider (the case of `PriceParameter` for example) or indirectly using the QoS ontology (the case of `DownloadTimeParameter` in our case study). These parameters and predicates will be used in a future SLA monitoring stage.

V. CONCLUSION AND FUTURE WORK

Service level contracts become increasingly complex with detailed terms and conditions containing highly technical information. Consequently, clients are experiencing many difficulties to understand the clauses of these contracts. In many cases, the clients have predefined and non understandable choices defined by the providers. Clients are generally not specialists to understand the details of these choices. In this work, our objective is to bridge the gap between the client's intention and the provider offer to facilitate the contract establishment. In this paper, we proposed a QoS contract generation approach by matching the client intention with the provider offer. We defined these intentions and offers as ontologies in order to perform semantic inferences on their asserted knowledge. Indeed, our matching algorithms are based on SWRL rules using the WordNet ontology to establish semantic correspondences between the client intention and the provider offers. We also defined and used a QoS ontology to detect correspondences that are based on mathematical relations between QoS terms. After that, we defined SWRL rules to evaluate the matching results between these ontologies

and to automatically generate an SLA in case of compatibility. We presented an illustrative digital video library example through the sections of this paper to give a concrete idea about our approach. According to this example, the generated correspondences between the client intention and the provider offers are complete and correct but they are only based on the existence or the absence of WordNet relations between the client terms and the provider terms. Actually, we are working on refining the establishment of these detected correspondences by using fuzzy logic techniques to have a better evaluation than the current binary existence or absence of WordNet relations linking the client terms and the provider terms. This will help in automatically comparing and classifying several providers offers with a same client intention.

REFERENCES

- [1] Wooldridge, M., Paurobally, S., Tamma, V.: Cooperation and Agreement between Semantic Web Services. In: W3C Workshop on Frameworks for Semantics in Web Services.
- [2] Hasselmeyer, P., Qu, C., Schubert, L., Koller, B., Wieder, P.: Towards Autonomous Brokered SLA Negotiation. In: Proceedings of eChallenges e-2006 Conference, Barcelona (October 2006).
- [3] BREIN project website, <http://www.eu-brein.com/>
- [4] Joanna Zieba, Bartosz Kryza, Renata Slota, Lukasz Dutka, Jacek Kitowski: Ontology Alignment for Contract Based Virtual Organizations Negotiation and Operation. PPAM 2007: 835-842.
- [5] 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), <https://forge.gridforum.org/projects/graap-wg/>.
- [6] Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic WSAgreement Partner Selection. In: Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW 2006, pp. 697-706. ACM Press, New York (2006).
- [7] Dobson, G., Sanchez-Macian, A.: Towards Unified QoS/SLA Ontologies. In: Proceedings of the IEEE Services Computing Workshop (SCW 2006), August 2006, pp. 169-174 (2006).
- [8] Zhou, C., Chia, L.-T., Lee, B.-S.: Web services discovery with DAMLQoS ontology. International Journal of Web Services Research (IJWSR) 2(2), 44-67 (2005).
- [9] The Protégé Ontology Editor, <http://protege.stanford.edu/>, Mai 2009.
- [10] H. Kanso, C. Soulé-Dupuy and S. Tazi, "Representing author's intentions of scientific documents", Funchal, Madeira (Portugal), 12-16 Juin 2007.
- [11] Tomic, V., Ma, W., Pagurek, B., Esfandiari, B.: Web Service Offerings Infrastructure (WSOI) - a management infrastructure for XML Web services. In: Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP 1, 817-830 (2004).
- [12] Chia, Bu-Sung Lee. "QoS Measurement Issues with DAML-QoS". IEEE InterChen Zhou, Likang-Tien national Conference on e-Business Engineering (ICEBE'05) pp. 395-403.
- [13] G. Dobson, R. Lock, I. Sommerville. "QoSOnt: a QoS Ontology for Service-Centric System", EUROMICRO Conference on Software Engineering and Advanced Applications, Porto, Portugal, Aug. 2005.
- [14] Steffen Bleul, Thomas Weise, Kurt Geihss. "An Ontology for Quality-Aware Service Discovery". Special Edition Editorial: Engineering Design and Composition of Service-Oriented



Applications, Computer Systems Science & Engineering, Volume 5, Number 21 – 2006.

- [15] Tian, M., Gramm, A., Ritter, H., and Schiller, J. "Efficient Selection and Monitoring of QoS-aware Web services with the WSQoS Framework". IEEE/WIC/ACM International Conference on Web Intelligence (WI'04), Beijing, China, 2004.
- [16] Foundation for Intelligent Physical Agents "FIPA Quality of Service Ontology Specification", Geneva, Switzerland, Nov. 2002.
- [17] H.M. Kim, A. Sengupta and J. Evermann. "MOQ: Web Services Ontologies for QoS and General Quality Evaluations", European Conference on Information Systems, Regensburg, Germany, May 2005. [18] ProtegeWiki: SQWRL (available online: <http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL>). 2009.
- [19] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, 21 May 2004.
- [20] K. Fakhfakh, T. Chaari, S. Tazi, K. Drira, and M. Jmaiel. A Comprehensive Ontology-Based Approach for SLA Obligations Monitoring. In Proceedings of the 2008 the Second international Conference on Advanced Engineering Computing and Applications in Sciences - Volume 00 (September 29 - October 04, 2008). ADVCOMP 2008. IARIA. Published by the IEEE Computer Society Press, pp. 217-222.
- [21] Mc Guinness D. & ZSPLITZ van Harmelen F. (2004). OWL Web Ontology Language Overview, W3C Recommendation <http://www.w3.org/TR/owlfeatures>