Documento aggiornato di Analisi e modellazione della Knowledge Base e del Reasoner

Deliverable 2.9.2

Versione 0.3
Date 08/01/2014

Progetto iCaro
La piattaforma cloud per l'accelerazione del business delle PMI toscane
[CUP 6408.30122011.026000074]

COMPUTER GROSS

liberologico.com

UNIVERSITÀ DEGLI STUDI FIRENZE
DINFO
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DISIT

altro lavoro
Agenzia per il lavoro

Informazioni sul documento

ID Deliverable	D2.9.2
Titolo Deliverable	Analisi e modellazione della Knowledge Base e del Reasoner
ID Attività	2.4
N. Versione / Revisione	0.3
Natura: Bozza / Definitivo	Bozza
Partner responsabile	DINFO
Distribuzione: Riservato / Pubblico	Riservato
Riferimenti Autore	Pierfrancesco Bellini
Data redazione	08/01/2014
Riferimenti revisore	Paolo Nesi
Data revisione	15/01/2014
Riferimenti soggetto che approva	Paolo.nesi@unifi.it
Data approvazione e consegna	15/01/2014

Controllo delle revisioni

Oggetto	Numero	Data
Stesura	0.1	8/01/2014
Revisione	0.2	15/01/2014
Chiusura	0.3	15/01/2014

Nota di riservatezza

Il presente documento sarà utilizzato esclusivamente ai fini del progetto ICARO, ha carattere riservato e non potrà quindi essere divulgato se non in seguito ad esplicita autorizzazione scritta da parte dell'ATS, salvo il caso in cui di richieste di ottemperare ad obblighi di legge o a richieste di pubbliche autorità.

Indice

Indice figure

Indice tabelle

Legenda Acronimi e sigle

Acronimo / Sigla	Dettaglio
RDF	Resource Description Framework
OWL	Ontology Web Language
SLA	Service Level Agreement
SCE	Smart Cloud Engine
SM	Supervisor & Monitor
BP	Business Producer
CM	Configuration Manager

1. Introduzione

Questo documento e' la versione aggiornata del deliverable D2.9.1 *Analisi e modellazione della Knowledge Base e del Reasoner*, inoltre per la parte relativa alla definizione della ontologia aggiorna anche quanto riportato nel deliverable D3.4 *Specifica di dettaglio dei sistemi SCE, SM, Knowledge Base e Reasoner.*

In questo documento viene riportata la definizione dell'ontologia per la descrizione delle entità che entrano in gioco nella KnowledgeBase della piattaforma ICARO, questa definizione e' stata fatta sulla base dei requisiti e test case proposti in altri deliverable del progetto oltre che di conoscenza specifica del dominio. Inoltre il documento riporta una analisi dei Reasoner che potrebbero essere usati e l'individuazione dei requisiti tecnici principali per l'inferenza e il controllo di consistenza delle istanze dell'ontologia.

2. Knowledge Base

La Knowledge Base è un componente dell'architettura Icaro che contiene le informazioni relative alla piattaforma. E' utilizzata da Smart Computing Engine (SCE), Supervisor & Monitor (SM), Business Producer (BP), Configuration Manager (CM) e Reasoner per ottenere informazioni su i servizi attivi e quelli disponibili, sulle macchine virtuali ed il loro stato, sugli utenti registrati, etc.

2.1 Analisi stato dell'arte

Per rappresentare la conoscenza in passato sono stati definiti diversi formalismi/linguaggi come Prolog, KL-ONE, Classic, etc. basati su logica del primo ordine, reti semantiche, ontologie, logiche descrittive ma recentemente con il crescente interesse per il Semantic Web le ontologie e le logiche descrittive sono ritornate in auge. In particolare lo standard OWL[owl] poi esteso con OWL2 [owl2] definito da W3C permette la definizione di ontologie descrittive che possono essere condivise e facilmente usate ed estese per descrivere le risorse accessibili su web ma anche oltre.

Per la modellazione delle entità che entrano in gioco nel cloud, sono presenti delle ontologie generali sul computing, per esempio l'ontologia https://geni-orca.renci.org/owl/owl-test/compute.rdf# definita dal progetto ORCA https://geni-orca.renci.org definisce alcuni aspetti ma e' focalizzata soprattutto nella definizione di una gerarchia e alla modellazione di aspetti di networking. Nelle seguenti figure e' riportata l'ontologia che e' orientata a definire una gerarchia e non sono definite relazioni tra le entità.

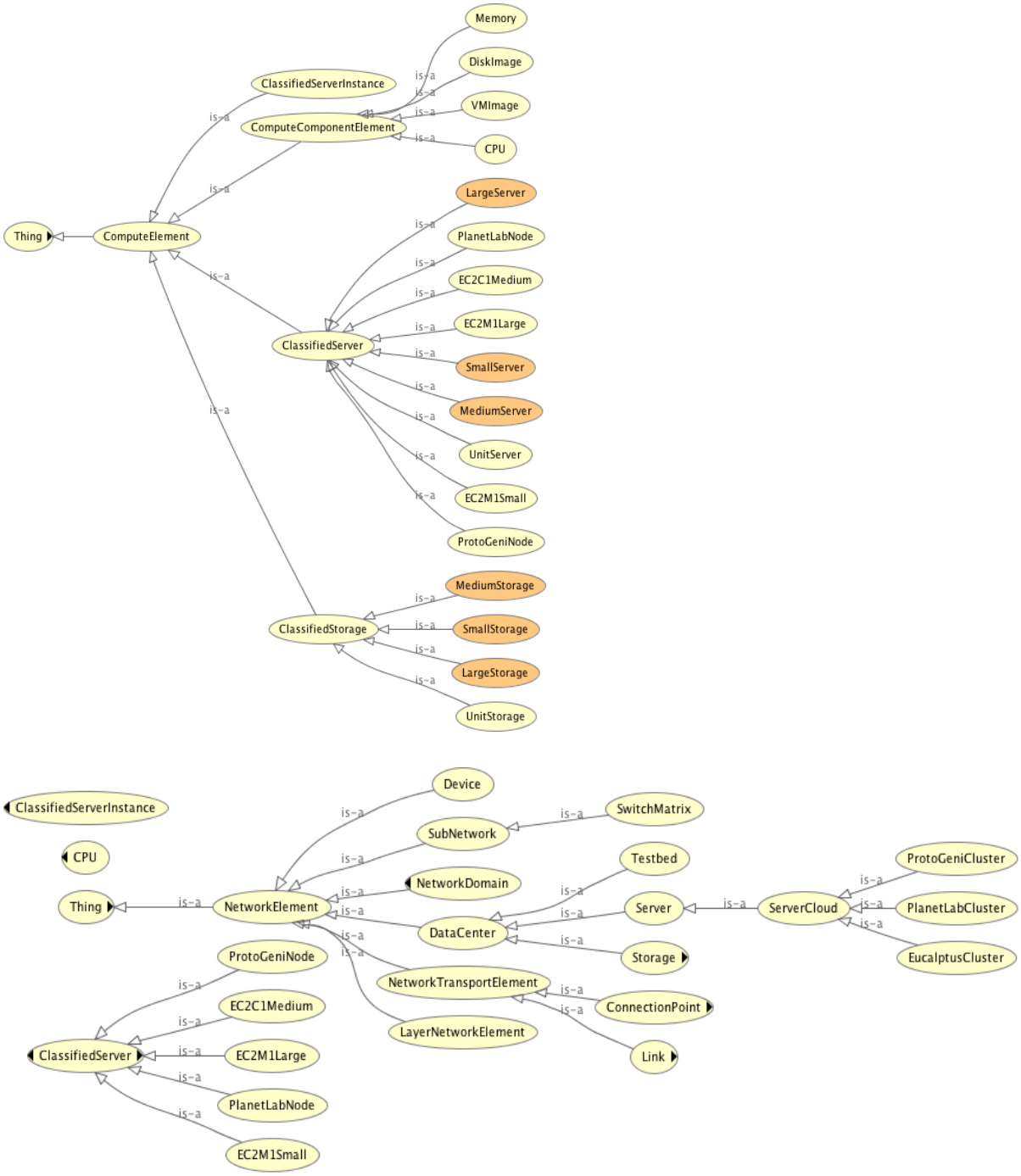


Figura 1 - Schema di Ontologia progetto ORCA

In genere le ontologie vengono usate per la classificazione di risorse esistenti, per facilitare la ricerca, come ad esempio in [Zhang2012], dove viene presentata l'ontologia **CoCoOn** che integrata con altre ontologie come QoSOnt [Dobson2005] per la descrizione dei parametri di servizio permette il 'service discovery'. Nella figura seguente e' riportato il diagramma delle classi definite da questa ontologia. L'ontologia include anche un insieme di *object properties* e *data properties*. Da notare che le classi e proprietà sono definite come in RDFS, definendo dominio e rango e non usando i costrutti specifici di OWL per definire le restrizioni.

Per la descrizione dei servizi offerti come WebServices e' stato proposta l'ontologia OWL-S (http://www.ai.sri.com/daml/services/owl-s/1.2/) che permette di modellare il servizio, il profilo del servizio, il processo ma anche il legame con la descrizione espressa con WSDL.

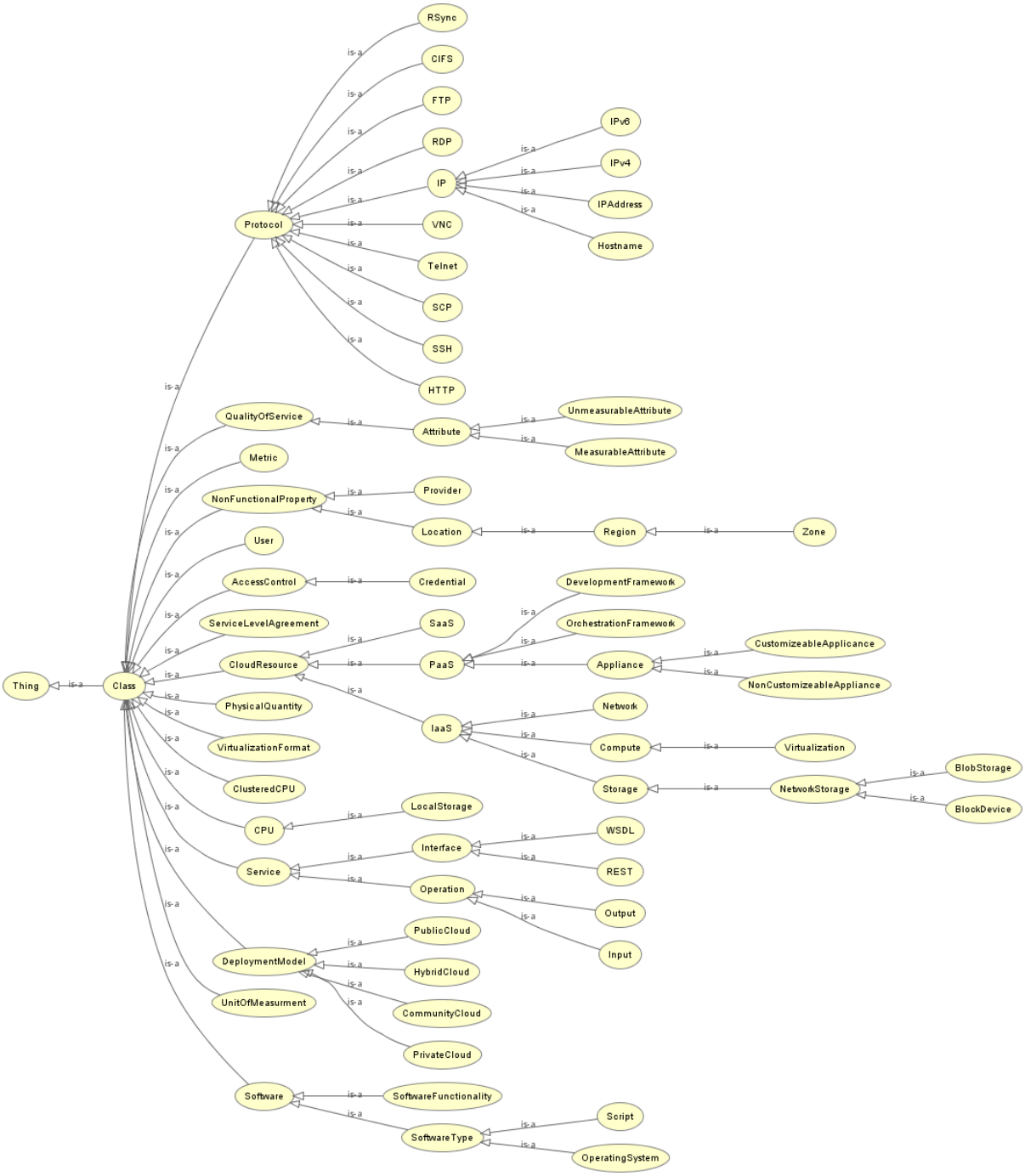


Figura 2 – Ontologia CoCoOn: diagramma delle classi

Altro progetto interessante e' il progetto FI-WARE (http://www.fi-ware.eu/) che usa Linked USDL (Unified Service Description Language) http://linked-usdl.org/ per descrivere i servizi disponibili sulla rete. Linked-USDL e' una rimodellazione del linguaggio USDL (http://www.internet-of-services.de)sviluppato precedentemente come vocabulary RDF(S) e riusa anche altri vocabulary RDF(S) come GoodRelations (http://www.heppnetz.de/projects/goodrelations/), Minimal Service

Model (http://iserve.kmi.open.ac.uk/wiki/index.php/IServe_vocabulary), FOAF (http://www.foaf-project.org/) e altri. Linked-USDL e' diviso in parti:

- **core**, per la descrizione di base del servizio;
- **sla**, per la descrizione delle SLA;
- **sec**, per la descrizione degli aspetti di sicurezza;
- **price**, per la descrizione del modello di prezzo del servizio offerto;
- **ipr**, per la descrizione di aspetti legati alla proprietà intellettuale;

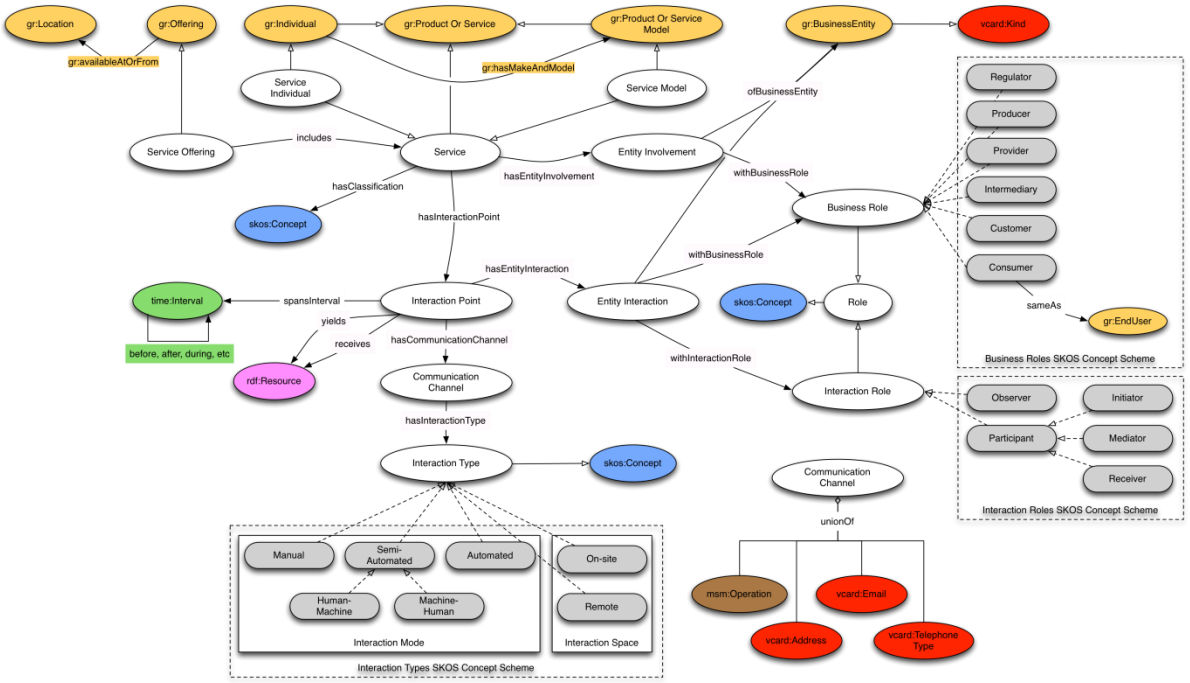


Figura 3 - Modello Linked-USDL core

Altro progetto che usa una descrizione semantica dei servizi disponibili su cloud e' mOSAIC (http://www.mosaic-cloud.eu/), l'ontologia che e' stata sviluppata e' basata su OWL-S e sulla tassonomia definita da NIST nella Cloud Reference Architecture [Moscato2011], l'ontologia sembra piuttosto articolata ma attualmente non e' accessibile la definizione come OWL, inoltre e' piu' orientata a rappresentare i servizi disponibili tramite cloud per poterne effettuare match semantico (esiste un altro cloud che offre lo stesso servizio?).

Per la parte relativa alle SLA e' presente una specifica per la descrizione delle SLA relative ai WebServices, **WSLA** [Ludwig2003]. La descrizione viene fatta usando un XML schema ed e' generale e permette di definire e comporre metriche relative ai servizi realizzati. Nella figura seguente e' riportato il modello generale di una SLA.

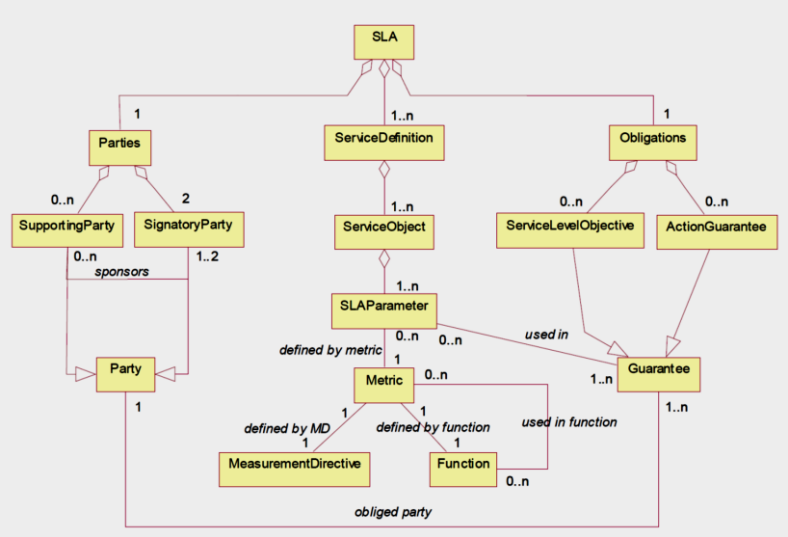


Figura 4 - Modello generale di una SLA

In cui vengono definiti i servizi (ServiceDefinition) sulla base di parametri (SLAParameter) che mettono sono definiti tramite metriche (Metric) che sono definite tramite funzioni (Function) che usano a loro volta anche altre metriche. Ad una SLA sono associate le Obligations che descrivono quali sono gli obiettivi del livello di servizio garantito. Per esempio:

```
<ServiceLevelObjective name="Conditional SLO For AvgThroughput">
   <Obliged>ACMEProvider</Obliged>
   <Validity>
      <Start>2001-11-30T14:00:00.000-05:00</Start>
      <End>2001-12-31T14:00:00.000-05:00</End>
   </Validity>
   <Expression>
      <Implies>
         <Expression>
            <Predicate xsi:type="Less">
               <SLAParameter>OverUtilization</SLAParameter>
               <Value>0.3</Value> <!-- 30% -->
            </Predicate>
         </Expression>
         <Expression>
            <Predicate xsi:type="Greater">
               <SLAParameter>AvgThroughput</SLAParameter>
               <Value>1000</Value>
            </Predicate>
         </Expression>
      </Implies>
   </Expression>
   <EvaluationEvent>NewValue</EvaluationEvent>
</ServiceLevelObjective>
```

WS-Agreement [Andrieux2007] e' stato sviluppato da "Grid Resource Allocation Agreement Protocol Working Group" (GRAAP-WG) il cui scopo e' di produrre un insieme di specifiche per descrivere metodi e mezzi per definire delle SLA tra entità differenti in un ambiente distribuito. La *WS-Agreement Specification V1.0*, definisce un protocollo basato su Web Services per stabilire degli agreements tra due servizi, e' stato pubblicato nel Maggio 2007 come una *Open Grid Forum Proposed Recommendation.* La specifica e' composta da tre parti, uno schema per specificare un agreement, uno schema per specificare un agreement template, e un insieme di "port type" e "operation" per gestire il ciclo di vita degli agreement includendo creazione, terminazione e controllo dello stato degli agreement.

Un Agreement e' formato da un contesto (*Context*) e da dei *Terms* divisi in *ServiceTerms* e *GuaranteeTerms*, quest'ultimi contengono le condizioni che devono essere garantite dal servizio. Le

condizioni possono essere specificate con un valore target per un certo parametro oppure usando delle espressioni ma il formalismo usato per esprimere queste espressioni non e' definito nella specifica, ma invece può essere usato un qualsiasi altro formalismo espresso in XML (o anche come testo), questo limita fortemente l'interoperabilità' del sistema.

In [Oldham2006]partendo da WS-Agreement viene definita una ontologia (non accessibile) per realizzare un sistema per fare matching tra richiesta e offerta di servizi, l'ontologia fa anche uso di *QoSOnt* [Maxemilien2004] per definire gli aspetti di Quality of Service e TimeOnt per gli aspetti temporali. Questo servizio (SWAPS) e' basato su tecnologie semantiche come SNOWBASE di IBM per la gestione della ontologia e ABLE sempre di IBM per il reasoning e per scrivere le regole di inferenza. Da notare che per definire le condizioni di servizio vengano usate le espressioni WSLA che sono facilmente modellabili in OWL. Per esempio il seguente frammento in WS-Agreement:

```
<ServiceLevelObjective>duration1+duration2 < 5 s</ServiceLevelObjective>
```

Viene rappresentato in SWAPS come:

```
<ServiceLevelObjective>
  <Expression>
    <Predicate type="less">
      <Parameter>duration1+duration2</Parameter>
      <OntConcept>qos::responseTime</OntConcept>
      <Value>5</Value>
      <Unit>time:seconds</Unit>
    </Predicate>
  </Expression>
</ServiceLevelObjective>
```

SLAng [Lamanna2003] e' un altro XML schema per definire SLA, lo schema definito (non accessibile) e' molto meno generico di WSLA e WS-Agreement come si vede dall'esempio che segue:

```
<SLAng xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Vertical>
      <Hosting>
         <Id sls_id="49258" service_id="Replication"/>
         <Client>
            <Name>CPXe-Dir</Name>
            <Place>Los Angeles</Place>
            <Clients mean_number="24000" maximum_number="40000" arrival_rate="113.2"/>
            <Availability>95%</Availability>
         </Client>
         <Server>
            <Name>WebApp2Go.com</Name>
            <Place>London</Place>
            <Provision disk_space="500" memory_usage="512"/>
            <High_availability>99.6%</High_availability>
            <Maintenance recovery_time="2" scheduled_outages="20" routine_maintenances="12"/>
            <Performance response_time="2.6" peak_time_latency="4.7"
               successful_transactions="98%" processing_speed="843"/>
            <Cluster_throughput containers="9" active_clients="310
               method_invocation="53.141"/>
            <Security data_protection="true" encryption_method="RSA"
               certificate="true" user_authentication="true"
               intrusion_detection="false" eavesdrop_prevention="false"/>
            <Backup solution="REOBack" complete_backup_interval="24"
               incremental_backup_interval="6" data_types="User configuration data"
               archiving_form="rar" client_access="true" backup_encryption="false"
               individual_client_backup="true"/>
            <Monitoring tracking_system="IDX System" report_method="XML"
               report_frequency="48" reporting_on_demand="false"/>
         </Server>
         <Mutual>
            <Service_schedule start="2002-12-13" end="2003-12-13"/>
            <Failure_clauses compensation="(100%-availability)*4.6"
               exclusion_clauses="Client caused outages"/>
         </Mutual>
      </Hosting>
```

```
   </Vertical>
</SLAng>
```

Il progetto europeo **NextGrid** per la definizione di una piattaforma europea per il grid computing ha definito una SLA basta essenzialmente su WS-Agreement. Una SLA e' definita con un Name, Context e Terms. Il contesto contiene informazioni sulle parti in gioco (ServiceProvider, ServiceConsumer ed un eventuale SupportingParty), sui limiti di validità temporale e un eventuale RelatedAgreement. La sezione dei Terms e' divisa in DynamicTerms e StaticTerms, i termini dinamici sono composti da una serie di metriche (Metric) che rappresentano le metriche di basso livello che poi vengono collegate attraverso una MappingFunction (viene proposto di usare XQuery per scrivere la funzione) in uno SLAParameter per il quale viene specificato un valore target ed eventuali valori di Premium/Compensation che saranno pagati dal consumer/provider se il parametro non rientra nel target.

Inoltre nel contesto del progetto FI-WARE e' stata definito il modello *SLAware*, che definisce in modo formale la semantica delle SLA usando la Transparent Intensional Logic (una logica modale, temporale) mentre il modello dati e' definito usando UML. Comunque il supporto a questo formalismo sembra fermato ad un draft del 2012 (). Sempre nel contesto del progetto FI-WARE e' presente in *LinkedUSDL* una parte relativa alla rappresentazione delle SLA che e' molto piu' semplice rispetto a SLAware ma che per esempio non definisce come le variabili in gioco nella SLA sono combiante in un espressione per definire un vincolo.

2.2 Requisiti

NR	Ruolo Utente	Requisito	Descrizione	Priorità
KB1	--	Struttura fisica	La KnowledgeBase deve contenere informazioni sulla configurazione fisica del datacenter	media
KB2	--	Struttura	La KB deve contenere informazioni sulle macchine virtuali presenti nel datacenter	media
KB3	--	Servizi	La KB deve contenere informazioni sui servizi disponibili sulla piattaforma	alta
KB4	--	Applicazioni	La KB deve contenere informazioni sulle applicazioni disponibili sulla piattaforma	alta
KB5	--	Multitenant	La KB deve contenere informazioni anche sui servizi/applicazioni che sono acquisibili in modalità multitenant	media

KB6	--	Configurazioni	La KB deve contenere informazioni sulla Configurazione definita dall'utente integrando applicazioni e servizi	alta
KB7	--	SLA	La KB deve contenere informazioni su la SLA richiesta/negoziata dall'utente per una Configurazione, Servizio o Applicazione	alta
KB8	--	Monitoring	La KB deve contenere informazioni sull'uso delle risorse, provenienti dal monitoring	media
KB9	--	Query	La KB deve permettere query semantiche	media
KB10	--	Storia	La KB deve mantenere la storia delle modfiche apportate alla knowledge-base	media

La Knowledge Base deve supportare inoltre query semantiche che permettano di:

- Identificare gli host poco utilizzati
- Identificare gli host troppo utilizzati
- Identificare macchine virtuali che usano troppe risorse (CPU, memoria)
- Identificare servizi/applicazioni che usano troppe risorse (CPU, memoria)
- Determinare se una nuova istanza di una applicazione possa essere realizzata

2.3 Ontologia Icaro

In questa sezione vengono riportate le entità principali e le loro relazioni dell'ontologia icaro per la rappresentazione delle informazioni presenti nella Knowledge Base, questa e' una prima versione che poi verrà rivista in fase di progettazione e quindi in seguito durante l'implementazione e test.

L'ontologia sarà divisa in più parti :

- Una parte per la descrizione della struttura fisica e virtuale del data center;
- Una parte per la descrizione generica della parte software;
- Una parte per la descrizione della SLA;
- Una parte per la descrizione delle applicazioni e dei servizi che sono utilizzabili dagli utenti.

E' questa ultima parte che conterrà la descrizione delle applicazioni utilizzabili su cloud icaro che sarà la più dinamica e sarà arricchita da nuove applicazioni che saranno utilizzabili sul cloud.

2.2.1 Entità

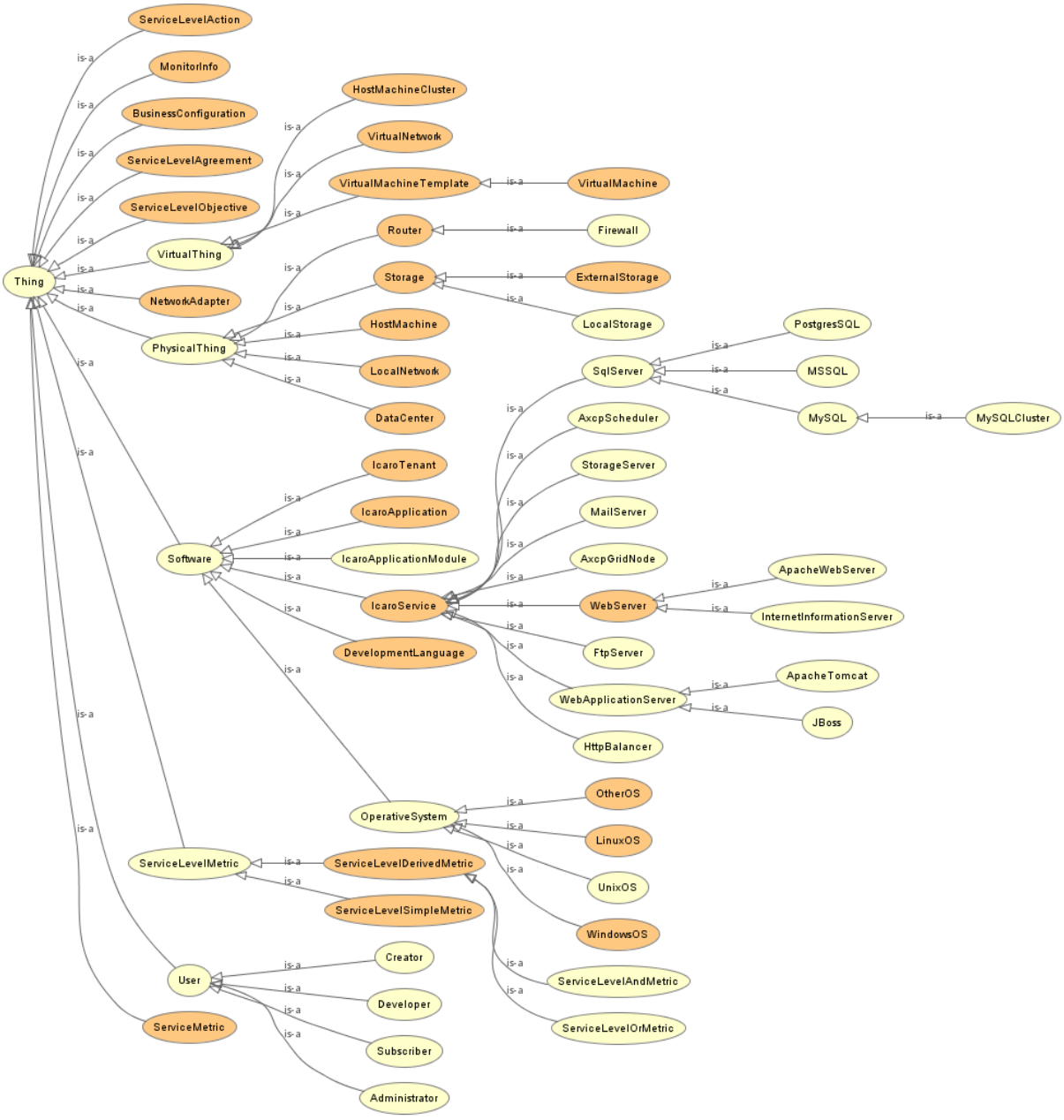
Nella figura seguente sono riportate le entità identificate, organizzate secondo la gerarchia IS-A. Ci sono le entità che rappresentano i **dispositivi fisici** (DataCenter, HostMachine, Storage, LocalNetwork) e le **virtualizzazioni** (VirtualMachine, VirtualMachineTemplate, VirtualNetwork), le entità legate alla parte **software** (IcaroService, IcaroApplication, OperativeSystem, DevelopmentLanguage etc.) e quelle per la definizione della **SLA** e per la misura di **metriche** dei vari servizi (ServiceLevelAgreement, ServiceMetric, etc.) e per la descrizione degli utenti secondo il loro ruoli (Creator, Developer, Administrator, Subscriber).

Nel seguito sono descritte le classi principali usando la sintassi OWL2 Manchester (http://www.w3.org/TR/owl2-manchester-syntax) inoltre sono riportati dei piccoli esempi scritti usando RDF-XML (http://www.w3.org/TR/rdf-syntax-grammar/).

Per la identificazione delle entita', si usano gli urn nella forma:

urn:cloudicaro:<tipo>**:**<identificatore>

Dove <*tipo*> rappresenta il tipo della entità e corrisponde al nome della classe a cui appartiene e <*identificatore*> un identificatore univoco almeno a livello di tipo, negli esempi che seguono viene usato come identificatore un numero progressivo all'interno dello stesso tipo.



2.2.1.1 DataCenter

Il **DataCenter**, aggrega un insieme di *HostMachine* e delle *LocalNetwork*

DataCenter EquivalentTo
PhysicalThing and
(hasPart some (HostMachine or HostMachineCluster) and
(hasPart some LocalNetwork) and
hasPart only (HostMachine orLocalNetwork or HostMachineCluster) and
(*hasIdentifier* exactly 1 string) and
(*hasName* exactly 1 string)

Esempio:

```
<icr:DataCenter rdf:about="urn:cloudicaro:DataCenter:01">\
  <icr:hasName>ICARO data center</icr:hasName>
  <icr:hasIdentifier>01</icr:hasIdentifier>
  <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:001" />
  <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:002" />
  <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:003" />
  <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:003" />
  <icr:hasPart rdf:resource="urn:cloudicaro:LocalNetwork:01" />
</icr:DataCenter>
```

2.2.1.2 HostMachine

Una **HostMachine,** aggrega un insieme di *VirtualMachineTemplate* e quindi di *VirtualMachines*

HostMachine EquivalentTo
PhysicalThing and
(hasPart some LocalStorage) and
(hasPart some VirtualMachineTemplate) and
(hasPart only (LocalStorage orVirtualMachineTemplate)) and
(isPartOf exactly 1 (DataCenter or HostMachineCluster)) and
(hasNetworkAdapter some NetworkAdapter) and
(hasNetworkAdapter only NetworkAdapter) and
(*hasIdentifier* exactly 1 string) and
(*hasName* exactly 1 string) and
(*hasCPUCount* exactly 1 positiveInteger) and
(*hasMemorySize* exactly 1 decimal) and
(hasMonitorInfo only MonitorInfo) and
(*hasAuthUserName* max 1 string) and
(*hasAuthUserPassword* max 1 string) and
(*isInDomain* max 1 string) and
(*hasMonitorState* max 1 string) and
(*hasCapacity* max 1 decimal)

Esempio:

```
<icr:HostMachine rdf:about="urn:cloudicaro:HostMachine:001">
  <icr:hasName>HOST1</icr:hasName>
  <icr:hasIdentifier>HST01</icr:hasIdentifier>
  <icr:hasCPUCount rdf:datatype="
      http://www.w3.org/2001/XMLSchema#positiveInteger">8</icr:hasCPUCount>
  <icr:hasMemorySize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">8</icr:hasMemorySize>
  <icr:hasNetworkAdapter>
    <icr:hasIPAddress>192.168.1.100</icr:hasIPAddress>
    <icr:boundToNetwork rdf:resource="urn:cloudicaro:LocalNetwork:01" />
  </icr:hasNetworkAdapter>
  <icr:hasPart rdf:resource="urn:cloudicaro:LocalStorage:001_001" />
  <icr:hasPart rdf:resource="urn:cloudicaro:VirtualMachine:001" />
  <icr:hasPart rdf:resource="urn:cloudicaro:VirtualMachine:002" />
</icr:HostMachine>
```

2.2.1.2 NetworkAdapter

Un **NetworkAdapter,** rappresenta un adottatore di rete che ha un indirizzo IP ed e' collegato ad una rete

NetworkAdapter EquivalentTo
hasIPAddress exactly 1 string and

(boundToNetwork exactly 1 LocalNetwork) and
(*hasType* max 1 string)

2.2.1.3 Storage

Uno **Storage** rappresenta un'unità di memorizzazione che potrà essere all'interno della macchina host oppure esterna su un dispositivo specifico (es. NAS, SAN, etc.), e' caratterizzata da avere una dimensione (in GB)

Storage EquivalentTo
PhysicalThing that
(*hasDiskSize* exactly 1 decimal)

2.2.1.4 LocalStorage

Una unità di memorizzazione su una macchina host

LocalStorage SubClassOf Storage

Esempio:

```
<icr:LocalStorage rdf:about="urn:cloudicaro:LocalStorage:001_001">
  <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">500</icr:hasDiskSize>
  <icr:isPartOf rdf:resource="urn:cloudicaro:HostMachine:001" />
</icr:LocalStorage>
```

2.2.1.5 ExternalStorage

Una unità di memorizzazione esterna accessibile nella rete locale

ExternalStorage EquivalentTo
Storage and
hasName exactly 1 string and
hasIdentifier exactly 1 string and
hasModelName max 1 string and
hasNetworkAdapter some NetworkAdapter and
hasNetworkAdapter only NetworkAdapter and
hasMonitorInfo only MonitorInfo and
hasMonitorState max 1 string

Esempio:

```
<icr:ExternalStorage rdf:about="urn:cloudicaro:ExternalStorage:001">
  <icr:hasName>EXTDISK 1</icr:hasName>
  <icr:hasIdentifier>DISK1</icr:hasIdentifier>
  <icr:hasNetworkAdapter>
    <icr:hasIPAddress>192.168.1.100</icr:hasIPAddress>
    <icr:boundToNetwork rdf:resource="urn:cloudicaro:LocalNetwork:01" />
  </icr:hasNetworkAdapter>
  <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">2000</icr:hasDiskSize>
</icr:ExternalStorage>
```

2.2.1.6 VirtualMachineTemplate

Una **VirtualMachineTemplate,** un template di macchina virtuale, implementa uno o più *IcaroService* o *IcaroApplication,* ha associato un sistema operativo, uno spazio disco, un numero di CPU e una memoria

VirtualMachineTemplate EquivalentTo
VirtualThing that
isPartOf exactly 1 HostMachine and
isPartOf only (HostMachine or LocalNetwork or VirtualNetwork) and
hasNetworkAdapter some NetworkAdapter and
hasNetworkAdapter only NetworkAdapter and
hasOS exactly 1 OperativeSystem and
hasName exactly 1 string and
hasIdentifier exactly 1 string and
hasDiskSize some decimal and
*hasCPU*Count exactly 1 positiveInteger and
hasMemorySize exactly 1 decimal and

Esempio:

```
<icr:VirtualMachineTemplate rdf:about="urn:cloudicaro:VirtualMachineTemplate:001">
   <icr:hasCPUCount rdf:datatype="
      http://www.w3.org/2001/XMLSchema#positiveInteger">2</icr:hasCPUCount>
   <icr:hasMemorySize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">2</icr:hasMemorySize>
   <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">200</icr:hasDiskSize>
   <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">100</icr:hasDiskSize>
   <icr:hasNetworkAdapter>
     <icr:NetworkAdapter>
       <icr:hasIPAddress>192.168.1.123</icr:hasIPAddress>
        <icr:boundToNetwork rdf:resource="urn:cloudicaro:LocalNetwork:01"/>
     </icr:NetworkAdapter>
   </icr:hasNetworkAdapter>
   <icr:hasOS rdf:resource="http://www.cloudicaro.it/cloud_ontology/core#windowsXP" />
   <icr:isPartOf rdf:resource="urn:cloudicaro:HostMachine:001" />
</icr:VirtualMachineTemplate>
```

2.2.1.7 VirtualMachine

Una **VirtualMachine,** una macchina virtuale, specializza un template ed e' istanza di un *VirtualMachineTemplate* ed ha associato uno o piu' indirizzi IP.

VirtualMachine EquivalentTo
VirtualMachineTemplate and
isInstanceOf max 1 VirtualMachineTemplate and
(hasMonitorInfo only MonitorInfo) and
(isPartOf only (HostMachine or LocalNetwork or VirtualNetwork)) and
(*hasAuthUserName* max 1 string) and
(*hasAuthUserPassword* max 1 string) and
(*isInDomain* max 1 string) and
(*hasMonitorState* max 1 string)

Esempio:

```
<icr:VirtualMachine rdf:about="urn:cloudicaro:VirtualMachine:001">
   <icr:hasCPUCount rdf:datatype="
      http://www.w3.org/2001/XMLSchema#positiveInteger">2</icr:hasCPUCount>
   <icr:hasMemorySize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">2</icr:hasMemorySize>
   <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">200</icr:hasDiskSize>
   <icr:hasDiskSize rdf:datatype="
      http://www.w3.org/2001/XMLSchema#decimal">100</icr:hasDiskSize>
   <icr:hasNetworkAdapter>
     <icr:NetworkAdapter>
       <icr:hasIPAddress>192.168.1.23</icr:hasIPAddress>
        <icr:boundToNetwork rdf:resource="urn:cloudicaro:LocalNetwork:01"/>
     </icr:NetworkAdapter>
   </icr:hasNetworkAdapter>
   <icr:hasOS rdf:resource="http://www.cloudicaro.it/cloud_ontology/core#windowsXP" />
   <icr:isPartOf rdf:resource="urn:cloudicaro:HostMachine:001" />
   <icr:isInstanceOf rdf:resource="urn:cloudicaro:VirtualMachineTemplate:001" />
</icr:VirtualMachine>
```

2.2.1.8 HostMachineCluster

Un **HostMachineCluster,** rappresenta un insieme di macchine host che sono raggruppate in un cluster.

HostMachineCluster EquivalentTo
VirtualThing and
isPartOf max 1 DataCenter and
hasPart some HostMachine and
hasPart only HostMachine and
hasName exactly 1 string and
hasIdentifier exactly 1 string

Esempio:

```
<icr:HostMachineCluster rdf:about="urn:cloudicaro:HostMachineCluster:001">
   <icr:hasName>XMLS cluster</icr:hasName>
   <icr:isPartOf rdf:resource="urn:cloudicaro:DataCenter:01" />
   <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:003" />
   <icr:hasPart rdf:resource="urn:cloudicaro:HostMachine:004" />
</icr:HostMachineCluster>
```

2.2.1.9 IcaroService

Un servizio realizzato da una macchina virtuale

IcaroService EquivalentTo
Software and
runsOnVM some VirtualMachine and
runsOnVM only VirtualMachine and
hasMonitorInfo only MonitorInfo and
hasName exactly 1 string and
hasIdentifier exactly 1 string

derivati da questa classe sono i servizi usati sulla piattaforma come (elenco preliminare):

- *WebServer*, che possono essere ulteriormente specializzati in:

    - *ApacheWebServer*
    - *InternetInformationServer*
- *WebApplicationServer*, specializzati in:
    - *JBoss*
    - *ApacheTomcat*
- *HttpBalancer*
- *SqlServer*, specializzati in:
    - *PostgresSql*
    - *MySql, specializzato in*
        - *MySqlClusterNode*
    - *MSSql*
- *FtpServer*
- *MailServer*
- *StorageServer*
- *AxcpScheduler*
- *AxcpGridNode*
- …

Specifiche versioni dei software potranno essere, se necessario, definite come specializzazioni es. ApacheTomcat5.5

Esempio:

```
<icr:ApacheWebServer rdf:about="urn:cloudicaro:ApacheWebServer:001">
    <icr:runsOnVM rdf:resource="urn:cloudicaro:VirtualMachine:001" />
    <icr:supportsLanguage rdf:resource="http://www.cloudicaro.it/cloud_ontology/core#php" />
    <icr:hasMonitorInfo>
        <icr:MonitorInfo>
            …
        </icr:MontorInfo>
    </icr:hasMonitorInfo>
</icr:ApacheWebServer>
```

2.2.1.10 IcaroApplication

Una applicazione utilizzabile tramite la piattaforma Icaro, utilizza per il suo funzionamento un certo insieme di macchine virtuali, ha bisogno eventualmente di altre applicazioni o servizi o moduli software specifici della applicazione, e' stata creata da un utente creatore, ed e' stata sottoscritta da utenti che hanno acquisito il diritto all'uso della applicazione ed ha associata una SLA che definisce i livelli di servizio della applicazione.

IcaroApplication EquivalentTo
Software that
useVM some VirtualMachine and
useVM onlyVirtualMachine and
needs only (IcaroApplication or IcaroApplicationModule or IcaroService) and
hasMonitorInfo only MonitorInfo and
subscribedBy only Subscriber and
createdBy exactly 1 Creator and
developedBy exactly 1 Developer and
hasSLA exactly 1 ServiceLevelAgreement and

hasName exactly 1 string and
hasIdentifier exactly 1 string and
hasCapacity max 1 decimal

derivate da *IcaroApplication* sono le applicazioni fruibili sulla piattaforma, ad esempio:

- Inaz, Amico, Sigla++, XLMS, OggiWeb, Joomla, OpenBravo etc.

Ogni applicazione potrà essere ulteriormente derivata per considerare le varie configurazioni e ogni applicazione avrà associate le dipendenze da altre applicazioni e servizi, per esempio XLMS (Crossmedia Learning Management System) che sarà definita come:

XLMS EquivalentTo
IcaroApplication that
developedBy value disit and
needs exactly 1 (ApacheWebServer and supportsLanguage value php) and
needs exactly 1 ApacheTomcat and
needs exactly 1 MySQL and
needs exactly 1 AxcpScheduler and
needs min 1 AxcpGridNode and
needs max 1 StorageServer and
needs only (ApacheTomcat or ApacheWebServer or AxcpGridNodeor AxcpScheduler or MySQL or StorageServer)

Indica che XLMS e' una *IcaroApplication* che e' stata sviluppata dall'utente '*disit*' ha bisogno di un *ApacheWebServer* che supporta PHP, un *ApacheTomcat*, un database *MySql*, un *AxcpScheduler* e al minimo un nodo AXCP (per il processing dei contenuti audio/video), e di al massimo uno *StorageServer*, inoltre viene indicato che questi servizi sono i soli dei quali ha bisogno.

Esempio:

```
<app:XLMS rdf:about="urn:cloudicaro:XLMS:001">
    <icr:developedBy rdf:resource="urn:cloudicaro:User:disit" />
    <icr:createdBy rdf:resource="urn:cloudicaro:User:xyz" />
    <icr:subscribedBy rdf:resource="urn:cloudicaro:User:acme" />
    <icr:needs rdf:resource="urn:cloudicaro:ApacheWebServer:001" />
    <icr:needs rdf:resource="urn:cloudicaro:ApacheTomcat:001" />
    <icr:needs rdf:resource="urn:cloudicaro:MySQL:001" />
    <icr:needs rdf:resource="urn:cloudicaro:AxcpScheduler:001" />
    <icr:needs rdf:resource="urn:cloudicaro:AxcpGridNode:001" />
    <icr:hasSLA rdf:resource="urn:cloudicaro:ServiceLevelAgreement:001" />
</app:XLMS>
```

2.2.1.11 IcaroMultiTenantApplication

Una applicazione usabile come divisa in tenat.

IcaroMultiTenantApplication EquivalentTo
IcaroApplication that
hasTenant some IcaroTenant and
hasTenant only IcaroTenant

2.2.1.12 IcaroTenant

Rappresenta una porzione di una applicazione divisa in tenant.

IcaroTenant EquivalentTo
Software that
isTenatntOf exactly 1 IcaroMultiTenantApplication and
isTenantOf only IcaroMultiTenantApplication and
hasMonitorInfo only MonitorInfo and
subscribedBy only Subscriber and
createdBy exactly 1 Creator and
administeredBy only Administrator and
hasSLA max 1 ServiceLevelAgreement and
hasName exactly 1 string and
hasIdentifier exactly 1 string and
hasMonitorState max 1 string

2.2.1.13 BusinessConfiguration

Una configurazione che integra un insieme di macchine virtuali per realizzare una o più applicazioni sfruttando una serie di servizi

BusinessConfiguration EquivalentTo
hasPart only (IcaroApplication or IcaroTenant) and
createdBy exactly 1 Creator and
administeredBy only Administrator
hasName exactly 1 string and
hasIdentifier exactly 1 string

Esempio:

```
<icr:BusinessConfiguration rdf:about="urn:cloudicaro:BusinessConfiguration:001">
    <icr:hasPart rdf:resource="urn:cloudicaro:XLMS:001" />
    <icr:createdBy rdf:resource="urn:cloudicaro:User:xyz" />
</icr:BusinessConfiguration>
```

2.2.1.14 OperativeSystem

Indica l'insieme dei sistemi operativi supportati dalla piattaforma, sono classificati in LinuxOS, UnixOS, WindowsOS e OtherOS

OperativeSystem SubClassOf Software
OperativeSystem DisjointUnionOf LinuxOS, UnixOS, WindowsOS, OtherOS
LinuxOS EquivalentTo {ubuntu, redhat}
UnixOS EquivalentTo {…}
WindowsOS EquivalentTo {windows2003Server , windows7 , windowsVista , windowsXP}
OtherOS EquivalentTo {os400}

2.2.1.15 DevelopmentLanguage

Indica l'insieme dei linguaggi di sviluppo supportati dalla piattaforma

DevelopmentLanguage EquivalentTo {java, php, csharp, ruby}

2.2.1.16 ServiceLevelAgreement

Insieme di condizioni che descrivono il livello di servizio richiesto dall'utente per la *Applicazione*

ServiceLevelAgreement EquivalentTo
hasSLObjective some ServiceLevelObjective and
hasSLObjective only ServiceLevelObjective and
hasStartTime exactly 1 dateTime and
hasEndTime exactly 1 dateTime

Esempio:

```
<icr:ServiceLevelAgreement rdf:about="urn:cloudicaro:ServiceLevelAgreement:001">
   <icr:hasSLObjective>
     <icr:ServiceLevelObjective>
       …
     </icr:ServiceLevelObjective>
   </icr:hasSLObjective>
   <icr:hasStartTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2013-01-
     01T00:00:00</icr:hasStartTime>
   <icr:hasEndTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-01-
     01T00:00:00</icr:hasEndTime>
</icr:ServiceLevelAgreement>
```

2.2.1.17 ServiceLevelObjective

Una condizione su un livello di servizio, ha associata una metrica o più metriche di servizio e una o più azioni da effettuare in caso di fallimento del servizio.

ServiceLevelObjective EquivalentTo
hasSLAction some ServiceLevelAction and
hasSLMetric some ServiceLevelMetric and
hasSLAction only ServiceLevelAction and
hasSLMetric only ServiceLevelMetric

Esempio:

```
<icr:ServiceLevelObjective rdf:about="urn:cloudicaro:ServiceLevelObjective:001">
    <icr:hasSLAction>
      <icr:ServiceLevelAction>
         <icr:hasName>Send e-mail</icr:hasName>
      </icr:ServiceLevelAction>
    </icr:hasSLAction>
    <icr:hasSLMetric>
      <icr:ServiceLevelSimpleMetric>
         <icr:hasMetricName>avgResponseTime</icr:hasMetricName>
         <icr:hasMetricValueLessThan rdf:datatype="
            http://www.w3.org/2001/XMLSchema#decimal">5</icr:hasMetricValueLessThan>
         <icr:hasMetricUnit>seconds</icr:hasMetricUnit>
         <icr:dependsOn rdf:resource="urn:cloudicaro:ApacheWebServer:001" />
      </icr:ServiceLevelSimpleMetric>
    </icr:hasSLMetric>
</icr:ServiceLevelObjective>
```

2.2.1.18 ServiceLevelMetric

Una generica condizione su una metrica di servizio

2.2.1.19 ServiceLevelSimpleMetric

Una condizione semplice sul valore di una metrica di servizio

ServiceLevelSimpleMetric EquivalentTo
ServiceLevelMetric that
hasMetricName exactly 1 string and
(*hasMetricValue* exactly 1 decimal or
hasMetricValueLessThan exactly 1 decimal or
hasMetricValueMoreThan exactly 1 decimal) and
hasMetricUnit exactly 1 string and
dependsOn some (IcaroApplication or IcaroService or PhysicalThing or VirtualMachine) and
dependsOn only (IcaroApplication or IcaroService or PhysicalThing or VirtualMachine)

2.2.1.20 ServiceLevelDerivedMetric

Una condizione composta da almeno una ServiceLevelMetric.

ServiceLevelDerivedMetric EquivalentTo
ServiceLevelMetric that
dependsOn some ServiceLevelMetric and
dependsOn only ServiceLevelMetric

2.2.1.21 ServiceLevelAndMetric

Una ServiceLevelDerivedMetric in cui tutte le metriche associate sono in AND

ServiceLevelAndMetric SubClassOf ServiceLevelDerivedMetric

2.2.1.22 ServiceLevelOrMetric

Una ServiceLevelDerivedMetric in cui tutte le metriche associate sono in OR

ServiceLevelOrMetric SubClassOf ServiceLevelDerivedMetric

2.2.1.23 ServiceLevelAction

Una ServiceLevelAction rappresenta l'azione da compiere quando la ServiceLevelObjective a cui e' legata risulta falsa. Attualmente e' descritta esclusivamente dal nome ma si possono avere delle sotto classi in cui si hanno per esempio delle penalita' economiche o solo l'indicazione che deve essere inviata una mail.

ServiceLevelAction EquivalentTo
Thing that
hasName exactly 1 string

2.2.1.24 ServiceMetric

Rappresenta il valore di una metrica di un servizio o di una applicazione.

ServiceMetric EquivalentTo
dependsOn some (IcaroApplication or IcaroService or PhysicalThing or VirtualMachine) and
dependsOn only (IcaroApplication or IcaroService or PhysicalThing or VirtualMachine) and
atTime exactly 1 dateTime and
hasMetricName exactly 1 string and
hasMetricUnit exactly 1 string and

hasMetricValue exactly 1 decimal

Esempio:

```
<icr:ServiceMetric rdf:about="urn:cloudicaro:ServiceMetric:001">
   <icr:atTime rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2013-10-
      2T14:00:34</icr:atTime>
   <icr:hasMetricName>avg CPU% 5min</icr:hasMetricName>
   <icr:hasMetricUnit>percent</icr:hasMetricUnit>
   <icr:hasMetricValue
      rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">20</icr:hasMetricValue>
   <icr:dependsOn rdf:resource="urn:cloudicaro:VirtualMachine:001" />
</icr:ServiceMetric>
```

2.2.1.25 MonitorInfo

Rappresenta le informazioni necessarie al monitoraggio di un servizio.

MonitorInfo EquivalentTo

hasMetricName exactly 1 string and
hasArguments max 1 string and
hasCheckInterval max 1 decimal and
hasCriticalValue max 1 decimal and
hasMaxCheckAttempts max 1 positiveInteger and
hasWarningValue max 1 decimal and
hasMonitorState max 1 string and
hasCheckMode max 1 string

Esempio:

```
<icr:MonitorInfo rdf:about="urn:cloudicaro:MonitorInfo:001">
   <icr:hasMetricName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">metric1
      </icr:hasMetricName>
   <icr:hasCriticalValue
      rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">20</icr:hasCriticalValue>
</icr:ServiceMetric>
```

2.2.2 Proprietà

Le proprietà tra oggetti sono le seguenti:

- **administeredBy –** indica che una entità e' amministrata da un Administrator
  - Range: Administrator
- **createdBy –**indica che una entità e' stata creata da un Creator
  - subProperty of adminsteredBy
  - Range: Creator
- **developedBy –** indica che una IcaroApplication e' stata sviluppata da un Developer
  - Domain: IcaroApplication
  - Range: Developer
- **subscribedBy** – indica che una entità e' stata sottoscritta da un Subscriber
  - Range: Subscriber

- ***hasPart*** – indica che una entità ha come sotto parte un'altra entità
  - inverse of isPartOf
- ***isPartOf*** **–** indica che una entità e' parte di un'altra entità
  - inverse of hasPart

- **dependsOn** – indica una dipendenza tra entità

- **hasOS** – indica che una VirtualMachineTemplate ha un certo sistema operativo installato
  - Domain: VirtualMachineTemplate
- **supportsLanguage** – indica che un certo servizio supporta un certo linguaggio di sviluppo
  - Domain: IcaroService

- **hasSLA** – indica quale e' la SLA supportata da una applicazione
  - Range: ServiceLevelAgreement
- **hasSLObjective** – indica il ServiceLevelObjective di una SLA
  - Domain: ServiceLevelAgreement
  - Range: ServiceLevelObjective
- **hasSLMetric** – indica una ServiceLevelMetric associata a uno SLO
  - Domain: ServiceLevelObjective
  - Range: ServiceLevelMetric
- **hasSLAction** – indica una azione associata a uno SLO
  - Domain: ServiceLevelObjective
  - Range: ServiceLevelAction
- **hasMonitorInfo** – indica le informazioni di Monitoring da usare per l'elemento
  - Range: MonitorInfo

- **needs** – indica che una IcaroApplication fa uso di un IcaroService o di un'altra IcaroApplication
  - Domain: IcaroApplication
  - Range: IcaroApplication or IcaroService or IcaroApplicationModule
- **runs** – indica una VirtualMachine esegue un certo IcaroService
  - inverse of: runsOnVM
  - Domain: VirtualMachine
  - Range: IcaroService
- **runsOnVM** – indica che un certo IcaroService viene eseguito su una certa VirtualMachine
  - inverse of: runs
  - Domain: IcaroService
  - Range: VirtualMachine
- **useVM** – indica quali VirtualMachines sono usate da una IcaroApplication
  - Property chain: needs o runsOnVM

Mentre le proprietà che associano un oggetto a dati semplici (DataProperties) sono le seguenti:

- *atTime* – indica la data e ora associata ad una entità
  - Range: xsd:dateTime
- *hasCPU* – indica il numero di CPU configurate per un certa VirtualMachine
  - Domain: VirtualMachineTemplate
  - Range: xsd:integer
- *hasDiskSize* – indica la dimensione in GB del disco
  - Domain: VirtualMachineTemplate
  - Range: xsd:decimal
- *hasIPAddress* – indica l'indirizzo IP associato ad una VM
  - Range: xsd:string
- *hasMemorySize* – indica la memoria associata a una VM

    - Domain: VirtualMachineTemplate
    - Range: xsd:integer
- *hasActionName* – indica il nome di una azione associata ad una ServiceLevelAction
    - Domain: ServiceLevelAction
    - Range: xsd:string
- *hasMetricName* – indica il nome di una metrica
    - Domain: ServiceLevelMetric or ServiceMetric
    - Range: xsd:string
- *hasMetricUnit* – indica l'unita' di misura di una metrica
    - Domain: ServiceLevelMetric or ServiceMetric
    - Range: xsd:string
- *hasMetricValue* – indica il valore di una metrica
    - Domain: ServiceLevelMetric or ServiceMetric
    - Range: xsd:decimal
- *hasMetricValueLessThan* – indica che il valore deve essere inferiore ad un valore fornito
    - Domain: ServiceLevelMetric
    - Range: xsd:decimal
- *hasMetricValueMoreThan* – indica che il valore di una metrica deve essere superiore a quello fornito
    - Domain: ServiceLevelMetric
    - Range: xsd:decimal
- *usesTcpPort* – indica che un servizio usa una o più porte TCP
    - Domain: IcaroService
    - Range: xsd:integer
- *usesUdpPort* – indica che un servizio usa una o più porte UDP
    - Domain: IcaroService
    - Range: xsd:integer
- *hasName* – indica il nome di una entita'
    - Range: xsd:string
- *hasIdentifier* – indica un identificatore di una entita'
    - Range: xsd:string
- *hasType* – indica il tipo di una entita'
    - Range: xsd:string
- *hasCapacity* – indica la capacita' di una entita'
    - Range: xsd:decimal
- *hasCheckMode* – indica la modalita' di check di una metrica da monitorare
    - Domain: MonitorInfo
    - Range: xsd:string
- *hasMonitorState* – indica lo stato di monitoring della entita' (enabled/disabled)
    - Range: xsd:string
- *hasArguments* – indica gli argomenti da usare per il plugin di monitoring
    - Range: xsd:string
- *hasAuthUsername* – indica il nome utente per autenticarsi al servizio da monitorare
    - Range: xsd:string
- *hasAuthUserPassword* – indica la password da usare per autenticarsi al servizio da monitorare
    - Range: xsd:string
- *hasCheckInterval* – indica il numero di minuti da aspettare prima di controllare nuovamente il servizio
    - Range: xsd:decimal

- *hasMaxCheckAttempts* – indica il numero massimo di volte in cui provare a effettuare il monitoraggio del servizio
    - Range: xsd:nonNegativeInteger
- *hasCriticalValue* – indica il valore critico da usare per il monitoraggio del servizio
    - Range: xsd:decimal
- *hasWarningValue* – indica il valore di attenzione da usare per il monitoraggio del servizio
    - Range: xsd:decimal
- *has*ProcessName – indica il nome del processo da usare per il monitoraggio del servizio
    - Range: xsd:string
- *isInDomain* – indica dominio dove si trova il servizio da monitorare
    - Range: xsd:string

3. Reasoner

3.1 Analisi stato dell'arte

Il Reasoner ha la necessità di elaborare le informazioni nella KB per dedurre nuove informazioni ma soprattutto per evidenziare inconsistenze.
Nel caso di una KB realizzata come un RDF store la conoscenza è rappresentata come triple <soggetto> <predicato> <oggetto> secondo una ontologia definita. Un'ontologia può essere definita tramite lo standard OWL [owl] che si basa sulla Description Logic [dl] che e' più espressiva della logica proposizionale e ha una procedura di decisione più efficiente della logica del primo ordine. La Description Logic è in realtà una famiglia di linguaggi più o meno espressivi che hanno procedure di decisione più o meno efficienti, anche lo standard OWL definisce una serie di profili OWL-Full, OWL-DL e OWL-Lite in ordine decrescente di espressività e complessità. OWL-Full ha il massimo della espressività ma non ha una procedura di decisione decidibile, quando per esempio c'e' sovrapposizione tra TBOX e ABOX cioè quando qualcosa e' sia una classe sia un'istanza di una classe, OWL-DL rappresenta invece la parte di OWL che rappresenta la Description Logic che ha procedura di decisione decidibile mentre OWL-Lite semplifica OWL-DL con alcune limitazioni per rendere più efficiente il reasoning.
Inoltre anche OWL2 [owl2][[owl2-profiles] definisce tre profili tali da avere procedure di decisione di complessità ragionevole, i profili sono: OWL2-EL, OWL2-QL, OWL2-RL.

OWL-EL è pensato per ontologie che hanno un gran numero di classi organizzate in una gerarchia complessa. I seguenti costrutti NON sono supportati da OWL2 EL:

- universal quantification to a class expression (ObjectAllValuesFrom) or a data range (DataAllValuesFrom)
- cardinality restrictions (ObjectMaxCardinality, ObjectMinCardinality, ObjectExactCardinality, DataMaxCardinality, DataMinCardinality, and DataExactCardinality)
- disjunction (ObjectUnionOf, DisjointUnion, and DataUnionOf)
- class negation (ObjectComplementOf)
- enumerations involving more than one individual (ObjectOneOf and DataOneOf)
- disjoint properties (DisjointObjectProperties and DisjointDataProperties)
- irreflexive object properties (IrreflexiveObjectProperty)
- inverse object properties (InverseObjectProperties)
- functional and inverse-functional object properties (FunctionalObjectProperty and InverseFunctionalObjectProperty)
- symmetric object properties (SymmetricObjectProperty)
- asymmetric object properties (AsymmetricObjectProperty)

OWL2 QL e' pensato per ontologie che hanno un gran numero di istanze e un numero limitato di classi. I seguenti costrutti NON sono supportati da OWL 2 QL:

- existential quantification to a class expression or a data range (ObjectSomeValuesFrom and DataSomeValuesFrom) in the subclass position
- self-restriction (ObjectHasSelf)
- existential quantification to an individual or a literal (ObjectHasValue, DataHasValue)
- enumeration of individuals and literals (ObjectOneOf, DataOneOf)
- universal quantification to a class expression or a data range (ObjectAllValuesFrom, DataAllValuesFrom)
- cardinality restrictions (ObjectMaxCardinality, ObjectMinCardinality, ObjectExactCardinality, DataMaxCardinality, DataMinCardinality, DataExactCardinality)
- disjunction (ObjectUnionOf, DisjointUnion, and DataUnionOf)
- property inclusions (SubObjectPropertyOf) involving property chains
- functional and inverse-functional properties (FunctionalObjectProperty, InverseFunctionalObjectProperty, and FunctionalDataProperty)
- transitive properties (TransitiveObjectProperty)
- keys (HasKey)
- individual equality assertions and negative property assertions

Il profilo **OWL 2 RL** e' orientato ad applicazioni che hanno bisogno di reasoning scalabile senza sacrificare troppo la potenza espressiva. Questo e' realizzato definendo un sottoinsieme della sintassi di OWL2 che e' implementabile usando tecnologie di reasoning basate su regole.

I reasoner per OWL si basano su alcuni assunti:

- *Open World Assumption (OWA)*, con il quale le cose non esplicitamente asserite non sono conosciute quindi potrebbero essere vere o false, mentre la *Closed World Assumption* (CWA) le cose non esplicitamente asserite sono false. Questo comporta che i reasoner che si basano su OWA non rilevano delle inconsistenze sulla esistenza e cardinalità delle relazioni, in quanto il fatto che non sia stata asserita un tipo di relazione non vuol dire che non esista.
- *Not Unique Name Assumption,* con la quale ogni entità può essere identificata da più di una URL, anche questa assunzione rende più difficoltoso il controllo di consistenza per esempio quando si hanno vincoli sulla cardinalità massima di una relazione. Se per una relazione e' stata specificata una cardinalità massima di due e per una istanza viene asserita la relazione usando tre URL diverse, non necessariamente questa e' una inconsistenza in quanto due URL potrebbero riferirsi alla stessa entità.

Nel caso si voglia utilizzare un OWL reasoner per identificare inconsistenze nella definizione di istanze dell'ontologia la *Open World Assumption* ed anche la *Not Unique Name Assumption* rendono praticamente impossibile raggiungere dei risultati soddisfacenti. Alcuni reasoner supportano la CWA su alcune parti dell'ontologia come TrOWL [Ren2010]. In [Jiao2009], viene proposto di generare delle query SPARQL sulla base dei vincoli imposti nella definizione dell'ontologia per identificare potenziali problemi nelle istanze.
Recentemente l'RDF Store *StarDog* (sviluppato da Clark & Parsia, gli stessi che sviluppano il reasoner *Pellet*) supporta la possibilità di definire dei vincoli di integrità espressi in OWL ed anche in questo caso per fare queste verifiche usano query SPARQL.

Per le Description Logic sono disponibili alcuni reasoner:

- **BaseVISor**, a versatile forward chaining inference engine specialized to handle facts in the form of RDF triples with support for OWL 2 RL and XML Schema Datatypes (BaseVISor).

- **Bossam** (software), a RETE-based rule engine with native supports for reasoning over OWL ontologies, SWRL rules, and RuleML rules.
- **Cyc** inference engine, a forward and backward chaining inference engine with numerous specialized modules for high-order logic. ([1] ResearchCyc) ([2] OpenCyc)
- **Hoolet**, reasons over OWL-DL ontologies by translating them to full first-order logic and then applying a first-order theorem prover (Hoolet)
- **Pellet**, an open-source Java OWL DL reasoner. (Pellet, AGPL version 3, commercial option available)
- **KAON2** is an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies.
- **RacerPro**, a semantic web reasoning system and information repository (RacerPro)
- **FaCT**, a Lisp-based description logic (DL) classifier. (FaCT, GNU GPL)
- **FaCT++,** the new generation of FaCT OWL-DL reasoner, based on C++. (FaCT++, GNU Lesser GPL)
- **SweetRules**, an integrated set of tools for Semantic web rules and ontologies. (SweetRules, GNU Lesser GPL)
- **OWLIM**, a high-performance semantic repository developed in Java and available in three versions: OWLIM-lite (which is free to download and use) and the commercial OWLIM-se and OWLIM-enterprise. Supports OWL2-rl semantics, which can be configured through rule-set definition and selection (OWLIM)
- **OntoBroker**, highly scalable SemanticWeb middleware (OntoBroker).
- **HermiT**, the first publicly available OWL reasoner based on a novel "hypertableau" calculus which provides much more efficient reasoning than any previously known algorithm. (HermiT, GNU Lesser GPL)

Nella tabella seguente viene riportata una comparazione tra i reasoner

	BaseVISor	Bossam	Cyc	Hoolet	Pellet	KAON2	RacerPro	FaCT	FaCT++	OWLIM	OntoBroker	HermiT
OWL-DL Entailment	No	Unknown	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Supported expressivity for reasoning	R-entailment, OWL 2 RL	Unknown	High-order	Unknown	SROIQ(D)	SHIQ(D)	SHIQ(D-)	SHIQ	SROIQ(D)	R-entailment, OWL 2 RL	**OWL:** SHIQ(D) (for OntoBroker 6.1: Subset of OWL-RL); **F-logic:** normal logic, wellfounded semantics	SHOIQ+
Reasoning algorithm	Rule-based, Rete	Rule-based	First-order with high-order extensions	First-order prover	Tableau	Resolution & Datalog	Tableau	Tableau	Tableau	Rule-based	**OWL:** Resolution & Datalog; **F-logic:** Rule-based (BottomUp, MagicSet, Dynamic-Filtering, QSQ)	Hypertableau
Consistency checking	Yes	Unknown	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DIG Support	No	No	No	No	Deprecated	Deprecated	Deprecated	Yes	Yes	No	Yes	No
Rule Support	Yes (Own rule format)	Yes (SWRL & own rule format)	Yes (Own rule format)	Yes (SWRL)	Yes (SWRL—DL Safe Rules)	Yes (SWRL—DL Safe Rules)	Yes (SWRL—not fully supported	No	No	Yes (Own format)	Yes (SWRL, RIF, F-logic, ObjectLogic)	Yes (SWRL—DL Safe Rules)

							) & own rule format					
Version	2.0	0.9b45	Unknown	Unknown	2.0.2	2008-06-29	2.0 preview	Unknown	1.6.0	2.x/3.x	6.1	1.3.6
Licensing	Free to use / closed-source	Free to use / closed-source	Free to use / closed-source (academic and research only) & Non-Free / closed-source	Free / open-source	Free / open-source & Non-Free/ closed-source	Free to use / closed-source	Non-Free/ closed-source	Free / open-source	Free / open-source	Non-Free/ closed-source	Non-Free/ closed-source	Free / open-source

Tabella 1- Comparazione tra reasoner (Sorgente Wikipedia)

Per la classificazione delle logiche descrittive viene usata una sigla in cui ogni lettera ha un significato (http://en.wikipedia.org/wiki/Description_logic):

- $\mathcal{AL}$: indica la logica degli attributi e introduce gli operatori di congiunzione e di quantificatori universale ed esistenziale;
- $\mathcal{C}$: descrive la possibilità di usare l'operatore di negazione;
- $\mathcal{S}$: estende la DL $\mathcal{ALC}$ con l'ulteriore possibilità di definire la chiusura transitiva di un ruolo;
- $\mathcal{H}$: fornisce la possibilità di definire gerarchie tra ruoli;
- $\mathcal{R}$: permette di definire ruoli riflessivi/irriflessivi, disgiunzione tra ruoli;
- $\mathcal{O}$: asserisce la presenza dell'operatore di enumerazione;
- $\mathcal{I}$: permette di riferirsi al ruolo inverso;
- $\mathcal{F}$, $\mathcal{N}$ e $\mathcal{Q}$ caratterizzano le possibilità di definire cardinalità rispettivamente funzionale, semplice e qualificata (in ordine di espressività crescente);
- $\mathcal{D}$ descrive la possibilità di riferirsi a domini concreti.

In quest'ottica, ad esempio, il sottolinguaggio di OWL "OWL DL" di può essere considerato equivalente ad una logica descrittiva di tipo $\mathcal{SHOIN}(\mathcal{D})$.

Per quanto riguarda l'efficienza del reasoning alcuni risultati sono presenti in letteratura [Dentler2010], [Bock2008], riguardano reasoning a livello di TBox, cioè sulla struttura dell'ontologia e considerano soddifacibilita' (possono esistere individui in una classe?), sussunzione (una classe e' contenuta in un'altra?) e reasoning a livello di ABox, cioè sulle istanze e considerano principalmente consistenza (le istanze sono consistenti con il TBox?) e *conjuntive queries* (quali istanze soddisfano una query?).
Inoltre va considerato che i reasoner sono legati al livello espressivo delle ontologie che riescono a gestire, quindi nella scelta di un reasoner bisogna tenere conto del livello espressivo usato nell'ontologia.

OWL API (http://owlapi.sourceforge.net/), e' un insieme di interfacce Java che astraggono dal tipo specifico di reasoner e permettono alle applicazioni di poter usare diversi reasoner senza cambiare il codice in modo pesante, inoltre permettono di manipolare una ontologia (aggiungere/togliere elementi). E' supportata da:

- FaCT++.
- HermiT
- Pellet
- RacerPro (via OWLLink)

Anche la *DIG Interface* (http://dig.sourceforge.net) e' un'altra interfaccia java che può essere usata per interfacciarsi con i reasoner.

OWLLink (http://www.owllink.org) è un altro modo per interfacciarsi con i reasoner attraverso un protocollo basato su HTTP/XML.

3.2 Requisiti funzionali

NR	Ruolo Utente	Requisito	Descrizione	Priorità
RSN1	--	Controllo consistenza	Il Reasoner deve controllare se una configurazione e' consistente internamente cioè se i servizi usati sono validi e compatibili tra loro	alta
RSN2	--	Scalabilità	Il Reasoner deve controllare se le condizioni di scaling/descaling per una data applicazione (se presenti) sono soddisfatte ed indicare la regola da eseguire per attivare un nuovo host o per disattivarlo	media
RSN3	--	SLA checking	Il Reasoner deve controllare se le SLA delle varie applicazioni sono state invalidate e in caso positivo indicare l'azione da fare	alta
RSN4	--	Ottimizzazione	Il reasoner deve permettere anche la ottimizzazione delle risorse hardware, identificare una nuova configurazione equivalente tale da usare meno macchine e risparmiare energia mantenendo inalterato il livello di servizio	alta

3.3 Requisiti non funzionali

NR	Ruolo Utente	Requisito non funzionale	Descrizione	Priorità
RSN5	--	Performance	Il reasoner deve avere una ottima/buona performance	alta

4. Bibliografia

[Zhang2012] Zhang, M.; Ranjan, R.; Haller, A.; Georgakopoulos, D.; Menzel, M.; Nepal, S., "An ontology-based system for Cloud infrastructure services' discovery," 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 , pp.524,530, 14-17 Oct. 2012

[Moscato2011] Moscato, F.; Aversa, R.; Di Martino, B.; Fortis, T.; Munteanu, V., "An analysis of mOSAIC ontology for Cloud resources annotation," Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on , vol., no., pp.973,980, 18-21 Sept. 2011

[Dobson2005] G. Dobson, et al., "QoSOnt: a QoS ontology for service-centric systems," in Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on, 2005, pp. 80-87

[Ludwig2003] H. Ludwig, A. Keller, A. Dan, et al, Web Service Level Agreement (WSLA) Language Specification, January 28 2003, available at: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[Andrieux2007] Andrieux, K. Czajkowski, A. Dan, et al, Web Services Agreement Specification (WS-Agreement), March 14 2007, available at: http://www.ogf.org/documents/GFD.107.pdf

[Lamanna2003] D.D. Lamanna, J. Skene & W. Emmerich, SLAng: A language for defining service level agreements, in Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS, 2003, pages 100-106

[Kearney2010] Kearney, K.T.; Torelli, F.; Kotsokalis, C., "SLA*: An abstract syntax for Service Level Agreements," Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on , pp.217,224, 25-28 Oct. 2010

[Buscemi2007] M. Buscemi & U. Montanari, CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements, in Programming Languages and Systems, 2007, pages 18-32.

[Patel2009] Patel, Pankesh, Ajith Ranabahu, and Amit Sheth. "Service level agreement in cloud computing." Cloud Workshops at OOPSLA. 2009.

[Wieder2011] Service Level Agreements for Cloud Computing, Wieder, P.; Butler, J.M.; Theilmann, W.; Yahyapour, R. (Eds.), Springer, 2011

[Oldham2006] Oldham, N., Verma, K., Sheth, A., and Hakimpour, F. 2006. Semantic WS-agreement partner selection. In *Proceedings of the 15th International Conference on World Wide Web* (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 697-706.

[Maxemilien2004] Maxemilien, M., Singh, M., A Framework and Ontology for Dynamic Web Services Selection. IEEE Internet Computing 8(5):84-93, September-October 2004

[owl] OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/

[dl] http://en.wikipedia.org/wiki/Description_logic

[owl2] OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation 11 December 2012, http://www.w3.org/TR/owl2-overview/

[owl2-profiles] OWL 2 Web Ontology Language Profiles (Second Edition), W3C Recommendation 11 December 2012, http://www.w3.org/TR/owl2-profiles/

[Dentler2010] Kathrin Dentler, Ronald Cornet, Annette ten Teije and Nicolette de Keizer, "Comparison of Reasoners for large Ontologies in the OWL 2 EL Profile", Semantic Web – Interoperability, Usability, Applicability *an IOS Press Journal, 2010*

[Bock2008] Jürgen Bock , Peter Haase , Qiu Ji , Raphael Volz, "Benchmarking OWL Reasoners", Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (June 2008)

[Ren2010] Yuan Ren, Jeff Z. Pan, Yuting Zhao, "Closed World Reasoning for OWL2 with NBox*", Tsinghua Science & Technology, Volume 15, Issue 6, December 2010, Pages 692-701

[Jiao2009] Jiao Tao; Li Ding; McGuinness, D.L., "Instance Data Evaluation for Semantic Web-Based Knowledge Management Systems," System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on , vol., no., pp.1,10, 5-8 Jan. 2009