

Appendice B

Descrizione OWL-S

Redatto da:

Bianca Lo Cascio
Davide Rizzo

Sommario

1	INTRODUZIONE	3
2	DESCRIZIONE OWL-S.....	4
2.1	SERVICE PROFILE.....	4
2.2	SERVICE MODEL.....	8
2.3	SERVICE GROUNDING.....	10
3	RISORSE.....	12
3.1	TIPI DI ALLOCAZIONE DELLE RISORSE.....	12
3.2	TIPI DI CAPACITÀ.....	13
3.3	COMPOSIZIONE DELLE RISORSE.....	13
3.4	ALGORITMO PER IL MATCHING.....	14
3.5	TRADUZIONE IN JADE.....	17
3.6	CONCLUSIONI SULL'APPLICAZIONE DI OWL-S AL PROGETTO ATENA.....	21

1 Introduzione

Nel campo del Web è in atto un cambio radicale di prospettiva, infatti si sta assistendo al passaggio dalla concezione della rete Internet intesa come collezione di pagine Web alla concezione di Internet intesa come collezione di servizi che interoperano. I servizi Web suggeriscono un nuovo modello del Web, in cui i siti a richiesta si scambiano in modo dinamico delle informazioni. Soprattutto questa nuova prospettiva interessa l'e-business, in quanto permetterebbe di condurre gli affari in modo più efficiente e veloce.

Serve un linguaggio per esprimere le caratteristiche dei servizi.

Come parte del programma DARPA Agent Markup Language, è stato sviluppato dagli stessi sviluppatori un'ontologia dei servizi, chiamata OWL-S, la cui versione precedente era il DAML-S. Noi ci riferiremo all'ultima versione dell'OWL-S, ovvero ad OWL-S 1.0, disponibile al <http://www.daml.org/services>.

Gli scopi dell'OWL-S sono:

- **Individuazione automatica delle risorse Web**, che soddisfino determinate condizioni. Un modo per ottenere ciò è possedere un server con comportamento *proattivo*, che usi un registro dei servizi, chiamato *middle-agent*.
- **Invocazione automatica dei Web service**. L'OWL-S fornisce delle API dichiarative e interpretabili da un computer per l'esecuzione automatica delle chiamate delle funzioni. Un agente software sarà capace di interpretare le richieste che gli sopravvivono, in modo da fornire le informazioni richieste e il servizio, tutto automaticamente.
- **Composizione e interoperazione dei servizi Web automatici**. Comprende la selezione, la composizione e l'interoperazione dei servizi Web per il compimento di qualche task. Per realizzare ciò OWL-S deve esplicitare i prerequisiti e le conseguenze di ogni servizio.
- **Monitoraggio dell'esecuzione dei servizi Web automatici**. Può essere utile per l'utente conoscere lo stato della sua richiesta, o può richiedere delle modifiche alla richiesta. Per questo motivo l'OWL-S fornisce dei descrittori dichiarativi per lo stato dell'esecuzione dello stato.

Ogni programma, sensore, dispositivo accessibile via Web deve essere dichiarato come servizio.

La forma più diffusa per la registrazione dei servizi Web, che presenta parecchie analogie con XML, è UDDI (Universal Description, Discovery and Integration), che è un registro che descrive i servizi (soprattutto legati al business) tramite i loro attributi concreti come il nome, indirizzo e i servizi offerti (stile database).

Inoltre le descrizioni in UDDI sono arricchite da un insieme di attributi chiamati TModels, che descrivono caratteristiche aggiuntive come la classificazione entro una tassonomia come N/AICS (North American Industry Classification System), tramite la quale un servizio può specificare la sua categoria di appartenenza.

L'UDDI ha lo scopo di creare un ampio registro Internet dei servizi Web. Tramite i TModels, UDDI supporta la registrazione degli attributi dei servizi. Un TModel è un *metadato* che fornisce informazioni sul servizio. In genere i TModels hanno due funzioni:

- etichettare il tipo di servizi pubblicizzato, specificando delle convenzioni da usare con UDDI.
- fornire chiavi d'accesso astratte da associare ad uno specifico valore di servizio.

L'integrazione tra UDDI e OWL-S consiste nel pubblicizzare i servizi tramite descrizioni semantiche OWL-S all'interno del registro UDDI, e nel trovare tramite le parole-chiave dell'UDDI.

L'OWL-S si presenta molto differente da linguaggi simili all'XML, come SOAP e WSDL che sono stati progettati per fornire descrizioni dei meccanismi di trasporto dei messaggi e per descrivere le interfacce usate da ogni servizio, infatti né SOAP né WSDL forniscono le basi per la collocazione automatica dei servizi in base alle loro caratteristiche.

Al fine di identificare i servizi dal punto di vista semantico non possono venire in aiuto i linguaggi di tipo XML perché ad esempio due descrizioni XML identiche possono avere un significato diverso in base al contesto. Ovviamente bisogna essere realistici e non aspettarsi una perfetta corrispondenza tra le richieste e le offerte dei servizi.

OWL-S collabora con DAML+OIL per la rappresentazione semantica dei servizi. DAML+OIL supporta il ragionamento a sussunzione sulle tassonomie di concetti e permette la definizione di relazioni tra i concetti. La maggior limitazione di DAML+OIL è la mancanza di formule ben formate.

2 Descrizione OWL-S

Vengono ora presentate le principali componenti della descrizione OWL-S.

2.1 Service Profile

Lo scopo di un Service Profile è di descrivere le funzionalità che un servizio Web fornisce alla comunità degli agenti.

La figura 1 mostra l'ontologia di alto livello per il Service Profile:

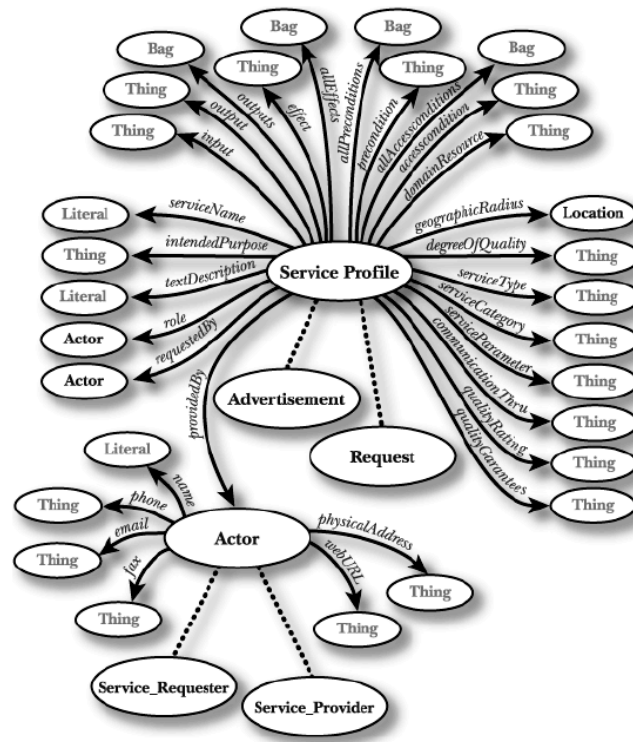


Figura 1 - Upper Ontology del Service Profile

La figura 1 è divisa logicamente in tre parti:

- in alto la figura presenta una *Functional Description* del servizio. Essa descrive i servizi in termini di input e output, precondizioni e effetti.
- al centro si descrivono i *Functional Attributes* come il *Quality Rating*, cioè la categoria assegnata al servizio, oppure come il *Geographic Radium*, che specifica se ci sono vincoli geografici nel servizio.
- in basso vi è la definizione dell'*Actor*, in cui si registrano le informazioni sul fornitore del servizio.

Riferendoci alla Figura 1, non riempiamo i campi relativi ad Actor, in quanto non vi è mai interazione tra una servizio ed un attore, poiché questi si limita (in taluni casi) a lanciare il servizio stesso e manca un'interazione diretta tra i due fintantoché il servizio stesso non è portato a termine. Per chiarire tale definizione vengono forniti due esempi: nel caso della fornitura di un servizio per aste elettroniche, è plausibile ipotizzare l'impiego di un agente che contratti i prezzi di acquisto di determinati lotti interagendo con altre entità (agenti, attori, ecc) e che, saltuariamente, richieda l'interazione con l'attore che ha lanciato il servizio stesso (ad esempio se i parametri di input non permettono l'acquisto, il servizio potrebbe chiedere asincronamente al proprio attore la possibilità di aumentare il proprio capitale a disposizione). In tal caso quindi il servizio, per il suo normale processo, può avere necessità di interagire con un attore. Nel secondo caso (programma Atena), i servizi forniti interagiscono (eventualmente) con un attore soltanto alle interfacce del servizio stesso, e mai durante le operazioni ad esso associate.

La richiesta consiste in una descrizione di un ipotetico servizio che svolge un compito richiesto dal richiedente.

Un profilo OWL-S descrive un servizio mediante tre informazioni:

1. quale *organizzazione* fornisce il servizio: fornisce informazioni sulle entità che vengono contattate nel prestare un servizio, come ad esempio l’informazione sull’operatore che fornisce assistenza.
2. quale *funzione* svolge il servizio: la descrizione funzionale del servizio è espressa in termini della trasformazione prodotta dal servizio. In particolare specifica gli input richiesti dal servizio e gli output generati, quindi descrive le precondizioni e gli effetti.
3. *caratteristiche* del servizio: comprendono anche il livello di qualità.

2.1.1 Tipo di organizzazione fornita dal servizio (descrizione non funzionale)

serviceName	ChangeExamDate	GetInscriptionList	GetQuestionsList	SendMail	UpdateInscriptionList	AccessToDBMS
textDescription	Servizio che permette il cambio di una data d’esame, tra le date disponibili. Bisogna fornire in ingresso la data dell’esame da cambiare e quella che si propone in alternativa.	Servizio per l’ottenimento della lista degli iscritti ad un esame. Bisogna fornire in ingresso l’esame di cui si desidera ricevere la lista degli iscritti.	Servizio per l’ottenimento della lista dei quesiti d’esame. Bisogna fornire in ingresso la materia di cui si vogliono visualizzare i quesiti d’esame.	Servizio per l’invio di e-mail. Bisogna fornire in ingresso un identificativo dell’utente a cui inviare la e-mail.	Servizio per l’aggiornamento della lista degli iscritti. Bisogna fornire in ingresso la nuova lista degli iscritti ad un esame.	Servizio per accedere al DBMS in lettura e in scrittura.
contactInformation	Atena&C.	Atena&C.	Atena&C.	Atena&C.	Atena&C.	Atena&C.
geographicRadius	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo	Container registrati nella piattaforma appartenenti alla intranet dell’Università di Palermo
degreeOfQuality (QoS)	Best effort	Best effort	Best effort	Best effort	Best effort	Best effort
qualityGuarantees		Authoritative	Authoritative		Authoritative	Authoritative
Altri parametri	N/A	N/A	N/A	N/A	N/A	N/A

Nel seguente diagramma delle classi (Figura 2) si evidenziano i servizi offerti all’interno del sistema Atena, gli agenti richiedenti e gli agenti fornitori di un dato servizio.

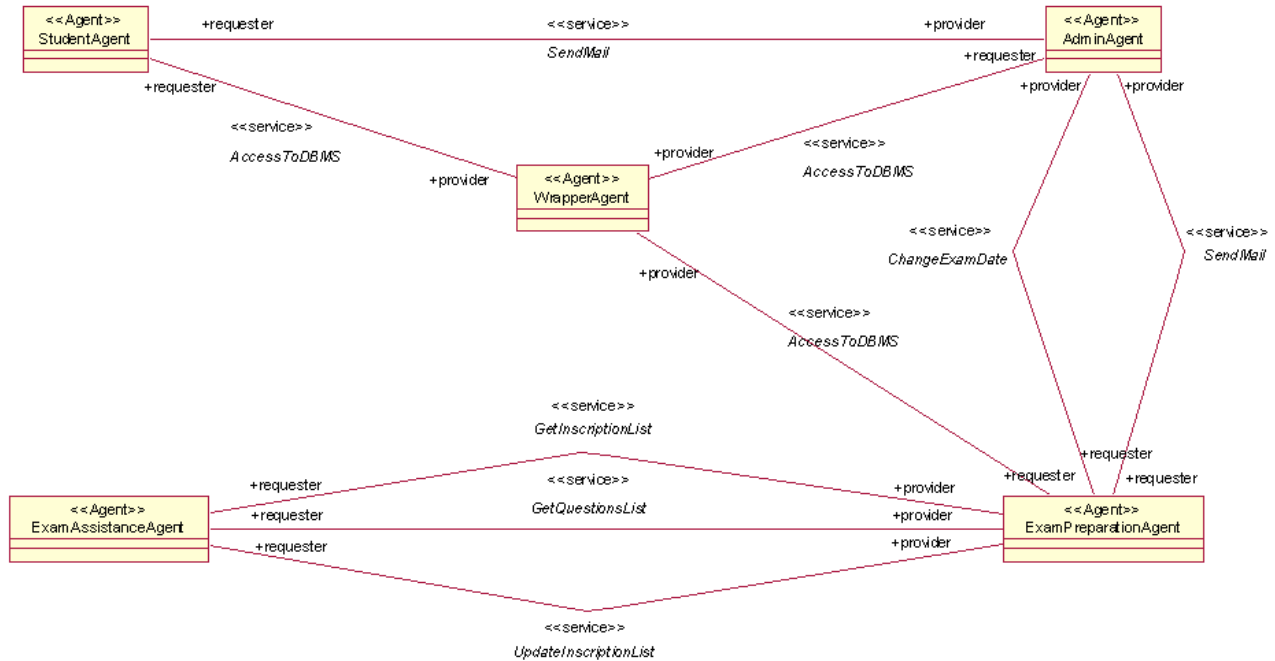


Figura 2 - I servizi in ATENA

2.1.2 Funzioni fornite dal servizio (descrizione funzionale)

Il profilo OWL-S presenta due aspetti della funzionalità di un servizio: l’informazione sulla trasformazione (input e output) e il cambio di stato prodotto dall’esecuzione del servizio (precondizioni e effetti).

serviceName	ChangeExamDate	GetInscriptionList	GetQuestionsList	SendMail	UpdateInscriptionList	AccessToDBMS
Input	Data precedente, data nuova	Esame	Materia	Utente	Nuova lista degli iscritti	Dati che identifichino un record di un database da leggere o da scrivere
Output	Esito dell’operazione	Lista degli iscritti all’esame dato.	Lista dei quesiti della materia data.	Esito dell’operazione	Esito dell’operazione	Dato letto dal database o esito della scrittura
Precondizioni	Deve esistere almeno una data d’esame per la materia in questione	Deve esistere l’esame di cui si chiede la lista degli iscritti.	Deve esistere la materia.	L’utente deve avere fornito il proprio indirizzo email.	Deve essere schedulato l’esame di cui si vuole aggiornare l’elenco degli iscritti.	Deve esistere il record coinvolto in lettura o scrittura
Effetti	Cambio della data di un esame	Ottenimento della lista degli iscritti all’esame dato	Ottenimento della lista dei quesiti della materia data.	Invio di una e-mail all’utente desiderato.	Aggiornamento della lista degli iscritti ad un esame	Letture o scrittura di uno o più record

Profile attributes

serviceParameter: lista di proprietà, ognuna delle quali istanzia ServiceParameter.

serviceCategory: si riferisce ad un elemento di un’ontologia o una tassonomia.

ServiceParameter

serviceParameterName: nome del parametro attuale, che può essere costituito da un letterale o da un URI.

sParameter: punta al valore del parametro dentro la OWL ontology.

ServiceCategory

Descrizione delle categorie di servizi sulla base di una classificazione indipendente sia da OWL-S che da OWL.

categoryName: è il nome della categoria attuale.

taxonomy: registra un riferimento allo schema della tassonomia.

value: punta al valore della specifica tassonomia. Ci possono essere più di un valore per ogni tassonomia.

code: ad ogni tipo di servizio si registra il codice associato ad una tassonomia.

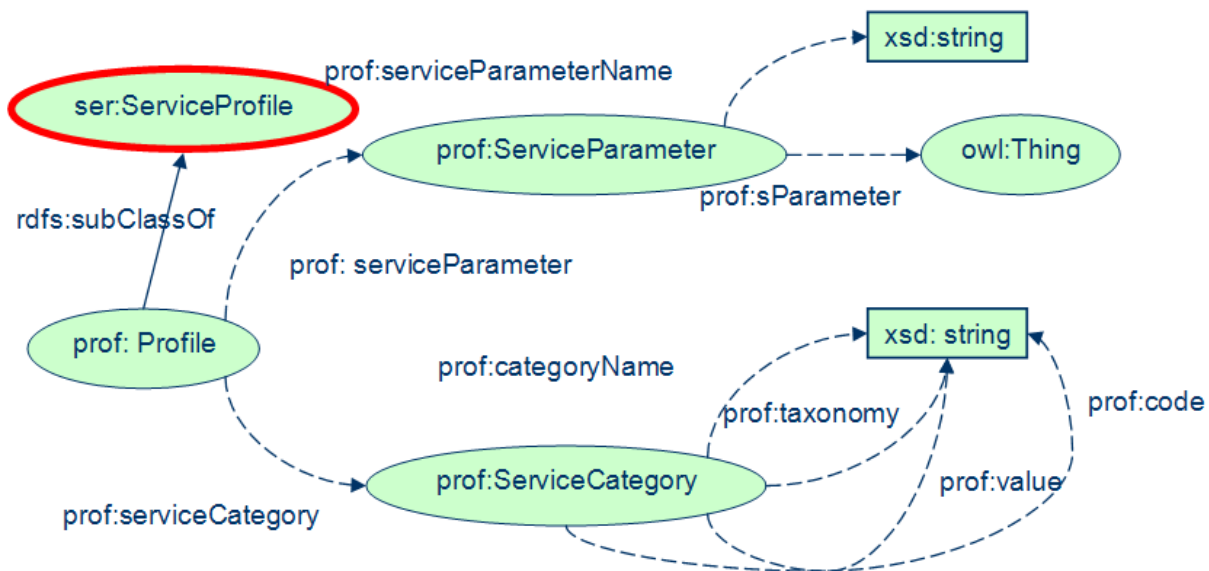


Figura 3 - Tassonomia del Service Profile

Nel progetto Atena, non essendovi una esplicita ontologia dei servizi, i campi sopra menzionati non sono definibili.

2.2 Service Model

L'entità principale del Service Model è il processo (Figura 4). In OWL-S il processo può essere concepito in due modi:

- processo come *trasformazione* dei dati da un insieme di input ad un insieme di output;

- processo come una *transizione* effettuata nel mondo da uno stato all'altro, date alcune precondizioni e prodotti alcuni effetti.

Si distinguono tre tipi di processo: *atomic*, *simple* e *composite*.

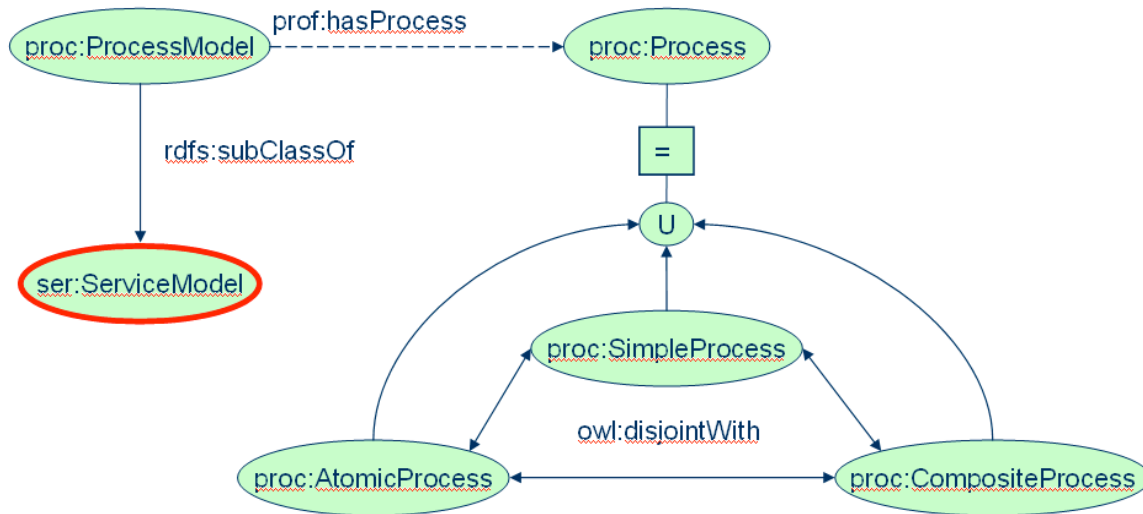


Figura 4 - Tassonomia del Service Model

2.2.1 Processo atomico

I processi atomici sono direttamente invocabili (tramite passaggio di messaggi appropriati). Essi sono eseguiti in un unico passo.

2.2.2 Processo semplice

Essi non sono invocabili direttamente e non sono associati al grounding, ma come quelli atomici sono eseguiti in un unico passo. I processi semplici sono usati come elementi di un'astrazione al fine di fornire una vista di un processo atomico o di uno composto, mediante le relazioni *realizedBy* e *expandsTo*, per le quali:

- il processo semplice è *realizedBy* il processo atomico
- il processo semplice *expandsTo* il processo composto

2.2.3 Processo composto

I processi composti sono scomponibili in altri processi, che a sua volta possono essere composti. La loro composizione viene realizzata tramite costrutti di composizione o di controllo come SEQUENCE e IF-THEN-ELSE. Nella composizione sono identificabili gli input e gli output dei singoli processi componenti. Un processo può essere a vari livelli di astrazione, scegliendo la vista più utile in quel dato

contesto. Ogni costrutto di controllo, usato per la composizione, è associato ad una proprietà chiamata *components* che indica l'ordine e le condizioni d'esecuzione dei sottoprocessi.

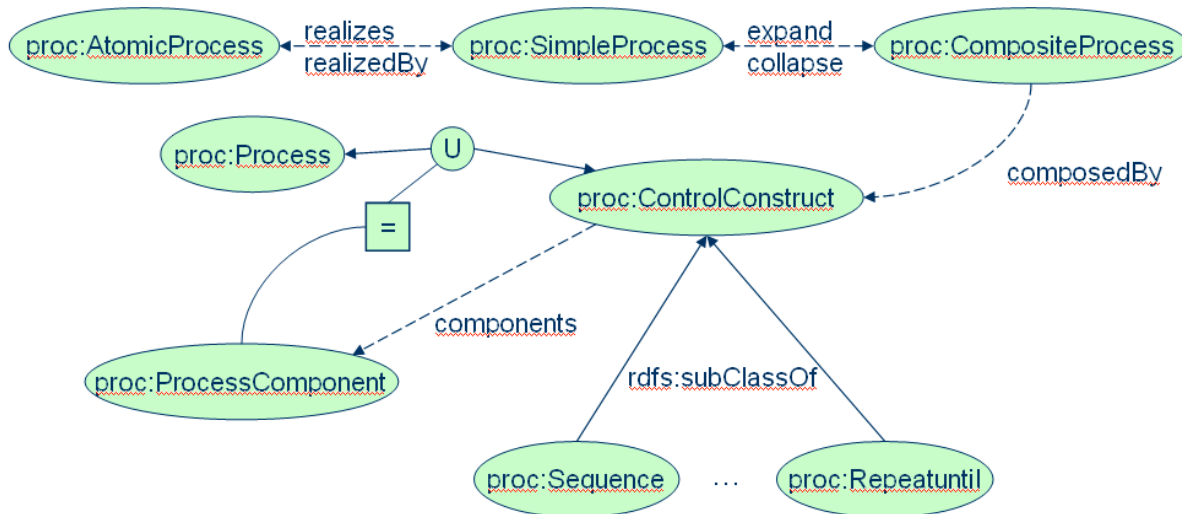


Figura 5 - Tassonomia dei Processi

I servizi Atena sono tutti costituiti da processi atomici, poiché data la relativa semplicità del dominio, non sono stati necessari processi composti.

2.3 Service Grounding

Il ServiceGrounding specifica in dettaglio come un agente può accedere ad un servizio. Esso specifica il protocollo di comunicazione, i formati dei messaggi e altri dettagli come il numero delle porte usate nella comunicazione.

Il grounding si può intendere crea una corrispondenza tra una specifica astratta (quella del ServiceProfile e del ServiceModel) e una concreta (ServiceGrounding). OWL-S non specifica costrutti astratti, piuttosto il contenuto astratto è specificato implicitamente dalle proprietà degli input e degli output di qualche processo atomico. In tal modo i processi atomici possono essere pensati come le primitive della comunicazione di un processo.

I messaggi concreti sono specificati nel grounding. La funzione principale dell'OWL-S è quella di mostrare come gli input e gli output astratti di un processo atomico siano concretizzati sottoforma di messaggi, ovvero in un formato comunicabile.

Nel caso del progetto Atena, tutti i nostri servizi si scambiano comunicazioni nel protocollo FIPA e il formato dei messaggi è RDF.

2.3.1 OWL-S e WSDL

Per descrivere in modo completo il grounding è necessario usare sia l'OWL-S che l'WSDL, perché essi sono complementari.

Tramite l'OWL-S si descrivono i tipi astratti, mentre tramite il WSDL si specificano i dettagli del formato dei messaggi.

Il WSDL (Web Services Description Language) definisce i servizi come una collezione di terminali di rete o porte. Ogni porta è descritta da un indirizzo di rete; insiemi di porte definiscono i servizi.

Esso è in formato XML per descrivere i servizi di rete come un insieme di nodi (*endpoint*) che operano sui messaggi che contengono informazioni sia orientate ai documenti che orientate alle procedure. Le operazioni e i messaggi sono descritti astrattamente e poi legati ad un protocollo concreto di rete e al formato del messaggio per definire un nodo. I nodi concreti relativi sono combinati in nodi astratti (servizi).

Il WSDL non supporta una descrizione semantica dei servizi, bensì si sofferma sul grounding dei servizi e nonostante abbia un concetto di tipi di input e di output definiti da XSD, non supporta la definizione logica dei vincoli tra i parametri di input e di output. Quindi il supporto che fornisce il WSDL per la ricerca e invocazione dei servizi è meno versatile di quello offerto da OWL-S. Per la collaborazione di WSDL e OWL-S, bisogna considerare le seguenti corrispondenze:

1. Un processo atomico OWL-S corrisponde ad un'operazione WSDL.
2. L'insieme di input e di output di un processo atomico OWL-S corrisponde al concetto WSDL di messaggio. Gli input corrispondono alle parti di un messaggio di input di un'operazione WSDL e gli output OWL-S corrispondono alle parti di un messaggio di output di un'operazione WSDL.
3. I tipi (classi OWL) degli input e degli output di un processo atomico OWL-S corrispondono alla nozione WSDL di tipo astratto.

Per costruire un grounding OWL-S/WSDL per prima cosa bisogna identificare in WSDL i messaggi e le operazioni tramite cui si può accedere ad un processo atomico e poi le corrispondenze.

Per realizzare una descrizione in WSDL bisogna definire tutte le sue parti:

- **Type:** è il contenitore per le definizioni di tipi di dato. Ogni definizione fa riferimento ad un particolare sistema di tipi, come ad esempio XSD.
- **Message:** una definizione astratta e tipizzata dei flussi di dati da e verso il servizio
- **Operation:** una descrizione astratta di un'azione svolta dal servizio
- **Port type:** un insieme di operazioni supportate da uno o più nodi
- **Binding:** un protocollo concreto ed una specifica del formato dei dati per un particolare port type
- **Port:** un singolo nodo definito come la combinazione di un binding e di un indirizzo di rete
- **Service:** una collezione di nodi (interdipendenti)

	Type	Message	Operation	Port type	Binding	Port	Service
ChangeExamDate	ElencoEsamiPerData.xsc	previousDate: Date newDate: Date	Vedi pseudocodice1	N/A	N/A	N/A	N/A
GetInscriptionList	ElencoIscritti.xsd	exam: Exam	Vedi pseudocodice2	N/A	N/A	N/A	N/A
GetQuestionsList	ListaDomande.xsd	subject: Subject	Vedi pseudocodice3	N/A	N/A	N/A	N/A
SendMail	DatiUniversitari.xsd ElencoDocenti.xsd Email.xsd Recapito.xsd	user: User	Vedi pseudocodice4	N/A	N/A	N/A	N/A
UpdateInscriptionList	ElencoIscritti.xsd	newInscriptionList: InscriptionList	Vedi pseudocodice5	N/A	N/A	N/A	N/A
AccessToDBMS	N/A	value: Object	Vedi pseudocodice6	N/A	N/A	N/A	N/A

Nota: gli ultimi quattro parametri non sono definiti in quanto i servizi forniti non sono distribuiti ma risiedono di volta in volta su un unico nodo.

Il grounding WSDL fornisce una ontologia per associare ogni frammento OWL-S ad un frammento WSDL

- Un processo atomico con input e output corrisponde a una operazione request-response.
- Un processo atomico con input ma senza output corrisponde ad una operazione one-way.
- Processi con output ma senza input corrispondono alle operazioni di notifica.
- Un processo composto con input ed output, e con l’invio dell’output precedente alla ricezione dell’input è un’operazione solicit-response.

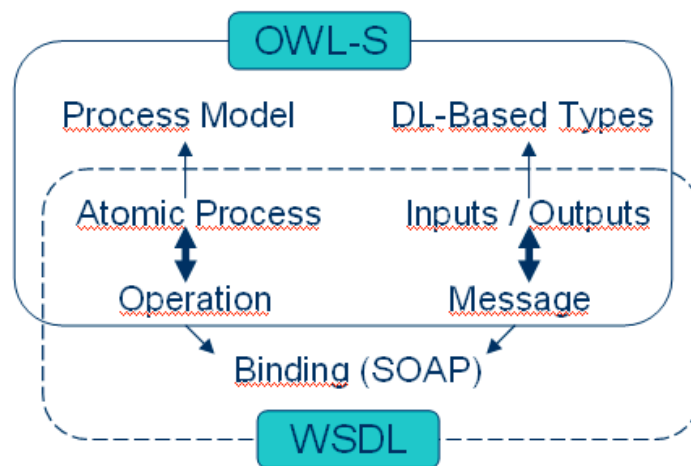


Figura 6 - Ralazione tra OWL-S e WSDL

3 Risorse

I servizi sono offerti dai processi e i processi generalmente richiedono risorse. Quindi un’ontologia delle risorse è un’importante componente di un’ontologia dei servizi.

L’OWL-S propone un’ontologia delle risorse ad alto livello in modo da permettere la descrizione di tutti i tipi di risorse come quelle fisiche, temporali e computazionali.

Individuiamo due tipi di risorse:

- *resources types*, come la benzina.
- *resources (tokens)*, come il combustibile del serbatoio di una particolare automobile.

Inoltre consideriamo la *capacity*, ovvero la quantità di un *token resource* disponibile in un dato momento.

3.1 Tipi di allocazione delle risorse

Le risorse sono allocate sia per le attività che per i processi. Una distinzione delle risorse deriva dal loro stato dopo il loro impiego:

- *ConsumableAllocation*: se la risorsa, dopo che viene impiegata, non è più utilizzabile essendo terminata. Alcuni esempi sono dati dal cibo, dalla carica in una batteria, dal carburante, dal denaro e dal tempo. In alcuni casi le risorse consumabili possono essere reintegrate. La scadenza è un vincolo indiretto su di una risorsa consumabile (come ad es. per il cibo).
- *ReusableAllocation*: se la risorsa, dopo che viene impiegata, rimane. Alcuni esempi sono dati dai dispositivi, da un agente, da una regione di spazio e da una porzione di banda.

Spesso le precondizioni ad un processo consistono nella disponibilità di alcune risorse.

Ovviamente le risorse di Atena non essendo risorse fisiche, sono tutte riusabili.

3.2 Tipi di capacità

Ad ogni istante le risorse hanno una precisa misura della quantità. La misurazione della risorsa deve essere compiuta in modo continuo o discreto. Quindi una risorsa ha un *CapacityType* che può essere:

- *ContinuousCapacity*
- *DiscreteCapacity*

Un'altra proprietà delle risorse è la *capacityGranularity* che specifica la grandezza delle unità secondo cui una risorsa è misurata.

3.3 Composizione delle risorse

Una risorsa può essere atomica o composta. Quindi le sottoclassi della classe *Resource* sono:

- *AtomicResource*
- *AggregateResource*. Si distinguono in:
 - *ConjunctiveAggregateResource*: in cui tutti gli elementi devono essere allocati alla stessa attività.
 - *DisjunctiveAggregateResource*: deve essere allocato un sottoinsieme degli elementi.

Alcune risorse atomiche possono essere condivise da diverse attività. Alcune attività ad esempio hanno bisogno di condividere la stessa tabella. Riguardo la disponibilità delle risorse distinguiamo le seguenti sottoclassi della classe *AtomicResource*:

- *UnitCapacityResource*: che non ammettono condivisione della risorsa.
- *BatchCapacityResource*: che ammettono condivisione.

	Risorsa	Allocazione	Capacità	Composizione
Exam	resource type	reusable	discrete	atomic
Subject	resource type	reusable	discrete	atomic
Question	resource type	reusable	discrete	atomic
QuestionsList	resource type	reusable	discrete	atomic
User	resource type	reusable	discrete	Atomic
Student	tokens	reusable	discrete	Atomic
Teacher	tokens	reusable	discrete	Atomic

3.4 Algoritmo per il matching

Oltre ad un linguaggio per esprimere le definizioni dei servizi offerti e di quelli richiesti, occorre un algoritmo che realizzi il matching tra le richieste e le offerte di servizi.

La ricerca dei servizi Web si basa sull'uso dei registri che funzionano come directory. Questi registri catalogano gli annunci dei servizi che viaggiano on-line e supporta la ricerca dei servizi che forniscano l'insieme di funzionalità richieste.

Si ha la corrispondenza tra un annuncio e una richiesta, quando un annuncio descrive un servizio che è sufficientemente simile al servizio richiesto. Il problema di questa definizione è di specificare quando una corrispondenza è accettabile. Poiché richieste e offerte dei servizi non corrisponderanno mai esattamente, occorre ricorrere ad un matching più flessibile, quindi dobbiamo stabilire una scala che quantifica quanto un servizio offerto corrisponda a quello richiesto. Se si concede poca flessibilità, si riduce la probabilità di trovare il servizio richiesto, ovvero si minimizzano i falsi positivi, ma crescono i falsi negativi. Aumentando esageratamente la flessibilità, si ha l'effetto opposto.

Il meccanismo per il matching deve soddisfare i seguenti requisiti:

- Deve supportare un matching semantico tra gli annunci e le richieste, sulla base di ontologie disponibili ai servizi e al motore di matching.
- Deve minimizzare sia i falsi positivi che i falsi negativi.
- Deve incoraggiare gli espositori e i richiedenti ad essere onesti nelle loro descrizioni a costo di pagare il prezzo di non essere contattati o di essere contattati erroneamente.
- Il matching deve essere efficiente e veloce.

Il matching semantico è basato sulle ontologie OWL-S, quindi le richieste e le offerte si riferiscono ai concetti OWL-S. Usando OWL-S, il processo di matching può eseguire inferenze sulla gerarchia delle sussunzioni. Il matching è permesso soltanto nelle descrizioni dei servizi in cui vengono usati concetti delle ontologie OWL-S. Le descrizioni OWL-S permettono di identificare dei gradi di matching.

Algoritmo

L'idea base dell'algoritmo è quella secondo la quale un annuncio corrisponde ad una richiesta se il servizio fornito può essere utile a chi ha inoltrato la richiesta. Un annuncio corrisponde ad una richiesta quando corrispondono rispettivamente tutti gli input e tutti gli output.

Consideriamo il caso in cui, una richiesta corrisponde a tutti gli annunci registrati sul registro, in tal caso serve calcolare un grado di matching e serve organizzare questi risultati.

Il grado del matching è determinato dalla minima distanza tra i concetti nell'albero della tassonomia. Da notare che poiché DAML+OIL supporta l'ereditarietà multipla, tra due nodi ci può essere più di un percorso.

Identifichiamo quattro livelli di matching, dati un output di un richiedente, R, e un output del servizio esposto, A.

- 1) Se output identico, abbiamo una *corrispondenza esatta*.
- 2) Se l'output di R è sottoclasse del modello A, abbiamo una *corrispondenza esatta*
- 3) Se l'output di A sussume l'output R allora è un *plug-in*, cioè A può essere impiegato al posto di R.
- 4) Se l'output R sussume l'output di A, allora *sussume*, ciò significa che la richiesta non viene pienamente soddisfatta.

La scala di preferenza per il matching è:

- 1) corrispondenza esatta
- 2) plug-in
- 3) sussume
- 4) fallimento = non corrispondenza

Per l'ordinamento delle offerte in base al loro livello di matching, innanzitutto si considera il grado di corrispondenza degli output; vengono considerati gli input solo nel caso di corrispondenza dello stesso livello negli output. In caso di errore nella fornitura dell'offerta, si può re-interrogare il registro per ricevere un altro fornitore.

Esempio: ricerca di automobili.

Il servizio pubblicizzato è una vendita di automobili che dato un prezzo fornisce la lista delle auto acquistabili a quel prezzo.

Segue la descrizione in RDF del servizio:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"/>

<profile:Profile rdf:ID="CarSellingService">
  <profile:serviceName>CarSellingService</profile:serviceName>
  <profile:providedBy>...</profile:providedBy>
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">
      <profile:parameterName>Price</profile:parameterName>
      <profile:restrictedTo rdf:resource="Concepts.daml#Price"></profile:restrictedTo>
    </profile:ParameterDescription>
  </input>
  <output>
    <profile:ParameterDescription rdf:ID="Car_Output">
      <profile:parameterName>Car</profile:parameterName>
      <profile:restrictedTo rdf:resource="Vehicle.daml#Car"></profile:restrictedTo>
    </profile:ParameterDescription>
  </output>
</profile:Profile>
```

Nel codice è specificato che gli input attesi dal servizio sono soltanto istanze del concetto Price per come è definito nell'ontologia dei Concepts, mentre gli output che il servizio genera sono istanze del concetto Car come è definito nell'ontologia Vehicle. Questi concetti si riferiscono all'ontologia:

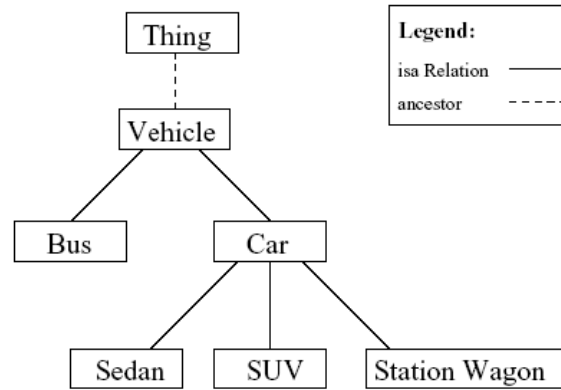


Figura 7 - Frammento di Vehicle Ontology

Una possibile richiesta di servizio potrebbe essere:

```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"/>

<profile:Profile rdf:ID="RequestSedanSellingService">
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">
      <profile:parameterName>Price</profile:parameterName>
      <profile:restrictedTo rdf:resource="Concepts.daml#Price"></profile:restrictedTo>
    </profile:ParameterDescription>
  </input>
  <output>
    <profile:ParameterDescription rdf:ID="Sedan_Output">
      <profile:parameterName>Sedan</profile:parameterName>
      <profile:restrictedTo
rdf:resource="file:data/Vehicle.daml#Sedan"></profile:restrictedTo>
    </profile:ParameterDescription>
  </output>
</profile:Profile>
    
```

Essa specifica che in uscita ha Sedan (berline). In questo caso poiché Sedan è una sottoclasse di Car, allora vi è una perfetta corrispondenza.

Esempio: in Atena richiesta e fornitura del servizio SendMail.

Esposizione servizio:

```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"/>

<profile:Profile rdf:ID="SendMail">
<profile:serviceName>SendMail</profile:serviceName>
<profile:providedBy>...</profile:providedBy>
<input>
  <profile:ParameterDescription rdf:ID="User_Input">
    <profile:parameterName>User</profile:parameterName>
    <profile:restrictedTo rdf:resource="Concepts.daml#User"></profile:restrictedTo>
  </profile:ParameterDescription>
</input>
<output>
  <profile:ParameterDescription rdf:ID="Result_Output">
    
```

```

    <profile:parameterName>Result</profile:parameterName>
    <profile:restrictedTo rdf:resource="Concepts.daml#Result"></profile:restrictedTo>
  </profile:ParameterDescription>
</output>
</profile:Profile>

```

Richiesta di un servizio.

```

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" />
<profile:Profile rdf:ID="SendMailToTeachers">
  <input>
    <profile:ParameterDescription rdf:ID="Teacher_Input">
      <profile:parameterName>Teacher</profile:parameterName>
      <profile:restrictedTo rdf:resource="Concepts.daml#Teacher"></profile:restrictedTo>
    </profile:ParameterDescription>
  </input>
  <output>
    <profile:ParameterDescription rdf:ID="Result_Output">
      <profile:parameterName>Result</profile:parameterName>
      <profile:restrictedTo
rdf:resource="file:data/Vehicle.daml#Result"></profile:restrictedTo>
    </profile:ParameterDescription>
  </output>
</profile:Profile>

```

Nell’esempio esposto, l’algoritmo permette di realizzare un matching di tipo plug-in. Lo schema generale nel caso del progetto Atena è il seguente:

matching	ChangeExamDate Corrispondenza esatta	GetInscriptionList Corrispondenza esatta	GetQuestionsList Corrispondenza esatta	SendMail Corrispondenza esatta + Plug-in	UpdateInscriptionList Corrispondenza esatta	AccessToDBMS Corrispondenza esatta
----------	---	---	---	---	---	---

3.5 Traduzione in Jade

L’utilizzo diretto della descrizione OWL-S precedentemente introdotta non è stato possibile poiché la piattaforma Jade, a cui è conforme il sistema PASSI, non supporta tale rappresentazione. Si è quindi reso necessario una fase di adeguamento delle strutture individuate in un formato compatibile.

Il template previsto in Jade per la descrizione dei servizi, processato dall’agente Director Facilitator (DF), prevede la seguente struttura:

```

DFAgentDescription
  Name:      AID // Required for registration
  Protocols: set of Strings
  Ontologies: set of Strings
  Languages: set of Strings
  Services: set of {
    { Name: String // Required for each service specified
      Type: String // Required ...
      Owner: String
      Protocols: set of Strings
      Ontologies: set of Strings
      Languages: set of Strings
      Properties: set of
        { Name: String
          Value: String

```

```

    }
}

```

Come si nota tale struttura risulta sottodimensionata rispetto alla forza espressiva della controparte OWL-S, quindi si è resa necessaria una fase di traduzione da una descrizione all'altra, che ha coinvolto l'eliminazione di alcuni campi (i meno rilevanti) precedentemente individuati e l'accorpamento di altri in un unico elemento.

Campi OWL-S

textDescription
 contactInformation
 geographicRadius
 degreeOfQuality (QoS)
 qualityGarantees
 Input
 Output
 Precondizioni
 Effetti
 Type
 Message
 Operation
 Port type
 Binding
 Port
 Service
 Risorsa
 Allocazione
 Capacità
 Composizione

Campi corrispondenti in Jade

nessuno
 nessuno
 nessuno
 nessuno
 nessuno
 Properties.name, Properties.value
 Properties.name, Properties.value
 nessuno
 nessuno
 Dipendente dal Task
 Services.languages
 Dipendente dal Behaviour
 nessuno
 nessuno
 nessuno
 Services
 nessuno
 nessuno
 nessuno
 nessuno

3.5.1 AdminAgent

Questo agente fornisce due servizi.

```

DFAgentDescription
    Name:      "admin"
    Protocols: RDF
    Ontologies: ExamOntology
    Languages:  FIPA-OS
    Services: set of {
        { Name:  "ChangeExamDate"
          Type:  null
          Owner: AdminAgent
          Protocols: RDF
          Ontologies: ExamOntology.ChangeExamDate
          Languages: FIPA-OS
          Properties: set of
            { Name: exam
              Value: Exam
            }
        }
    }

```

```

        { Name:    newDate
          Value:   String
        }
    }

    { Name:    "SendMail"
      Type:    null
      Owner:   AdminAgent
      Protocols:   RDF
      Ontologies: ExamOntology.GetMailAddress
      Languages:   FIPA-OS
      Properties: set of
        { Name:    users
          Value:   List
        }

        { Name:    teacher
          Value:   Teacher
        }
    }
}

```

3.5.2 ExamAssistanceAgent

DFAgentDescription

```

Name:    "exam_codiceMateria_dataEsame"
Protocols:   RDF
Ontologies: ExamOntology
Languages:   FIPA-OS
Services: null

```

3.5.3 ExamPreparationAgent

DFAgentDescription

```

Name:    "teacher"
Protocols:   RDF
Ontologies: ExamOntology
Languages:   FIPA-OS
Services: set of {
  { Name:    "GetInscriptionList"
    Type:    null
    Owner:   ExamPreparationAgent
    Protocols:   RDF
    Ontologies: ExamOntology.ChangeExamDate
    Languages:   FIPA-OS
    Properties: set of
      { Name:    value
        Value:   List
      }

      { Name:    exam
        Value:   Exam
      }
  }
}

```

```

    }

    { Name:    "GetQuestionsList"
      Type:    null
      Owner:   ExamPreparationAgent
      Protocols:  RDF
      Ontologies: ExamOntology.GetMailAddress
      Languages:  FIPA-OS
      Properties: set of
        { Name:    value
          Value:   List
        }

        { Name:    subject
          Value:   Subject
        }
    }

    { Name:    "UpdateInscriptionList"
      Type:    null
      Owner:   ExamPreparationAgent
      Protocols:  RDF
      Ontologies: ExamOntology.GetMailAddress
      Languages:  FIPA-OS
      Properties: set of
        { Name:    exam
          Value:   Exam
        }

        { Name:    newList
          Value:   List
        }
    }
}

```

3.5.4 StudentAgent

```

DFAgentDescription
  Name:    "student"
  Protocols:  RDF
  Ontologies: ExamOntology
  Languages:  FIPA-OS
  Services:  null

```

3.5.5 WrapperAgent

```

DFAgentDescription
  Name:    "wrapper"
  Protocols:  RDF
  Ontologies: ExamOntology
  Languages:  FIPA-OS

```

```

Services:    set of {
  { Name:    "AccessToDBMS"
    Type:    null
    Owner:   WrapperAgent
    Protocols:  RDF
    Ontologies: ExamOntology.ChangeExamDate
    Languages:  FIPA-OS
    Properties: set of
      { Name:    exam
        Value:   Exam
      }

      { Name:    newDate
        Value:   String
      }
    }
  }
}

```

3.6 Conclusioni sull'applicazione di OWL-S al progetto Atena

In questo paragrafo conclusivo vengono tratte le conclusioni sull'utilità dell'impiego della metodologia OWL-S alla gestione dei servizi del programma Atena.

La descrizione OWL-S di tipo non funzionale, ovvero quella relativa all'organizzazione del servizio, si è dimostrata più applicabile alla nostra applicazione rispetto alla descrizione funzionale, in quanto risulta utile, anche se non innovativo, il campo *serviceName*, mentre gli altri parametri come *contactInformation*, *geographicRadius*, *degreeOfQuality* e *qualityGuarantees* potrebbero essere applicati, in seguito ad un'operazione di codifica che permetta il funzionamento di un algoritmo di matching tra stringhe di caratteri. Una nota a parte richiede il campo *textDescription*, il quale risulterebbe di immediata utilità ad un eventuale operatore umano, ma che per essere coinvolto nel processo automatico richiederebbe un applicativo basato sull'intelligenza artificiale, capace di trarne il contenuto o almeno i termini chiave della descrizione del servizio.

La descrizione funzionale, seppur interessante nel suo insieme, non è risultante molto applicabile al nostro caso, in quanto necessiterebbe un radicale cambio di prospettiva volto a concepire il software come un sistema che abbia un proprio stato, caratterizzato da ingressi ed uscite e da cambiamenti di stato. In tal senso il programma verrebbe inteso come un'entità dinamica. L'individuazione dei parametri funzionali è stata volta più che ad una implementazione ad una verifica della correttezza dei parametri di ingresso e d'uscita dei servizi e alla verifica dei requisiti non funzionali individuati nella fase della raccolta dei requisiti (RAD).

Lo studio della composizione dei servizi non ci ha riguardato in quanto, data la relativa semplicità del dominio di applicazione, non sono stati impiegati servizi composti.

Inoltre la struttura per la descrizione WSDL si sono presentati come eccessivamente legati al dettaglio dell'implementazione e volti all'applicazione telematica. Tuttavia è stato possibile individuare il tipo di dati, che nel nostro specifico caso trovavano un corrispettivo nelle strutture dati da noi individuate per il funzionamento interno del programma.

In genere poi si è sentita la necessità di un'ontologia dei servizi, a noi non disponibile, che sarebbe stata utile per identificare al suo interno la collocazione dei servizi Atena. Inoltre non è stato possibile classificare i servizi da noi individuati nelle tassonomie esistenti come UDDI e NAICS, in quanto i nostri servizi risultano troppo specifici rispetto a quelli considerati dalle tassonomie esistenti che si propongono di racchiudere tutte le possibili categorie di servizi su scala mondiale.