UNIVERSITÀ DEGLI STUDI DI NAPOLI

"FEDERICO II"

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INFORMATICA E SISTEMISTICA

TESI DI LAUREA

IN

SISTEMI INFORMATIVI

"ANALISI E CARATTERIZZAZIONE DI SISTEMI DI PERVASIVE COMPUTING: UN APPROCCIO ONTOLOGY-BASED PER LA RISOLUZIONE DI PROBLEMI SEMANTICI"

RELATORI

Ch.mo Prof. A. d'Acierno

Ch.mo Prof. G. De Pietro

Ch.mo Prof. A. Coronato

CANDIDATO

Esposito Massimo

matr. 41/2820

Anno Accademico 2002-03

Ringraziamenti

Un ringraziamento particolare ai miei relatori per l'aiuto e la disponibilità dimostrata durante lo sviluppo della tesi.

A tutti i componenti di San Giorgio Net che hanno reso piacevoli e poco pesanti questi mesi di lavoro.

A tutti gli amici dell'università per gli indimenticabili anni trascorsi insieme.

A Guido per aver condiviso con me questa esperienza e per le tante discussioni costruttive che sono state fondamentali per portare a termine questo impegno.

A tutti gli amici per il sostegno e l'affetto.

Ai miei genitori per i loro sacrifici, il loro fondamentale appoggio morale e per tutti i preziosi consigli ed insegnamenti che hanno saputo trasmettermi in questi anni.

Ai miei fratelli, ai miei nonni, agli zii e a tutti gli altri parenti che mi sono stati vicini.

Massimo Esposito

Alla mia famiglia

Abstract

Introduzione

Il proliferarsi di computer sempre più piccoli e poco costosi a cui si è assistito negli ultimi anni, accoppiato con i progressi delle tecnologie di comunicazione, ha portato ad una nuova forma di Computing, il Pervasive Computing: esso si basa su ambienti dotati di proprietà computazionali e comunicative e che contengono un gran numero di componenti hardware e software, autonomi ed eterogenei, che necessitano di cooperare e che tendono ad essere altamente dinamici.

Nonostante siano stati sviluppati vari tipi di middleware ed infrastrutture per rendere possibile la comunicazione tra questi componenti, nessuno di questi è riuscito a garantire l'interoperabilità semantica tra essi.

Obiettivi

Scopo di questa tesi è la definizione e l'utilizzo di modelli ontologici all'interno di un sistema pervasivo per rendere possibile l'interazione e la collaborazione tra le differenti parti che lo compongono.

Per rendere possibile la comunicazione tra entità di un ambiente pervasivo, occorre necessariamente che interfacce, protocolli e formato dei messaggi siano comuni. E' difficile, però, aspettarsi che la semantica dell'ambiente e delle entità contenute in esso sia unica e comprensibile da tutti. Mentre per sistemi semplici o chiusi, tutti gli schemi richiesti per interpretare il contenuto dei messaggi possono essere compilati nei componenti del sistema stesso, in un sistema aperto occorre un modello che renda possibile la scoperta e l'utilizzo degli schemi quando occorrono e mentre il sistema è in esecuzione.

In questo lavoro sono state analizzate le principali infrastrutture per la comunicazione distribuita, evidenziando soprattutto gli aspetti semantici, ed è stato proposto l'utilizzo dei concetti sviluppati dal Web Semantico ed in particolare delle ontologie come modello descrittivo aperto, valutandone l'applicabilità all'interno di un prototipo di infrastruttura pervasiva basata su tecnologie Web.

Coerentemente con lo stato dell'arte e tenendo presente modelli e specifiche di uso già diffuso, sono state definite ontologie, in linguaggio DAML+OIL, che descrivono tutti i componenti di un ambiente pervasivo, ossia utenti, dispositivi, risorse, servizi, e tutte le informazioni relative al contesto che caratterizza l'ambiente stesso.

In particolare, poi, è stato progettato ed implementato, in linguaggio Java, un Ontology Service, ossia un Web Service che rappresenta un componente di sistema del prototipo di infrastruttura pervasiva proposta e che fornisce un'interfaccia generica per gestire ontologie, espresse in DAML+OIL.

Organizzazione del lavoro

Il capitolo 1 analizza le caratteristiche e le problematiche fondamentali appartenenti ai sistemi di Pervasive Computing e presenta lo "stato dell'arte" relativo alla realtà di interesse, illustrando i principali progetti realizzati da industrie e università di diverse parti del mondo.

Il capitolo 2 fornisce una descrizione delle principali tecnologie di comunicazione tra oggetti distribuiti e dei linguaggi di markup dello stack del Web semantico.

Il capitolo 3 propone un approccio basato sulle ontologie per la risoluzione dei problemi semantici relativi alla comunicazione tra componenti eterogenei, applicato ad un prototipo di ambiente pervasivo basato su tecnologie Web.

Il capitolo 4 contiene la progettazione e l'implementazione dell'Ontology Service, e mostra come esso utilizza le ontologie.

Nelle conclusioni, infine, viene fatto il punto su quanto è stato proposto e realizzato, evidenziandone sia gli aspetti positivi che i limiti ed indicando i possibili sviluppi futuri.

Indice

Capitolo 1

Il Pervasive Computing

1.1 Introduzione

Le tecnologie più profonde sono quelle che svaniscono. Esse si fondono nella vita di tutti i giorni fino a divenire indistinguibile da essa [3].

Così, più di dieci anni fa, Mark Weiser iniziò il suo articolo ”The Computer for the 21st Century”, in cui compariva, per la prima volta, il termine Ubiquitous Computing, associato ad una visione futuristica del computing accentrata sull’ubiquità dei personal computer.

I computer ed il computing in generale, come li conosciamo oggi, stanno lentamente diventando storia passata. Negli anni sessanta, alla parola *“computer”* si associavano grandi e costosi mainframe con un grosso numero di utenti che ne condividevano le risorse. Si parlava di paradigma *“many people per computer”*: molti utenti per una sola macchina. I continui progressi tecnologici hanno poi consentito la realizzazione dei *personal computer* che hanno significativamente modificato il tipo di utilizzo di sistemi di calcolo, trasformando il paradigma in *“one person per computer”*: una persona per una macchina. Nell’ultimo decennio la diffusione di laptop, Personal Digital Assistant (PDA), telefoni cellulari, dispositivi portatili dotati di

microprocessori e di capacità di immagazzinare dati, ha mutato ulteriormente il rapporto uomo-computer ed il paradigma è diventato "*many computers per person*": tanti elaboratori per una sola persona.

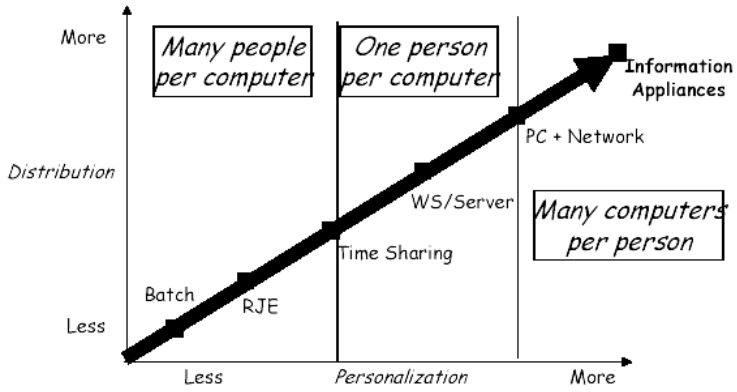

Figura 1.1:Evoluzione del computing

Questo proliferarsi di elaboratori, che sono diventati sempre più piccoli e poco costosi, accoppiato con i progressi delle tecnologie di comunicazione, ha indotto a pensare che, in un futuro non troppo lontano, i computer possano far parte integrante di ogni oggetto della realtà quotidiana, favorendo la comunicazione e l'elaborazione di informazioni in maniera naturale, in qualsiasi posto ci si trovi ed in qualsiasi momento. E' questa la nuova visione del computing ed a questo scenario ci si riferisce quando si parla di Ubiquitous Computing, oggi ribattezzato Pervasive Computing.

Un sistema di Pervasive Computing, in base a quanto detto, risulta essere caratterizzato da due attributi fondamentali [41]:

- *Ubiquità*: l'interazione con il sistema è disponibile dovunque l'utente ne abbia bisogno;
- *Trasparenza*: il sistema non è intrusivo ed è integrato negli ambienti della vita quotidiana.

In accordo con questa visione [40], è possibile identificare due dimensioni che forniscono una più chiara definizione dei sistemi pervasivi ed esprimono le relazioni esistenti con le altre aree di ricerca emergenti:

- *Mobilità dell'utente*: riflette la libertà che l'utente ha di muoversi quando interagisce con il sistema;
- *Trasparenza di interfaccia*: si applica all'interfaccia del sistema e riflette lo sforzo consapevole e l'attenzione che il sistema richiede all'utente, sia per operare su di esso che per percepirne i suoi output.

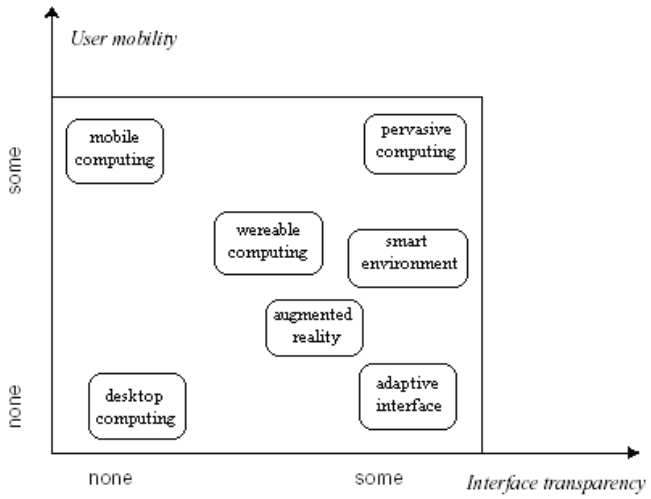

Figura 1.2: Attributi fondamentali di un sistema pervasivo

In sintesi, quindi, il Pervasive Computing mira alla realizzazione di un mondo non più vincolato alle scrivanie, ma composto da una molteplicità di ambienti dotati di capacità computazionali e comunicative e talmente integrati con l'utente da diventare una "tecnologia che svanisce" [2], utilizzabile in maniera trasparente ed inconscia. Non una realtà virtuale, in cui le persone sono inserite in un mondo generato dai computer, ma piuttosto una virtualità reale che porta i computer stessi a vivere nel mondo reale, assieme alle persone [3].

Anche se molti dei componenti necessari a realizzare tali ambienti sono già di d'uso diffuso, la mancanza di integrazione e coordinamento tra tali componenti e di collaborazione e comprensione tra applicazioni, utenti e le loro attività,

non ha ancora consentito la realizzazione di un vero e proprio sistema pervasivo [39]. Quindi, nonostante siano trascorsi più di dieci anni dalla sua formalizzazione, il mondo del Pervasive Computing, così come lo aveva pensato Weiser, risulta essere ancora oggi una visione futuristica.

Tuttavia, grazie ai continui progressi tecnologici ottenuti negli ultimi anni, numerose iniziative sono state realizzate in diverse università ed industrie di tutto il mondo, tutte con lo scopo comune di gettare le basi affinché il Pervasive Computing diventi presto una realtà.

1.2 Evoluzione del Pervasive Computing

Per garantire i requisiti di "ubiquità e trasparenza" questa nuova "tecnologia" deve essere necessariamente fortemente dinamica e disaggregata, ossia capace di gestire utenti estremamente mobili e servizi forniti da collezioni di componenti distribuiti che collaborano tra loro [29]. Il Pervasive Computing rappresenta, quindi, la tappa dell'evoluzione tecnologica che si colloca immediatamente e naturalmente dopo il Distributed Computing ed il Mobile Computing, ereditandone parecchi aspetti e ridefinendone degli altri [2].

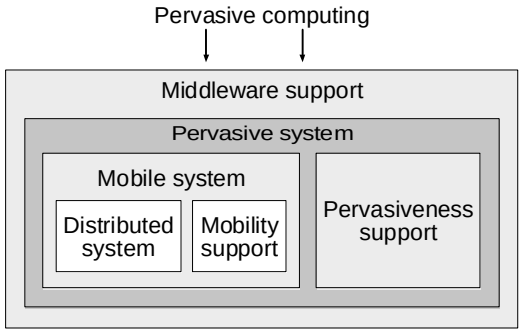

Figura 1.3: Un modello architetturale per sistemi pervasivi

Il Distributed Computing mira a ripartire dati e capacità computazionali su componenti indipendenti, sparsi su macchine diverse, che comunicano tra loro attraverso reti di interconnessione. La possibilità di comunicare ed accedere ai dati e agli strumenti di calcolo da postazioni remote, le tecniche della

replicazione e ridondanza dei dati che favoriscono una maggiore disponibilità degli stessi e che aumentano l'affidabilità del sistema nel complesso, rappresentano oggetto di studio per questa forma di computing.

Con l'introduzione di vincoli e problematiche legate al concetto di mobilità, è stato necessario pensare a nuove soluzioni tecnologiche che hanno portato alla creazione di una nuova forma di computing, ossia il Mobile Computing [42]. Non si hanno più nodi di rete fissi, con connessioni stabili e veloci, ma nodi costituiti da dispositivi mobili che accedono e abbandonano la rete continuamente ed in maniera del tutto imprevedibile, forniti di connessioni precarie e contraddistinte da forti cambiamenti sulle caratteristiche di banda. Le limitate capacità di calcolo e di memoria dei dispositivi mobili ed il funzionamento fortemente condizionato dallo stato di carica delle batterie, rappresentano ulteriori aspetti di cui il Mobile Computing si sta occupando.

I sistemi di pervasive computing, che, come detto, sono a loro volta anche sistemi distribuiti e mobili, presentano tutta una serie di problematiche, molte delle quali sono simili a quelle di questi sistemi e di cui si è discusso precedentemente, ma parecchie delle soluzioni trovate non sono adottabili in questo nuovo contesto.

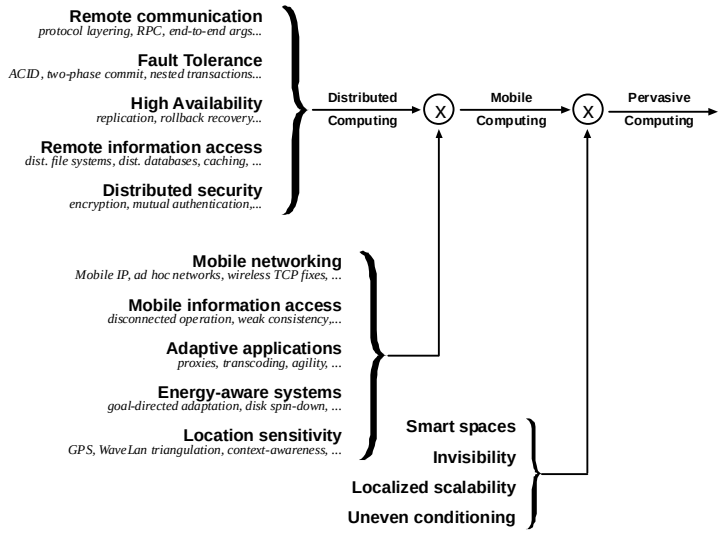

Figura 1.4:Evoluzione dei sistemi di computer

L'aumento di complessità introdotto da questi nuovi sistemi, infatti, è risultato essere moltiplicativo piuttosto che additivo, il che ha comportato una maggiore difficoltà a progettare e a realizzare un sistema pervasivo che un sistema distribuito di comparabile robustezza e maturità [2].

1.3 Background e stato dell'arte

I numerosi progetti realizzati in tutto il mondo per implementare un'infrastruttura in grado di trasformare un semplice ambiente in un ambiente di computing pervasivo evidenziano chiaramente il massiccio impiego di tempo e di risorse che è stato profuso negli ultimi anni. Ognuno di questi progetti ha indirizzato la propria attenzione su uno specifico tipo di ambiente con determinate applicazioni e servizi per particolari utenti (ad esempio abitazioni, uffici, stanze per conferenze) essendo ancora irrealizzabile l'idea di sviluppare un'unica infrastruttura di tipo generale capace di essere usata in tutti gli ambienti esistenti.

Di seguito sono, quindi, analizzati, singolarmente, i principali esempi di sistemi pervasivi implementati, evidenziando caratteristiche principali e modelli architetturali di ognuno di essi.

1.3.1 Aura

Aura [30] mira a fornire all'utente una ambiente di computing "libero da distrazioni", dove le persone possono accedere ai servizi o effettuare le proprie attività senza interventi sul sistema o sull'ambiente.

Aura utilizza essenzialmente due concetti per realizzare i suoi obbiettivi:

- *Pro-activity* rappresenta la capacità, a livello di sistema, di anticipare richieste provenienti da un livello più alto;
- *Self-tuning* rappresenta la capacità di adattarsi osservando le richieste fatte e di modificare le prestazioni e l'utilizzo delle risorse in base ad esse.

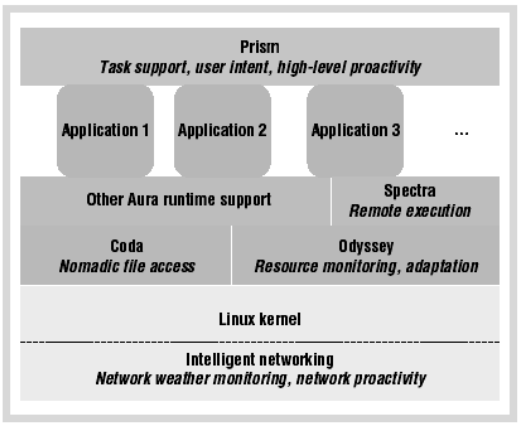

Figura 1.5:L'architettura di Aura

Aura è composta da cinque componenti principali:

- *Odyssey* è una collezione di estensioni del sistema operativo, che consentono al sistema di monitorare le risorse ed alle applicazioni di adattarsi in base alla loro disponibilità;
- *Coda* è un sistema di gestione di file distribuiti che consente l'accesso continuato ai dati anche in presenza di malfunzionamenti delle rete o dei server;
- *Spectra* è un meccanismo di esecuzione remota ed adattativa che usa il contesto per decidere come meglio eseguire le invocazioni remote;
- *Prism* si occupa di catturare e gestire le intenzioni dell'utente ed in più fornisce un supporto di alto livello per realizzare gli aspetti di proactivity e di self-tuning.

Aura riusa, in pratica, tecnologie vecchie per creare nuovi sistemi adatti ad essere inseriti in ambienti pervasivi [30].

1.3.2 Gaia

Gaia [31],[43]è un'infrastruttura middleware, creata dai ricercatori del Dipartimento di Computer Science nell'Università di Illinois, che mira a gestire "Spazi Attivi" *(Active Spaces)*.

Nell'ambito di questo progetto di ricerca, vengono definiti i concetti di: "Spazio fisico" (*Physical Space*) una regione geografica con confini fisici limitati e ben definiti, contenente oggetti e dispositivi eterogenei collegati in rete e popolato da utenti che svolgono attività; "Spazio attivo" uno luogo fisico coordinato da un'infrastruttura software sensibile al contesto che consente agli utenti mobili di interagire e configurarsi con l'ambiente fisico e digitale in maniera automatica.

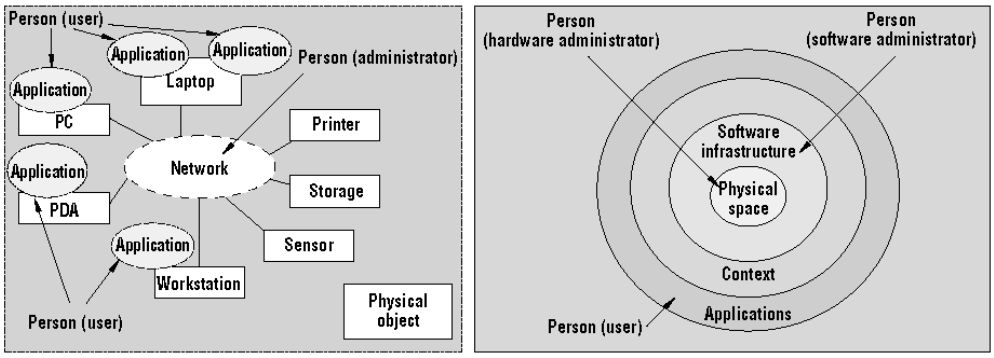

Figura 1.6:Spazi fisici e spazi attivi

L'architettura di Gaia è formata essenzialmente da quattro componenti:

- *Gaia Kernel* è un sistema di gestione e sviluppo di oggetti distribuiti (in particolare oggetti CORBA) ed è costituito da un insieme interconnesso di servizi di base di supporto per le applicazioni: *context service*, *event manager*, *presence service*, *security service* ed *component repository*;
- *Gaia Application Framework* modella le applicazioni come collezioni di componenti distribuiti, prendendo in conto le risorse hardware disponibili nello spazio attivo o quelle relative ad un particolare dispositivo; fornisce funzionalità per alterare la composizione delle applicazioni dinamicamente; è *context–sensitive*; implementa un meccanismo che supporta la creazione di applicazioni indipendenti dallo spazio attivo e fornisce politiche per adattarsi a differenti aspetti delle applicazioni, inclusa la mobilità;

- *QoS Service Framework* si occupa della gestione delle risorse per le applicazioni sensibili alla QoS e adatta dinamicamente tali applicazioni, determinando i nodi appropriati per la loro istanziazione, in base ad una selezione tra configurazioni, in accordo con la disponibilità di risorse, con il servizio di discovery e con protocolli di assegnazione di risorse multiple, ossia traducendo i requisiti di alto livello relativi alla QoS in requisiti di sistema;
- *Application layer* che contiene le applicazioni e fornisce le funzionalità per registrare, gestire, e controllare queste applicazioni attraverso i servizi del Kernel di Gaia.

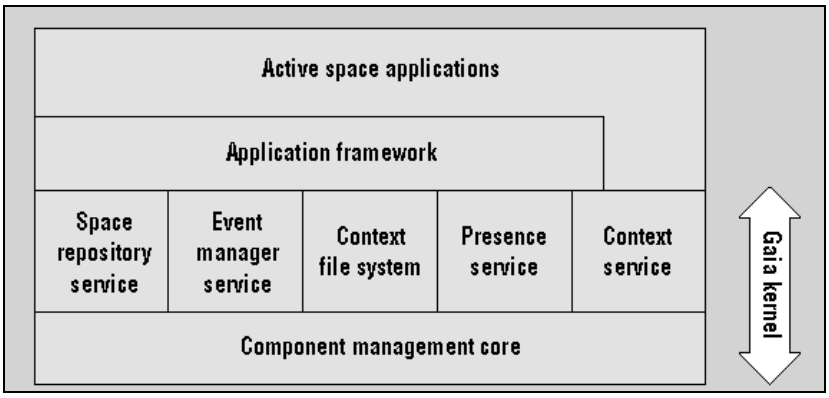

Figura 1.7:L'architettura di Gaia

Gaia, quindi, cerca di portare le funzionalità di un sistema operativo in uno spazio attivo, come gli eventi, i segnali, il file system, la sicurezza, i processi, naturalmente estendendo questi concetti ed inserendo in più il nuovo concetto di contesto.

1.3.3 Cooltown

Il progetto realizzato presso i laboratori della Hewlett-Packard, Cooltown, offre un modello di sistema di pervasive computing che combina tecnologie web, reti wireless e dispositivi portatili per creare un ponte virtuale tra utenti mobili, entità fisiche e servizi elettronici [32].

Nella visione di Cooltown, ogni dispositivo fisico deve possedere una homepage (localizzata su un web server dedicato) attraverso la quale può essere controllato dall'utente. Il link alla homepage può essere acquisito usando segnali radio (Bluetooth), infrarossi, codici a barre o semplicemente fornendo l'URL in formato testuale. Nel caso in cui solo un ID è ottenuto (come per i codici a barre), l'URL deve essere risolta da un server, favorendo la possibilità di avere più revolver, a seconda del contesto.

La ragione per cui HP ha scelto il web come tecnologia alla base del suo prototipo di ambiente pervasivo è che è certamente di più facile adozione, essendo l'ambiente di distributed computing più diffuso, rispetto ad una qualsiasi altra tecnologia da sviluppare ex novo.

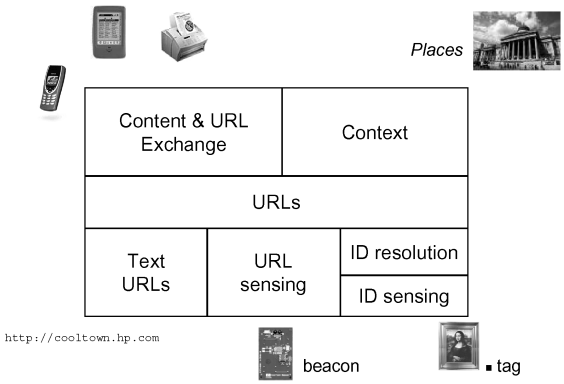

Figura 1.8:La visione di Cooltown

In Cooltown, dove il protocollo di comunicazione è HTTP, è possibile comunicare con i dispositivi anche se non si è agganciati su una rete globale adoperando ad esempio tecnologie come Bluetooth o IrDA, che risultano sufficienti per individuare, ad esempio, una stampante e dirle di stampare un documento. Esso, inoltre, supporta gli spazi attivi e quindi è consapevole di chi si trova in un posto e si comporta di conseguenza [34]. Le caratteristiche dei dispositivi dell'ambiente HP sono descritte in una web-form indipendente dal tipo di dispositivo stesso, il che consente una semplice interazione tra

dispositivi, adoperando solo l'URL, senza la necessità di dover istallare driver per ogni tipo di dispositivo incontrato.

Tuttavia l'interazione macchina-macchina tra dispositivi sconosciuti è un aspetto ancora irrisolto poiché le web-interfaces sono comprensibili, essenzialmente, solo dall'uomo e non sono adatte ad una elaborazione automatica [33].

1.3.4 Centaurus

Centaurus [35] costituisce un'infrastruttura ed un protocollo di comunicazione per servizi software ed hardware eterogenei che devono essere resi disponibili agli utenti ovunque essi ne abbiano bisogno. Tali servizi, inoltre, debbono essere capaci di comprendere le necessità dell'utente ed usare ragionamenti logici per fornirgli un migliore supporto.

L'architettura di Centaurus consiste di cinque componenti:

- *CentaurusComm Trasport Protocol* è un protocollo di trasporto efficiente, basato su messaggi, e che può essere utilizzato indipendentemente dal tipo di mezzo trasmissivo;
- *Communication Manager* gestisce tutta la comunicazione con i client usando differenti moduli del *CentaurusComm Protocol*;
- *Service Manager* controlla l'accesso ai servizi ed agisce da gateway tra i servizi ed i client;
- *Services* sono oggetti che offrono certe funzionalità ai client Centaurus. I servizi contengono informazioni necessarie a localizzare il più vicino Service Manager ed a registrarsi ad esso. Una volta registrato può essre richiesto da qualsiasi client comunicando attraverso un qualsiasi Communication Manager;
- *Clients* forniscono un'interfaccia all'utente per interagire con i servizi. Un client può accedere ai servizi forniti dal più vicino sistema

Centaurus che si comporta come un proxy soddisfacendo le richieste del client.

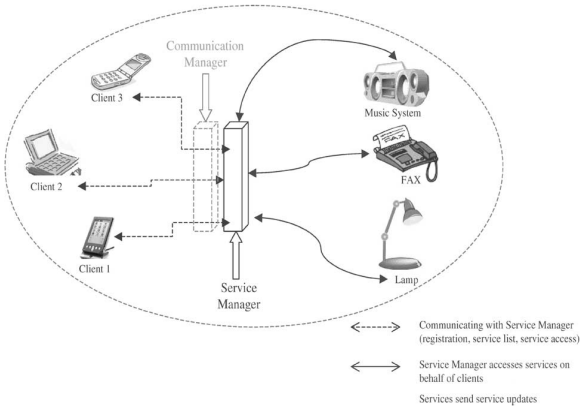

Figura 1.9:L'architettura di Centaurus

Tutti i componenti di questo modello usano un linguaggio proprietario basato su XML, il *Centaurus Communication Markup Language* (CCML), per la comunicazione, dando al sistema un'interfaccia uniforme e maggiormente adattabile.

1.3.5 CoBrA

Il progetto CoBrA [36] mira a sviluppare un'architettura di pervasive computing sensibile al contesto, per consentire ad agenti, servizi e dispositivi di comportarsi in maniera intelligente in base al contesto in cui si trovano.

Il cuore dell'architettura è caratterizzato da un'entità server specializzata ed intelligente, il Context Broker (da cui il nome CoBrA: Context Broker Architecture), che riceve informazioni legate al contesto dai dispositivi e dagli agenti presenti nell'ambiente, le relaziona definendo un modello centralizzato e condiviso dell'ambiente e di tutto ciò che si trova ad operare al suo interno e si preoccupa di mantenerlo, nel tempo, coerente e privo di inconsistenze in seguito a nuove acquisizioni di informazioni. Un punto chiave nella realizzazione di questa architettura è lo sviluppo e l'utilizzo di una serie di ontologie comuni, ossia di descrizioni formali di concetti di un dominio del

discorso, attraverso le quali agevolare la comunicazioni fra i diversi agenti e rappresentare lo stato del sistema. Sono utilizzate le tecnologie del Web Semantico, che includono linguaggi per costruire ontologie e tool per processare e ragionare sulle informazioni descritte in esse. In particolare CoBrA descrive la propria ontologia adoperando OWL, modellando concetti come persona, agente, luogo, evento, e le proprietà e le relazioni tra essi.

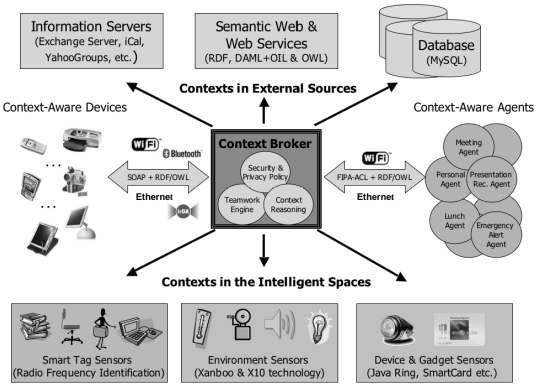

Figura 1.10:Architettura di CoBrA

L'architettura del context broker è costituita da quattro componenti funzionali:

- *Context Knowledge Base* immagazzina in maniera persistente la conoscenza del contesto. Fornisce un set di API per consentire agli altri componenti di accedere alla conoscenza memorizzata. Contiene anche l'ontologia dello specifico ambiente e delle regole euristiche associate all'ambiente stesso;
- *Context Reasoning Engine* è un motore inferenziale che ragiona sulla conoscenza di contesto immagazzinata. Si possono avere inferenze che usano le ontologie per dedurre conoscenza di contesto e inferenze che usano la conoscenza per rilevare e risolvere inconsistenze;
- *Context Acquisition Module* è una libreria di procedure che rappresenta una sorta di middleware per l'acquisizione di informazioni di contesto;
- *Policy Management Module* è un insieme di regole di inferenza definite sia per decidere i permessi giusti, associati alle differenti entità, per

condividere un particolare pezzo delle informazioni di contesto, sia per ricevere notifiche dei cambi del contesto stesso.

1.4 Caratteristiche di un sistema pervasivo

La visione del pervasive computing, che si deduce dagli esempi di progetti riportati, si fonda sulla presenza di ambienti dotati di computer inseriti in ogni oggetto della vita quotidiana e su sensori capaci di catturare qualsiasi informazione dal contesto. Questi ambienti sono caratterizzati da specifiche proprietà computazionali che generalmente li distinguono dagli altri sistemi di computing, e contengono un gran numero di componenti hardware e software che necessitano di cooperare, e che tendono ad essere altamente dinamici.

Progettare sistemi che si riferiscono a particolari scenari e implementare prototipi, così come ha fatto e sta facendo la ricerca mondiale, ha dato inizio ad un ciclo di sviluppo evoluzionario: molte delle idee e soluzioni che vengono trovate nascono come risposta a problemi e difficoltà specifiche e solo in seguito sono applicate anche in altri contesti [38].

A partire da questi progetti è possibile, quindi, individuare una serie di aspetti di carattere generale ed indicazioni di fondamentale importanza per l'analisi, lo sviluppo e l'implementazione di un sistema di pervasive computing.

1.4.1 Context information e Run-time Adaptation

Una prerogativa di un sistema pervasivo è che sia in grado di ottenere informazioni sugli utenti e sullo stato dell'ambiente, come ad esempio, la posizione e l'identità dei singoli utenti e la disponibilità delle risorse.

Un aspetto fondamentale del *pervasive computing* consiste, quindi, nel collezionare dati grezzi da una moltitudine di sorgenti, processare i dati, trasformali in informazioni di contesto e disseminare le informazioni tra le diverse applicazioni in esecuzione sui vari dispositivi, il tutto cercando di

mantenere la scalabilità, di garantire la sicurezza delle informazioni, inibendo accessi non autorizzati e rispettando la privacy individuale.

In base a queste informazioni [43], le applicazioni possono comportarsi in maniera diversa, effettuando, a seconda dei casi, un adattamento funzionale o strutturale.

Un esempio di adattamento funzionale può essere rappresentato da un'applicazione che fornisce news in un ambiente: il tipo di notizie che vengono fornite potrebbe essere determinato in base a chi si trova nell'ambiente o all'ora della giornata.

Un esempio di adattamento strutturale, invece, può essere rappresentato da un'applicazione musicale: a seconda se l'utente è solo oppure no all'interno dell'ambiente, l'applicazione potrebbe utilizzare il sistema audio del laptop dell'utente, oppure quello dell'ambiente stesso.

Per realizzare quanto appena detto, occorre, quindi, avere una percezione dell'ambiente ragionevolmente accurata, che passa necessariamente attraverso la rilevazione e la correzione delle informazioni di contesto inattendibili o contrastanti fornite dai diversi sensori.

1.4.2 Task Recognition e Pro-activity

Un sistema di pervasive computing dovrebbe essere capace, a partire dalle informazioni di contesto collezionate, di ragionare sullo stato corrente e sulle intenzioni dell'utente e modificare dinamicamente il proprio comportamento, per assisterne i task.

Diversamente dai sistemi di computing convenzionale in cui il comportamento del computer è principalmente composto di risposte all'interazione con l'utente, il pervasive computing mira a realizzare il modello opposto in cui i dispositivi sono la parte attiva. Quindi se la tecnologia corrente si basa su persone che indicano ai computer ciò essi che debbono fare, la nuova generazione di tecnologie dovrebbe essere basata su computer capaci di

comprendere quello che le persone stanno facendo e quello che esse desiderano.

Modelli che tengono conto dell'esperienza, del passato, rappresentano un importante aspetto che può caratterizzare un sistema pervasivo, perché favoriscono la *pro-activity* del sistema, ossia consentono di determinare, in accordo con i precedenti comportamenti dell'utente, le azioni ottimali che l'utente stesso deve eseguire in una certa situazione. Per ottenere questo risultato occorre combinare disparati pezzi di conoscenza relativi ai task dell'utente e all'ambiente, senza coinvolgere l'utente stesso. Ad esempio, combinare una forma di conoscenza di alto livello, come l'ora di imbarco di un utente ad una porta dell'aeroporto, con una di più basso livello, come la congestione dei collegamenti wireless a quell'ingresso, il tutto associato alla conoscenza del task dell'utente, ad esempio l'invio di una e-mail, dovrebbe consentire al sistema di raccomandare all'utente un luogo meno congestionato per completare il task nell'intervallo di tempo disponibile prima della partenza [38].

1.4.3 Resource Abstraction e Discovery

Le risorse di un sistema pervasivo dovrebbero essere rappresentate in maniera astratta (magari attraverso le caratteristiche anziché col nome) cosicché esse possano facilmente essere selezionate in base a requisiti di tipo generale o specifico. Potrebbe essere necessario definire alcune convenzioni di nomi per evitare che differenti risorse descrivano se stesse adoperando gli stessi termini ma offrendo servizi diversi o viceversa che si descrivano con termini diversi pur offrendo gli stessi servizi.

Dovrebbero essere previsti, anche, dei meccanismi per scoprire, interrogare ed interagire con le risorse nell'ambiente e per consentire introduzioni di nuovi componenti senza richiedere una configurazione a priori o una reconfigurazione di componenti esistenti [38].

1.4.4 Eterogeneity e Service UI Adaptation

Un ambiente pervasivo è caratterizzato da una collezione di dispositivi eterogenei, ed in numero variabile, come laptop, PDA, telefoni mobili, la cui interazione sta interessando una scala sempre più vasta. Alla riduzione delle dimensione dei dispositivi dell'ambiente corrisponde, infatti, una maggiore crescita del numero di dispositivi connessi e dell'intensità dell'interazione uomo-macchina.

Lo sviluppo tradizionale fornisce servizi tipicamente distribuiti e istallati separatamente per ogni classe di dispositivo e famiglia di processore. Gli scenari di computing pervasivo, invece, portano alla conclusione che distribuire ed istallare servizi per ogni classe e famiglia diventa ingestibile, specialmente in un area geografica estesa.

E' importante [38], quindi, che i servizi forniscano agli utenti interfacce che possano adattarsi alle caratteristiche del dispositivo client, senza intaccare le funzionalità del servizio stesso. Ad esempio, un servizio adibito al controllo dell'illuminazione in una stanza è libero di fornire un'interfaccia utente di tipo grafico per un client PDA, ma deve necessariamente garantire anche una rappresentazione testuale della stessa UI per un utente che adopera un telefono cellulare, senza, peraltro, dover cambiare l'implementazione del servizio stesso.

1.4.5 Fault-tolerance e Scalability

Gli ambienti pervasivi costituiscono sistemi "perennemente" attivi. Pertanto, un componente che subisce un guasto non deve compromettere il funzionamento generale dell'intero sistema, né richiedere una complessa tecnica di gestione.

I componenti che cadono dovrebbero automaticamente ripartire, laddove possibile, magari adoperando, ad esempio, memorie di stato persistenti che consentano di effettuare resume rapidi ed efficaci.

Abbiamo visto che gli ambienti pervasivi sono caratterizzati anche da una forte dinamicità: dispositivi possono aggiungersi all'ambiente ed abbandonarlo in qualsiasi momento; alcuni servizi possono cadere e presentarsene altrettanti nuovi; gli stessi utenti possono entrare ed uscire dall'ambiente secondo la propria volontà.

Il sistema deve garantire scalabilità, ossia essere in grado di gestire e assicurare il suo funzionamento anche in seguito all'aggiunta di componenti; allo stesso tempo, i nuovi dispositivi e servizi introdotti nell'ambiente non dovrebbero interferire con quelli esistenti [38].

1.4.6 Security e Privacy

Un sistema pervasivo è in possesso di una grossa quantità di dati, molti dei quali acquisiti tramite sensori, che riguardano gli utenti dell'ambiente.

Gli utenti desiderano che la loro privacy non sia intaccata e che sia garantita la sicurezza per queste informazioni. Ad esempio, risulta fastidioso sapere che la stazione di polizia più vicina sia in grado di sapere in quale stanza ci si trovi, attraverso i rilevatori di moto del sistema d'allarme, o quanto alcool si sta consumando, inferendo questa informazione dal sistema che gestisce l'inventario degli alimenti. D'altra parte una certa perdita di privacy può essere tollerata in molte situazioni, ad esempio in caso di incendio, o per modificare la temperatura o l'illuminazione dell'ambiente in base alle proprie preferenze.

Per aggiungere sicurezza alle informazioni, i servizi nell'ambiente dovrebbero, inoltre, non consentire accessi non autorizzati: ad esempio, a casa propria, non dovrebbe essere possibile per una persona non autorizzata, ad esempio esterna, aumentare il riscaldamento o controllare la TV [44].

Capitolo 2

Tecnologie per il Pervasive Computing

2.1 La comunicazione nei sistemi distribuiti

Un sistema distribuito [46] consiste in un insieme di computer che comunicano su di una rete per coordinare le azioni e i processi di un'applicazione. Le tecnologie per la realizzazione di un sistema distribuito hanno riscosso molto interesse negli ultimi anni, grazie alla proliferazione dei sistemi e servizi basati sul Web.

Tecnologie consolidate come la comunicazione tra processi e l'invocazione remota, i naming service, la sicurezza crittografica, i file system distribuiti, la replicazione dei dati e i meccanismi di transazione distribuita forniscono l'infrastruttura di run-time che supporta le applicazioni di rete di oggi [45].

Il modello dominante è ancora la tradizionale architettura client-server, ma lo sviluppo di applicazioni per i sistemi distribuiti si sta basando sempre più sia su supporti middleware, che forniscono astrazioni di più alto livello come oggetti distribuiti condivisi, sia su servizi vari, come servizi per la comunicazione sicura o per l'autenticazione.

Inoltre lo stesso Internet con i suoi protocolli base ed anche, ad un livello più alto, il World Wide Web stanno diventando una piattaforma standard per le

applicazioni distribuite. Difatti Internet e le sue risorse possono essere viste come un ambiente globale in cui la computazione ha luogo. Di conseguenza, protocolli e standard di alto livello, come XML, entrano in gioco, mentre passano in secondo piano aspetti di basso livello.

2.2 Paradigmi di comunicazione

Esistono diversi modi attraverso i quali componenti di applicativi software risiedenti su macchine differenti possono comunicare fra loro adoperando una rete.

Una tecnica di basso livello è quella di utilizzare direttamente le interfacce offerte dal livello trasporto, come il meccanismo delle socket, assieme ad un protocollo di comunicazione pensato ad hoc per l'utilizzo specifico. Comunque programmare a questo livello di astrazione è consigliabile solo in particolari circostanze poiché demanda completamente al programmatore la gestione di problemi complessi come la sicurezza, l'eterogeneità e la concorrenza.

Nella maggior parte dei casi, invece, è preferibile scegliere tra una serie di protocolli e ambienti di più alto livello quello che meglio si adatta alle proprie esigenze. Alcuni di questi protocolli sono self-contained e possono essere usati in qualsiasi programma applicativo con un overhead addizionale basso o addirittura nullo. Altri protocolli ed ambienti, invece, sono vincolati a specifici linguaggi di programmazione o a particolari piattaforme di esecuzione [46].

2.2.1 Remote Procedure Call

Un classico schema di comunicazione che utilizza il modello client-server sono le chiamate a procedure remote (RPC). In questo modello, un componente agisce da client quando richiede un servizio ad un altro componente, da server quando risponde a richieste di un altro client. RPC effettua una chiamata ad una procedura esterna che risiede in un differente nodo della rete quasi con la stessa semplicità con cui invoca una procedura

locale. Argomenti e valori di ritorno sono automaticamente impacchettati in un formato definito dall'architettura e mandati tra procedure locali e remote.

Per ogni procedura remota, il framework RPC sottostante ha bisogno di una procedura stub dal lato client (che agisce da proxy) e di un oggetto simile lato server. Il ruolo dello stub è prendere i parametri passati attraverso una regolare procedura locale e passarli al sistema RPC (che deve risiedere su entrambi i nodi). Dietro le quinte, il sistema RPC coopera con gli stub di ambo i lati per trasferire argomenti e valori di ritorno sulla rete.

Per facilitare la creazione di stub, sono stati realizzati speciali tool. Il programmatore fornisce i dettagli di una chiamata RPC in forma di specifiche, espresse attraverso l'Interface Definition Language (IDL), poi viene utilizzato un compilatore IDL che genera, a partire da tali specifiche, gli stub in maniera automatica.

I framework RPC, anche se sono tipicamente invisibili ai programmatori, sono diventati una tecnica consolidata poiché essi rappresentano i meccanismi di trasporto su cui si basano le più generali piattaforme middleware di cui si parlerà in seguito [46].

2.2.2 XML-based RPC

Sebbene i sistemi RPC trattino esplicitamente aspetti di interoperabilità in sistemi aperti, i programmi client e server che fanno uso di questo principio sono vincolati ad un singolo framework RPC: ognuno di questi, infatti, definisce la propria tecnica di codifica per le strutture dati. A dispetto di queste differenze, le semantiche base della maggior parte dei sistemi RPC sono simili poiché si fondano su chiamate a procedure sincrone in un formato espresso in sintassi C-like.

L'idea nuova è stata quella di utilizzare XML per definire la sintassi delle richieste e delle risposte RPC, consentendo a differenti sistemi RPC di poter comunicare tra loro. In pratica XML è usato per definire un sistema di tipi che può essere adoperato per comunicare dati tra client e server. Questo sistema

specifica tipi primitivi, come interi, floating point, stringhe di testo, e fornisce i meccanismi per aggregare istanze di questi ultimi in tipi composti per ottenere nuove categorie di dati.

Uno dei primi framework RPC basati su XML è stato SOAP (Simple Object Access Protocol) [50], definito inizialmente da un consorzio di compagnie tra le quali figuravano Microsoft, IBM, SAP, ma che oggi è invece divenuto un progetto open source in fase di standardizzazione presso il WorldWideWebConsortium (W3C).

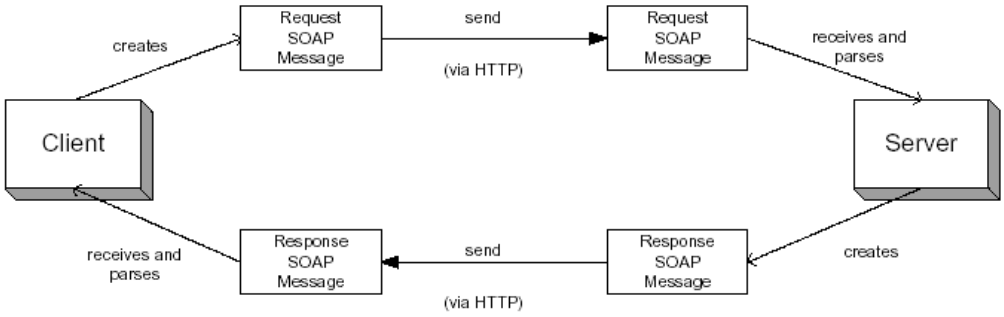

Figura 2.1:Interazione client-server usando SOAP

SOAP definisce solo la struttura del messaggio ed alcune regole per elaborare quest'ultimo, rimanendo, quindi, ad un livello alto e completamente indipendente dal protocollo di trasporto sottostante. Aspetti chiave di SOAP sono, quindi, la sua estendibilità dovuta all'uso di schemi XML e l'utilizzo del protocollo HTTP come meccanismo di trasporto tra client e server, e quindi del Web come infrastruttura di comunicazione [46].

2.2.3 Remote Method Invocation

Mentre RPC è ragionevolmente ben adattabile al paradigma di programmazione procedurale, non è direttamente applicabile allo stile di programmazione object oriented che ha riscosso molta popolarità negli ultimi anni.

Ed è qui che entra in gioco RMI: simile a RPC, integra il modello ad oggetti distribuiti nel linguaggio Java in modo naturale, lavorando direttamente sugli oggetti esistenti ed evitando di descriverne i metodi in un file di definizione diverso.

In un classico sistema RPC, il codice dello stub lato client deve essere generato e linkato nel client prima che una procedura remota possa essere chiamata; RMI, invece, è più dinamico perché, sfruttando la capacità di Java di trasferire codice, gli stub necessari per l'invocazione possono essere scaricati a tempo di esecuzione da una locazione remota, per esempio direttamente dal server. Internamente, RMI fa uso della serializzazione degli oggetti per trasmettere tipi di oggetti arbitrari sulla rete, e poiché il codice scaricato può essere dannoso per il sistema, usa un security manager per controllarlo [46].

2.3 Middleware ed infrastrutture di comunicazione

I middleware e le infrastrutture software per sistemi distribuiti forniscono strumenti per la comunicazione ai componenti applicativi e gestiscono aspetti come l'eterogeneità di piattaforma dovuta a differente hardware, software, o linguaggio di programmazione. Inoltre forniscono un set di servizi standard che sono tipicamente utili alle applicazioni distribuite come servizi di directory, file distribuiti, sicurezza crittografica [46].

2.3.1 CORBA

Una delle più usate infrastrutture per i sistemi distribuiti basata sul modello object-oriented è CORBA (Common Object Request Broker Architecture), che è supportata da un esteso consorzio di industrie.

Il primo standard CORBA è stato introdotto nel 1991 ed è stato oggetto di continue e significative revisioni.

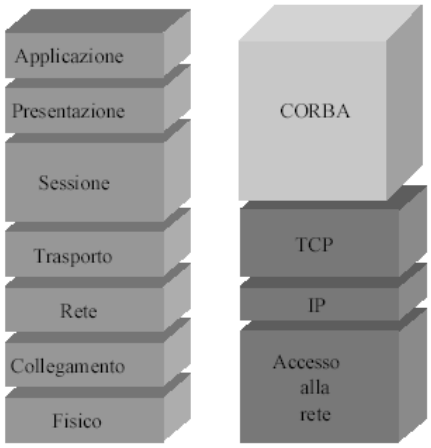

Figura 2.2:CORBA rispetto alla pila OSI

Una parte delle sue specifiche descrive l'Inteface Definition Language (IDL) che tutte le implementazioni di CORBA debbono supportare. L'IDL di CORBA si rifà al C++ ed è usata dalle applicazioni per definire i metodi di un oggetto che è possibile richiamare esternamente ed i tipi di dati sia statici che dinamici utilizzati da essi. E' simile all'IDL di RMI. Tramite un compilatore IDL si ottengono file contenenti del codice di supporto per l'aggancio all'ORB, lo *stub* (per il client), *lo skeleton* (per il server). Lo stub realizza le funzionalità di marshalling, lo skeleton quelle di unmarshalling e si differenzia dallo stub poiché coopera con un Object Adapter per le operazioni che riguardano l'attivazione dell'oggetto.

Il componente centrale di un sistema CORBA è l'object request broker (ORB), una sorta di bus software Esso fornisce un meccanismo per comunicare in maniera trasparente richieste di client alle implementazioni degli oggetti server. Questo semplifica la programmazione distribuita, disaccoppiando il client dai dettagli di invocazione dei metodi: quando, infatti, un client invoca un'operazione, l'ORB si preoccupa di trovare l'implementazione dell'oggetto, di attivarlo se necessario adoperando l'OA, di inoltrargli la richiesta, e ritornare una eventuale risposta al chiamante.

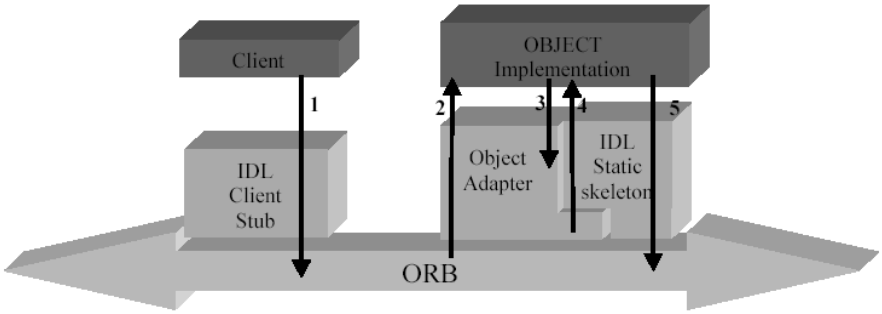

Figura 2.3:Interazione client-server in CORBA

CORBA prevede sia collegamenti di tipo statico che di tipo dinamico tra client e server. Nel caso statico, codice proveniente dal server (stub del client) viene compilato con il client stesso. Di conseguenza il codice dell'oggetto server deve essere conosciuto dal client a tempo di compilazione. Nel caso dinamico è introdotta la possibilità di costruire ed invocare richieste su oggetti non conosciuti a tempo di compilazione.

La comunicazione tra ORB nella rete avviene tramite i protocolli GIOP e IIOP che implementano il marshalling, unmarshalling e la rappresentazione esterna dei dati.

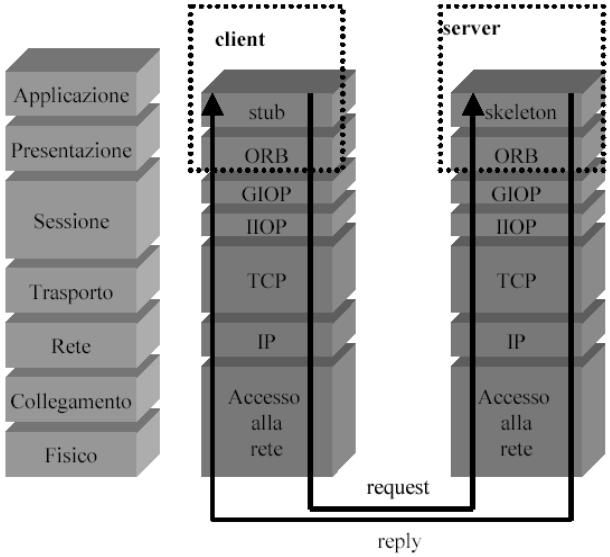

Figura 2.4: Collocazione dei protocolli GIOP-IIOP rispetto alla pila OSI

GIOP è un protocollo astratto che specifica l'insieme di tipi di messaggi utilizzabili tra un client e un server, una sintassi standard nel trasferimento dati delle interfacce IDL, un formato standard per ogni messaggio utilizzabile. In più GIOP richiede un sistema di trasporto con connessione: IIOP è il mapping specifico di GIOP sul protocollo TCP/IP.

CORBA supporta, quindi, sia la trasparenza di comunicazione, mascherando le chiamate remote come semplici invocazioni locali, sia quella di locazione, nascondendo al client tutti i dettagli sulla localizzazione degli oggetti sulla rete e sul loro linguaggio di implementazione.

In aggiunta all'ORB, CORBA definisce un framework di oggetti, limitandosi a fornire le specifiche ma non le implementazioni:

- gli oggetti dei servizi (*Object Services*) costituiscono un insieme di servizi di uso generico per l'implementazione di applicazioni distribuite;
- le facilitazioni comuni (*Common Facilities*) sono un insieme di componenti che forniscono funzionalità applicative di uso comune, come stampa, accesso a DB, accesso a servizi di email;
- le interfacce di dominio (*Domain Interfaces*) specificano componenti che forniscono servizi specifici per particolari domini applicativi;
- gli oggetti delle applicazioni (*Application Interfaces*) corrispondono alla nozione classica di applicazione prodotta da una particolare organizzazione e di conseguenza non sono standardizzati.

Di seguito è fornita una descrizione sommaria di due object service legati ai sistemi distribuiti, il Naming Service ed il Trading Service.

Il Naming Service permette ad un client di trovare un oggetto attraverso il suo nome e ad un server di registrare i suoi oggetti assegnandogli un nome.

Il Trading Service è un servizio di locazione di oggetti, come il Naming Service, ma lavora ad un livello più astratto. Nel Naming Service si cerca un

oggetto a partire da un nome. Nel Trading Service si fanno delle richieste per avere una lista di oggetti che soddisfano certe caratteristiche o proprietà (inserite nella richiesta) [47].

CORBA ha riscontrato molto successo nelle industria e nel campo della ricerca: infatti, implementazioni dello standard sono disponibili presso un gran numero di rivenditori ed esistono persino versioni freeware. CORBA supporta tutti i maggiori linguaggi di programmazione ed è adattabile per quasi tutte le combinazioni di hardware e sistemi operativi. Tuttavia non è adatto a piccoli dispositivi ed a sistemi altamente dinamici, per i quali sono adottati altri sistemi come Jini.

2.3.2 Jini

Jini [46] è un'infrastruttura che si fonda su Java e su RMI per creare una federazione di dispositivi e componenti software che implementano servizi. Essa abilita qualsiasi dispositivo che gira su una Java Virtual Machine a interoperare con gli altri offrendo ed usando servizi.

Un servizio è definito come un'entità che può essere usata da una persona, da un programma o da un altro servizio. Tipici esempi di servizi sono la stampa di un documento o la traduzione di dati da un formato ad un altro, ma anche dispositivi con funzionalità hardware sono considerati servizi.

I client possono usare un particolare servizio senza avere una conoscenza a priori della sua implementazione. L'abilità di Jini di creare spontaneamente una federazione di servizi che sia robusta e tollerante ai guasti, è basata su una serie di concetti fondamentali:

- *Discovery*: processo con cui un client o un servizio localizza il sistema di nomi all.interno della rete;
- *Join*: meccanismo che permette di registrare un nuovo servizio sul sistema di nomi locale o remoto;

- *Lookup*: processo attraverso il quale il client interroga il sistema di nomi per trovare il servizio richiesto.

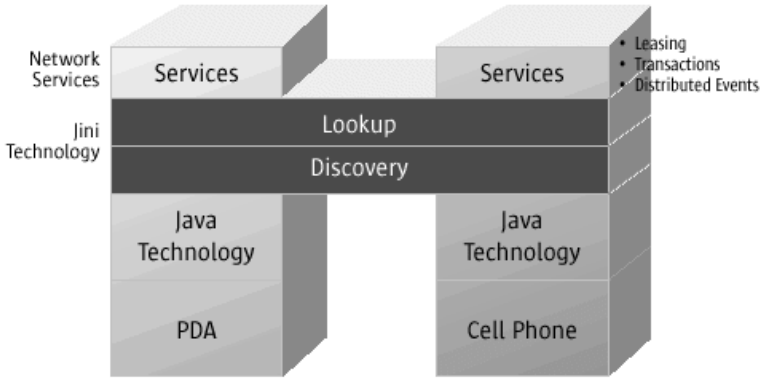

Figura 2.5:Architettura di Jini

2.3.2.1 Discovery

I servizi usano un meccanismo standard per registrarsi nella federazione: innanzitutto, interrogano la rete locale per localizzare *il Lookup Service*. Questo servizio è un sistema che tiene traccia dei servizi che hanno notificato la loro presenza sulla rete. Poiché questo componente è di vitale importanza, è previsto che, per aumentare la robustezza e l'affidabilità del sistema, vi possano essere più Lookup Service in esecuzione contemporaneamente su macchine diverse.

La ricerca avviene, a seconda che si cerchi in una rete locale (LAN) o geografica (WAN), attraverso richieste multicast o unicast. Il multicast discovery è utilizzato per trovare un Lookup Service che contiene servizi che appartengono alla comunità locale Jini, mentre l'unicast discovery permette di localizzare Lookup Service che contengono servizi remoti.

2.3.2.2 Join

Quando un servizio ha terminato la fase di Discovery e ha trovato un Lookup Service, ha la necessità di registrasi presso esso. Il servizio invia un oggetto proxy *(service object)* e gli attributi ad esso associati. Il service object

contiene la descrizione dell'interfaccia per l'utilizzo del servizio, nonché l'implementazione dei metodi che verranno utilizzati dalle applicazioni.

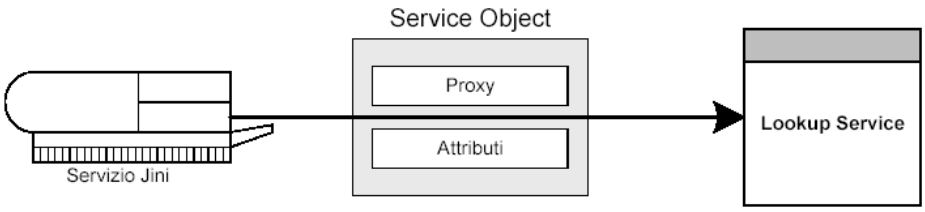

Figura 2.6:Registrazione del servizio

I servizi sono tenuti a rinnovare la propria registrazione dopo un tempo stabilito al momento della registrazione stessa (leasing). Allo scadere del tempo, se il servizio non rinnova la registrazione, viene cancellato dalle entry del servizio di Lookup.

2.3.2.3 Lookup

Questa fase avviene quando un client ha già effettuato la fase di Discovery con successo e vuole richiedere un servizio Jini che abbia le caratteristiche richieste. La richiesta al Lookup Service avviene specificando uno o più dei seguenti campi:

- *attributi*: viene effettuata una comparazione con gli attributi registrati all'atto del Join dei servizi;
- *interfaccia*: si specifica un'interfaccia che il servizio poi implementa. Si può, ad esempio, specificare un'interfaccia stampante per poi scegliere tra i service provider che la implementano;
- *ID number*: ad ogni servizio Jini, all'atto della registrazione, è assegnato un identificativo che può essere utilizzato nella fase di Lookup per richiedere un servizio ben preciso.

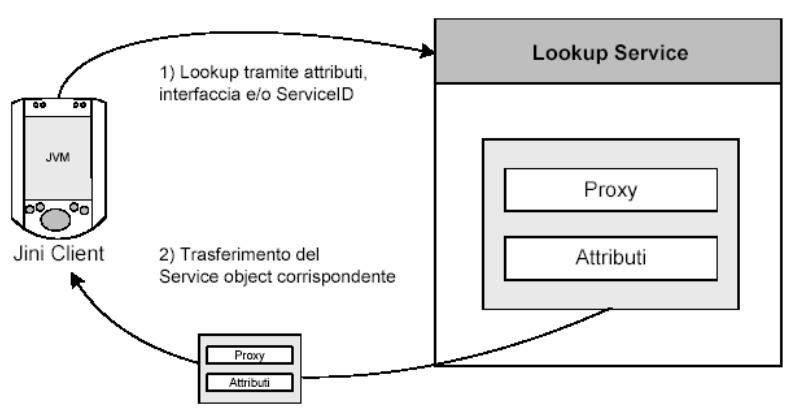

Figura 2.7:Lookup di un servizio

Se gli attributi specificati dal client corrispondono ad un servizio registrato nel Lookup Service, quest'ultimo risponde con il service object corrispondente che verrà usato per invocare il servizio [48].

La tecnologia Jini, anche se particolarmente adatta per lo sviluppo di sistemi dinamici, presuppone che ogni componente della federazione sia in grado di eseguire una Java Virtual Machine. L'ipotesi non è banale e non è sempre possibile, non solo per mancanza di risorse, ma anche per questioni di conflittualità di interessi. Per questi motivi, sono stati ricercati nuovi modelli basati su tecnologie aperte e popolari, come ad esempio i *Web Services*.

2.3.3 Web Services

Il modello dei Web Services è un insieme di standard per la pubblicazione, il discovery e la composizione di servizi indipendenti in una rete aperta [1].

Il concetto di Web Service è molto simile a quello di oggetto, nel significato con cui lo si usa nella Object-Oriented Programming: un oggetto è un modulo software che offre una serie di funzioni utilizzabili dall'esterno da parte di altro software, tramite una interfaccia di comunicazione dichiarata dall'oggetto stesso. Così come per CORBA ed RMI, anche un Web Service

offre una funzionalità (servizio) ad altri client sulla rete attraverso un'interfaccia ben definita. La differenza è che, in questo caso, il servizio è posto sul web, e l'integrazione con il servizio avviene attraverso lo scambio di messaggi sulla rete [49].

La forza dei Web Services è di utilizzare un set base di protocolli disponibili ovunque, permettendo l'interoperabilità tra piattaforme molto diverse e mantenendo comunque la possibilità di utilizzare protocolli più avanzati e specializzati per effettuare compiti specifici. I protocolli alla base del modello dei Web Services sono quattro [4]:

- XML è lo standard usato per rappresentare i dati trasportati;
- SOAP è lo standard usato per definire il formato dei messaggi scambiati;
- WSDL è lo standard usato per descrivere il formato dei messaggi da inviare al Web Service, quali sono i metodi esposti, quali sono i parametri ed i valori di ritorno;
- UDDI è lo standard promosso dall'omonimo consorzio, che ha come scopo quello di favorire lo sviluppo, la scoperta e l'interoperabilità dei servizi Web.

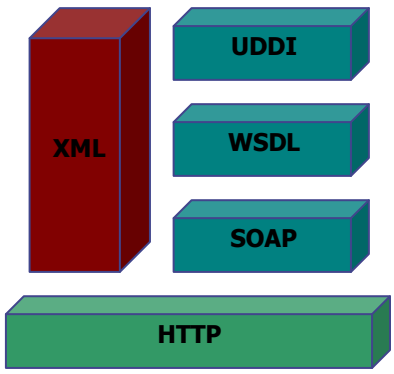

Figura 2.8:Protocolli standard dei Web Services

2.3.3.1 WSDL

I web services sono visti come software disponibile sul web, utilizzabile da altri web services o da utenti: di conseguenza essi debbono essere descritti cosicché gli altri componenti possano usarli facilmente, disaccoppiando interfaccia ed implementazione.

Il Web Service Description Language [51] è il linguaggio standard per la descrizione delle interfacce dei web services (l'equivalente di IDL per CORBA). Esso presenta le caratteristiche architetturali di molti altri linguaggi basati su XML nati di recente: infatti fa riferimento e si integra con standard esistenti, evitando di ridefinire ciò che è già stato definito ed inoltre predilige l'uso di XML Schema per il type system e di SOAP per la definizione dei messaggi.

WSDL si presenta, quindi, come un formato XML per descrivere servizi di rete come un insieme di punti terminali, detti *porte*, operanti su messaggi contenenti informazioni di tipo "documentale" o "procedurale". Esso opera una chiara distinzione tra i messaggi e le porte: i messaggi, ossia la sintassi e la semantica proprie di un servizio Web, sono sempre astratti, mentre le porte, l'indirizzo di rete grazie al quale è possibile richiamare il servizio Web desiderato, sono sempre concrete.

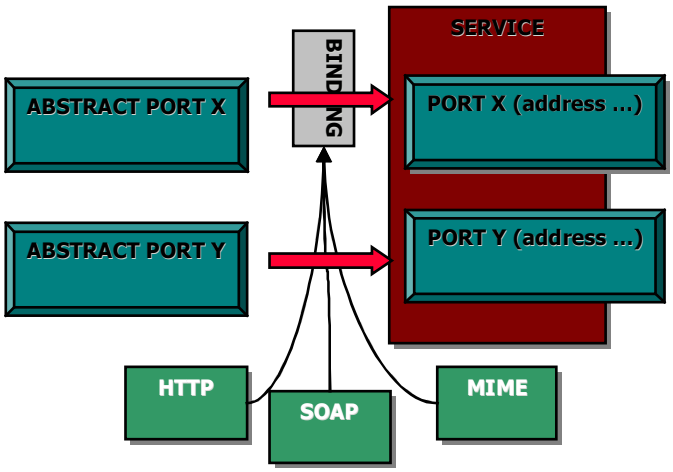

Figura 2.9:Dalla descrizione astratta a quella concreta

All'utente non viene chiesto di fornire informazioni sulla porta in un file WSDL. Un file WSDL può contenere unicamente informazioni di interfaccia astratte e non può fornire dati di implementazione concreta. Sono questi i requisiti da rispettare affinché un file WSDL sia considerato valido.

WSDL separa, dunque, gli aspetti "astratti" della descrizione di un servizio da quelli "concreti" che lo legano a particolari protocolli di rete o di RPC, consentendo ad uno stesso servizio di poter avere implementazioni diverse, basate sulla stessa descrizione astratta, garantendo in questo modo che i sistemi possano comunicare tra di loro e favorendo il riutilizzo delle descrizioni astratte per la creazione di nuovi servizi [49].

2.3.3.2 UDDI

L'Universal Description, Discovery and Integration (UDDI) Service è un'iniziativa supportata da IBM, Microsoft, e HP, e fornisce ai client un meccanismo per trovare dinamicamente web services.

UDDI è un registro pubblico progettato per contenere informazioni strutturate sulle aziende e i rispettivi servizi. Lo scenario nel quale UDDI trova pieno utilizzo può essere ricondotto a tre fasi fondamentali:

- pubblicazione (*publishing*): il fornitore del servizio per renderlo pubblico contatta il service registry che provvede ad inserirlo nel registry tramite UDDI;
- ricerca (*finding*): alla richiesta di un servizio, il service registry provvede a cercare quelli che meglio rispondono alle esigenze del richiedente;
- collegamento (*binding*): nel momento in cui il service registry fornisce la risposta può stabilirsi il collegamento tra il richiedente del servizio web e il fornitore.

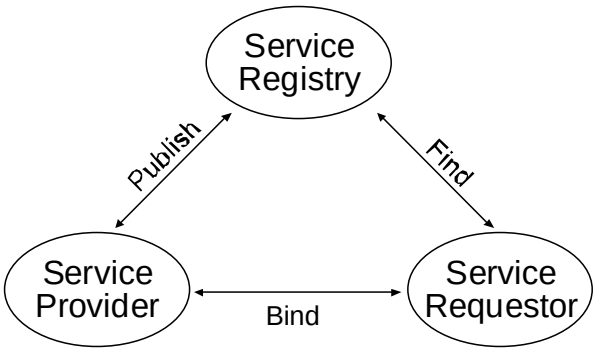

Figura 2.10: Scenario di utilizzo di UDDI

La pubblicazione sul registro UDDI è un processo relativamente semplice e consiste nel raccogliere alcune informazioni di base relative al modo in cui si desidera modellare in UDDI la voce che definisce l'azienda e i relativi servizi, dopo di che si passa alla registrazione vera e propria, adoperando un'interfaccia utente basata su web o tramite un programma. Le informazioni alla base della registrazione sono così classificate:

- *white pages*, che forniscono il nome dell'azienda, una breve descrizione, se necessario in più lingue, le persone da contattare per ottenere informazioni sui servizi Web offerti ed altri identificativi;
- *yellow pages*, che classificano azienda e servizi attraverso tassomonie standard. Le tassonomie attualmente supportate sono NAICS (North American Industry Classification System), UNSPSC (Universal Standard Products and Services Codes), ISO 3166, SIC (Standard Industry Classification) e GeoWeb Geographic Classification;
- *green pages*, che contengono informazioni tecniche circa i web service che sono esposte attraverso il concetto di tModel. La struttura tModel, abbreviazione di "Technology Model", rappresenta le tracce tecniche, le interfacce e i tipi astratti dei metadati. I file WSDL sono esempi perfetti dei tModel UDDI.

Per ricercare il servizio sono utilizzate una serie di API XML basate su SOAP che vengono esposte per fare browsing su un repository site e per prelevare le

informazioni che occorrono riguardo al servizio desiderato. Esistono anche API, destinate a chi pubblica nuovi web service, per l'inserimento di nuove informazioni in questo repository.

Infine, una volta trovato il web service di cui si aveva bisogno, viene effettuato il binding tra interfaccia e implementazione. Anche UDDI, infatti, opera una distinzione simile a quella fatta da WSDL tra astrattezza e implementazione e lo fa attraverso il concetto di tModel. A supporto dei tModel, vi sono i modelli di binding, ossia le implementazioni concrete di uno o più tModel. È nel modello di binding che si registra il punto di accesso per un'implementazione specifica di un tModel.

Il registro UDDI e il linguaggio WSDL fungono da specifiche complementari per la definizione di configurazioni software basate sui servizi web. Il linguaggio WSDL fornisce un metodo formale, non legato a produttori specifici, per definire i servizi web, rendendo possibile l'utilizzo delle chiamate di procedure remote dell'ultima generazione, mentre UDDI fornisce una vasta infrastruttura standardizzata che consente agli utenti di descrivere e individuare i servizi web. L'utilizzo combinato dei due standard consente la nascita di un vero ecosistema di servizi web [52].

2.4 La comunicazione in ambienti pervasivi

Gli ambienti pervasivi o ubiqui sono caratterizzati da un gran numero di entità autonome dette *agenti*, che contribuiscono a trasformare uno spazio fisico in un ambiente intelligente e computazionalmente attivo. Queste entità possono essere dispositivi, servizi, utenti.

Nonostante siano stati sviluppati vari tipi di middleware ed infrastrutture per rendere possibile la comunicazione tra agenti, tra cui quelli presentati precedentemente, nessuno di questi è riuscito a garantire l'interoperabilità semantica tra essi [1].

Nei sistemi distribuiti, lo scambio di messaggi rappresenta, come si è visto, una aspetto fondamentale. I messaggi, che contengono descrizioni di entità, servizi, eventi ed altri concetti, possono essere raggruppati essenzialmente in tre categorie:

- *Advertisement*: corrisponde all'invio di descrizioni di servizi offerti, di dispositivi, interfacce per registri;
- *Notification*: corrisponde all'invio di descrizioni di eventi, ad esempio l'arrivo di un'entità nell'ambiente;
- *Query*: corrisponde alla richiesta di servizi o entità che rispondono a certi requisiti e alla ricezione delle descrizioni di servizi ed entità che li soddisfano.

Per consentire lo scambio di tali messaggi tra due agenti occorre necessariamente che il loro formato sia comune ed inoltre che siano comuni interfacce e protocolli. E' difficile, poi, aspettarsi che le diverse entità comprendano la semantica dell'ambiente e delle altre entità quando interagiscono fra loro. Deve essere quindi conosciuta o scoperta anche la semantica dei messaggi, ossia il vocabolario dei messaggi, che include i nomi e i valori validi degli elementi.

Mentre per sistemi semplici o chiusi, tutti gli schemi richiesti per interpretare il contenuto dei messaggi sono compilati nei componenti del sistema stesso, in un sistema aperto le parti che comunicano sono autonome, eterogenee ed evolvono nel tempo. Occorre, quindi, un modello aperto che renda possibile la scoperta e l'utilizzo degli schemi quando occorrono e mentre il sistema è in esecuzione [1].

2.4.1 L'interoperabilità semantica: esigenza di nuovi strumenti

I registri di oggetti, come il CORBA Naming Service, forniscono un meccanismo base per la ricerca di servizi della cui esistenza si è già a conoscenza. Brokers, come il CORBA Trading Service, forniscono la capacità

di localizzare servizi in base ad alcuni attributi degli stessi. Altri servizi che forniscono caratteristiche simili sono, ad esempio, JINI, LDAP, Microsoft's Registry. Questi standard definiscono interfacce e formati per le descrizioni ma non ne definiscono il contenuto, la semantica.

Ad esempio, il CORBA Trading Service è un'interfaccia standard per un broker, ma non definisce né le proprietà dei servizi registrati né i valori legali di tali proprietà. Le regole di matching sono limitate e applicabili sono agli attributi definiti esplicitamente. Inoltre mentre per le query esiste una sintassi, non esiste un linguaggio dichiarativo per definire le tipologie di servizi e le loro istanze.

Allo stesso modo, JINI Discovery Service definisce architettura, protocolli e interfacce per lo scambio di tutte le tipologie di messaggi viste precedentemente. I servizi e le entità sono descritte dai Service Entry, ma come nel caso di CORBA, la loro definizione è lasciata agli applicativi. Non sono definiti, inoltre, né un linguaggio standard per la definizione di schemi né meccanismi per la gestione e la validazione delle Service Entry [1].

I Web Services, sebbene siano nati soprattutto per superare i problemi di interoperabilità tra piattaforme, stanno cercando di risolvere anche questi problemi di service discovery e di gestione di descrizioni provenienti da entità autonome. Tuttavia anche per i Web Services non è ancora stato definito il livello semantico, ossia non sono stati definiti standard per la definizione, validazione e scambio di schemi per le descrizioni di entità e servizi e dei loro modelli tecnici. Il Web Semantico è destinato a definire questi aspetti.

2.5 Web Semantico

Il Web Semantico è un insieme di standard aperti, indipendenti dalle tecnologie, nati per consentire lo scambio di descrizioni di entità e relazioni per migliorare le capacità di ricerca nel Web, ma che sono ben adattabili ad alcuni dei requisiti di un sistema pervasivo[1].

Il Web Semantico rappresenta un'estensione del Web attualmente esistente che, in aggiunta, mira alla cooperazione tra uomo e calcolatori [5]. Gran parte del contenuto di Internet è progettato per essere letto da esseri umani e non per essere trattato da programmi ed è ben lontano dal poter fornire una solida piattaforma che renda possibile un'interpretazione e una comprensione semantica da parte di agenti automatici.

Questo è ciò che viene fornito dal Web Semantico, termine coniato per la prima volta da Tim Berners-Lee, l'ideatore del WWW, e che può essere riassunto come segue [6]:

- l'informazione deve essere *machine-readable*, ossia non più pensata per essere letta direttamente dall'uomo, ma mirata ad essere in un formato facilmente elaborabile dalla macchina, da agenti intelligenti, servizi specializzati, siti web personalizzati e motori di ricerca potenziati semanticamente;
- deve essere fornito un supporto per *l'interoperabilità sintattica*, intesa come la facilità di leggere dati e ottenere una rappresentazione utilizzabile dalle applicazioni;
- deve essere fornito un supporto per *l'interoperabilità a livello semantico*: non sono più sufficienti standard per la forma sintattica dei documenti, ma anche per il loro contenuto semantico; interoperabilità semantica significa definire mapping tra termini sconosciuti e termini conosciuti nei dati;
- il formato utilizzato per lo scambio dei dati deve permettere di poter esprimere qualsiasi forma di dati, poiché non è possibile anticiparne tutti i suoi usi potenziali (*potere espressivo universale*). Per raggiungere questo obiettivo, è necessario basarsi su un modello comune di grande generalità. Solo cosÏ qualsiasi "prospettiva" può trovare espressione all'interno del modello.

Il Web Semantico, quindi, mira a riportare chiarezza, formalità e organizzazione dei dati, collegando le informazioni a concetti astratti organizzati in una gerarchia, a sua volta descritta in un meta-documento; permettendo a vari agenti automatici di cogliere il contesto semantico di una fonte informativa interpretando le varie relazioni esistenti tra le risorse, formulando asserzioni sulle stesse, nonché controllando la loro attendibilità.

2.5.1 Architettura e linguaggi del Web Semantico

Nella visione di Tim Berners-Lee, il Web Semantico è un'architettura strutturata su almeno quattro livelli:

- il livello dei dati (un semplice modello dei dati e una sintassi per i metadati);
- il livello schema (una base per la definizione di una vocabolario);
- il livello ontologico (per la definizione delle ontologie);
- il livello logico (supporto al ragionamento).

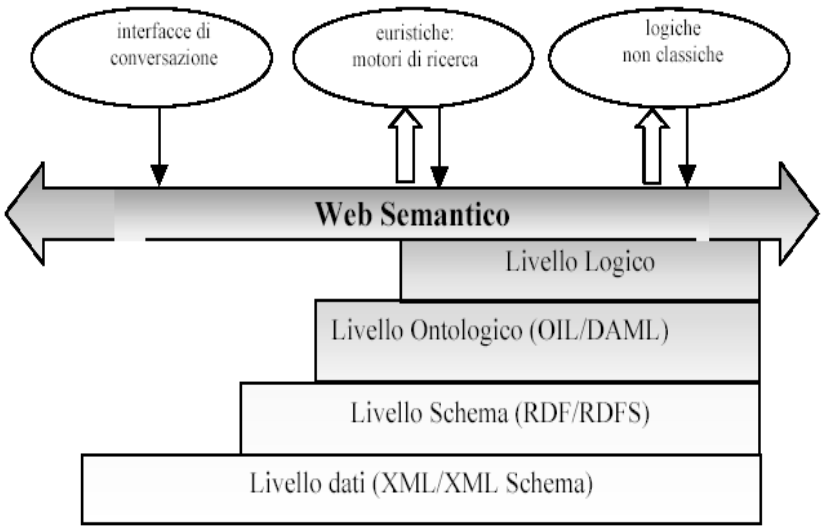

Figura 2.11:Architettura del Web Semantico

Il Web Semantico, a differenza del Web che si fonda su documenti, si fonda su risorse. Esse sono descritte adoperando i linguaggi RDF e RDF Schema, entrambi basati su tecnologie XML. Questo tuttavia non è sufficiente perché occorre catturare la semantica delle risorse descritte e delle relazioni tra esse e per fare ciò occorre introdurre vocabolari ontologici. Una volta che le risorse sono state ben definite non solo sintatticamente ma anche semanticamente, è possibile estendere questi vocabolari ontologici con logiche che supportano ragionamenti automatici.

2.5.2 XML e XML Schema

XML (eXtension Markup Language) [7][8] è un linguaggio di markup destinato alla descrizione di documenti, strutturati in maniera arbitraria. A differenza di HTML, che è utilizzato per descrivere la modalità con cui documenti ipertestuali con strutture fisse sono visualizzati, XML separa il contenuto del documento dalla sua modalità di visualizzazione. E' dunque uno standard elaborato per garantire l'interoperabilità sintattica, ossia per fare in modo che le informazioni non siano semplicemente formattate per facilitarne il reperimento per un utente umano, ma anche facilmente elaborate da agenti software.

XML definisce una struttura ad albero per i documenti dove ciascun nodo individua un tag ben definito (*elemento*) mediante il quale è possibile in qualche modo interpretare le informazioni che esso racchiude (*attributi*).

```
<?xml version = "1.0"?>
<person>
        <name>Jim</name>
        <personal_info>
                <office_number>768</office_number>
                <phone_number>1024</phone_number>
        </personal_info>
</person>
```

Figura 2.12:Semplice documento XML

É possibile imporre delle restrizioni con cui i tag XML possono essere usati e quali annidamenti di tali tag sono permessi mediante l'uso di due tecnologie:

- DTD (Document Tye Definition): Contengono le regole che definiscono i tag usati nel documento XML, in altre parole ne definiscono la struttura. Questi possono essere dei file esterni o specificati direttamente all'interno del documento;
- XML Schema [9]: Spesso i documenti condividono una struttura specifica per un certo dominio, per consentire sia al software sia agli umani di riconoscere il contenuto dell'XML si richiede che tali strutture siano documentate in un formato comprensibile ad entrambi. A tal scopo è stato introdotto XML Schema, il quale permette di definire un vocabolario utilizzabile per descrivere documenti XML facendo uso della loro stessa sintassi. Le sue specifiche assumono che si faccia uso di almeno due documenti, un'istanza ed uno schema. La prima contiene le informazioni che interessano realmente, mentre il secondo descrive struttura e tipo della precedente.

XML, tuttavia, non gestisce la semantica dei contenuti: essa non è specificata in modo esplicito, ma è "incorporata" nei nomi dei tag, ossia il vocabolario degli elementi e le loro combinazioni non sono prefissati ma possono essere definiti ad hoc per ogni applicazione. Tale semantica, quindi, non è definita formalmente e può risultare eventualmente comprensibile solo all'uomo e non alla macchina: un entità software riconosce i contenuti, ma non è in grado di attribuire loro un significato.

XML, quindi, fornisce l'interoperabilità sintattica ma non riesce a garantire quella semantica né a fornire meccanismi di classificazione o ragionamento e quindi deve necessariamente essere affiancato ad altri linguaggi più potenti.

2.5.3 RDF e RDF Schema

RDF (Resource Description Framework) [11] è lo standard che consente l'aggiunta di semantica a un documento XML e quindi si pone ad un

livello direttamente superiore rispetto ad esso. Esso è in un certo senso un'applicazione di XML: se XML è un'estensione del documento, RDF può essere visto come un'estensione dei dati introdotti da XML.

Il modello base dei dati RDF è composto da:

- *Risorse*: con questo termine si intende qualsiasi cosa possa essere descritta. Una risorsa può essere ad esempio una pagina Web oppure un qualsiasi oggetto anche se non direttamente accessibile via Web (ad esempio un libro, una persona). Una risorsa viene identificata univocamente attraverso un URI.
- *Proprietà*: una proprietà è una caratteristica, una relazione che descrive una risorsa. Il significato, l'insieme di valori che può assumere, i tipi di risorse a cui può riferirsi sono tutte informazioni reperibili dallo schema RDF in cui essa è definita.
- *Asserzioni*: una asserzione è costituita da un soggetto (la risorsa descritta) , un predicato (la proprietà) e un oggetto (il valore attribuito alla proprietà), dove l'oggetto può essere una semplice stringa o un'altra asserzione.

Volendo fare un paragone con un database relazionale potremmo dire che una riga di una tabella è una risorsa RDF, il nome di un campo (una colonna) è il nome di una proprietà RDF, il valore del campo è il valore della proprietà.

Un semplice esempio di utilizzo del modello di RDF può essere fornito dalla seguente asserzione: **" Jim's phone_number is 1024 "**, che può essere schematizzata graficamente così:

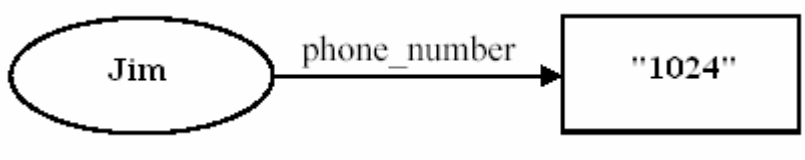

Figura 2.13:Semplice esempio di un'asserzione RDF

La precedente asserzione è formalizzata in RDF/XML così:

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://www.example.com/props/"
        xmlns:base="http://www.organization.com/people">
        <rdf:Description rdf:about="Jim">
                <s:Phone_number>1024</s:Phone_number>
        </rdf:Description>
</rdf:RDF>
```

Figura 2.14:Un semplice documento RDF/XML

Sintatticamente, i concetti espressi con RDF vengono serializzati mediante XML. RDF definisce, quindi, un semplice modello dei dati per descrivere proprietà e relazioni fra risorse. In RDF però non esistono livelli di astrazione: ci sono le risorse e le loro relazioni, tutte organizzate in un grafo piatto. In altri termini non è possibile definire tipi (o classi) di risorse con loro proprietà specifiche.

Vista l'utilità di poter definire classi di risorse, RDF è stato arricchito con un semplice sistema di tipi detto RDF Schema. Il sistema di tipi RDF Schema ricorda i sistemi di tipi dei linguaggi di programmazione object-oriented (tipo Java). Una risorsa può, per esempio, essere definita come istanza di una classe (o di più classi) e le classi possono essere organizzate in modo gerarchico.

RDF Schema utilizza il modello RDF stesso per definire il sistema di tipi RDF, fornendo un insieme di risorse e proprietà predefinite che possono essere utilizzate per definire classi e proprietà a livello utente. E' possibile, inoltre, definire vincoli di dominio e di range sulle proprietà ed alcuni tipi di relazioni (comprese quelli di sottoclasse di una risorsa e sottotipo di una proprietà). L'insieme di tali elementi è detto vocabolario dell'RDF Schema.

Il linguaggio di specifica RDFS è un linguaggio dichiarativo poco espressivo ma molto semplice da implementare. Si esprime attraverso la sintassi di

serializzazione RDF/XML e si avvale del meccanismo dei namespace XML per la formulazione delle URI che identificano in modo univoco le risorse (definite nello schema stesso) consentendo il riutilizzo di termini definiti in altri schemi. Concetti e proprietà già dichiarati per un dominio possono essere impiegati di nuovo o precisati per incontrare le esigenze di una particolare comunità di utenti.

Sebbene, in prima battuta, RDF e RDF Schema sembrano essere un buon strumento per la definizione di un linguaggio di markup per il Web Semantico (ad esempio, per determinare le relazioni semantiche tra termini differenti),in realtà essi mostrano di non avere sufficiente potere espressivo: non consentono, infatti, di specificare le proprietà delle proprietà, le condizioni necessarie e sufficienti per l'appartenenza alle classi e gli unici vincoli che si possono definire sono quelli di dominio e range delle proprietà.

Il loro utilizzo nei sistemi di rappresentazione della conoscenza è limitato, inoltre, da un'altra caratteristica: non permettono di specificare meccanismi di ragionamento, ma rappresentano semplicemente un sistema a frame. I meccanismi di ragionamento devono essere costruiti, quindi, ad un livello superiore.

2.5.4 DAML+OIL

Il linguaggio DAML+OIL [18] è uno standard che consente la rappresentazione delle informazioni in modo che il loro significato sia comprensibile alle macchine. I due risultati più interessanti degli ultimi anni in fatto di sviluppo verso il Web Semantico, DAML-ONT e OIL, si sono fusi in un unico linguaggio, DAML+OIL appunto, che incorpora caratteristiche provenienti dal lavoro del gruppo americano (DARPA) e del gruppo europeo (progetto On-To-Knowledge, IST).

DAML+OIL riesce a garantire l'interoperabilità sintattica e semantica sfruttando la sintassi e le caratteristiche dei linguaggi che si trovano ai livelli più bassi dello stack del WEB Semantico, ossia XML e RDF, ed in più

aggiunge, rispetto a questi, una maggiore forza espressiva ottenuta a partire da primitive di modellazione ereditate dai sistemi basati su Frame e la capacità di produrre asserzioni in logica formale ed effettuare ragionamenti in modo automatico, traendo spunto dalle Description Logics.

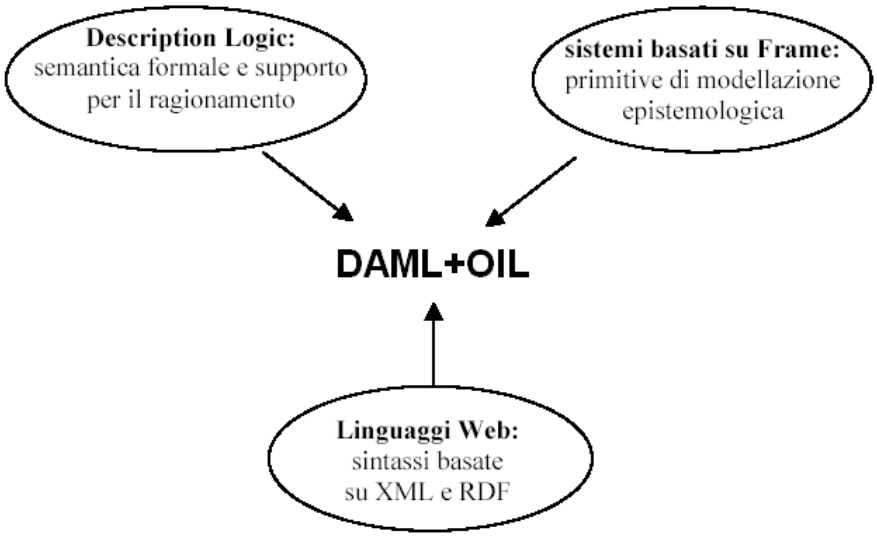

Figura 2.15:Il modello di DAML+OIL

2.5.4.1 Sistemi basati su Frame

I sistemi basati su Frame somigliano agli approcci orientati agli oggetti, ma considerano le cose da un punto di vista differente. Le primitive di modellazione sono le classi (o frame), ognuna delle quali ha determinate proprietà (o slot), chiamate attributi. Essi non hanno una visibilità globale, ma sono applicabili alle sole classi per cui sono definiti. Un frame fornisce un certo contesto per la modellazione di un aspetto del dominio in esame.

Dai sistemi basasti su Frame, DAML+OIL eredita le primitive di modellazione essenziali. DAML+OIL, infatti, è basato sulla nozione di classe, la definizione delle sue proprietà e delle sue superclassi e sottoclassi. Le relazioni possono essere definite come entità indipendenti aventi un certo dominio e intervallo. Come le classi, anche le relazioni possono essere organizzate in una gerarchia.

Ad incrementare il potere espressivo contribuiscono, poi, due aspetti: i costruttori e i tipi di assiomi. I costruttori permettono la creazione, oltre che di classi intese nel senso classico, con un nome ed associate ad un URI, anche di classi intese come espressioni, cioè come il prodotto di un certo numero di operatori. Una classe può essere ottenuta dall'unione o dall'intersezione di altre classi, elencandone gli elementi, può essere complementare di un'altra, può essere definita come collezione degli elementi che hanno almeno o al più un certo numero di valori distinti di una proprietà.

Il secondo aspetto è rappresentato dagli assiomi, che permettono la dichiarazione di relazioni di classificazione o equivalenza tra classi e/o proprietà, la disgiunzione tra classi, l'equivalenza o meno di oggetti diversi, le proprietà delle proprietà.

2.5.4.2 Description Logics

Le Description Logics [17] rappresentano una classe di logiche che sono specificatamente designate a modellare vocabolari. Esse descrivono la conoscenza in termini di concetti (paragonabili ai frame) e restrizioni (paragonabili agli slot) che sono utilizzate per derivare automaticamente classificazioni tassonomiche.

DAML+OIL eredita dalle Description Logics la semantica formale e il supporto per il ragionamento, stabilendo una corrispondenza tra concetti logici e tag DAML+OIL che li rappresentano. A partire da un documento DAML+OIL è possibile, infatti, creare una Knowledge Base: una KB è un database con capacità di ragionare in maniera automatica, in grado non solo di rispondere a query grazie a dei match ma anche effettuando ragionamenti.

Una KB è fatta di due componenti:

- *Intensionale*: uno schema che definisce classi, proprietà e relazioni tra classi (Tbox, ossia Terminological knowledge).
- *Estensionale*:un'istanza (parziale) dello schema, contenente asserzioni su individui (Abox, ossia Assertional knowledge)

Il Tbox è il modello di ciò che può essere vero, l'Abox è il modello di ciò che correntemente è vero. Effettuando ragionamenti sul KB, è possibile ottenere risposte a questioni importanti come:

- la *satisfiability* di un concetto: se un concetto esiste;
- la *sussunzione*: se un concetto è un caso di un altro concetto;
- la *consistence*: se l'intera KB verifica la satisfiability;
- l'*instance checking*: se un'asserzione verifica la satisfiability.

Uno dei motivi principali per cui si usano le DL come supporto per il ragionamento in DAML+OIL è che la satisfiability e la sussunzione possono essere ricavate in modo computazionalmente efficiente, attraverso algoritmi non complessi: essi convertono le asserzioni in una forma normale e poi costruiscono un insieme di vincoli che sono analizzati per verificare se esistono contraddizioni o meno.

2.5.5 OWL

OWL (Ontology Web Language) [16] è un'estensione di DAML+OIL ottenuta attingendo da esso gli insegnamenti tratti dal suo uso applicativo, ed in virtù di questo molto simile ad esso. Come DAML+OIL, anche OWL estende RDF ed RDF Schema e usa la sintassi XML.

Esso fornisce tre sottolinguaggi:

- *OWL Full*: consente di combinare OWL con RDF e RDF Schema. E' destinato agli utenti che vogliono la massima espressività e la libertà sintattica di RDF senza alcuna garanzia computazionale. Il suo vantaggio è che è pienamente compatibile con RDF , sia sintatticamente che semanticamente, cioè qualsiasi documento RDF legale è anche un documento OWL Full legale.
- *OWL DL*: inserisce alcuni vincoli sul modo di combinare OWL con RDF Schema. E' destinato agli utenti che vogliono la massima

espressività senza perdere l'efficienza e la completezza computazionali, e i benefici dei sistemi che ragionano. E' stato realizzato per supportare le Description Logics esistenti.

- *OWL Lite*:è un sottoinsieme di OWL DL che è facilmente utilizzabile ed implementabile. Aggiunge a RDF Schema molte funzionalità utili per supportare le applicazioni web. Comprende molte delle caratteristiche più usate di OWL ed è destinato agli sviluppatori che vogliono utilizzare OWL ma vogliono iniziare con un set relativamente semplice di caratteristiche del linguaggio.

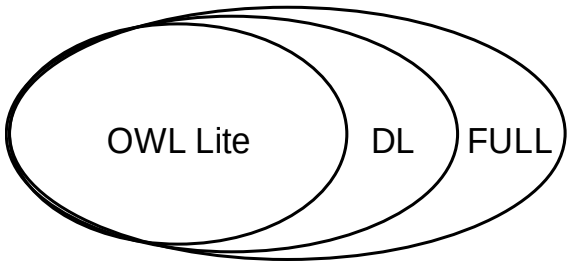

Figura 2.16:Sottolinguaggi di OWL

Capitolo 3

Le ontologie

3.1 Introduzione alle ontologie

Attraverso i linguaggi del web semantico, capaci di rappresentare e di ragionare circa la semantica delle informazioni, è possibile definire vocabolari descrittivi di domini applicativi di tipo standard, le ontologie.

"Ontologia" è un termine carico di valenze teoretiche nella tradizione del pensiero occidentale e, per avere un punto di riferimento sintetico ma attendibile, è preferibile sfogliare le pagine del Dizionario di filosofia di Nicola Abbagnano: "…La seconda concezione fondamentale [di metafisica] è quella della metafisica come ontologia o dottrina che studia i caratteri fondamentali dell'essere: quei caratteri che ogni essere ha e non può non avere".

Sebbene l'ontologia sia nata in ambito filosofico, negli ultimi anni, si è affermata una nuova scuola di pensiero che propone una caratterizzazione logica rigorosa delle categorie ontologiche fondamentali, applicata ai sistemi di computing, con lo scopo di aumentarne la trasparenza semantica e l'interoperabilità. Tale approccio, che coinvolge attività di modellazione concettuale e di ingegneria della conoscenza in una prospettiva fortemente interdisciplinare, ha portato alla formulazione di nuove definizioni del termine ontologia, distinte da quella prettamente filosofica:

- Un'ontologia identifica i termini basilari e le relazioni di un determinato dominio, definendone in questo modo il vocabolario, e le regole per combinare tali termini e tali relazioni, andando oltre il vocabolario stesso [22].
- Un'ontologia è un insieme di termini gerarchicamente strutturati per descrivere un dominio che può essere usato come fondamenta per una base di conoscenza [23].
- Un'ontologia è un mezzo per descrivere esplicitamente la concettualizzazione presente dietro la conoscenza rappresentata in una base di conoscenza [24].
- Le ontologie sono punti di incontro tra le concettualizzazione condivise. Le concettualizzazioni condivise includono framework concettuali per modellare la conoscenza di dominio, protocolli specifici per la comunicazione tra gli agenti e accordi circa la rappresentazione di particolari teorie di dominio. In un contesto di condivisione della conoscenza, le ontologie sono specificate nella forma di un vocabolario di termini [25].

Ma la definizione maggiormente adottata, soprattutto nell'ambito delle "artificial intelligence communities", è quella proposta da Gruber, secondo cui un'ontologia è una "*specifica esplicita e formale di una concettualizzazione condivisa*" [21].

Una *concettualizzazione* è l'insieme di oggetti, concetti e altre entità, che si assume esistano in una particolare area di interesse, e delle relazioni che sussistono tra essi; per *esplicita* si intende che i tipi di concetti usati e i vincoli sul loro uso sono esplicitamente definiti; *formale* si riferisce al fatto che l'ontologia dovrebbe essere "machine-readable"; *condivisa* riflette il fatto che l'ontologia cattura la conoscenza consensuale, cioè quella non propria di un individuo, ma accettata da un gruppo.

Tutte le definizioni proposte, comunque, mettono in evidenza alcuni aspetti fondamentali che un'ontologia deve possedere, affinché essa risulti utile come strumento di interoperabilità e condivisione della conoscenza tra applicazioni:

- *Chiarezza*: un'ontologia dovrebbe comunicare in modo oggettivo e non ambiguo il significato dei termini, indipendentemente dal contesto per cui sono definiti. Il mezzo attraverso il quale è possibile perseguire tale fine è la formalizzazione. Ogni dichiarazione dovrebbe, inoltre, essere affiancata da una spiegazione in linguaggio naturale.
- *Coerenza*: le informazioni dedotte dagli assiomi devono essere consistenti con le definizioni contenute nell'ontologia..
- *Estensibilità*: un'ontologia dovrebbe essere progettata per un uso condiviso del vocabolario in modo che sia possibile definire nuovi termini e specializzare quelli esistenti senza dovere modificare le definizioni esistenti.
- *Indipendenza dalla codifica*: la concettualizzazione dovrebbe essere definita a livello di conoscenza senza alcuna dipendenza da una particolare linguaggio di codifica. Questo perché gli agenti e le applicazioni potrebbero essere implementate facendo riferimento a sistemi di rappresentazioni diversi.

Riassumendo quanto detto, quindi, si può dire che un'ontologia consiste di termini o concetti, di termini generali che esprimono le categorie principali in cui è organizzato il mondo, come *cosa, entità, sostanza, uomo, oggetto fisico,* ecc. oppure di termini particolari che descrivono un dominio d'applicazione specifico (ontologie di dominio), delle definizioni delle relazioni che li associano o che impongono particolari vincoli.

3.2 Applicazione delle ontologie in ambienti pervasivi

L'adozione di modelli ontologici all'interno di sistemi di Pervasive Computing ha fondamentalmente lo scopo di rendere possibile l'interazione e la collaborazione tra le differenti parti che compongono l'ambiente pervasivo stesso. Di seguito sono descritti gli aspetti di tali interazioni che presentano un maggior numero di problematiche e per i quali l'utilizzo delle ontologie può rappresentare una valida soluzione.

3.2.1 Definizione dei termini usati nell'ambiente

Molto spesso le entità di un ambiente distribuito non conoscono il significato dei vari termini usati nelle interfacce o come le differenti parti del sistema interagiscono fra loro. Questo problema assume maggiore rilevanza negli ambienti pervasivi, dove si ha a che fare con una miriade di dispositivi e servizi. E' facile, quindi, perdersi in questi ambienti, soprattutto se non sia ha un chiaro modello di come il sistema lavori.

Le ontologie rappresentano una possibile soluzione a questo problema, descrivendo tutti i termini che possono essere usati nell'ambiente, assegnandovi una precisa semantica e definendo le relazioni esistenti tra essi. Nel caso in cui differenti entità dell'ambiente abbiano diverse idee su cosa significhi un particolare termine, esse possono adoperare le ontologie per essere sicure riguardo la sua precisa definizione, evitando in questo modo ambiguità di tipo semantico.

Per ottenere questo risultato, occorre quindi associare ad ogni termine sia una descrizione human-readable che una machine-readable, in modo da consentire sia alle entità umane che a quelle automatiche di avere una chiara conoscenza di tutto ciò con cui hanno a che fare nell'ambiente, abilitando l'interoperabilità semantica tra utenti, agenti software e sistema.

3.2.2 Validazione delle descrizioni

Adoperando le ontologie per modellare un ambiente pervasivo e ciò che esso contiene, è possibile effettuare una validazione delle descrizioni associate ad ogni sua entità per controllare se esse rispettano gli schemi definiti nelle ontologie stesse, garantendo così una corretta caratterizzazione dell'ambiente.

Questa operazione di verifica della consistenza logica della descrizione di un'entità è un'operazione computazionalmente intensiva, poiché viene eseguita solo la prima volta che un'entità è introdotta nell'ambiente o nel caso in cui la descrizione cambia. In entrambe le situazioni, la descrizione viene confrontata con le ontologie esistenti per controllare se è consistente con i concetti descritti in esse. Se viene rilevata un'inconsistenza logica, viene richiesta una revisione della descrizione, magari andando a modificare le proprietà dell'entità descritta. Ad esempio, un'ontologia potrebbe imporre che la potenza di un lampadina dell'ambiente debba avere valore compreso tra 20 e 50 Watt. Nel caso in cui si provi ad istallare una lampadina con potenza di 100 Watt, allora la descrizione della stessa dovrebbe risultare inconsistente con l'ontologia e un messaggio dovrebbe essere generato per evitare che vengano violati aspetti legati alla sicurezza.

3.2.3 Discovery Semantico e Matchmaking

Un ambiente pervasivo è un ambiente aperto, in cui i componenti sono eterogenei ed autonomi. Prima che le entità possano interagire e collaborare per rilasciare servizi, esse debbono scoprirsi fra loro.

I registri di oggetti e i cosiddetti protocolli di discovery, di cui si è già parlato diffusamente, forniscono varie tecniche per la scoperta di oggetti nell'ambiente, ma tutte risultano essere limitate da un punto di vista semantico.

Non è realistico aspettarsi che la richiesta di un oggetto e l'advertisement ad esso relativo siano uguali, poiché offerente e richiedente possono vedere uno stesso oggetto attraverso prospettive diverse o semplicemente averne una

differente conoscenza. Potrebbe, inoltre, non esistere un oggetto che risponda esattamente ai requisiti del richiedente.

Occorre, dunque, modificare le tecniche di discovery esistenti, per rendere più efficiente la ricerca di oggetti cercando di scoprire tutte e sole le entità rilevanti, senza tuttavia conoscerle a priori. Per rilevanti si intendono tutte quelle entità le cui caratteristiche meglio si adattano ai requisiti richiesti da chi è alla ricerca di un oggetto nell'ambiente.

Il discovery semantico, anche chiamato matchmaking, mira a sfruttare la conoscenza dell'ambiente e delle entità in esso contenute, fornita attraverso le ontologie, per determinare, anche nel caso in cui sono utilizzati termini diversi, se request e advertisement sono relazionate, o se addirittura sono semanticamente equivalenti.

Il matchmaking, inoltre, attraverso l'utilizzo delle ontologie, è in grado di filtrare i risultati ottenuti in seguito alla richiesta di un oggetto, ed in particolare di un servizio, in base alle caratteristiche di chi lo richiede, andando nei fatti a valutare una serie di prerequisiti di tipo hardware e software che l'utilizzatore del servizio deve necessariamente possedere per averne una corretta fruizione.

3.2.4 Interazione con i componenti dell'ambiente

I differenti componenti di un sistema pervasivo forniscono una serie di funzionalità utilizzabili da chi popola l'ambiente. Per consentire l'utilizzo di tali funzionalità in modo del tutto generico, ossia senza dover avere un client ad hoc per interagire con ogni componente, occorre avere definizioni formali ed universali che in particolare descrivano i singoli servizi offerti da ogni risorsa e per ognuno di essi i parametri necessari per utilizzarli. Adoperando queste informazioni è possibile non solo invocare una funzionalità di un componente ma anche comporre più funzionalità, anche di più componenti diversi, tra loro con lo scopo di realizzare un unico task.

La necessità di un linguaggio e di un modello capaci di rappresentare le informazioni, anche in questo caso, finisce col condurre alla scelta delle

ontologie come strumento più adatto, confermando l'importanza che esse rivestono. In altri termini, se l'esecuzione di una singola funzionalità può essere vista come una chiamata a procedura, allora le ontologie forniscono una sorta di API in un linguaggio machine-readable, adoperate dagli agenti software che ne interpretano il contenuto ed acquisiscono la conoscenza necessaria ad effettuare le invocazioni in maniera corretta ed a coordinarsi fra loro.

3.3 Ontologie per le entità di un ambiente pervasivo

Un ambiente pervasivo contiene un grosso numero di entità di tipo diverso:

- *dispositivi*, che vanno dai piccolo dispositivi portatili ai grandi schermi e ai server potenti;
- *risorse*, che contribuiscono al funzionamento dell'ambiente, come risorse che si occupano di problemi di accesso o di autenticazione o che forniscono altre funzionalità, come player musicali, viewer Pdf o Power Point;
- *utenti* dell'ambiente.

Per ognuna di esse sono state sviluppare delle ontologie che le definiscono, che descrivono come esse interagiscono fra loro, le relazioni che esistono e gli assiomi sulle loro proprietà che devono essere sempre soddisfatti.

Esse, tuttavia, rappresentano delle ontologie base, dei punti di partenza, che hanno come scopo quello di descrivere, modellare solo alcune delle entità che possono caratterizzare un ambiente pervasivo.

Un ambiente pervasivo è, infatti, un ambiente estremamente dinamico, nuovi tipi di entità possono aggiungersi all'ambiente in qualsiasi momento: per questo è possibile aggiungere nuove classi e proprietà alle ontologie esistenti utilizzando dei concetti ponte che le relazionino alle classi e alle proprietà già

esistenti. Tali concetti ponte sono tipicamente relazioni di sussunzione che definiscono la nuova classe come sottoclasse di una già esistente. E' possibile, quindi, estendere le ontologie esistenti acquisendo conoscenza su nuovi aspetti della realtà che si sta modellando, e rappresentando tali aspetti in maniera formalizzata.

In aggiunta a queste ontologie, che forniscono metadati relativi a classi di individui, sono sviluppate, per le singole entità, descrizioni che ne definiscono le proprietà e che devono essere consistenti con la descrizioni delle classi di cui sono istanze.

Sia le ontologie che le varie descrizioni di istanze sono state scritte in DAML+OIL: questa scelta, che è stata motivata dalla possibilità di poter usufruire di software con interfacce aperte disponibile gratuitamente, non rappresenta, però, una limitazione poichè è possibile passare facilmente ad OWL, essendo questo un'estensione di DAML+OIL.

3.3.1 Le specifiche FIPA e le ontologie dei dispositivi

Per assegnare una semantica definita e universalmente riconosciuta alle descrizioni specifiche di dispositivi e per supportare i ragionamenti su di esse, è stato necessario realizzare un'ontologia che ne definisce le caratteristiche hardware e software.

Esistono molti esempi, realizzati da diverse organizzazioni ed università, di ontologie che descrivono dispositivi: ognuna di esse è stata sviluppata per soddisfare esigenze precise, dipendenti essenzialmente dal contesto in cui vanno ad essere inserite.

Tra queste, l'ontologia che meglio si sposa con il nuovo modo di vedere e di utilizzare i dispositivi e le loro descrizioni all'interno di ambienti di Pervasive Computing è quella proposta attraverso una specifica FIPA.

FIPA (Foundation For Intelligent Physical Agents) [26] è un'organizzazione internazionale che si occupa di promuovere l'industria degli agenti intelligenti

sviluppando specifiche aperte che consentono l'interoperabilità tra agenti e applicazioni basate su agenti. L'ontologia dei dispositivi proposta da FIPA definisce in maniera generale le caratteristiche principali di un insieme di differenti dispositivi e consente di realizzare, a partire essa, una serie di profili, associati agli specifici dispositivi, che vengono poi validati attraverso un confronto con l'ontologia stessa.

A partire da questa proposta è stata implementata una nuova ontologia dei dispositivi, che in parte adotta le specifiche FIPA, mantenendo quindi la compatibilità con un vocabolario già definito e di uso diffuso, ed in parte la estende introducendo nuovi concetti e proprietà, con lo scopo di fornire una migliore e più completa caratterizzazione dei dispositivi stessi.

Le classi di questa ontologia e le relazioni che le legano sono schematizzate nella figura seguente:

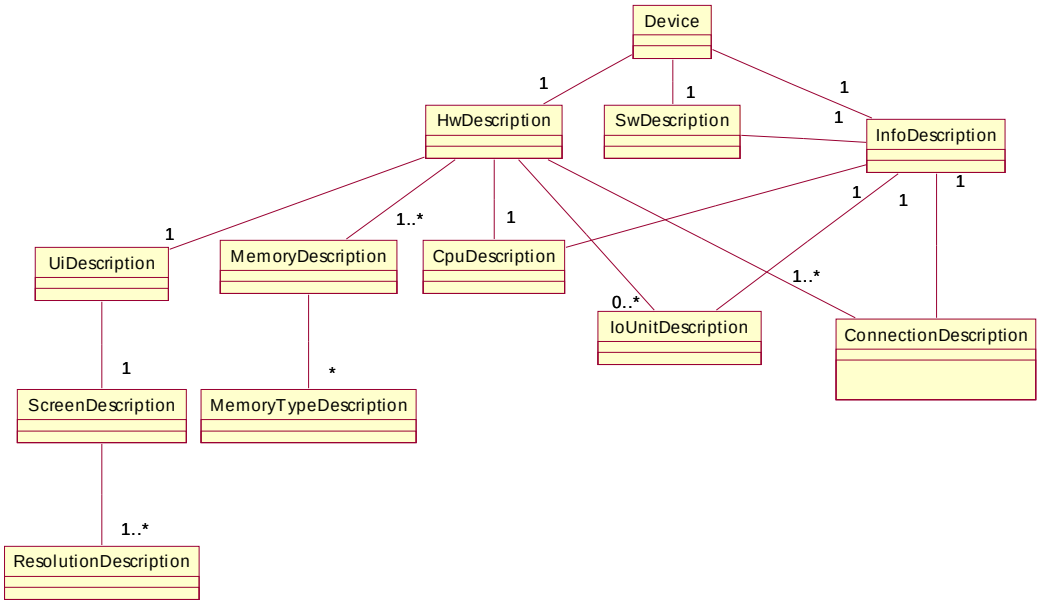

Figura 3.1:Ontologia dei device:una visione di insieme

Di seguito sono analizzate le singole classi, le proprietà di ognuna ed i valori che tali proprietà possono assumere.

La classe *Device* descrive un dispositivo ed è caratterizzata dal seguente insieme di proprietà:

- *information*, che fornisce informazioni generali su un device;
- *devicetype*, che specifica il tipo di device, ad esempio PDA, laptop, cellulare;
- *hwproperties*, che fornisce la lista di proprietà che ne descrivono le caratteristiche hardware;
- *swproperties*, che fornisce la lista di proprietà che ne descrivono le caratteristiche software;
- *offers*, che fornisce la lista di risorse che un device offre;
- *ownedBy*, che associa un device all'utente che lo possiede.

La classe *InfoDescription* descrive un determinato oggetto, che può essere un device, la sua cpu, una sua periferica di I/O, una connessione, il suo sistema operativo, un'applicazione, fornendo alcune informazioni di tipo generale:

- *infoname*, che specifica il nome dell'oggetto;
- *infovendor*, che specifica il venditore dell'oggetto;
- *infoversion*, che specifica la versione dell'oggetto.

La classe *SwDescription* fornisce informazioni che definiscono alcune caratteristiche software di un device:

- *os*, che associa ad un device le informazioni relative al suo sistema operativo;
- *applcation*, che associa ad un device le informazioni relative alle applicazioni che esso possiede.

La classe *HwDescription* descrive un device a partire dalle sue caratteristiche hardware, attraverso queste proprietà:

- *connection*, che fornisce la lista di connessioni che un device possiede;

- *cpu*, che fornisce la descrizione della cpu di un device;
- *memory*, che associa ad un device la descrizione della sua memoria massima teorica e di quella invece disponibile;
- *ui*, che fornisce l'interfaccia utente che un device offre;
- *io*, che associa ad un device la lista di periferiche di I/O che esso possiede.

La classe *IODescription* fornisce informazioni generali relative a periferiche di I/O del device, attraverso la proprietà *"information"*. A questa classe è possibile aggiungere nuove proprietà, o attraverso relazioni di sussunzione, agganciare nuove classi per fornire una caratterizzazione più dettagliata.

La classe *ConnectionDescription* fornisce i dettagli relativi alle connessioni del device, ed in particolare:

- *information*, che fornisce informazioni generali relative alla connessione;
- *bandwidth*, che specifica la larghezza di banda associata alla connessione;
- *bandwidthunit*, che specifica l'unità di misura usata per esprimere la bandwidth (Kbit/s, Mbit/s, Gbit/s).

La classe *CpuDescription* fornisce i dettagli relativi alla cpu del device, ossia:

- *information*, che fornisce informazioni generali relative alla cpu;
- *cpuspeed*, che specifica la velocità della cpu;
- *cpuspeedunit*, che specifica l'unità di misura usata per esprimere la cpuspeed (MHz,GHz).

Le classi *MemoryDescription* e *MemoryTypeDescription* descrivono la quantità di memoria di un device, ed in particolare la massima memoria del device e quella invece disponibile, l'unità di misura in cui è espressa e il tipo di utilizzo.

In particolare, la classe *MemoryDescription* fornisce la descrizione della memoria massima e di quella disponibile di un device, adoperando le proprietà:

- *available*, che fornisce una descrizione della quantità di memoria disponibile;
- *maximum*, che fornisce una descrizione della massima quantità di memoria.

La classe *MemoryTypeDescription*, invece, descrive la memoria di un device fornendo le seguenti informazioni:

- *memoryamount*, che specifica la quantità di memoria di un device;
- *memoryunit*, che specifica l'unità di misura usata per esprimere la memoryamount (Kb, Mb, Gb);
- *memoryusagetype*, che specifica il tipo di memoria, ossia se è Storage o Application (memoria su disco fisso o memoria Ram).

La classe *UiDescription* descrive l'interfaccia utente che un device offre, attraverso le seguenti proprietà:

- *screen*, che associa ad un device la descrizione delle caratteristiche del suo schermo;
- *audio-input*, che specifica se un device è capace di ricevere audio in input (ossia se è dotato, ad esempio, di microfono);
- *audio-output*, che specifica se un device è capace di fornire audio in output (ossia se è dotato, ad esempio, di casse);
- *video-input*, che specifica se un device è capace di ricevere video in input (ossia se è dotato, ad esempio, di una webcam).

La classe *ScreenDescription* fornisce le caratteristiche di uno schermo, ossia:

- *resolution*, che associa ad uno schermo la lista delle sue possibili risoluzioni;

- *width*, che specifica la larghezza dello schermo;
- *height*, che specifica l'altezza dello schermo;
- *unit*, che specifica l'unità di misura in cui sono espresse height e width (cm, mm, pollici);
- *color*, che specifica se lo schermo è a colori o meno.

La classe *ResolutionDescription* fornisce i dettagli relativi alla risoluzione di uno schermo, ossia:

- *resolutionwidth*, che specifica la risoluzione orizzontale;
- *resolutionheight*, che specifica la risoluzione verticale;
- *resolutionunit*, che specifica l'unità di misura in cui sono espresse resolutionwidth e resolutionheight (pixel o caratteri);
- *resolutionbpp*, che specifica il numero di bit per pixel;
- *resolutiongraphics*, che specifica se è possibile visualizzare solo testo o anche grafica.

Di seguito è mostrato un esempio, in formato schematico, di descrizione di un laptop attraverso le proprietà e le classi dell'ontologia appena definita:

Descrizione del Device UbiLaptop				
Info Description	**Infoname**			**Massimo**
	Infovendor			**Toshiba**
	Infoversion			**Satellite 1400-103**
Devicetype				**Laptop**
Offers				**PdfViewer**
OwnedBy				**Massimo Esposito**
Hw Description	**Connection Description**	**InfoDescription**	**Infoname**	**Modem 56K**
		Bandwidth		**56**
		Bandwidthunit		**Kbit/s**
	Connection Description	**InfoDescription**	**Infoname**	**Lan Ethernet**
		Bandwidth		**100**

			Bandwidthunit		**Mbit/s**
	Cpu Description		**InfoDescription**	**Infoname**	**Intel Celeron**
			Cpuspeed		**1.33**
			Cpuspeedunit		**GHz**
	Ui Description	**Screen Description**	**Width**		**28.25**
			Height		**22**
			Unit		**Cm**
			Color		**True**
			Resolution Description	**Resolutionwidth**	**1024**
				Resolutionheight	**768**
				Resolutionunit	**Pixel**
				Resolutionbpp	**32**
				Resolutiongraphics	**True**
		Audio-input			**False**
		Audio-output			**True**
		Video-input			**False**
	Memory Description	**Maximum**	**MemoryType Description**	**Memoryamount**	**20**
				Memoryunit	**Gb**
				Memoryusagetype	**Storage**
		Available	**MemoryType Description**	**Memoryamount**	**6**
				Memoryunit	**Gb**
				Memoryusagetype	**Storage**
	Memory Description	**Maximum**	**MemoryType Description**	**Memoryamount**	**128**
				Memoryunit	**Mb**
				Memoryusagetype	**Application**
Sw Description	**Os Description**	**InfoDescription**		**Infoname**	**Windows Xp**
				Infovendor	**Microsoft**
				Infoversion	**Home**
	Application Description	**InfoDescription**		**Infoname**	**Windows Media Player**
				Infovendor	**Microsoft**
				Infoversion	**8.0**

Figura 3.2:Esempio di descrizione di un laptop adoperando l'ontologia dei dispositivi

3.3.2 DAML-S e le ontologie delle risorse e dei servizi

L' ontologia delle risorse e dei servizi proposta di seguito, insieme con le altre che sono state realizzate, sono state progettate per essere adoperate all'interno di un prototipo di sistema pervasivo, proposto più avanti nella trattazione. Questo sistema utilizza il Web, ed in particolare i Web Services, come infrastruttura di comunicazione tra i componenti distribuiti dell'ambiente, e quindi tutti i servizi che esso fornisce sono stati implementati come risorse Web. Per far uso di tali risorse, ed in particolare delle singole funzionalità da esse offerte, occorre fornire delle descrizioni contenenti un numero di informazioni sufficiente ad abilitarne il discovery, l'invocazione e la composizione in maniera automatica.

Anche se la maggior parte dei Web Services è descritta attraverso un documento WSDL, e seppur esso rappresenti un ottimo strumento per utilizzare un servizio, se non si ha una chiara idea di ciò che si cerca non è possibile a partire dalla sola WSDL ottenere un'estesa e strutturata descrizione riguardante ciò che fa il servizio. UDDI cerca di colmare questo gap, ma le descrizioni da esso fornite sono maggiormente indirizzate agli sviluppatori di servizi e non agli utenti finali. Una soluzione a questo problema è fornita da DAML-S [27].

DAML-S è un'ontologia, espressa in DAML+OIL, che supporta un insieme di classi e proprietà base per la definizione e descrizione di servizi e procedure sul Web, attraverso un linguaggio di markup non ambiguo e machine-readable.

Questa ontologia intende modellare tre tipi diversi di informazioni relative ai servizi, ed in particolare attraverso il *Service Profile* fornisce informazioni relative a ciò che il servizio fa, attraverso il *Service Model* informazioni relative a come lavora il servizio ed infine attraverso il *Service Grounding* informazioni relative al modo attraverso il quale è possibile accedere al servizio. Ogni istanza di un servizio può essere vista come la dichiarazione di un API, come un entry point associato al sevizio che un provider vuole rendere accessibile: essa può presentare zero o più profili e può essere descritta da

nessuno o al più da un modello. Se esiste un modello allora esso può supportare 1 o più grounding.

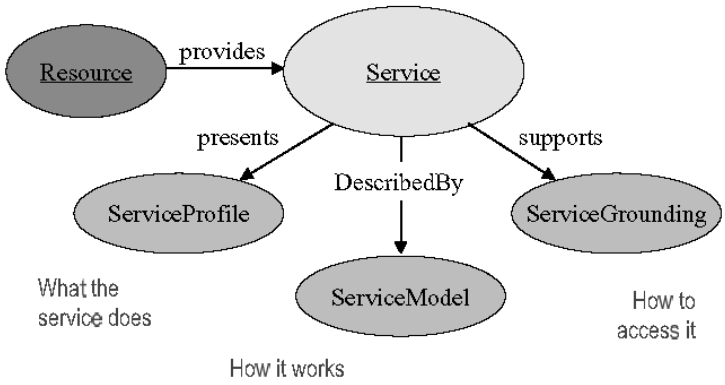

Figura 3.3:Visione d'alto livello dell'ontologia fornita da DAML-S

Attraverso il Service Profile vengono fornite le informazioni necessarie ad un'entità per effettuare discovery di servizi, mentre presi insieme, il Service Model e il Service Grounding forniscono le informazioni necessarie per consentire l'uso dei servizi stessi, ed in particolare il primo consente di comporre più descrizioni di servizi per effettuare un unico task, di monitorare l'esecuzione di un servizio, di coordinare le attività di diversi partecipanti durante la fruizione del servizio, di fornire una più profonda analisi dei requisiti del servizio, mentre il secondo specifica il protocollo di comunicazione, il formato dei messaggi, ed altri dettagli specifici del servizio come il numero della porta usata per contattare il servizio stesso.

Sebbene DAML-S fornisca una descrizione completa dei servizi Web, esso non intende sostituirsi alla WSDL, che rimane lo strumento fondamentale per la descrizione di servizi Web, ma vi si lega attraverso le specifiche definite nel grounding sfruttandone quindi le informazioni contenute: in pratica la WSDL fornisce informazioni su come invocare il servizio, mentre attraverso DAML-S si ottengono le informazioni riguardanti ciò che il servizio fa, il perché si vuole utilizzarlo e cosa ci si aspetta dal suo utilizzo.

Anche in questo caso, così come è stato fatto per l'ontologia dei dispositivi, per realizzare un'ontologia delle risorse che ben si adatti ai requisiti di un ambiente pervasivo ma che mantenga una certa compatibilità con progetti già realizzati e di uso diffuso, è stata effettuata la scelta di utilizzare alcuni dei concetti espressi in DAML-S, ed in particolare è stato ripreso il concetto di Service Profile, di estenderne certi aspetti e di adattarne degli altri in accordo con nuove esigenze identificate.

Nella figura seguente sono schematizzate le classi di questa ontologia e le relazioni che le legano:

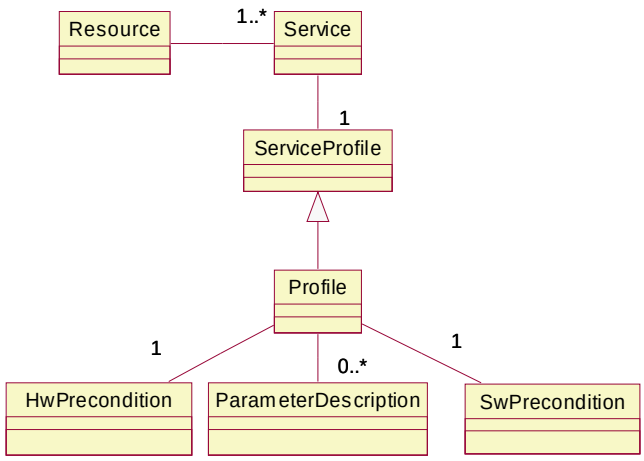

Figura 3.4:Ontologia delle risorse:una visione d'insieme

Di seguito sono analizzate le singole classi dell'ontologia, le proprietà di ognuna ed i valori che tali proprietà possono assumere.

La classe *Resource* descrive una risorsa ed è caratterizzata dal seguente insieme di proprietà:

- *provides*, che associa ad ogni risorsa una serie di servizi, dove per servizi, si intendono le funzionalità atomiche, le procedure fornite dalla risorsa in questione;
- *resourceName*, che associa un nome alla risorsa;

- *resourceUrl*, che fornisce l'url associata alla risorsa;
- *resourcePageUrl*, che fornisce l'url di una pagina Web attraverso la quale è possibile utilizzare la risorsa ed i sui servizi;
- *resourceTextDescription*, che fornisce una descrizione testuale della risorsa;
- *offeredBy*, che associa una risorsa al device che la offre.

Attraverso le classi *Service*, *ServiceProfile* e *Profile* si associa ad ogni singolo servizio un profilo. Esistono vari modi di caratterizzare un servizio con un profilo e quella di seguito riportata è solo una possibile soluzione ed è stata modellata attraverso la classe Profile: per questo tale classe è vista come sottoclasse della più generica ServiceProfile. La classe Profile mostra quali sono le funzionalità del servizio, ciò che il servizio fa: essa fornisce ad un entità che ricerca un servizio il tipo di informazioni necessarie a comprendere se esso risponde ai suoi bisogni:

- *serviceName*, che associa un nome ad un servizio;
- *textDescription*, che associa al servizio una descrizione testuale;
- *parameter*, che fornisce i parametri del servizio;
- *input*, che fornisce i parametri di input del servizio: è una sottoproprietà di parameter;
- *output*, che fornisce i parametri di output del servizio: è una sottoproprietà di parameter;
- *hwPrecondition*, che fornisce le precondizioni hardware che occorre possedere per poter utilizzare il sevizio in questione;
- *swPrecondition*, che fornisce le precondizioni hardware che occorre possedere per poter utilizzare il sevizio in questione.

La classe *ParameterDescription* caratterizza un parametro, sia esso di input o di output, associandogli le seguenti informazioni:

- *parameterName*, che specifica il nome del parametro;
- *restrictedTo*, che specifica il range di valori che il parametro può assumere (ad esempio stringhe, interi, reali).

La classe *HwPrecondition* descrive i prerequisiti hardware associati al servizio e necessari per la sua corretta fruizione attraverso le seguenti proprietà :

- *connection*, che fornisce la larghezza di banda minima richiesta dal servizio;
- *cpu*, che fornisce la velocità minima di cpu richiesta;
- *audio-input*, che richiede la capacità del device di ricevere audio in input (ossia se il device è dotato, ad esempio, di microfono);
- *audio-output*, che richiede la capacità del device di fornire audio in output (ossia se il device è dotato, ad esempio, di casse);
- *resolution*, che fornisce la minima risoluzione richiesta;
- *memory*, che fornisce la minima quantità di memoria richiesta, sia Ram che su disco fisso;
- *video-input*, che richiede la capacità del device di ricevere video in input (ossia se il device è dotato, ad esempio, di una webcam)

La classe *SwPrecondition* descrive i prerequisiti software associati al servizio, ossia il tipo di sistema operativo che il device che lo usa deve possedere e applicativi necessari all'esecuzione del servizio stesso.

E' fornita, di seguito, la descrizione di una risorsa Web che ha le funzionalità di un viewer di file Pdf.

Descrizione della risorsa Web UbiPdfViewer	
ResourceName	**UbiPdfViewer**
ResourceTextDescription	**UbiPdfViewer è una risorsa Web capace di eseguire file Pdf.**
ResourcePageUrl	**http://ubinet.it/UbiPdfViewer/Page**
ResourceUrl	**http://ubinet.it/UbiPdfViewer**

Executes				**PdfFile**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_View**
		TextDescription		**Questo servizio consente di aprire fe visualizzare file Pdf.**
		Input	**ParameterName**	**Filename**
			Restricted To	**String**
		Opens		**PdfFile**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Close**
		TextDescription		**Questo servizio consente di chiudere file Pdf aperti precedentemente.**
		Closes		**PdfFile**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Nextpage**
		TextDescription		**Questo servizio consente di passare alla pagina successiva all'interno di un documento Pdf.**
		Goesto		**NextPage**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Prevpage**
		TextDescription		**Questo servizio consente di passare alla pagina precedente all'interno di un documento Pdf.**
		Goesto		**PrevPage**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Pageup**
		TextDescription		**Questo servizio consente di spostarsi di una schermata verso l'alto all'interno di un pagina di un documento Pdf.**
		Goesto		**Pageup**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Print**
		TextDescription		**Questo servizio consente di stampare file Pdf.**
		Prints		**PdfFile**
Service	**Profile**	**ServiceName**		**UbiPdfViewer_Pagedown**
		TextDescription		**Questo servizio consente di spostarsi di una schermata verso il basso all'interno di un pagina di un documento Pdf.**
		Goesto		**Pagedown**

Figura 3.5:Esempio di descrizione di un PdfViewer

Come è possibile vedere dalla schematizzazione fornita, sono state associate alla risorsa UbiPdfViewer e ad ogni suo servizio descrizioni testuali che ne definiscono le caratteristiche, assegnandovi così una precisa semantica comprensibile alle persone.

La risorsa UbiPdfViewer è stata definita come un oggetto in grado di eseguire file Pdf, mentre i suoi servizi hanno le funzionalità di aprire, chiudere, stampare file Pdf e muoversi tra pagine o al loro interno. L'utilizzo di tali descrizioni consente agli utenti di effettuare discovery semantico delle risorse dell'ambiente, consentendogli di scegliere quella che meglio si adatta alle proprie esigenze. Tuttavia, questa operazione non è effettuata in maniera automatica ma deve essere l'utente, leggendo le descrizioni, a comprendere di quale risorsa ha bisogno.

Ad effettuare discovery nell'ambiente, non sono, però, solo gli utenti ma anche le risorse possono aver bisogno, per completare un proprio task, di utilizzare altre risorse. Per fare questo, esse debbono scoprire dinamicamente quali sono le risorse tra quelle disponibili con cui debbono cooperare, e quindi utilizzarle.

Ad esempio, la risorsa UbiPdfViewer, per stampare i file pdf, potrebbe aver bisogno di utilizzare un servizio di stampa messo a disposizione dall'ambiente. Tuttavia, per poter capire quali tra le risorse disponibili sono capaci di stampare file pdf, non è possibile utilizzare una descrizione testuale poiché rappresenta un formalismo non comprensibile ad una risorsa software.

Per consentire, quindi, anche agli agenti software di poter comprendere senza ambiguità il significato delle risorse e dei servizi occorre fornire descrizioni in formato machine-understandable per ognuno di essi, aggiungendo all'ontologia proposta una serie di nuove classi e proprietà. Questi nuovi concetti, però, non possono essere completamente generali ma risultano fortemente legati al particolare tipo di risorsa e servizio che debbono descrivere e per questo occorre estendere l'ontologia esistente ogni qualvolta si vada ad aggiungere ad essa una nuova risorsa o un nuovo servizio di cui non se ne è ancora definito il significato.

Nell'esempio proposto, per descrivere la risorsa UbiPdfViewer è stato necessario introdurre i seguenti nuovi concetti e termini:

- *PdfViewer*, definita come la classe che comprende tutte le risorse in grado di eseguire file pdf;
- *File*, definita come la classe che comprende tutti i tipi di file possibili;
- *Functionality*, definita come la classe che comprende tutti i tipi di funzionalità che è possibile associare ad un servizio;
- *PdfFile*, definito come un oggetto di tipo File con estensione pdf;
- *NextPage*, *PrevPage*, *Pageup*, *Pagedown*, definiti come oggetti di tipo Functionality.
- Il termine *executes* che relaziona i concetti PdfViewer e PdfFile fra loro ed il termine *extension* che relaziona il concetto di PdfFile con la sua estensione;
- Il termine *goesto* che relaziona i singoli servizi con i concetti Nextpage, Prevpage, Pagedown e Pageup;
- I termini *opens*, *closes e printes* che relazionano i singoli servizi con il concetto PdfFile.

Di seguito è proposta anche la descrizione di una risorsa Web che offre un servizio di stampa di file pdf.

Descrizione della risorsa Web UbiPrintService				
ResourceName				**UbiPrintService**
ResourceTextDescription				**UbiPrintService è una risorsa Web capace di stampare file Pdf.**
ResourcePageUrl				**http://ubinet.it/PrintService/Page**
ResourceUrl				**http://ubinet.it/PrintService**
Prints				**PdfFile**
Service	**Profile**	**ServiceName**		**UbiPrintService_Pdf**
		TextDescription		**Questo servizio consente di stampare file Pdf.**
		Input	**ParameterName**	**Filename**

			RestrictedTO	**String**
		Prints		**PdfFile**

Figura 3.6:Esempio di descrizione di un servizio di stampa

Quando viene lanciata la stampa di un documento pdf, la risorsa UbiPdfViewer cerca in maniera dinamica nell'ambiente tutte le risorse che " prints PdfFile", individua la risorsa UbiPrintService, e va ad utilizzare, tra tutti i servizi che essa offre, quello che in particolare stampa file pdf.

Per fruire di un servizio, tutte le entità dell'ambiente utilizzano il Profile ad esso associato. A differenza di DAML-S, infatti, il Profile definito in questa ontologia consente non solo di comprendere ciò che un servizio fa, ma fornisce anche le informazioni sufficienti affinché il servizio stesso possa essere utilizzato. Ad ogni servizio è, infatti, stato associato il nome, i parametri di input e output, e per ognuno di questi è fornito il nome ed il tipo. Adoperando queste informazioni e l'url della risorsa Web che lo offre è possibile, quindi, invocare in maniera dinamica un servizio.

Affinché un servizio possa essere utilizzato, occorre, però, che i prerequisiti hardware e software ad esso associati, definiti col le classi *HwPrecondition* e *SwPrecondition*, siano verificati. Questa verifica viene effettuata andando a confrontare le caratteristiche, definite con l'ontologia dei dispositivi, del device attraverso il quale un utente o una risorsa software invoca un servizio, con questi prerequisiti. Se non c'è match tra essi, allora viene inibita la fruizione del servizio a chi cercava di farne uso.

E' fornita, di seguito, la descrizione di una risorsa Web che fornisce un servizio di videoconferenza, in cui un prerequisito, quello relativo alla quantità di memoria richiesta, non è verificato dalla descrizione del laptop proposta in precedenza.

Descrizione della risorsa Web UbiVideoConferencingService				
ResourceName				**UbiVideoConferencingService**
HwPrecondition	**Connection Description**		**Bandwidth**	**64**
			Bandwidthunit	**Kbit/s**
	Resolution Description		**Resolutionwidth**	**800**
			Resolutionheight	**600**
			Resolutionunit	**Pixel**
			Resolutionbpp	**32**
			Resolutiongraphics	**True**
	Audio-output			**True**
	MemoryType Description		**Memoryamount**	**196**
			Memoryunit	**Mb**
			Memoryusagetype	**Application**
ResourceTextDescription				**UbiVideoConferencingService è una risorsa Web che trasmette immagini in tempo reale.**
ResourcePageUrl				**http://ubinet.it/ UbiVideoConferencingService/Page**
ResourceUrl				**http://ubinet.it/ UbiVideoConferencingService**
Service	**Profile**	**ServiceName**		**UbiVideoConferencing_Leave**
		TextDescription		**Questo servizio consente di abbandonare una videoconferenza.**
Service	**Profile**	**ServiceName**		**UbiVideoConferencing_Join**
		TextDescription		**Questo servizio consente di partecipare a una videoconferenza.**
		Input	**ParameterName**	**Url**
			RestrictedTO	**String**

Figura 3.7:Esempio di descrizione di un servizio di videoconferenza

3.3.3 Ontologia degli utenti

Per assegnare una semantica definita e universalmente riconosciuta anche alle descrizioni specifiche di utenti e per supportare i ragionamenti su di esse, è stata realizzata ex novo, non esistendo standard o proposte di uso diffuso, un'ontologia, seppur semplice, che ne definisce le caratteristiche principali:

- *userName*, che definisce il nome di un utente;

- *userSurname*, che definisce il cognome di un utente;
- *userEmail*, che definisce l'indirizzo di posta elettronica di un utente

E' mostrato, qui di seguito un esempio di descrizione di un utente attraverso l'ontologia appena definita:

Descrizione dell'Utente Jim Huck	
userName	**Jim**
userSurname	**Huck**
userEmail	**jimhuck@ubinet.it**

Figura 3.8:Semplice esempio di descrizione di un utente adoperando l'ontologia degli utenti

Nella figura seguente è mostrata la schematizzazione, in una visione d'insieme, delle tre ontologie base realizzate e delle relazioni che esistono fra esse.

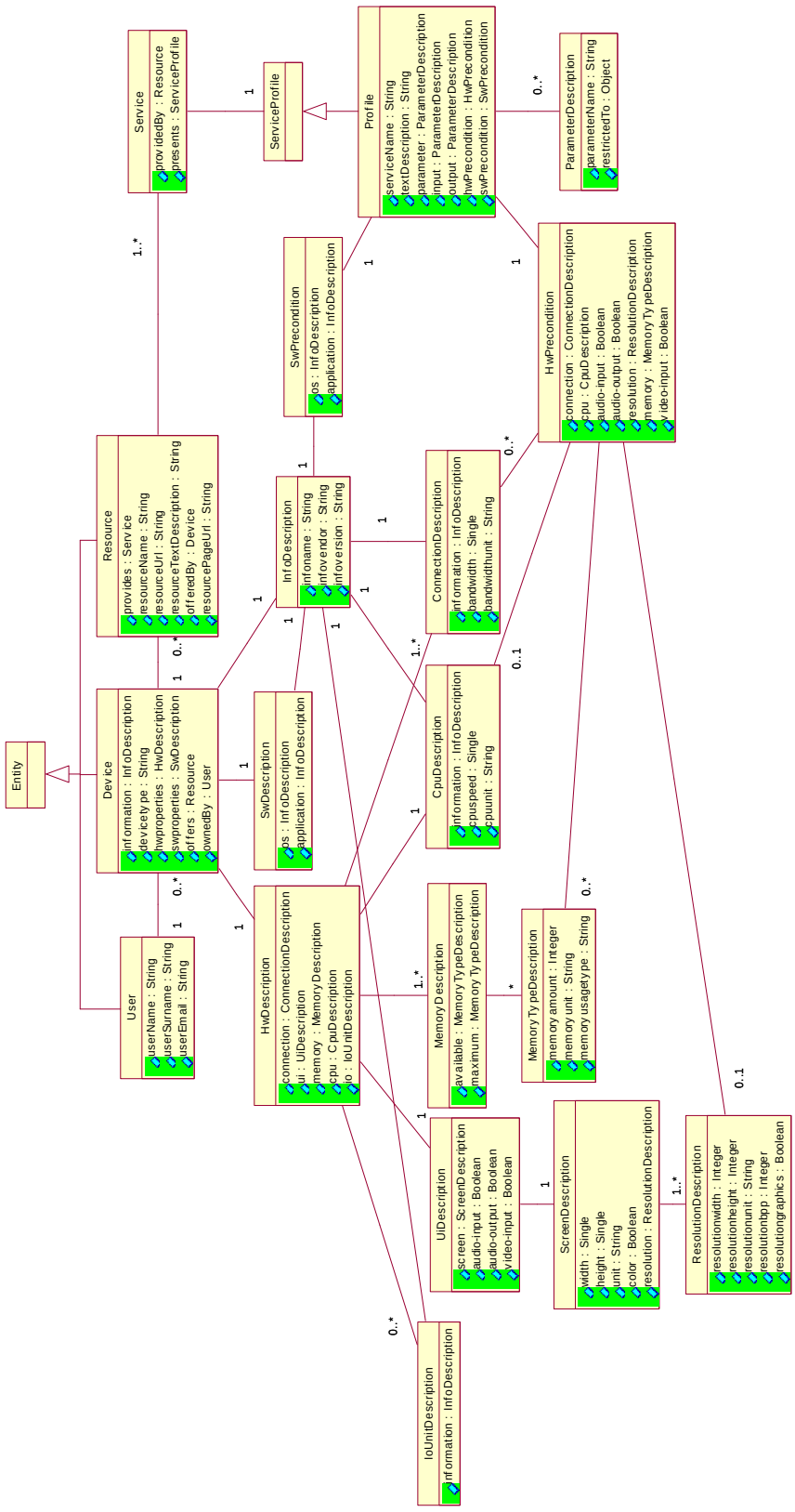

Figura 3.9:Ontologia delle entità dell'ambiente

3.4 Ontologie per le informazioni di contesto

Il contesto e la consapevolezza del contesto hanno assunto un ruolo di fondamentale importanza negli ambienti pervasivi.

Per *"contesto"* si intende qualsiasi informazione che può essere usata per caratterizzare il comportamento di una persona o di entità software [28]. Le persone agiscono in maniera diversa in relazione al contesto: sono capaci di percepire in che tipo di contesto si trovano ed adattano il loro modo di comportarsi in relazione al contesto corrente. Similmente anche le applicazioni di un ambiente pervasivo devono essere *"context-aware"*, ossia consapevoli del contesto e capaci di adattarsi a situazioni che cambiano rapidamente.

Ci possono essere diversi tipi di contesto che possono essere usati dalle applicazioni [1]:

- contesti fisici (posizione e tempo)
- contesti ambientali (aspetti meteorologici, livelli di luce e sonoro)
- contesti informazionali (quote azionarie, punteggi sportivi)
- contesti applicativi (email, siti web visitati)
- contesti di sistema (traffico della rete, stato delle stampanti).
- contesti personali (salute, attività, stati d'animo)
- contesti sociali (attività di gruppo, relazioni sociali, con chi si è in una stanza)

Per essere context-aware le applicazioni necessitano di un'infrastruttura composta di sensori che percepiscono le informazioni di contesto e di reasoner che inferiscono nuove informazioni di contesto a partire dai dati percepiti. Occorre, inoltre, che le varie entità che utilizzano le informazioni di contesto riescano a comprendere gli aspetti dell'ambiente per loro rilevanti, riescano a capire cioè il vocabolario di contesto.

Tale vocabolario deve essere ben definito semanticamente in modo che le differenti entità abbiano una conoscenza comune del contesto e non vi siano ambiguità. Anche in questo caso, per esprimere questo vocabolario, possono essere utilizzate le ontologie ed in particolare è stata proposta e sviluppata una nuova ontologia che definisce informazioni fisiche di contesto, andando a descrivere ambienti con confini geografici ben identificabili come edifici e stanze.

Non esistendo standard o modelli di uso diffuso che caratterizzano un tale ambiente, ma tutta una serie di proposte realizzate in diversi progetti, è stata effettuata la scelta di utilizzare alcuni dei concetti espressi nell'ontologia di contesto sviluppata dal progetto CoBrA, estendendone certi aspetti e ridefinendone degli altri.

Di seguito sono analizzate le singole classi dell'ontologia sviluppata, le proprietà di ognuna ed i valori che tali proprietà possono assumere.

La classe *Place* rappresenta la classe di tutti i possibili ambienti rappresentati nell'ontologia. Ad essa sono associate le seguenti proprietà:

- *placeName*, che fornisce il nome dell'ambiente.
- *contains*, che associa ad ogni ambiente tutti ciò esso contiene.
- *isContainedIn*, che associa ad un ambiente tutti quelli da cui è contenuto.

La classe *CompoundPlace* descrive gli ambienti composti, ossia ambienti ne che contengono altri e che possono essere a loro volta contenuti .

La classe *AtomicPlace* descrive gli ambienti atomici, e cioè ambienti che non contengono altri ambienti ma che sono contenuti in ambienti composti.

La classe *OpenPlace* descrive gli ambienti aperti, e cioè ambienti che non includono né sono inclusi spazialmente in altri ambiente.

A partire da queste tre classi piuttosto generali ne sono state definite delle altre, che derivano da esse e che caratterizzano in maniera più particolare ambienti reali.

La classe *Building* modella il concetto di edificio, definendolo come un ambiente composto caratterizzato dalle seguenti proprietà:

- *buildingAddress*, che fornisce l'indirizzo dell'edificio.
- *numberOfFloors*, che fornisce il numero di piani di cui è composto l'edificio.
- *contains*, che associa un edificio alle stanze che esso contiene.

La classe *Room* definisce una stanza come un ambiente atomico contenuto in un edificio e la descrive attraverso le seguenti proprietà:

- *floorNumber*, che fornisce il piano dell'edificio in cui si trova la stanza.
- *isContainedIn*, che associa la stanza all'edificio in cui si trova..
- *availableResource*, che fornisce tutte le risorse Web disponibili all'interno della stanza.
- *placedDevice*, che fornisce i dispositivi che si trovano all'interno della stanza.
- *locatedUser*, che fornisce gli utenti presenti all'interno della stanza.

Per consentire alle varie entità di conoscere la loro posizione all'interno degli ambienti appena descritti, occorre aggiungere alle classi *User*, *Device* e *Resource*, rispettivamente, le seguenti proprietà:

- *isLocatedIn*, che associa l'utente alla stanza in cui si trova. Un utente può trovarsi fisicamente in uno stesso intervallo di tempo in al più una stanza, o *AtomicPlace* in genere.
- *isPlacedIn*, che associa un dispositivo alla stanza in cui è posizionato. Anche un dispositivo può trovarsi fisicamente in uno stesso intervallo di tempo in al più una stanza, o *AtomicPlace* in genere.

- *isAvailableIn*, che associa una risorsa alla stanza in cui è possibile adoperarla. Una risorsa, a differenza di utenti e dispositivi, può essere disponibile in uno stesso intervallo di tempo anche in più una stanza, o *AtomicPlace* in genere.

Adoperando la semantica definita nelle tre classi CompoundPlace, AtomicPlace e OpenPlace è possibile accorgersi di incongruenze riguardo la posizione di utenti e dispositivi, dovute magari a malfunzionamenti dei sensori negli ambienti. Infatti, la presenza contemporanea in due posti, ad esempio, di un utente è semanticamente corretta se i due posti sono una stanza ed un edificio ed è giustificato dal fatto che un edificio può contenere una stanza, mentre rappresenta un'inconsistenza se i posti in questione sono due stanze poiché esse sono AtomicPlace e non possono contentersi fra loro.

Nella figura seguente sono schematizzate le classi di questa ontologia e le relazioni che le legano.

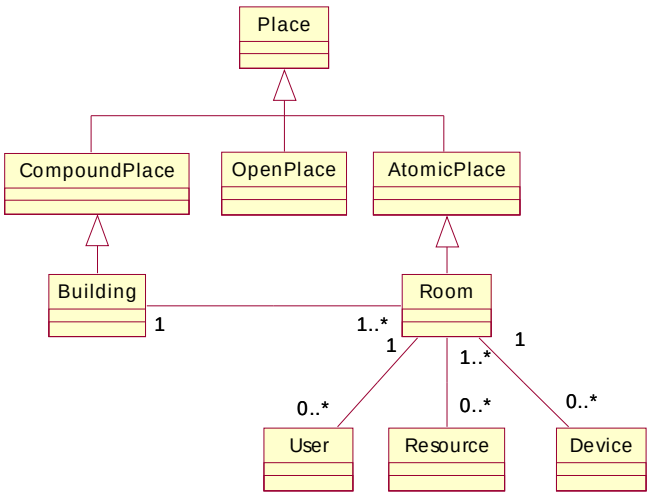

Figura 3.10:Ontologia di contesto

E' mostrato, qui di seguito un esempio di descrizione di un edificio, di una stanza e di ciò che essa contiene, attraverso l'ontologia appena definita:

Descrizione di un Building e delle Rooms contenute		
PlaceName		**UbiNet**
BuildingAddress		**Via Mille n. 24**
NumberOfFloors		**1**
Room	**PlaceName**	**Room1**
	FloorNumber	**1**
	availableResource	**UbiPrintService**
		UbiVideoConferencingService
	placedDevice	**Pc1**
		Pc2
	locatedUser	**Fabio**
		Massimo
		Guido
Room	**PlaceName**	**Room2**
	FloorNumber	**1**
	availableResource	**UbiPdfViewer**
		UbiPrintService
	placedDevice	**Pc3**
		UbiLaptop
	locatedUser	**Jim**
		Huck
		Pippo

Figura 3.11:Esempio di descrizione di un Building e delle sue Rooms adoperando l'ontologia di contesto

Attraverso le informazioni di contesto contenute in questa descrizione, è possibile mostrare, ad esempio a Jim, non tutte le risorse che offre l'edificio ma solo le risorse a cui può accedere trovandosi nella Room2, condizionando quindi il suo comportamento e le sue possibili scelte in base al contesto in cui è in quel momento inserito.

Capitolo 4

Implementazione di un Ontology Service

4.1 UbiSystem, un prototipo di infrastruttura pervasiva

Tutte le ontologie sviluppate, e che sono state presentate precedentemente, sono state integrate all'interno di un prototipo di infrastruttura pervasiva, UbiSystem, che ha rappresentato, di fatto, il test-bed per l'utilizzo di ontologie all'interno di sistemi di Pervasive Computing.

UbiSystem è, in pratica, un'infrastruttura Web-based che mira a gestire e rendere possibile la comunicazione tra entità di un ambiente pervasivo, cercando di garantire tutta una serie di requisiti, quali:

- Supporto per l'eterogeneità di dispositivi, piattaforme e linguaggi;
- Definizione e condivisione dello stato del contesto dell'ambiente;
- Discovery semantico dei servizi e supporto per l'integrazione;
- Gestione della dinamicità e della disponibilità delle risorse presenti.

Questa infrastruttura è costituita da una piattaforma di servizi di sistema che cerca di facilitare lo sviluppo e l'interazione con applicazioni e servizi d'utente, adoperando la tecnologia dei Web Services come middleware abilitante. UbiSystem si preoccupa, fondamentalmente, di consentire ad un utente, che

accede al sistema mediante un qualsiasi dispositivo portatile (palmari, cellulari, laptop), di poter usufruire, in seguito ad una fase di autenticazione, di una serie di servizi e risorse disponibili nell'ambiente senza dover procedere all'installazione di software aggiuntivo né di dover eseguire procedure di configurazione di alcuna sorta, ma adoperando un comune internet browser.

Dal punto di vista delle risorse e dei servizi, UbiSystem consente ad un provider di poter esporre i propri servizi facilmente a patto che essi siano dei Web Services. Tutti i provider che utilizzano tecnologie di comunicazione diverse, possono, comunque, offrire i propri servizi, creando appositi wrappers che li espongano come Web Services.

Il sistema, adoperando le ontologie descritte, riesce ad avere una conoscenza precisa dell'ambiente che intende gestire e di tutte e le entità da esso contenute. Tutte le entità che interagiscono col sistema, siano essi dispositivi, utenti, risorse e servizi, per essere riconosciuti e classificati debbono fornire una propria descrizione, in linguaggio DAML+OIL, che viene analizzata e confrontata con le ontologie esistenti. Solo se essa è validata allora un'entità entra a far parte dell'ambiente.

Dal punto di vista architetturale, è possibile individuare un insieme di componenti modulari che provvedono a fornire al sistema le funzionalità appena descritte:

- *l'Ubiquitous Gateway*, che rappresenta l'entry point per gli utenti del sistema pervasivo, attraverso il quale essi possono accedere alle risorse e ai servizi che offre l'ambiente;
- *il System Service Manager*, che si preoccupa di gestire l'interazione dell'utente e dei servizi con i servizi di supporto del sistema;
- *l'Utilities Manager*, che gestisce la registrazione e la cancellazione di tutti i servizi d'utente all'interno dell'ambiente, mantenendo costantemente aggiornata una ServiceDirectory, e fornisce meccanismi di discovery dei servizi stessi;

- *il Context Service,* che recepisce informazioni dall'ambiente e costruisce un modello di contesto che poi condivide con utenti e fornitori di servizi, utilizzando una ContextDirectory;
- *l'Ontology Service*, che gestisce le ontologie dell'ambiente;
- *il DHCP Service,* che gestisce dinamicamente gli indirizzi di rete di tutti i dispositivi, quando essi entrano nell'ambiente pervasivo, ma anche quando lo abbandonano, seguendone ogni spostamento;
- *l'Authentication Service*, che si preoccupa di problemi legati all'autenticazione degli utenti che accedono all'ambiente;
- tutta una serie di servizi che sono messi a disposizione degli utenti e che offrono diverse funzionalità.

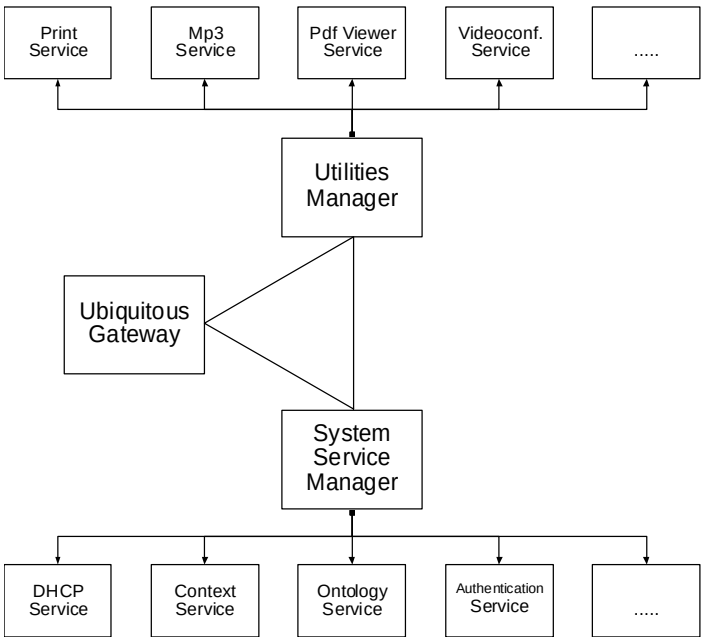

Figura 4.1:Modello architetturale di UbiSystem

Di seguito è proposta una descrizione funzionale e architetturale dettagliata del componente Ontology Service ed una sua possibile implementazione.

4.2 L'Ontology Service

L'Ontology Service è un Web Service, realizzato in linguaggio Java, che fornisce un'interfaccia generica per gestire ontologie espresse in DAML+OIL. In particolare esso gestisce un'unica ontologia, dinamicamente aggiornata e costantemente in crescita, che descrive un ambiente pervasivo, offre funzionalità per caricare e validare ontologie, a partire da file DAML+OIL, per comporre ontologie in un'unica ontologia di sistema e per effettuare alcune semplici query, il tutto utilizzando una Knowledge Base. Tra i vari motori, che implementano meccanismi logici e di ragionamento, fornendo una KB, è stato scelto FaCT (Fast Classification of Terminologies) [13], un classificatore di concetti open source molto popolare ed efficiente.

4.2.1 FaCT System e FaCT Server

Il FaCT system è un classificatore di logiche descrittive che implementa algoritmi computazionalmente efficienti per testare la satisfiability e la subsumption di una Knowledge Base, ossia per verificare la consistenza di tutte le classi dell'ontologia gestita dinamicamente dalla KB e per scoprire nuove relazioni di super-class e sub-class che sono implicite nelle definizioni dell'ontologia ma non esplicitamente formulate.

In particolare esso fornisce due resoner che implementano rispettivamente la logica SHF e quella SHIQ, ed in particolare, tra le due, la logica SHIQ è quella che meglio supporta l'espressività di DAML+OIL. SHIQ, infatti, come è mostrato nella figura seguente, include tutti i concetti forniti da DAML+OIL, con l'eccezione di quelli etichettati con le lettere "(D)" e "O", e cioè non supporta alcun tipo di dato concreto e quindi non fornisce nessun meccanismo per ragionare con questa parte del linguaggio, né prevede la possibilità di avere nominals e restrizioni con valori, che sono necessari per definire set di istanze di concetti e per definire proprietà con valori specifici.

DL Expressiveness	**DL Syntax**	**DAML/XMLS Syntax**	**Serv. Descript. Lang.**
ALC, also called *S* when transitively closed primitive roles are included	A	Daml:Class	Concept
	T	Daml:Thing	Thing (Top)
	⊥	Daml:Nothing	Nothing (Bottom)
	($C \subseteq D$)	Daml:subClassOf	Subsumption
	($C \equiv D$)	Daml:sameClassAs	Equivalence
	R	Daml:Property	Role:Properties
	R	Daml:ObjectProperty	ObjectProperties
	($C \cap D$)	Daml:intersectionOf	Conjunction
	($C \cup D$)	Daml:disjunctionOf	Disjunction
	$\neg C$	Daml:complementOf	Negation
	$\forall R.C$	Daml:toClass	Universal Role Rest.
	$\exists R.C$	Daml:hasClass	Existential Role Rest.
N	$\leq nR.\top$	Daml:maxCardinality	Non-Qualifed Card.
	$\leq nR.\top$	Daml:minCardinality	
	$= nR.\top$	Daml:cardinality	
Q	$\leq nR.C$	Daml:hasClassQ	Qualifed Cardinality
		Daml:minCardinalityQ	
	$\leq nR.C$	Daml:hasClassQ	
		Daml:maxCardinalityQ	
	$= nR.C$	Daml:hasClassQ	
		Daml:cardinalityQ	
I	R^-	Daml:inverseOf	Inverse Roles
H	$(R \subseteq S)$	Daml:subPropertyOf	Role Hierarchy: Subsumption of Roles
	$(R \equiv S)$	Daml:samePropertyOf	Equivalence of Roles
O	$\{o, p, \ldots\}$	XML Type + rdf:value	Nominals (Collection of values)
	$\exists T.\{o, p, \ldots\}$	Daml:hasValue	Value Restrictions
(D)	D	Daml:Datatype + XMLS	Datatype System

	T	Daml:datatypeProperty	Datatype Property
	$\exists$ *T.d*	Daml:hasClass+XMLSType	Exist. Datat. Rest.
	$\forall$ *T.d*	Daml:toClass+XMLSType	Univ. Datat. Rest.

Figura 4.2:Corrispondenza tra i concetti delle DL e di DAML+OIL

FaCT, inoltre, supporta i concetti richiesti per la definizione del Tbox, ma fornisce limitate funzionalità per supportare l'introduzione e la descrizione di individui: non mira, infatti, a supportare la produzione di una grossa KB estensionale ma ad usare gli individui solo nelle descrizioni delle classi.

Esso è implementato in Lisp, un linguaggio funzionale per l'elaborazione di strutture dati adatte ad esprimere espressioni formali, come formule della logica simbolica. Per consentire agli utenti di poter accedere al FaCT system, è stato realizzato un server CORBA, il FaCT Server, che fornisce un'interfaccia con le funzionalità necessarie ad interagire coi reasoner, senza dover adoperare client Lisp. Esso si registra presso il Naming Service, consentendo, ai client che intendono utilizzarlo, di poterlo facilmente individuare.

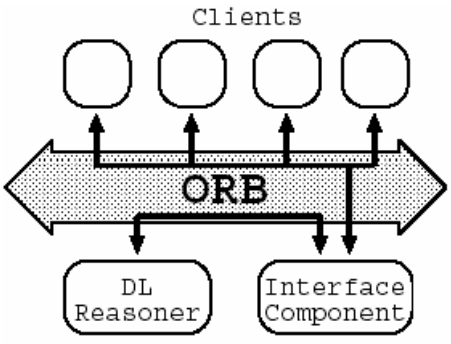

Figura 4.3:Architettura del FaCT Server

L'interfaccia del FaCT Server è fornita attraverso una CORBA IDL, che identifica i seguenti componenti fondamentali:

- *Classifier*, che incapsula il concetto di reasoner DL: ad ogni incarnazione del reasoner corrisponde un singolo oggetto classifer;

- *ClientHandler*, che si occupa degli aspetti legati alla connessione di un client con il classifer, dei comandi per la gestione della KB e delle query che è possibile formulare.

Questa caratterizzazione consente di poter distinguere tra client e fornisce la possibilità di avvertire i client se la KB è stata modificata da qualcun altro. E' fornito, inoltre, un semplice modello transazionale che si preoccupa di prevenire che più client si connettano contemporaneamente alla KB.

4.3 Modello architetturale dell'Ontology Service

L'Ontology Service è costituito, da un punto di vista architetturale, da tre componenti fondamentali: l'*OntoKB*, l'*OntoServer*, l'*OntoWrapper*.

Nella figura seguente sono mostrati i blocchi logici relativi a questi componenti, dei quali è fornita una dettagliata descrizione nei paragrafi successivi:

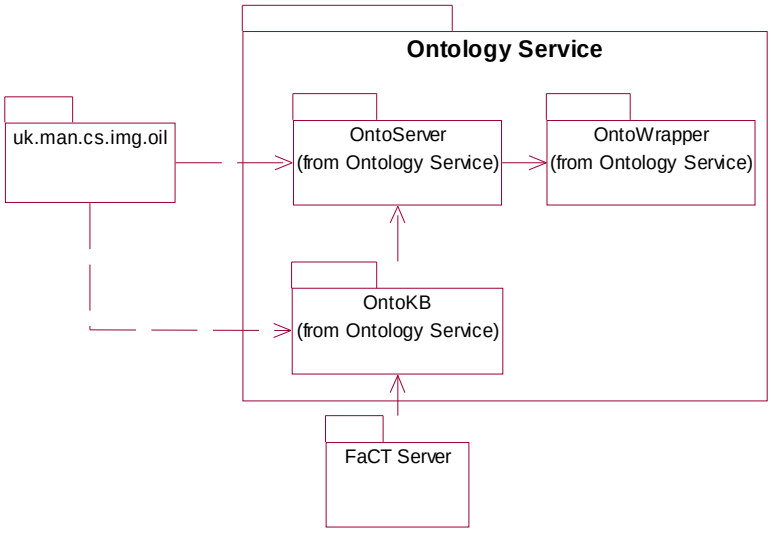

Figura 4.4:Modello architetturale dell'Ontology Service

4.3.1 OntoKB

L'OntoKB è il componente che interagisce direttamente con il FaCT Server, e quindi con la KB che esso implementa, ed in virtù di questo si preoccupa, fondamentalmente, di gestire gli aspetti legati alla connessione col FaCT Server, di creare la KB, a partire da una struttura dati utilizzata dall'OntoServer, e di fornire tutta una serie di metodi per controllare, modificare e interrogare la KB stessa.

L'OntoKB, quindi, disaccoppia l'OntoServer dalla particolare KB, nascondendo i dettagli relativi al componente specifico che la fornisce. Nel caso in cui tale componente dovesse essere sostituito, basterà modificare solo l'OntoKB, lasciandone inalterata l'interfaccia. In questo modo l'OntoServer potrà continuare ad utilizzare la KB, senza dover apportare alcuna modifica.

Dal punto di vista architetturale, l'OntoKB è stato suddiviso, a sua volta, in tre componenti, ognuno dei quali svolge una delle funzionalità menzionate:

- il *ClientConnector*, che gestisce gli aspetti legati alla connessione col FaCT Server;
- l'*OntoLoader*, che ha il compito di generare la KB;
- il *FactClient*, che fornisce le funzionalità per interagire con la KB.

Essi interagiscono tutti col FaCT Server ed essendo quest'ultimo un oggetto CORBA, per poter effettuare qualsiasi operazione su di esso, utilizzano stub e classi di supporto generate a partire dalla sua IDL.

Nella figura seguente sono mostrati i blocchi logici relativi a questi componenti, per i quali è fornita una caratterizzazione dettagliata nei paragrafi successivi.

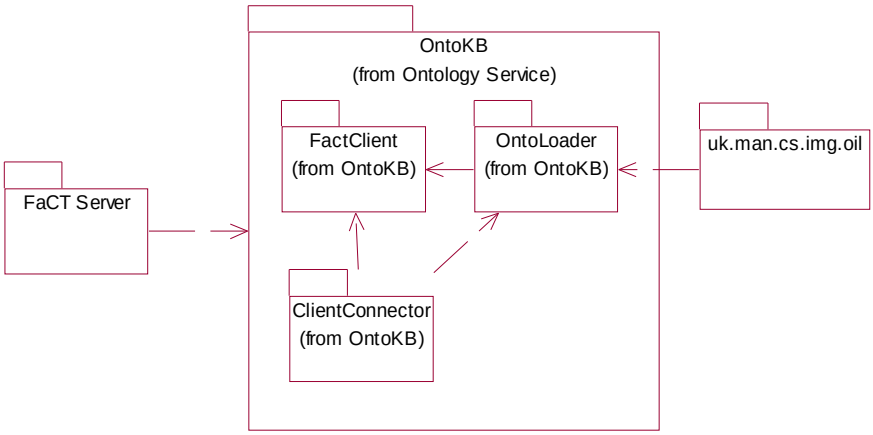

Figura 4.5:Modello architetturale dell'OntoKB

4.3.1.1 ClientConnector

Il ClientConnector è il componente che rende possibile la connesione tra FactClient e FaCTServer, utilizzando CORBA come middleware abilitante.

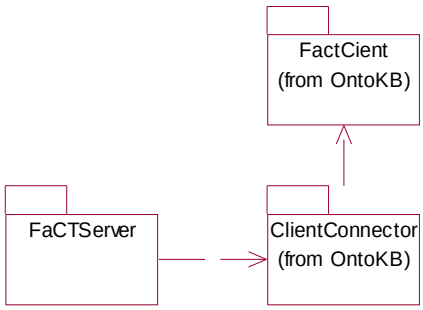

Figura 4.6: Modello architetturale del ClientConnector

Il FaCT Server, com'è già stato detto, viene fisicamente istanziato come un oggetto CORBA. Esso va a registrarsi al CORBA Naming Service in modo da poter essere facilmente individuato da altri oggetti CORBA che intendono utilizzarlo. Il ClientConnector, a partire dall'indirizzo di rete e di porta della macchina su cui è allocato il FaCT Server, riesce a ricostruire l'url su cui quest'ultimo pubblica l'IOR del Naming Service presso il quale si è registrato, e poi, adoperando il nome del server di cui è alla ricerca, riesce ad ottenere dal Naming Service stesso l'IOR del FaCT Server. A questo punto è in grado di

istanziare un oggetto Classifier, che in pratica è un'istanza del reasoner FaCT, ed un oggetto ClientHandler per questo Classifier, associato al client che ne ha richiesto l'utilizzo.

Il Client Connector è stato modellato ed implementato attraverso la classe in figura.

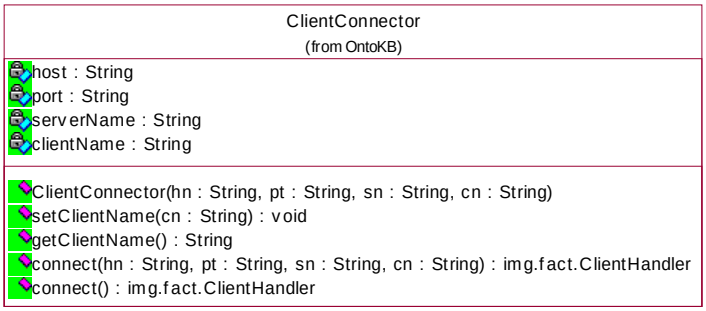

Figura 4.7: La classe ClientConnector

Essa offre le seguenti funzionalità:

- *ClientConnector* è il costruttore della classe, e permette di istanziare un oggetto Client Connector a partire dal nome del client che richiede la connessione, dall'indirizzo di rete, di porta e dal nome dell'oggetto CORBA con cui intende connettersi;
- *setClientName* consente di modificare il nome di chi richiede la connessione;
- *getClientName* consente di conoscere il nome di chi richiede la connessione;
- *connect* gestisce i dettagli di connessione di cui si è parlato poco prima. Può utilizzare i parametri necessari alla connessione definiti all'atto dell'istanziazione dell'oggetto o ridefinirne degli altri.

4.3.1.2 OntologyLoader

L'OntologyLoader è il componente che si occupa di creare una KB a partire da una struttura dati interna dell'OntoServer in cui è immagazzinata

un'ontologia. Il FaCT Server crea una KB a partire da asserzioni espresse in SHIQ. L'OntologyLoader, in pratica, riceve dall'OntoServer l'ontologia che si vuole caricare e la converte in un file SHIQ che poi compila, fornendo al FaCT Server i comandi necessari a costruire la KB.

Il comportamento di tale componente, come possibile vedere dalla figura in basso, è stato modellato ed implementato attraverso due classi:

- la classe *Compiler*, che compila il file SHIQ e crea la KB;
- la classe *OntologyLoader*, che converte l'ontologia in un file SHIQ e richiama la classe Compiler per costruire la KB.

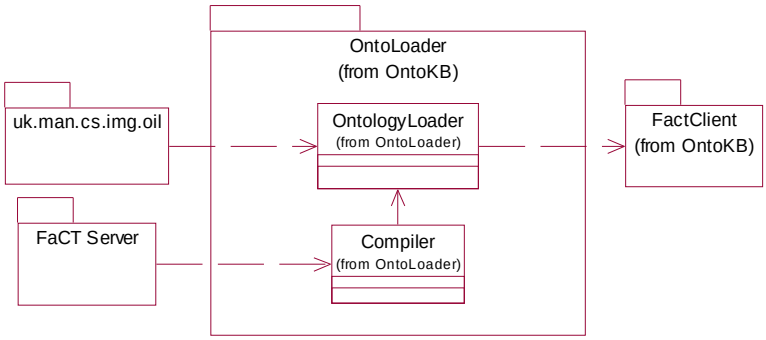

Figura 4.8:Modello architetturale dell'OntoLoader

La classe OntologyLoader offre le seguenti funzionalità:

- *OntologyLoader* è il costruttore della classe, ed istanzia un oggetto OntologyLoader associandolo allo specifico oggetto ClientHandler che lo ha richiesto;
- *ontoload* è la funzione che si occupa di caricare un'ontologia nel FaCT Server. In pratica essa riceve dal FactClient un oggetto della classe Ontology del package uk.ma.cs.img.oil, che rappresenta la struttura dati che gestisce l'ontologia che si vuole caricare, e di cui si parlerà diffusamente in seguito. Poi ne fa il render, ossia lo converte in un file SHIQ richiamando la funzione privata *renderOntology*, che a sua volta utilizza un renderer SHIQ contenuto nel package uk.ma.cs.img.oil.

Infine utilizza le funzionalità offerte dalla classe Compiler per mandare il file SHIQ al FaCT Server.

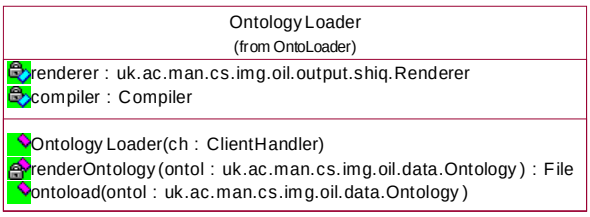

Figura 4.9:La classe OntologyLoader

La classe Compiler, rappresentata nella figura seguente, è stata fornita open source dagli sviluppatori del sistema FaCT e per questo motivo viene analizzata in maniera abbastanza generale.

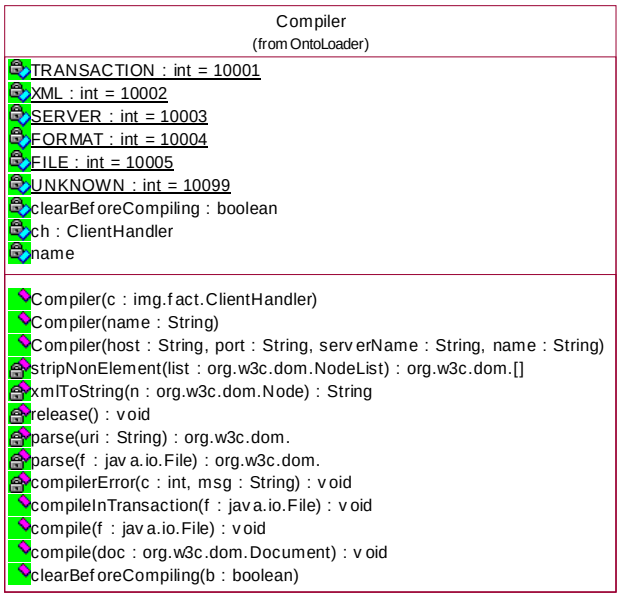

Figura 4.10:La classe Compiler

Essa fornisce le seguenti funzionalità:

- *Compiler* rappresenta il costruttore della classe e consente di connettersi al FaCT Server, attraverso la classe ClientConnector, fornendo indirizzo IP e di porta, nome del server e nome del client

richiedente, o in alternativa solo il nome del client o un oggetto ClientHandler.

- *compileInTransaction* compila un file SHIQ all'interno di una transazione;
- *clearBeforeCompiling* è una funzione che viene utilizzata per controllare se dalla KB sono state rimosse tutte le asserzioni prima di compilare;
- *release* è un metodo privato utilizzato per rilasciare il ClientHandler, dopo che il file è stato compilato;
- *parse* è un metodo privato utilizzato per effettuare il parsing di un file, trasformandolo in un documento xml;
- *compile* consente di compilare un documento xml, andando ad interpretare il contenuto dei singoli tag di ogni nodo xml e richiamando le funzioni opportune offerte dal FaCT Server per aggiungere alla KB nuovi concetti, proprietà, implicazioni e relazioni di uguaglianza tra concetti e proprietà;
- *compileError* è un metodo privato utilizzato per gestire errori di compilazione. I possibili tipi di errore previsti compaiono come attributi della classe Compiler;
- *stripNonElement* è un metodo privato utilizzato dalla funzione compile per estrarre da una lista di nodi xml tutti quelli testuali;
- *xmlToString* è un metodo privato utilizzato dalla funzione compile per convertire un nodo xml in una stringa di testo.

4.3.1.3 FactClient

Il FactClient è il componente che si occupa di interagire con la KB attraverso il FaCT Server, fornendo tutta una serie di funzionalità che la utilizzano in maniera diretta. Esso utilizza il Client Connector per stabilire la

connessione col FaCT Server, e poi il Classifier ed il ClientHandler associati all'istanza per poter accedere al reasoner che il FaCT offre. Inoltre, adopera l'OntoLoader per caricare un'ontologia nel FaCT Server.

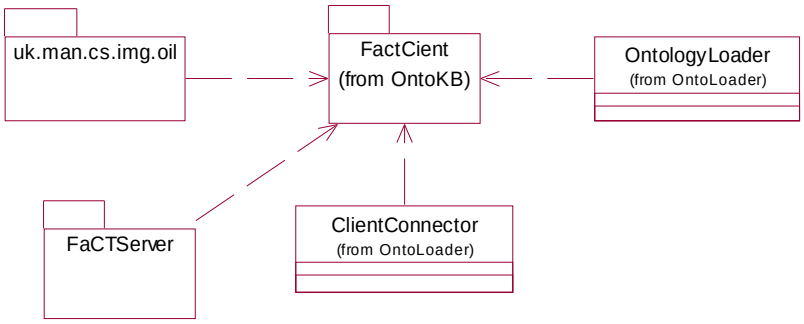

Figura 4.11:Modello architetturale del FactClient

Il FactClient è stato modellato ed implementato attraverso la classe in figura:

Figura 4.12:La classe FactClient

Esso offre le seguenti funzionalità per il controllo della KB, utilizzando il semplice modello transazionale messo a disposizione dal FaCT:

- *beginTransaction* è una richiesta per l'apertura di una transazione sulla KB. Non esiste coda di attesa e quindi, se il client non riesce ad iniziare la transazione, dovrà riprovarci in seguito;
- *resetTransaction* è una richiesta di riportare la corrente transazione aperta allo stato iniziale, ossia di scartare tutte le modifiche fatte come se non fossero mai state effettuate;
- *endTransaction* è una richiesta di chiusura di una transazione aperta. Essa va a buon fine solo se c'è una transazione aperta dal client richiedente e tutte le modifiche effettuate durante la transazione sono state commissionate con successo. In caso contrario lo stato della KB è riportato a come era prima che la transazione venisse lanciata;
- *abortTransaction* è una richiesta di abortire la corrente transazione aperta. Se c'è una transazione aperta dal client richiedente, allora tutte le modifiche in essa effettuate vengono scartate e lo stato della KB è riportato a come era prima che la transazione venisse lanciata;
- *transaction* indica se è in atto o meno una transazione.

Sono fornite, poi funzionalità che consentono di interrogare la KB:

- *allSubs* ritorna tutti i concetti che, rispetto alla tassonomia della KB, sono specializzazioni di un dato concetto;
- *allSupers* ritorna tutti i concetti che, rispetto alla tassonomia della KB, sono generalizzazioni di un dato concetto;
- *directSubs* ritorna tutti i concetti che, rispetto alla tassonomia della KB, sono specializzazioni dirette di un dato concetto;
- *directSupers* ritorna tutti i concetti che, rispetto alla tassonomia della KB, sono generalizzazioni dirette di un dato concetto;
- *checkConcept* valuta se un concetto è stato inserito nella tassonomia della KB, controllando se ha super o sub concetti;

- *checkSatisfiability* controlla la satisfiability di un concetto rispetto alla tassonomia della KB;
- *checkSubsumption* controlla la sussunzione tra due concetti rispetto alla tassonomia della KB;
- *checkEquivalence* controlla l'equivalenza tra due concetti rispetto alla tassonomia della KB;

Queste ultime tre funzioni utilizzano la funzione privata *getDescription* per poter ottenere la descrizione di un dato concetto in XML, in modo da poterla inserire nella KB per effettuare le valutazioni richieste.

Sono previste, inoltre, funzionalità che gestiscono la connessione col FaCT Server, adoperando il ClientConnector:

- *connect* è una richiesta di connessione al Fact Server. Può essere fornito un client handler oppure se ne può usare uno di default;
- *release* è una richiesta di rilascio del client handler. Se una transazione è aperta, essa viene abortita;
- *connected* indica se il client è connesso o meno al server.

Infine, sono fornite funzionalità che consentono di modificare la KB:

- *loadOntology* provvede a caricare un'ontologia espressa attraverso un file SHIQ all'interno del FaCT Server, adoperando l'OntologyLoader;
- *clear* rimuove tutte le asserzioni dalla KB.

4.3.2 OntoServer

L'OntoServer espone, attraverso un'interfaccia CORBA, una serie di funzionalità che si preoccupano di gestire l'ontologia corrente del sistema e di interagire con la KB tramite l'OntoKB. Questa interfaccia è formalizzata in una IDL, a partire dalla quale è possibile generare stub e le classi di supporto necessarie per interagire con l'OntoServer stesso.

Dal punto di vista architetturale, come è mostrato nella figura seguente, l'OntoServer è stato suddiviso, a sua volta, in due componenti, l'*OntoParser* e l'*OntologyServer*.

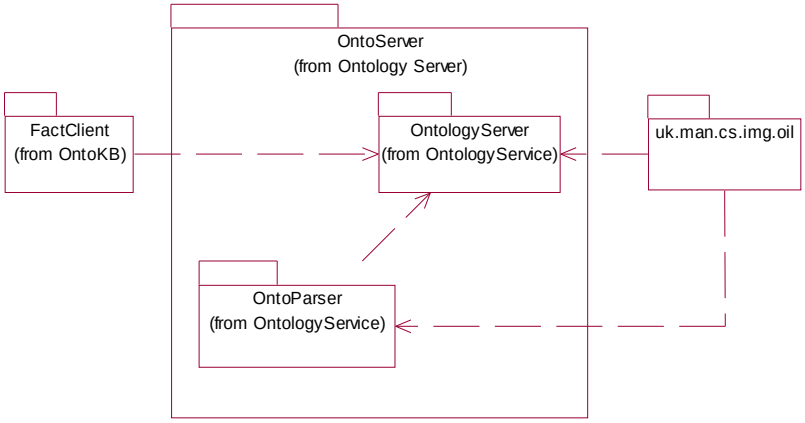

Figura 4.13Modello arhcitetturale dell'OntoServer

Questi blocchi logici sono descritti in modo dettagliato nei paragrafi successivi.

4.3.2.1 OntoParser

L'OntoParser è il componente che si preoccupa di estrarre un'ontologia o la descrizione di un'entità da un file scritto in DAML+OIL e di immagazzinare classi relazioni, assiomi in essa contenuti all'interno di una struttura dati fornita dal package uk.man.cs.img.oil, la classe Ontology.

Oggetti di questa classe vengono utilizzati sia per memorizzare temporaneamente ontologie, prima che esse vengano caricate nella KB, dopo di che essi vengono distrutti, sia per gestire l'ontologia che correntemente descrive il sistema, consentendo di potervi inserire facilmente ed in maniera dinamica nuovi concetti. E' possibile accedere a questa struttura dati per ottenere informazioni sulle classi e le proprietà che modellano il sistema, ma è necessario interagire con la KB se invece le informazioni di cui si ha bisogno hanno, in qualche modo, a che fare con la satisfiability o la subsumption.

L'OntoParser è stato modellato ed implementato attraverso la classe in figura:

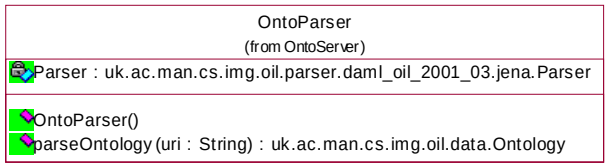

Figura 4.14:La classe OntoParser

La funzione *OntoParser* è il costruttore della classe, mentre la funzione *parseOntology* è quella che consente di effettuare il parsing di file DAML+OIL ossia di realizzare quanto detto sopra, adoperando a sua volta un parser DAML+OIL contenuto nel package uk.ma.cs.img.oil.

4.3.2.2 OntologyServer

L'OntologyServer è il componente che si occupa di caricare e validare ontologie espresse in DAML+OIL utilizzando la KB, di verificare la consistenza delle descrizioni delle istanze, di impostare l'ontologia corrente di sistema in un oggetto della classe Ontology e di inserire in essa nuove ontologie, di fornire la possibilità di effettuare alcune query di tipo logico.

Esso è stato modellato ed implementato attraverso la seguente classe:

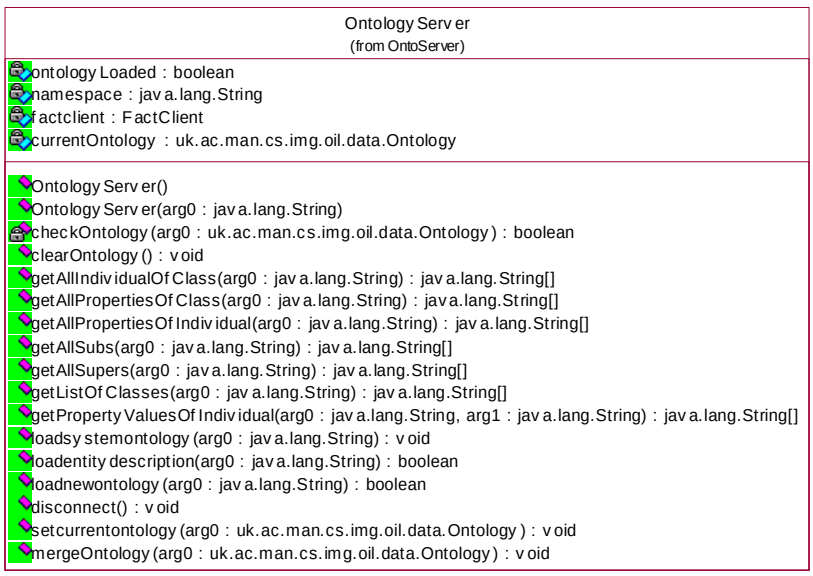

Figura 4.15:La classe OntologyServer

Essa fornisce funzionalità per operare su ontologie e descrizioni di entità:

- *loadsystemontology* inizializza un oggetto OntologyServer caricando un'ontologia DAML+OIL nella KB;
- *loadentitydescription* carica la descrizione di un'istanza, formulata in DAML+OIL, all'interno della KB per verificare se essa è consistente con l'ontologia del sistema;
- *loadnewontology* consente di inserire all'interno dell'ontologia di sistema una nuova ontologia DAML+OIL che introduce nuovi concetti che vanno a modellare aspetti in essa non descritti ma che sono necessari a classificare e validare descrizioni di entità che li utilizzano;
- *clearOntology* ha il compito di cancellare tutte le asserzioni inserite nella KB e tutte le classi, relazioni, assiomi ed istanze memorizzate nell'oggetto della classe Ontology che gestisce l'ontologia di sistema;
- *checkOntology* verifica la satisfiability dell'ontologia di sistema, andando a controllare la consistenza di ogni sua classe e di ogni sua istanza attraverso la KB;
- *setCurrentOntology* consente di impostare l'ontologia di sistema;
- *mergeOntology* consente di fondere fra loro due oggetti della classe Ontology;
- *disconnect* consente di disconnettersi dalla KB.

E' possibile, poi, formulare query logiche, adoperando le seguenti funzioni:

- *GetIndividualOfClass* consente di ricavare dalla KB tutte le istanze relative ad una data classe. Viene prima verificata l'appartenza di quest'ultima alla KB; poi, per ogni istanza della KB, viene valutato se esiste una relazione di sussunzione con essa. Le istanze della classe data saranno tutte quelle che avranno verificato la relazione di sussunzione;

- *GetAllPropertiesOfClass* consente di ricavare, dall'ontologia corrente di sistema, tutte le proprietà relative ad una data classe. Viene prima verificata l'appartenza di quest'ultima all'ontologia, dopo di che vengono estratte tutte le sue proprietà;
- *GetAllPropertiesOfIndividual* consente di ricavare, dall'ontologia corrente di sistema, tutte le proprietà relative ad una data istanza. Anche in questo caso viene prima verificata l'appartenza di quest'ultima all'ontologia, dopo di che vengono estratte tutte le sue proprietà;
- *GetPropertyValuesOfIndividual* consente di ricavare il valore specifico che assume una data proprietà, associata ad una particolare istanza, interrogando l'ontologia corrente di sistema;
- *GetListOfClasses* consente di ricavare, dall'ontologia corrente di sistema, la lista di tutte le classi che contengono una particolare sottostringa;
- *GetAllSubs* consente di ricavare tutte le sottoclassi di una data classe. In particolare viene interrogata la KB per verificare prima l'appartenenza della classe alla KB stessa e poi per individuare le sue sottoclassi;
- *GetAllSupers* consente di ricavare tutte le superclassi di una data classe. In particolare viene interrogata la KB per verificare prima l'appartenenza della classe alla KB stessa e poi per individuare le sue superclassi.

Di seguito sono forniti dei semplici modelli di interazione che mostrano come i componenti dell'Ontology Service comunicano fra loro. In particolare, i modelli proposti descrivono il comportamento delle funzioni *loadsystemontology, loadentitydescription* e *loadnewontology*. In essi, per motivi di leggibilità, non sono stati inseriti i componenti del FaCTServer ed i messaggi scambiati con essi.

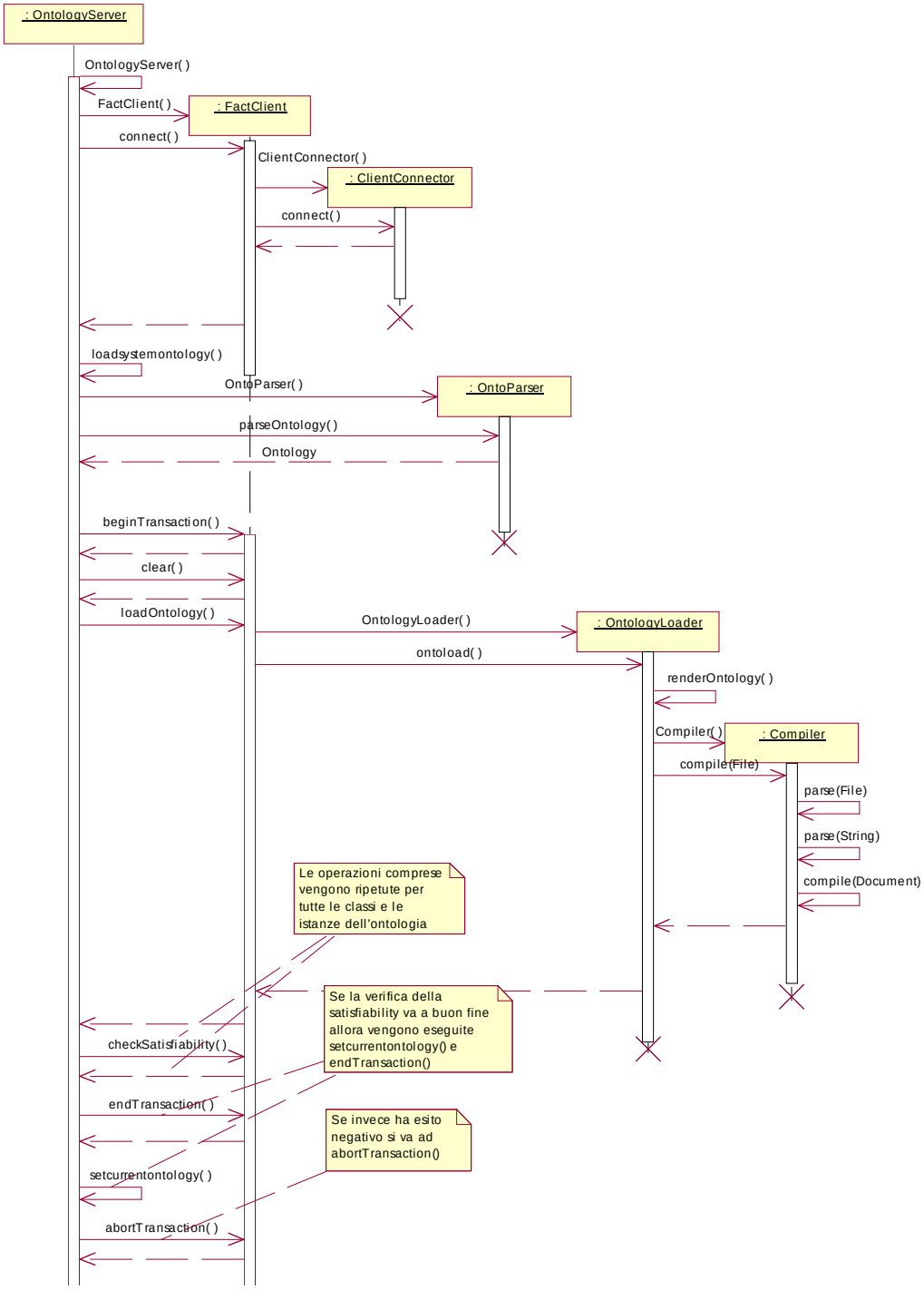

Figura 4.16:Modello di interazione per la funzione loadsystemontology

Il primo modello di interazione, rappresentato nella figura precedente, descrive il comportamento della funzione *loadsystemontology*. Essa, dopo aver cancellato tutte le asserzioni contenute nella KB, effettua il parsing del file

attraverso l'OntoParser, memorizzando l'ontologia in un oggetto della classe Ontology. Poi richiama l'OntologyLoader per convertirla in un file SHIQ ed il Compiler per costruire a partire da esso la KB. Infine invoca il FactClient per testare la satisfiability di ogni classe e istanza. Tutte queste operazioni vengono effettuate all'interno di una transazione. Se la satisfiability non è verificata non viene aggiornata l'ontologia corrente di sistema e la transazione viene abortita.

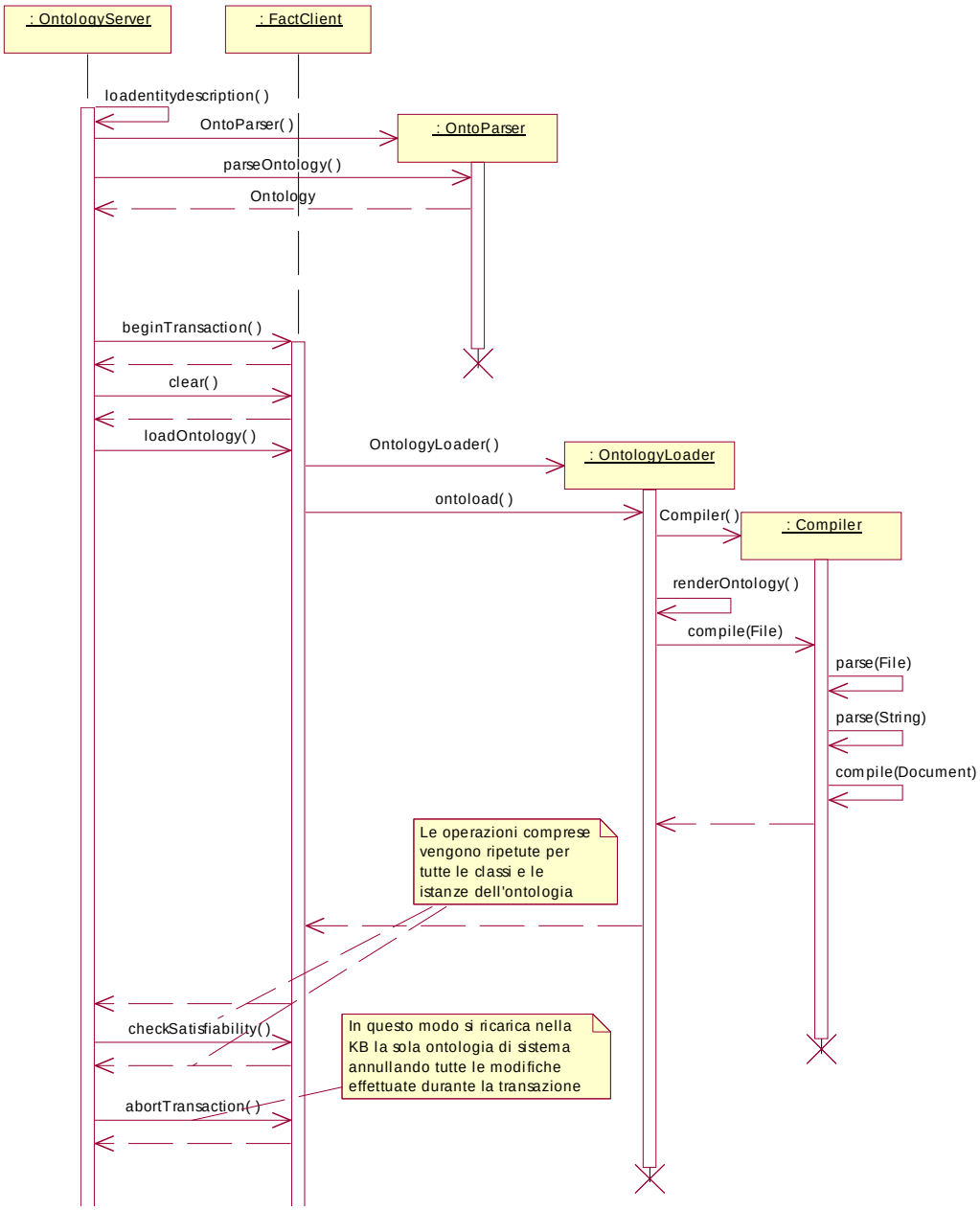

Figura 4.17:Modello di interazione per la funzione loadentitydescription

Il secondo modello di interazione, rappresentato nella figura precedente, descrive il comportamento della funzione *loadentitydescription.* Essa dopo aver caricato la descrizione dell'entità nella KB ed aver verificato la satisfiability, provvede a reinserire nella KB la sola ontologia di sistema. Anche per in questo caso tutte queste operazioni vengono effettuate all'interno di una transazione.

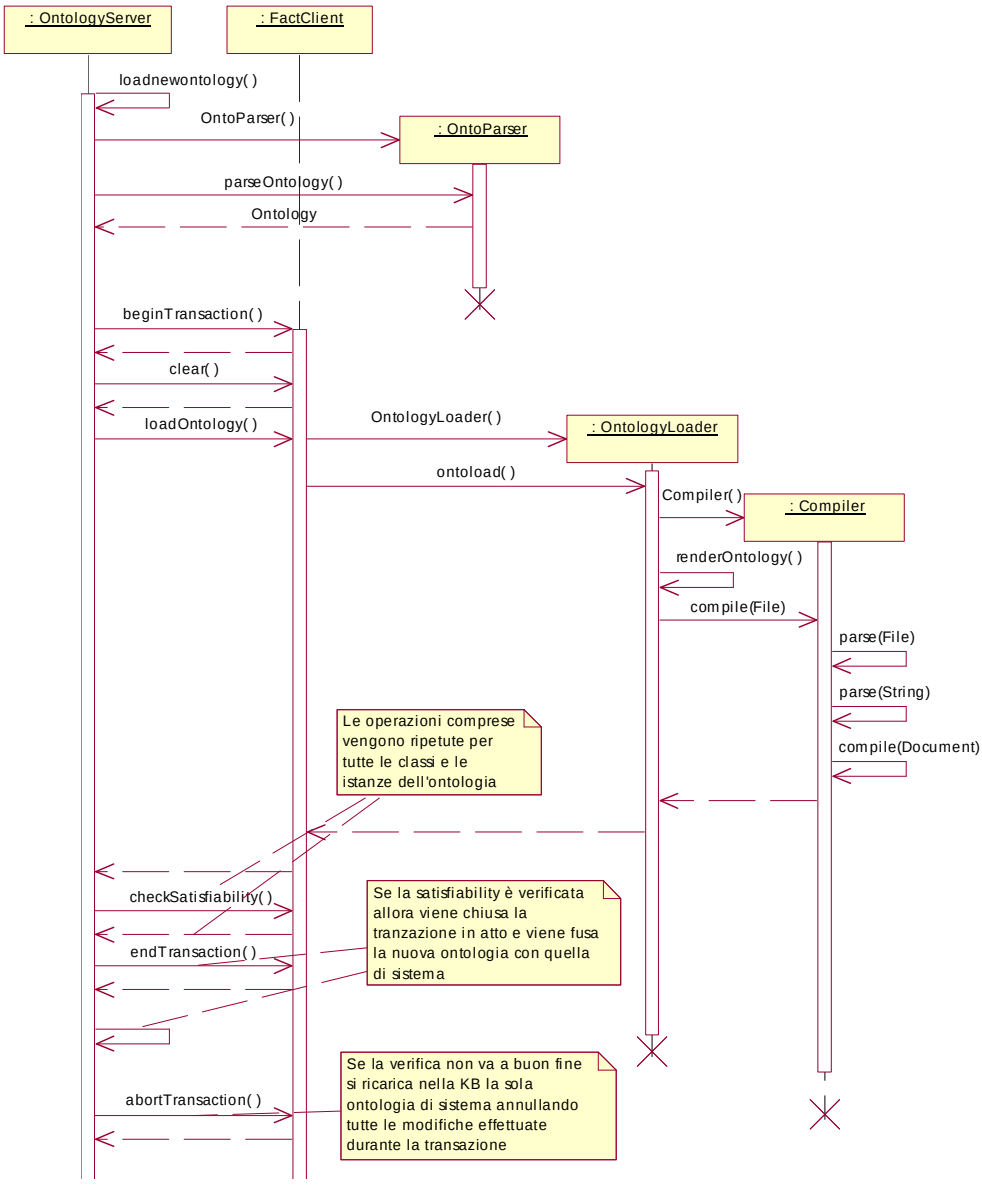

Figura 4.18:Modello di interazione per la funzione loadnewontology

Infine, l'ulitmo modello di interazione, rappresentato nella figura precedente, descrive il comportamento della funzione *loadnewontology*. Essa carica la nuova ontologia nella KB, inserendovi nuovi concetti e proprietà. In seguito viene verificata la satisfiability dell'intera KB, e nel caso in cui tale controllo va a buon fine viene fatta la fusione tra la nuova ontologia e quella di sistema nella struttura dati che mantiene quest'ultima. In caso contrario viene abortita la transazione.

4.3.3 OntoWrapper

L'OntoWrapper è il componente che si preoccupa di esporre l'OntoServer come Web Service, fornendo l'interfaccia attraverso la quale le altre entità dell'ambiente possono utilizzarlo.

L'OntoServer non si presenta direttamente come Web Services poiché è stato pensato ed implementato, in origine, come un servizio CORBA, attivo ed esistente di per sé, utilizzabile da altre entità attraverso un'interfaccia CORBA. Solo in seguito è stato inserito nell'ambiente UbiSystem, che, come è stato detto, usa appunto i Web Services come middleware abilitante, ed è quindi stato necessario introdurre un wrapper.

L'OntoWrapper è stato implementato attraverso la classe in figura:

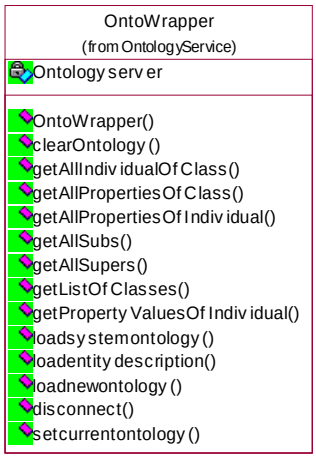

Figura 4.19:La classe OntoWrapper

Essa, in pratica, funge da client CORBA per l'OntologyServer: il suo costruttore adopera il Naming Service per ottenere il riferimento all'oggetto OntologyServer mentre tutti i suoi metodi richiamano, al loro interno, i metodi omonimi di quest'ultimo.

Questa classe è stata, poi, esposta come Web Service, utilizzando Tomcat [55] come Web Server. Gli stub, la WSDL e le altre classi necessarie per esporre un Web Service e per interagire con esso, sono state realizzate utilizzando le librerie di Apache Axis [54], che mettono a disposizione una serie di toolkit che rendono estremamente semplice la realizzazione di Web Service a partire da applicazioni Java.

Quando si interagisce con il Web Service ottenuto, si accede prima ad uno dei servizi da esso offerti, utilizzando SOAP su HTTP; poi, il servizio scelto, che è in pratica uno dei metodi dell'Ontowrapper, richiama il metodo corrispondente dell'OntologyServer attraverso CORBA.

Naturalmente, per poter utilizzare l'OntologyServer senza incorrere in malfunzionamenti, il Web Service ha bisogno, prima che una qualsiasi invocazione di un suo servizio venga effettuata, che l'Ontology Server stesso sia già attivo e che si sia già registrato al Naming Service.

Conclusioni

Questo lavoro di tesi, fondamentalmente, individua nell'integrazione delle tecnologie del web semantico all'interno dei sistemi di Pervasive Computing una possibile soluzione per risolvere i problemi di interoperabilità semantica tra le entità in essi presenti.

L'utilizzo delle ontologie come vocabolari condivisi che descrivono l'ambiente circostante e di DAML+OIL come liguaggio capace di fornire un formalismo universalmente interpretabile e semanticamente non ambiguo rappresentano gli aspetti chiave di questo approccio.

Il potere espressivo di cui è fornito DAML+OIL consente, inoltre, di utilizzare le ontologie non solo come uno strumento di conoscenza statico ma di poter effettuare inferenze sul contenuto informativo che esse offrono, traendone nuove informazioni o risolvendo inconsistenze logiche. Per effettuare questi ragionamenti a partire, appunto da file DAML+OIL, è stato implementato un Ontology Service che fornisce un'interfaccia standard per accedere ad una Knowledge Base, interrogando un reasoner, il FaCT System, basato sulla logica descrittiva SHIQ.

La forza teorica di questo approccio va, tuttavia a scontrarsi con aspetti di carattere pratico, che forniscono, almeno per ora, alcune limitazioni.

DAML+OIL e le logiche descrittive su cui esso si basa, infatti, non offrono, attualmente, le funzionalità sufficienti a soddisfare tutte le esigenze e a risolvere le molteplici problematiche dei sistemi di Pervasive Computing. In particolare, la logica che è stata utilizzata in questo lavoro, ossia SHIQ, non è adatta a trattare con concetti quantitativi, che rappresentano, però, un aspetto fondamentale del Pervasive Computing, né ad avere a che fare con gli individui.

Una logica in grado di sopperire a queste mancanze è SHOQ(D) [14] anche se essa, a causa di un'alta complessità computazionale, smette di trattare le proprietà inverse. E', tuttavia, più probabile, e comunque ragionevolmente più utile, che all'interno di ontologie vengano utilizzati dati concreti piuttosto che proprietà inverse e quindi la scelta di SHOQ(D) al posto di SHIQ come logica di supporto ai ragionamenti può essere ritenuta più che valida. Un reasoner che implementa tale logica, però, non è tuttora stato portato a termine, ma è ancora in fase di realizzazione.

Nonostante tali limitazioni, adoperando le ontologie, è stato possibile ottenere una corretta caratterizzazione di un ambiente pervasivo, di tutti i termini che possono essere usati e di ciò che esso contiene, controllando la consistenza delle informazioni con gli schemi definiti nelle ontologie stesse.

E' stato possibile, poi, effettuare discovery semantico di un servizio, ossia di scoprire tutti e soli gli oggetti rilevanti, senza tuttavia conoscerli a priori, basandosi su requisiti del richiedente e adattando i risultati in base alle sue caratteristiche.

Infine, le ontologie stesse sono state scelte come strumento per consentrire l'interazione, sia agli uomini che alle entità software, con i vari componenti dell'ambiente, utilizzandone le funzionalità o collaborando con loro.

Altri aspetti, come la sicurezza, la privacy e il controllo di accesso, che assumono una maggiore rilevanza in tali sistemi, completamente aperti e in cui ogni cosa è esposta per essere utilizzata, dovranno essere gestiti.

Anche per essi le ontologie potranno fornire una possibile soluzione, anche se, almeno per ora, i motori inferenziali esistenti non supportano politiche di sicurezza, né DAML+OIL fornisce funzionalità per limitare la visibilità dei concetti e delle proprietà che, attraverso esso, possono essere definite..

Appendice A

Ontologie in DAML+OIL

In questa appendice sono state formalizzate in DAML+OIL le ontologie definite precedentemente. Sono, poi, fornite le descrizioni, anch'esse in DAML+OIL, di alcune delle entità introdotte nel capitolo 3: in particolare è stata proposta la descrizione del laptop, del visualizzatore di file Pdf, in una versione completa e che è stata implementata all'interno di UbiSystem, di un utente e di un particolare contesto.

A.1 Ontologia per le entità di un ambiente pervasivo

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://www.ubinet.it/Ontology.daml">
  <!ENTITY DEFAULT "http://www.ubinet.it/Ontology.daml"> ]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd      =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">
```

```
  <daml:Ontology rdf:about="">
     <dc:title>&amp;quot;Ontology&amp;quot;</dc:title>
     <dc:date>20/12/2003</dc:date>
     <dc:creator>Esposito Massimo</dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
         descrivere le entità di un ambiente pervasivo.
     </dc:description>
     <dc:subject>Un&amp;apos;ontologia per ambienti multimediali.
     </dc:subject>
     <daml:versionInfo>5.0</daml:versionInfo>
  </daml:Ontology>

<!--  Entità dell'ambiente   -->

  <daml:Class rdf:about="&ontology;#Entity">
     <rdfs:label>Entity</rdfs:label>
     <rdfs:comment>Classe di tutti gli oggetti presenti nell'ambiente
          pervasivo.Sue sottoclassi sono: Device,Resource e User.
     </rdfs:comment>
  </daml:Class>

<!--  Device   -->

  <daml:Class rdf:about="&ontology;#Device">
     <rdfs:label>Device</rdfs:label>
     <rdfs:comment>Sottoclasse della classe Entity, rappresenta tutti
          i possibili dispositivi presenti nell'ambiente pervasivo.
     </rdfs:comment>
     <rdfs:subClassOf>
         <daml:Class rdf:about="&ontology;#Entity"/>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="&ontology;#information"/>
         </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onPropertyrdf:resource="&ontology;#hwproperties"/>
         </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onPropertyrdf:resource="&ontology;#swproperties"/>
         </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="&ontology;#devicetype"/>
         </daml:Restriction>
     </rdfs:subClassOf>
  </daml:Class>

  <daml:ObjectProperty rdf:about="&ontology;#information">
     <rdfs:comment>Questa proprietà fornisce le informazioni generali
             che caratterizzano un oggetto.
     </rdfs:comment>
     <rdfs:domain>
        <daml:Class>
           <daml:unionOf>
              <daml:List>
                 <daml:first>
                    <daml:Class
```

```
                  rdf:about="&ontology;#IoUnitDescription"/>
                </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
                         <daml:Class rdf:about="&ontology;#Device"/>
                   </daml:first>
                   <daml:rest>
                      <daml:List>
                         <daml:first>
                            <daml:Class
                              rdf:about="&ontology;#CpuDescription"/>
                         </daml:first>
                         <daml:rest>
                            <daml:List>
                               <daml:first>
                                  <daml:Class
                     rdf:about="&ontology;#ConnectionDescription"/>
                                </daml:first>
                                <daml:rest>
                                   <daml:nil/>
                                </daml:rest>
                              </daml:List>
                           </daml:rest>
                         </daml:List>
                      </daml:rest>
                    </daml:List>
                  </daml:rest>
               </daml:List>
            </daml:unionOf>
         </daml:Class>
      </rdfs:domain>
      <rdfs:range  rdf:resource="&ontology;#InfoDescription"/>
   </daml:ObjectProperty>

   <daml:Class rdf:about="&ontology;#InfoDescription">
      <rdfs:label>InfoDescription</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
        essere usata per definire name, version e vendor di un oggetto.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#infoname"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:maxCardinality="1">
             <daml:onProperty rdf:resource="&ontology;#infovendor"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:maxCardinality="1">
             <daml:onProperty rdf:resource="&ontology;#infoversion"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

   <daml:DatatypeProperty rdf:about="&ontology;#infoname">
      <rdfs:comment>Questa proprietà fornisce il name dell'oggetto che
              descrive.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#InfoDescription"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
```

```
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#infovendor">
   <rdfs:comment>Questa proprietà fornisce il vendor dell'oggetto
           che descrive.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#InfoDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#infoversion">
   <rdfs:comment>Questa proprietà fornisce la version dell'oggetto
           che descrive.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#InfoDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#devicetype">
   <rdfs:comment>Questa proprietà fornisce il devicetype.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Device"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:about="&ontology;#hwproperties">
   <rdfs:comment>Questa proprietà fornisce la lista di
           caratteristiche hardware di un device.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Device"/>
   <rdfs:range  rdf:resource="&ontology;#HwDescription"/>
</daml:ObjectProperty>

<daml:Class rdf:about="&ontology;#HwDescription">
   <rdfs:label>HwDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire le caratteristiche hardware di
           un device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#cpu"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:minCardinality="1">
          <daml:onProperty rdf:resource="&ontology;#connection"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:minCardinality="1">
          <daml:onProperty rdf:resource="&ontology;#ui"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:minCardinality="1">
          <daml:onProperty rdf:resource="&ontology;#memory"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>
```

```
<daml:ObjectProperty rdf:about="&ontology;#connection">
   <rdfs:comment>Questa proprietà fornisce il tipo di connessione
           che il device usa.
   </rdfs:comment>
   <rdfs:domain>
      <daml:Class>
         <daml:unionOf>
            <daml:List>
               <daml:first>
                  <daml:Class rdf:about="&ontology;#HwDescription"/>
            </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
                         <daml:Class
                              rdf:about="&ontology;#HwPrecondition"/>
                      </daml:first>
                      <daml:rest>
                         <daml:nil/>
                      </daml:rest>
                   </daml:List>
                </daml:rest>
             </daml:List>
          </daml:unionOf>
       </daml:Class>
   </rdfs:domain>
   <rdfs:range  rdf:resource="&ontology;#ConnectionDescription"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#ui">
   <rdfs:comment>Questa proprietà fornisce la lista delle user
           interface che il device offre.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#HwDescription"/>
   <rdfs:range  rdf:resource="&ontology;#UiDescription"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#io">
   <rdfs:comment>Questa proprietà fornisce la lista dei
           dispositivi di I/O che il device possiede.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#HwDescription"/>
   <rdfs:range  rdf:resource="&ontology;#IoUnitDescription"/>
</daml:ObjectProperty>

<daml:Class rdf:about="&ontology;#IoUnitDescription">
   <rdfs:label>IoUnitDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire i dispositivi di i/o di un
           device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#information"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:about="&ontology;#memory">
   <rdfs:comment>Questa proprietà fornisce la quantità di memoria
           che il device possiede.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#HwDescription"/>
```

```
   <rdfs:range  rdf:resource="&ontology;#MemoryDescription"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#cpu">
   <rdfs:comment>Questa proprietà fornisce il tipo di cpu che il
           device possiede.
   </rdfs:comment>
   <rdfs:domain>
       <daml:Class>
          <daml:unionOf>
             <daml:List>
                <daml:first>
                   <daml:Class
                        rdf:about="&ontology;#HwDescription"/>
         </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
                         <daml:Class
                            rdf:about="&ontology;#HwPrecondition"/>
                      </daml:first>
                      <daml:rest>
                         <daml:nil/>
                      </daml:rest>
                   </daml:List>
                </daml:rest>
             </daml:List>
          </daml:unionOf>
       </daml:Class>
  </rdfs:domain>
  <rdfs:range  rdf:resource="&ontology;#CpuDescription"/>
</daml:ObjectProperty>

<daml:Class rdf:about="&ontology;#CpuDescription">
   <rdfs:label>CpuDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire la cpu di un device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#information"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#cpuspeed"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#cpuunit"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:DatatypeProperty rdf:about="&ontology;#cpuspeed">
   <rdfs:comment>Questa proprietà specifica la velocità della cpu.
           Questo valore deve essere un numero non negativo.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#CpuDescription"/>
   <rdfs:range  rdf:resource="&xsd;#real"/>
</daml:DatatypeProperty>
```

```
<daml:DatatypeProperty rdf:about="&ontology;#cpuunit">
   <rdfs:comment>Questa proprietà specifica l'unità di misura per la
           velocità della cpu.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#CpuDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:Class rdf:about="&ontology;#ConnectionDescription">
   <rdfs:label>ConnectionDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire le caratteristiche di
           connessione di un device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#information"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#bandwidth"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#bandwidthunit"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:DatatypeProperty rdf:about="&ontology;#bandwidth">
   <rdfs:comment>Questa proprietà specifica la larghezza di banda
           della connessione. Questo valore deve essere positivo.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ConnectionDescription"/>
   <rdfs:range  rdf:resource="&xsd;#real"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#bandwidthunit">
   <rdfs:comment>Questa proprietà specifica l'unità di misura del
 parametro bandwidth.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ConnectionDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:Class rdf:about="&ontology;#UiDescription">
   <rdfs:label>UiDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire la user interface di un device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#screen"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#audio-input"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
```

```
        <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#audio-output"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#video-input"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:about="&ontology;#screen">
    <rdfs:comment>Questa proprietà fornisce le informazioni generali
            che caratterizzano lo screen di un device.
    </rdfs:comment>
    <rdfs:domain rdf:resource="&ontology;#UiDescription"/>
    <rdfs:range  rdf:resource="&ontology;#ScreenDescription"/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:about="&ontology;#audio-input">
    <rdfs:comment>Questa proprietà specifica se il device in
            questione è capace di ricevere input audio.
    </rdfs:comment>
    <rdfs:domain>
        <daml:Class>
           <daml:unionOf>
              <daml:List>
                 <daml:first>
                    <daml:Class
                          rdf:about="&ontology;#UiDescription"/>
         </daml:first>
                 <daml:rest>
                    <daml:List>
                       <daml:first>
                          <daml:Class
                             rdf:about="&ontology;#HwPrecondition"/>
                <daml:first>
                       <daml:rest>
                   <daml:nil/>
                <daml:rest>
                     </daml:List>
                  </daml:rest>
              </daml:List>
           </daml:unionOf>
        </daml:Class>
    </rdfs:domain>
    <rdfs:range  rdf:resource="&xsd;#boolean"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#video-input">
    <rdfs:comment>Questa proprietà specifica se il device in
            questione è capace di ricevere video in input.
    </rdfs:comment>
    <rdfs:domain>
        <daml:Class>
           <daml:unionOf>
              <daml:List>
                 <daml:first>
                    <daml:Class
                          rdf:about="&ontology;#UiDescription"/>
         </daml:first>
                 <daml:rest>
                    <daml:List>
```

```
                        <daml:first>
                           <daml:Class
                              rdf:about="&ontology;#HwPrecondition"/>
                     </daml:first>
                         <daml:rest>
                            <daml:nil/>
                         </daml:rest>
                      </daml:List>
                   </daml:rest>
                </daml:List>
             </daml:unionOf>
          </daml:Class>
      </rdfs:domain>
      <rdfs:range  rdf:resource="&xsd;#boolean"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#audio-output">
      <rdfs:comment>Questa proprietà specifica se il device in
              questione è capace di fornire output audio.
      </rdfs:comment>
      <rdfs:domain>
          <daml:Class>
             <daml:unionOf>
                <daml:List>
                   <daml:first>
                      <daml:Class
                                  rdf:about="&ontology;#UiDescription"/>
           </daml:first>
                   <daml:rest>
                      <daml:List>
                         <daml:first>
                            <daml:Class
                rdf:about="&ontology;#HwPrecondition"/>
             </daml:first>
                         <daml:rest>
                            <daml:nil/>
                           </daml:rest>
                      </daml:List>
                  </daml:rest>
             </daml:List>
           </daml:unionOf>
        </daml:Class>
      </rdfs:domain>
      <rdfs:range  rdf:resource="&xsd;#boolean"/>
   </daml:DatatypeProperty>

   <daml:Class rdf:about="&ontology;#ScreenDescription">
      <rdfs:label>ScreenDescription</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
              essere usata per definire le caratteristiche dello screen
              di un device.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#width"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#height"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
```

```
        <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#unit"/>
         </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#color"/>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:minCardinality="1">
           <daml:onProperty rdf:resource="&ontology;#resolution"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:DatatypeProperty rdf:about="&ontology;#width">
   <rdfs:comment>Questa proprietà specifica la larghezza dello
           screen. Questo valore deve essere positivo.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ScreenDescription"/>
   <rdfs:range  rdf:resource="&xsd;#real"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#height">
   <rdfs:comment>Questa proprietà specifica l'altezza dello screen.
           Questo valore deve essere positivo.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ScreenDescription"/>
   <rdfs:range  rdf:resource="&xsd;#real"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#unit">
   <rdfs:comment>Questa proprietà specifica l'unità di misura dei
           parametri width ed height.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ScreenDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#color">
   <rdfs:comment>Questa proprietà specifica se lo screen del device
           in questione è a colori o meno.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ScreenDescription"/>
   <rdfs:range  rdf:resource="&xsd;#boolean"/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:about="&ontology;#resolution">
   <rdfs:comment>Questa proprietà specifica la descrizione della
           risoluzione dello schermo.
   </rdfs:comment>
   <rdfs:domain>
       <daml:Class>
          <daml:unionOf>
             <daml:List>
                <daml:first>
                   <daml:Class
                           rdf:about="&ontology;#ScreenDescription"/>
         </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
```

```
                        <daml:Class
                            rdf:about="&ontology;#HwPrecondition"/>
            </daml:first>
                        <daml:rest>
                           <daml:nil/>
             </daml:rest>
                      </daml:List>
                  </daml:rest>
               </daml:List>
            </daml:unionOf>
         </daml:Class>
   </rdfs:domain>
   <rdfs:range  rdf:resource="&ontology;#ResolutionDescription"/>
</daml:ObjectProperty>

<daml:Class rdf:about="&ontology;#ResolutionDescription">
   <rdfs:label>ResolutionDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire i dettagli di risoluzione dello
           screen di un device o di un proiettore collegato ad esso.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty
                  rdf:resource="&ontology;#resolutionwidth"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty
                  rdf:resource="&ontology;#resolutionheight"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty
                  rdf:resource="&ontology;#resolutionunit"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:maxCardinality="1">
          <daml:onProperty rdf:resource="&ontology;#resolutionbpp"/>
         </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:maxCardinality="1">
          <daml:onProperty
                  rdf:resource="&ontology;#resolutiongraphics"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:DatatypeProperty rdf:about="&ontology;#resolutionwidth">
   <rdfs:comment>Questa proprietà specifica il numero di unità di
           risoluzione orizzontalmente. Questo valore deve essere
           positivo.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#ResolutionDescription"/>
   <rdfs:range  rdf:resource="&xsd;#integer"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#resolutionheight">
   <rdfs:comment>Questa proprietà specifica il numero di unità di
```

```
            risoluzione verticalmente. Questo valore deve essere
            positivo.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ResolutionDescription"/>
      <rdfs:range  rdf:resource="&xsd;#integer"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#resolutionunit">
      <rdfs:comment>Questa proprietà specifica l'unità di misura per la
            risoluzione.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ResolutionDescription"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#resolutionbpp">
      <rdfs:comment>Questa proprietà specifica il numero di bit per
            pixel.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ResolutionDescription"/>
      <rdfs:range  rdf:resource="&xsd;#integer"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#resolutiongraphics">
      <rdfs:comment>Questa proprietà specifica se lo screen del device
            in questione visualizza grafica o solo caratteri.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ResolutionDescription"/>
      <rdfs:range  rdf:resource="&xsd;#boolean"/>
   </daml:DatatypeProperty>

   <daml:Class rdf:about="&ontology;#MemoryDescription">
      <rdfs:label>MemoryDescription</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
            essere usata per definire la quantità di memoria di un
            device, l'unità di misura e il tipo di utilizzo.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#available"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#maximum"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

   <daml:ObjectProperty rdf:about="&ontology;#available">
      <rdfs:comment>Questa proprietà specifica la quantità di memoria
            disponibile.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#MemoryDescription"/>
      <rdfs:range  rdf:resource="&ontology;#MemoryTypeDescription"/>
   </daml:ObjectProperty>

   <daml:ObjectProperty rdf:about="&ontology;#maximum">
      <rdfs:comment>Questa proprietà specifica la massima quantità di
            memoria.
      </rdfs:comment>
      <rdfs:domain>
          <daml:Class>
```

```
            <daml:unionOf>
               <daml:List>
                  <daml:first>
                     <daml:Class
                          rdf:about="&ontology;#MemoryDescription"/>
          </daml:first>
                  <daml:rest>
                     <daml:List>
                        <daml:first>
                           <daml:Class
                              rdf:about="&ontology;#HwPrecondition"/>
                 </daml:first>
                          <daml:rest>
                             <daml:nil/>
                          </daml:rest>
                       </daml:List>
                    </daml:rest>
                 </daml:List>
              </daml:unionOf>
          </daml:Class>
      </rdfs:domain>
      <rdfs:range  rdf:resource="&ontology;#MemoryTypeDescription"/>
   </daml:ObjectProperty>

   <daml:Class rdf:about="&ontology;#MemoryTypeDescription">
      <rdfs:label>MemoryTypeDescription</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
              essere usata per definire la quantità di memoria, l'unità
              di misura e il tipo di utilizzo.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#memoryamount"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#memoryunit"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty
                     rdf:resource="&ontology;#memoryusagetype"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

   <daml:DatatypeProperty rdf:about="&ontology;#memoryamount">
      <rdfs:comment>Questa proprietà specifica la quantità di
              memoria.Questo valore deve essere un numero non negativo.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#MemoryTypeDescription"/>
      <rdfs:range  rdf:resource="&xsd;#integer"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#memoryunit">
      <rdfs:comment>Questa proprietà specifica l'unità di misura per la
              memoria.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#MemoryTypeDescription"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>
```

```
<daml:DatatypeProperty rdf:about="&ontology;#memoryusagetype">
   <rdfs:comment>Questa proprietà specifica il tipo di utilizzo
           della memoria.Può assumere i valori Application o
           Storage.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#MemoryTypeDescription"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:about="&ontology;#swproperties">
   <rdfs:comment>Questa proprietà fornisce la lista di
           caratteristiche software di un device.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Device"/>
   <rdfs:range  rdf:resource="&ontology;#SwDescription"/>
</daml:ObjectProperty>

<daml:Class rdf:about="&ontology;#SwDescription">
   <rdfs:label>SwDescription</rdfs:label>
   <rdfs:comment>Questa classe rappresenta la descrizione che può
           essere usata per definire le caratteristiche software di
           un device.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#os"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:about="&ontology;#os">
   <rdfs:comment>Questa proprietà fornisce le caratteristiche del
           sistema operativo di un device.
   </rdfs:comment>
   <rdfs:domain>
      <daml:Class>
         <daml:unionOf>
            <daml:List>
               <daml:first>
                  <daml:Class rdf:about="&ontology;#SwDescription"/>
            </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
                         <daml:Class
                             rdf:about="&ontology;#SwPrecondition"/>
               </daml:first>
                      <daml:rest>
                         <daml:nil/>
                      </daml:rest>
                    </daml:List>
              </daml:rest>
           </daml:List>
         </daml:unionOf>
       </daml:Class>
   </rdfs:domain>
   <rdfs:range  rdf:resource="&ontology;#InfoDescription"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#application">
   <rdfs:comment>Questa proprietà fornisce le caratteristiche delle
           applicazioni di un device.
   </rdfs:comment>
```

```
      <rdfs:domain>
          <daml:Class>
             <daml:unionOf>
                <daml:List>
                   <daml:first>
                     <daml:Class
                           rdf:about="&ontology;#SwDescription"/>
           </daml:first>
                   <daml:rest>
                      <daml:List>
                         <daml:first>
                            <daml:Class
                               rdf:about="&ontology;#SwPrecondition"/>
                 </daml:first>
                         <daml:rest>
                             <daml:nil/>
                         </daml:rest>
                      </daml:List>
                   </daml:rest>
                </daml:List>
             </daml:unionOf>
          </daml:Class>
      </rdfs:domain>
      <rdfs:range  rdf:resource="&ontology;#InfoDescription"/>
   </daml:ObjectProperty>

<!--  User   -->

   <daml:Class rdf:about="&ontology;#User">
      <rdfs:label>User</rdfs:label>
      <rdfs:comment>Sottoclasse di Entity, rappresenta tutti i
              possibili utenti dell'ambiente pervasivo.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Class rdf:about="&ontology;#Entity"/>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#userName"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#userSurname"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:minCardinality="1">
             <daml:onProperty rdf:resource="&ontology;#userEmail"/>
          </daml:Restriction>
      </rdfs:subClassOf>
    </daml:Class>

   <daml:DatatypeProperty rdf:about="&ontology;#userName">
      <rdfs:comment>Questa proprietà fornisce il nome dell'utente.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#User"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#userSurname">
      <rdfs:comment>Questa proprietà fornisce il cognome dell'utente.
      </rdfs:comment>
```

```
      <rdfs:domain rdf:resource="&ontology;#User"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#userEmail">
      <rdfs:comment>Questa proprietà fornisce l'indirizzo di posta
              elettronica dell'utente.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#User"/>
     <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

<!--  Resource   -->

   <daml:Class rdf:about="&ontology;#Resource">
      <rdfs:label>Resource</rdfs:label>
      <rdfs:comment>Sottoclasse della classe Entity,rappresenta tutte
              le risorse che forniscono una qualche forma di servizio.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Class rdf:about="&ontology;#Entity"/>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:minCardinality="1">
             <daml:onProperty rdf:resource="&ontology;#provides"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#offeredBy"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#resourceName"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#resourceUrl"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#resourcePageUrl"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
              <daml:onProperty
         rdf:resource="&ontology;#resourceTextDescription"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

   <daml:DatatypeProperty rdf:about="&ontology;#resourceName">
      <rdfs:comment>Questa proprietà fornisce il nome della risorsa.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#Resource"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

   <daml:DatatypeProperty rdf:about="&ontology;#resourceUrl">
```

```
      <rdfs:comment>Questa proprietà fornisce l'Url della risorsa.
      </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Resource"/>
     <rdfs:range  rdf:resource="&xsd;#string"/>
  </daml:DatatypeProperty>

  <daml:DatatypeProperty rdf:about="&ontology;#resourcePageUrl">
     <rdfs:comment>Questa proprietà fornisce l'Url di una pagina
             attraverso la quale essere utilizzata la risorsa.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Resource"/>
     <rdfs:range  rdf:resource="&xsd;#string"/>
  </daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#resourceTextDescription">
     <rdfs:comment>Questa proprietà fornisce una descrizione testuale
             della risorsa.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Resource"/>
     <rdfs:range  rdf:resource="&xsd;#string"/>
  </daml:DatatypeProperty>

  <daml:ObjectProperty rdf:about="&ontology;#offers">
     <rdfs:comment>Un device fornisce una risorsa.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Device"/>
     <rdfs:range rdf:resource="&ontology;#Resource"/>
     <daml:inverseOf rdf:resource="&ontology;#offeredBy"/>
  </daml:ObjectProperty>

  <daml:ObjectProperty rdf:about="&ontology;#offeredBy">
     <rdfs:comment>Una risorsa è fornita da un device.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Resource"/>
     <rdfs:range rdf:resource="&ontology;#Device"/>
     <daml:inverseOf rdf:resource="&ontology;#offers"/>
  </daml:ObjectProperty>

  <daml:ObjectProperty rdf:about="&ontology;#provides">
     <rdfs:comment>Una risorsa, come un sito Web, fornisce un
             servizio.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Resource"/>
     <rdfs:range rdf:resource="&ontology;#Service"/>
     <daml:inverseOf rdf:resource="&ontology;#providedBy"/>
  </daml:ObjectProperty>

  <daml:ObjectProperty rdf:about="&ontology;#providedBy">
     <rdfs:comment>Un servizio è fornito da una risorsa, come un sito
        Web.
     </rdfs:comment>
     <rdfs:domain rdf:resource="&ontology;#Service"/>
     <rdfs:range rdf:resource="&ontology;#Resource"/>
     <daml:inverseOf rdf:resource="&ontology;#provides"/>
  </daml:ObjectProperty>

<!-- Service -->

  <daml:Class rdf:about="&ontology;#Service">
     <rdfs:label>Service</rdfs:label>
     <rdfs:comment>Ogni servizio presenta ServiceProfile ed è fornito
             da una o più risorse.
     </rdfs:comment>
```

```
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#presents"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:minCardinality="1">
             <daml:onProperty rdf:resource="&ontology;#providedBy"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

<!-- Service Profile -->

   <daml:Class rdf:about="&ontology;#ServiceProfile">
      <rdfs:label>ServiceProfile</rdfs:label>
      <rdfs:comment>Il service profile indica ciò che fa il servizio;
              cioè da il tipo di informazioni di cui hai bisogno un
              agente che cerca un servizio per determinare se il
              servizio soddisfa le sue richieste.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
             <daml:onProperty rdf:resource="&ontology;#presentedBy"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

   <daml:ObjectProperty rdf:about="&ontology;#presents">
      <rdfs:comment>Ogni servizio presenta un ServiceProfile.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#Service"/>
      <rdfs:range rdf:resource="&ontology;#ServiceProfile"/>
      <daml:inverseOf rdf:resource="&ontology;#presentedBy"/>
   </daml:ObjectProperty>

   <daml:ObjectProperty rdf:about="&ontology;#presentedBy">
      <rdfs:comment>Un ServiceProfile è presentato da un service.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ServiceProfile"/>
      <rdfs:range rdf:resource="&ontology;#Service"/>
      <daml:inverseOf rdf:resource="&ontology;#presents"/>
   </daml:ObjectProperty>

<!--  Profile    -->

   <daml:Class rdf:about="&ontology;#Profile">
      <daml:label>Profile</daml:label>
      <rdfs:subClassOf rdf:resource="&ontology;#ServiceProfile" />
      <rdfs:comment>Questa proprietà fornisce un Profile (si usa
              Profile e non ServiceProfile per mantenere compatibilità
              con Daml-s).
      </rdfs:comment>
      <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="&ontology;#serviceName"/>
         </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
           <daml:onProperty rdf:resource="&ontology;#textDescription"/>
         </daml:Restriction>
       </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="&ontology;#hwPrecondition"/>
       </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="&ontology;#swPrecondition"/>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:DatatypeProperty rdf:about="&ontology;#serviceName">
   <rdfs:comment>Questa proprietà fornisce il nome del servizio.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Profile"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:about="&ontology;#textDescription">
   <rdfs:comment>Questa proprietà fornisce una descrizione testuale
           del servizio.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Profile"/>
   <rdfs:range  rdf:resource="&xsd;#string"/>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:about="&ontology;#parameter">
  <rdfs:comment>Questa proprietà fornisce i parametri associati al
           servizio.I paramtri possono essere input, output,
           precondizioni, effetti.
  </rdfs:comment>
  <rdfs:domain rdf:resource="&ontology;#Profile"/>
  <rdfs:range  rdf:resource="&ontology;#ParameterDescription"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#input">
   <rdfs:subPropertyOf rdf:resource="&ontology;#parameter" />
   <rdfs:comment>Un input è un tipo di parametro ed è una proprietà
           di profile.Gli input descrivono le informazioni attese
           dal servizio.
   </rdfs:comment>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#output">
   <rdfs:subPropertyOf rdf:resource="&ontology;#parameter" />
   <rdfs:comment>Un output è un tipo di parametro ed è una proprietà
           di profile.Gli output descrivono le informazioni generate
           dal servizio.
   </rdfs:comment>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#hwPrecondition">
   <rdfs:comment>Questa proprietà fornisce le precondizioni hardware
           da soddisfare per poter usufruire del servizio.
   </rdfs:comment>
   <rdfs:domain rdf:resource="&ontology;#Profile"/>
   <rdfs:range  rdf:resource="&ontology;#HwPrecondition"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#swPrecondition">
   <rdfs:comment>Questa proprietà fornisce le precondizioni software
           da soddisfare per poter usufruire del servizio.
```

```
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#Profile"/>
      <rdfs:range  rdf:resource="&ontology;#SwPrecondition"/>
   </daml:ObjectProperty>

   <daml:Class rdf:about="&ontology;#HwPrecondition">
      <rdfs:label>HwPrecondition</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
              essere usata per definire le caratteristiche hardware che
              un device deve avere per poter usufruire di un servizio.
      </rdfs:comment>
   </daml:Class>

   <daml:Class rdf:about="&ontology;#SwPrecondition">
      <rdfs:label>SwPrecondition</rdfs:label>
      <rdfs:comment>Questa classe rappresenta la descrizione che può
              essere usata per definire le caratteristiche software che
              un device deve avere per poter usufruire di un servizio.
      </rdfs:comment>
   </daml:Class>

   <daml:DatatypeProperty rdf:about="&ontology;#parameterName">
      <rdfs:comment>Questa proprietà fornisce il nome del parametro.
      </rdfs:comment>
      <rdfs:domain rdf:resource="&ontology;#ParameterDescription"/>
      <rdfs:range  rdf:resource="&xsd;#string"/>
   </daml:DatatypeProperty>

   <daml:ObjectProperty rdf:about="&ontology;#restrictedTo">
      <rdfs:domain rdf:resource="&ontology;#ParameterDescription"/>
      <rdfs:comment>Questa proprietà fornisce il range di valori
              ammesso dal parametro.
      </rdfs:comment>
   </daml:ObjectProperty>

   <daml:Class rdf:about="&ontology;#ParameterDescription">
      <rdfs:comment>Questa classe rappresenta la descrizione che può
              essere associata ad un parametro di un servizio.
      </rdfs:comment>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#parameterName"/>
          </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
          <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#restrictedTo"/>
          </daml:Restriction>
      </rdfs:subClassOf>
   </daml:Class>

</rdf:RDF>
```

A.2 Ontologia per le informazioni di contesto

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
   <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
```

```
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://www.ubinet.it/contextOntology.daml">
  <!ENTITY DEFAULT "http://www.ubinet.it/contextOntology.daml">]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd      =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">

  <daml:Ontology rdf:about="">
     <dc:title>&quot;Ontology&quot;</dc:title>
     <dc:date>15/03/2004</dc:date>
     <dc:creator>Esposito Massimo</dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
             descrivere un possibile contesto per un ambiente
             pervasivo.</dc:description>
     <dc:subject>Un&apos;ontologia di contesto.</dc:subject>
     <daml:versionInfo>2.0</daml:versionInfo>
  </daml:Ontology>

  <daml:Class rdf:about="&ontology;#Place">
     <rdfs:label>Place</rdfs:label>
     <rdfs:comment>Place è la classe che modella tutti i possibili
                ambienti.
     </rdfs:comment>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="1">
            <daml:onProperty rdf:resource="&ontology;#placeName"/>
         </daml:Restriction>
     </rdfs:subClassOf>
  </daml:Class>

  <daml:Class rdf:about="&ontology;#OpenPlace">
     <rdfs:label>OpenPlace</rdfs:label>
     <rdfs:comment>Un OpenPlace è un Place che non contiene nè è
             contenuto da altri Place.
     </rdfs:comment>
     <rdfs:subClassOf>
         <daml:Class rdf:about="&ontology;#Place"/>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="0">
            <daml:onProperty rdf:resource="&ontology;#contains"/>
         </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="0">
            <daml:onProperty rdf:resource="&ontology;#isContainedIn"/>
         </daml:Restriction>
     </rdfs:subClassOf>
  </daml:Class>

  <daml:Class rdf:about="&ontology;#CompoundPlace">
     <rdfs:label>CompoundPlace</rdfs:label>
     <rdfs:comment>Un CompoundPlace è un Place che contiene un
             AtomicPlace o un CompoundPlace ed è contenuto da un
             CompoundPlace.
     </rdfs:comment>
```

```
    <rdfs:subClassOf>
        <daml:Class rdf:about="&ontology;#Place"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <daml:Restriction daml:minCardinalityQ="1">
           <daml:onProperty rdf:resource="&ontology;#contains"/>
              <daml:hasClassQ>
                 <daml:Class>
                    <daml:unionOf>
                       <daml:List>
                          <daml:first>
                             <daml:Class
                                rdf:about="&ontology;#AtomicPlace"/>
                           </daml:first>
                           <daml:rest>
                              <daml:List>
                                 <daml:first>
                                    <daml:Class
                                  rdf:about="&ontology;#CompoundPlace"/>
                                  </daml:first>
                                  <daml:rest>
                                     <daml:nil/>
                                  </daml:rest>
                                </daml:List>
                             </daml:rest>
                          </daml:List>
                       </daml:unionOf>
                 </daml:Class>
              </daml:hasClassQ>
        </daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <daml:Restriction daml:maxCardinalityQ="1">
          <daml:onProperty
                    rdf:resource="&ontology;#isContainedIn"/>
             <daml:hasClassQ>
                    <daml:Class
rdf:about="&ontology;#CompoundPlace"/>
             </daml:hasClassQ>
        </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>

  <daml:Class rdf:about="&ontology;#AtomicPlace">
     <rdfs:label>AtomicPlace</rdfs:label>
     <rdfs:comment>Un AtomicPlace è un Place che è contenuto da un
             CompoundPlace e che non contiene nessun Place.
     </rdfs:comment>
     <rdfs:subClassOf>
         <daml:Class rdf:about="&ontology;#Place"/>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinalityQ="1">
            <daml:onProperty rdf:resource="&ontology;#isContainedIn"/>
               <daml:hasClassQ>
                  <daml:Class rdf:about="&ontology;#CompoundPlace"/>
               </daml:hasClassQ>
         </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
         <daml:Restriction daml:cardinality="0">
            <daml:onProperty rdf:resource="&ontology;#contains"/>
         </daml:Restriction>
```

```
   </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="&ontology;#Building">
   <rdfs:label>Building</rdfs:label>
   <rdfs:comment>Un Building è un CompoundPlace che contiene Rooms.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Class rdf:about="&ontology;#CompoundPlace"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:minCardinalityQ="1">
          <daml:onProperty rdf:resource="&ontology;#contains"/>
             <daml:hasClassQ>
                <daml:Class rdf:about="&ontology;#Room"/>
             </daml:hasClassQ>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty
                rdf:resource="&ontology;#buildingAddress"/>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty
                rdf:resource="&ontology;#numberOfFloors"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="&ontology;#Room">
   <rdfs:label>Room</rdfs:label>
   <rdfs:comment>Una Room è un AtomicPlace contenuto in un Building.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Class rdf:about="&ontology;#AtomicPlace"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinalityQ="1">
            <daml:onProperty
                  rdf:resource="&ontology;#isContainedIn"/>
              <daml:hasClassQ>
                 <daml:Class rdf:about="&ontology;#Building"/>
              </daml:hasClassQ>
       </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinality="1">
          <daml:onProperty rdf:resource="&ontology;#floorNumber"/>
       </daml:Restriction>
   </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="&ontology;#Device">
   <rdfs:label>Device</rdfs:label>
   <rdfs:comment>Un device è associato ad una stanza.
   </rdfs:comment>
   <rdfs:subClassOf>
       <daml:Restriction daml:cardinalityQ="1">
          <daml:onProperty rdf:resource="&ontology;#isPlacedIn"/>
             <daml:hasClassQ>
```

```
                <daml:Class rdf:about="&ontology;#Room"/>
             </daml:hasClassQ>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="&ontology;#Resource">
    <rdfs:label>Resource</rdfs:label>
    <rdfs:comment>Una risorsa è associata a più stanze.
    </rdfs:comment>
    <rdfs:subClassOf>
        <daml:Restriction daml:minCardinalityQ="1">
           <daml:onProperty rdf:resource="&ontology;#isAvailableIn"/>
              <daml:hasClassQ>
                 <daml:Class rdf:about="&ontology;#Room"/>
              </daml:hasClassQ>
        </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="&ontology;#User">
    <rdfs:label>User</rdfs:label>
    <rdfs:comment>Un utente può trovarsi al più in un Place in un
            preciso istante.
    </rdfs:comment>
    <rdfs:subClassOf>
        <daml:Restriction daml:cardinalityQ="1">
           <daml:onProperty rdf:resource="&ontology;#locatedUser"/>
              <daml:hasClassQ>
                 <daml:Class rdf:about="&ontology;#Place"/>
              </daml:hasClassQ>
           </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>

<daml:ObjectProperty rdf:about="&ontology;#contains">
    <rdfs:label>contains</rdfs:label>
    <rdfs:comment>Associa ad un Place quelli che esso contiene.
    </rdfs:comment>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#CompoundPlace"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class>
           <daml:unionOf>
              <daml:List>
                 <daml:first>
                    <daml:Class rdf:about="&ontology;#AtomicPlace"/>
                 </daml:first>
                 <daml:rest>
                    <daml:List>
                       <daml:first>
                          <daml:Class
                                rdf:about="&ontology;#CompoundPlace"/>
                       </daml:first>
                       <daml:rest>
                          <daml:nil/>
                       </daml:rest>
                    </daml:List>
                 </daml:rest>
              </daml:List>
           </daml:unionOf>
        </daml:Class>
```

```
   </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#isContainedIn">
   <rdfs:label>isContainedIn</rdfs:label>
   <rdfs:comment>Associa ad un Place quello in cui esso è contenuto.
   </rdfs:comment>
   <rdfs:domain>
       <daml:Class>
          <daml:unionOf>
             <daml:List>
                <daml:first>
                   <daml:Class rdf:about="&ontology;#AtomicPlace"/>
                </daml:first>
                <daml:rest>
                   <daml:List>
                      <daml:first>
                         <daml:Class
                               rdf:about="&ontology;#CompoundPlace"/>
                      </daml:first>
                      <daml:rest>
                         <daml:nil/>
                      </daml:rest>
                   </daml:List>
                </daml:rest>
             </daml:List>
          </daml:unionOf>
       </daml:Class>
   </rdfs:domain>
   <rdfs:range>
       <daml:Class rdf:about="&ontology;#CompoundPlace"/>
   </rdfs:range>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:about="&ontology;#placeName">
   <rdfs:label>placeName</rdfs:label>
   <rdfs:comment>Associa un nome ad un Place.
   </rdfs:comment>
   <rdfs:domain>
       <daml:Class rdf:about="&ontology;#Place"/>
   </rdfs:domain>
   <rdfs:range>
       <xsd:string/>
   </rdfs:range>
</daml:DatatypeProperty>

<daml:ObjectProperty rdf:about="&ontology;#availableResource">
   <rdfs:label>availableResource</rdfs:label>
   <rdfs:comment>Associa tutte le risorse alla stanza in cui sono
           disponibili.
   </rdfs:comment>
   <daml:inverseOf rdf:resource="&ontology;#isAvailableIn"/>
   <rdfs:domain>
       <daml:Class rdf:about="&ontology;#Room"/>
   </rdfs:domain>
   <rdfs:range>
       <daml:Class rdf:about="&ontology;#Resource"/>
   </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#locatedUser">
   <rdfs:label>locatedUser</rdfs:label>
   <rdfs:comment>Associa ad una stanza gli utenti che esso contiene.
```

```
    </rdfs:comment>
    <daml:inverseOf rdf:resource="&ontology;#isLocatedIn"/>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#Place"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class rdf:about="&ontology;#User"/>
    </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#placedDevice">
    <rdfs:label>placedDevice</rdfs:label>
    <rdfs:comment>Associa ad una stanza tutti i dispositivi che essa
            contiene.
    </rdfs:comment>
    <daml:inverseOf rdf:resource="&ontology;#isPlacedIn"/>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#Room"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class rdf:about="&ontology;#Device"/>
    </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#isLocatedIn">
    <rdfs:label>isLocatedIn</rdfs:label>
    <rdfs:comment>Fornisce la posizione di un utente.
    </rdfs:comment>
    <daml:inverseOf rdf:resource="&ontology;#locatedUser"/>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#User"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class rdf:about="&ontology;#Place"/>
    </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#isAvailableIn">
    <rdfs:label>isAvailableIn</rdfs:label>
    <rdfs:comment>Associa ad una risorsa le stanze dove è possibile
            utilizzarla.
    </rdfs:comment>
    <daml:inverseOf rdf:resource="&ontology;#availableResource"/>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#Resource"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class rdf:about="&ontology;#Room"/>
    </rdfs:range>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:about="&ontology;#isPlacedIn">
    <rdfs:label>isPlacedIn</rdfs:label>
    <rdfs:comment>Associa ad un dispositivo la stanza dove esso è
            posizionato.
    </rdfs:comment>
    <daml:inverseOf rdf:resource="&ontology;#placedDevice"/>
    <rdfs:domain>
        <daml:Class rdf:about="&ontology;#Device"/>
    </rdfs:domain>
    <rdfs:range>
        <daml:Class rdf:about="&ontology;#Room"/>
    </rdfs:range>
```

```
  </daml:ObjectProperty>

  <daml:DatatypeProperty rdf:about="&ontology;#floorNumber">
     <rdfs:label>floorNumber</rdfs:label>
     <rdfs:comment>Associa il numero di piano ad una stanza.
     </rdfs:comment>
     <rdfs:domain>
         <daml:Class rdf:about="&ontology;#Room"/>
     </rdfs:domain>
     <rdfs:range>
         <xsd:integer/>
     </rdfs:range>
  </daml:DatatypeProperty>

  <daml:DatatypeProperty rdf:about="&ontology;#buildingAddress">
     <rdfs:label>buildingAddress</rdfs:label>
     <rdfs:comment>Associa l'indirizzo ad un edificio.
     </rdfs:comment>
     <rdfs:domain>
         <daml:Class rdf:about="&ontology;#Building"/>
     </rdfs:domain>
     <rdfs:range>
         <xsd:string/>
     </rdfs:range>
  </daml:DatatypeProperty>

  <daml:DatatypeProperty rdf:about="&ontology;#numberOfFloors">
     <rdfs:label>numberOfFloors</rdfs:label>
     <rdfs:comment>Associa ad un edificio il numero dei piani di cui è
             composto.
     </rdfs:comment>
     <rdfs:domain>
         <daml:Class rdf:about="&ontology;#Building"/>
     </rdfs:domain>
     <rdfs:range>
         <xsd:integer/>
     </rdfs:range>
  </daml:DatatypeProperty>

</rdf:RDF>
```

A.3 Descrizione di un Laptop

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://ubinet.it/Ontology.daml">
  <!ENTITY instance "http://ubinet.it/Ontology.daml#UbiLaptop">
  <!ENTITY DEFAULT "http://ubinet.it/Ontology.daml">]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd      =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
```

```
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">

  <daml:Ontology rdf:about="">
     <dc:title>&quot;Ontology&quot;</dc:title>
     <dc:date>20/12/2003</dc:date>
     <dc:creator>Esposito Massimo</dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
             definire le caratteristiche di un laptop in un
             ambiente pervasivo.
     </dc:description>
     <dc:subject>Un&apos;istanza di un laptop.</dc:subject>
     <daml:versionInfo>3.1</daml:versionInfo>
  </daml:Ontology>

 <!--
########################################################################## --
>
  <!-- # Instance Definition of a Laptop # -->
  <!--
########################################################################## --
>

  <rdf:Description rdf:about="&instance;">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Device"/>
     </rdf:type>
     <ontology:info rdf:resource="&instance;_InfoDescription"/>
     <ontology:hwproperties rdf:resource="&instance;_HwDescription"/>
     <ontology:swproperties rdf:resource="&instance;_SwDescription"/>
     <ontology:devicetype>
      <xsd:string xsd:value="Laptop"/>
     </ontology:devicetype>
  </rdf:Description>

  <rdf:Description rdf:about="&instance;_InfoDescription">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#InfoDescription"/>
     </rdf:type>
     <ontology:infoname>
      <xsd:string xsd:value="Massimo"/>
     </ontology:infoname>
     <ontology:infovendor>
      <xsd:string xsd:value="Toshiba"/>
     </ontology:infovendor>
     <ontology:infoversion>
      <xsd:string xsd:value="Satellite 1400-103"/>
     </ontology:infoversion>
  </rdf:Description>

  <rdf:Description rdf:about="&instance;_HwDescription">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#HwDescription"/>
     </rdf:type>
     <ontology:connection
             rdf:resource="&instance;_ConnectionDescription1"/>
     <ontology:connection
             rdf:resource="&instance;_ConnectionDescription2"/>
     <ontology:ui rdf:resource="&instance;_UiDescription"/>
     <ontology:memory rdf:resource="&instance;_MemoryDescription1"/>
     <ontology:memory rdf:resource="&instance;_MemoryDescription2"/>
     <ontology:cpu rdf:resource="&instance;_CpuDescription"/>
  </rdf:Description>
```

```
<rdf:Description rdf:about="&instance;_CpuDescription">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#CpuDescription"/>
   </rdf:type>
   <ontology:cpuinfo>
         <rdf:Description
                  rdf:about="&instance;_Info_CpuDescription">
            <rdf:type>
              <daml:Class rdf:about="&ontology;#InfoDescription"/>
            </rdf:type>
            <ontology:infoname>
             <xsd:string xsd:value="Intel Celeron"/>
            </ontology:infoname>
         </rdf:Description>
   </ontology:cpuinfo>
   <ontology:cpuspeed>
     <xsd:real xsd:value="1.33"/>
   </ontology:cpuspeed>
   <ontology:cpuunit>
    <xsd:string xsd:value="GHz"/>
   </ontology:cpuunit>
</rdf:Description>

<rdf:Description rdf:about="&instance;_ConnectionDescription1">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#ConnectionDescription"/>
   </rdf:type>
   <ontology:information>
         <rdf:Description
             rdf:about="&instance;_Info_ConnectionDescription1">
            <rdf:type>
              <daml:Class rdf:about="&ontology;#InfoDescription"/>
            </rdf:type>
            <ontology:infoname>
             <xsd:string xsd:value="Lan Ethernet"/>
            </ontology:infoname>
         </rdf:Description>
   </ontology:information>
   <ontology:bandwidth>
     <xsd:real xsd:value="100"/>
   </ontology:bandwidth>
   <ontology:bandwidthunit>
     <xsd:string xsd:value="Mbit/s"/>
   </ontology:bandwidthunit>
</rdf:Description>

<rdf:Description rdf:about="&instance;_ConnectionDescription2">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#ConnectionDescription"/>
   </rdf:type>
   <ontology:information>
         <rdf:Description
              rdf:about="&instance;_Info_ConnectionDescription2">
            <rdf:type>
              <daml:Class rdf:about="&ontology;#InfoDescription"/>
            </rdf:type>
            <ontology:infoname>
             <xsd:string xsd:value="Modem56K"/>
            </ontology:infoname>
      </rdf:Description>
   </ontology:information>
   <ontology:bandwidth>
```

```
      <xsd:real xsd:value="56"/>
   </ontology:bandwidth>
   <ontology:bandwidthunit>
      <xsd:string xsd:value="Kbit/s"/>
   </ontology:bandwidthunit>
</rdf:Description>

<rdf:Description rdf:about="&instance;_UiDescription">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#UiDescription"/>
   </rdf:type>
   <ontology:screen rdf:resource="&instance;_ScreenDescription"/>
   <ontology:audio-output>
      <xsd:boolean xsd:value="true"/>
   </ontology:audio-output>
   <ontology:audio-input>
      <xsd:boolean xsd:value="false"/>
   </ontology:audio-input>
   <ontology:video-input>
      <xsd:boolean xsd:value="false"/>
   </ontology:video-input>
</rdf:Description>

<rdf:Description rdf:about="&instance;_ScreenDescription">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#ScreenDescription"/>
   </rdf:type>
   <ontology:resolution
          rdf:resource="&instance;_ResolutionDescription"/>
   <ontology:width>
      <xsd:real xsd:value="28,25"/>
   </ontology:width>
   <ontology:height>
      <xsd:real xsd:value="22"/>
   </ontology:height>
   <ontology:unit>
      <xsd:string xsd:value="cm"/>
   </ontology:unit>
   <ontology:color>
      <xsd:boolean xsd:value="true"/>
   </ontology:color>
</rdf:Description>

<rdf:Description rdf:about="&instance;_ResolutionDescription">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#ResolutionDescription"/>
   </rdf:type>
   <ontology:resolutionwidth>
      <xsd:integer xsd:value="1024"/>
   </ontology:resolutionwidth>
   <ontology:resolutionheight>
      <xsd:integer xsd:value="768"/>
   </ontology:resolutionheight>
   <ontology:resolutionunit>
      <xsd:string xsd:value="pixel"/>
   </ontology:resolutionunit>
   <ontology:resolutionbpp>
      <xsd:integer xsd:value="32"/>
   </ontology:resolutionbpp>
   <ontology:resolutiongraphics>
      <xsd:boolean xsd:value="true"/>
   </ontology:resolutiongraphics>
</rdf:Description>
```

```
<rdf:Description rdf:about="&instance;_MemoryDescription1">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#MemoryDescription"/>
   </rdf:type>
   <ontology:available
           rdf:resource="&instance;_MemoryTypeDescription1"/>
   <ontology:maximum
           rdf:resource="&instance;_MemoryTypeDescription2"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_MemoryDescription2">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#MemoryDescription"/>
   </rdf:type>
   <ontology:maximum
           rdf:resource="&instance;_MemoryTypeDescription3"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_MemoryTypeDescription1">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#MemoryTypeDescription"/>
   </rdf:type>
   <ontology:memoryamount>
      <xsd:integer xsd:value="6"/>
   </ontology:memoryamount>
   <ontology:memoryunit>
      <xsd:string xsd:value="Gb"/>
   </ontology:memoryunit>
   <ontology:memoryusagetype>
      <xsd:string xsd:value="Storage"/>
   </ontology:memoryusagetype>
</rdf:Description>

<rdf:Description rdf:about="&instance;_MemoryTypeDescription2">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#MemoryTypeDescription"/>
   </rdf:type>
   <ontology:memoryamount>
      <xsd:integer xsd:value="20"/>
   </ontology:memoryamount>
   <ontology:memoryunit>
      <xsd:string xsd:value="Gb"/>
   </ontology:memoryunit>
   <ontology:memoryusagetype>
      <xsd:string xsd:value="Storage"/>
   </ontology:memoryusagetype>
</rdf:Description>

<rdf:Description rdf:about="&instance;_MemoryTypeDescription3">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#MemoryTypeDescription"/>
   </rdf:type>
   <ontology:memoryamount>
      <xsd:integer xsd:value="128"/>
   </ontology:memoryamount>
   <ontology:memoryunit>
      <xsd:string xsd:value="Mb"/>
   </ontology:memoryunit>
   <ontology:memoryusagetype>
      <xsd:string xsd:value="Application"/>
   </ontology:memoryusagetype>
</rdf:Description>
```

```
<rdf:Description rdf:about="&instance;_SwDescription">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#SwDescription"/>
   </rdf:type>
   <ontology:os>
      <rdf:Description rdf:about="&instance;_OsDescription">
        <rdf:type>
          <daml:Class rdf:about="&ontology;#InfoDescription"/>
        </rdf:type>
        <ontology:infoname>
           <xsd:string xsd:value="Windows XP"/>
        </ontology:infoname>
        <ontology:infovendor>
           <xsd:string xsd:value="Microsoft"/>
        </ontology:infovendor>
        <ontology:infoversion>
           <xsd:string xsd:value="Home"/>
        </ontology:infoversion>
      </rdf:Description>
  </ontology:os>
  <ontology:application>
      <rdf:Description
           rdf:about="&instance;_ApplicationDescription">
        <rdf:type>
          <daml:Class rdf:about="&ontology;#InfoDescription"/>
        </rdf:type>
        <ontology:infoname>
           <xsd:string xsd:value="Windows Media Player"/>
        </ontology:infoname>
        <ontology:infovendor>
           <xsd:string xsd:value="Microsoft"/>
        </ontology:infovendor>
        <ontology:infoversion>
           <xsd:string xsd:value="8.0"/>
        </ontology:infoversion>
      </rdf:Description>
  </ontology:application>
</rdf:Description>

<daml:DatatypeProperty rdf:about="&ontology;#devicetype"/>
<daml:DatatypeProperty rdf:about="&ontology;#infoname"/>
<daml:DatatypeProperty rdf:about="&ontology;#infovendor"/>
<daml:DatatypeProperty rdf:about="&ontology;#infoversion"/>
<daml:DatatypeProperty rdf:about="&ontology;#audio-input"/>
<daml:DatatypeProperty rdf:about="&ontology;#audio-output"/>
<daml:DatatypeProperty rdf:about="&ontology;#video-input"/>
<daml:DatatypeProperty rdf:about="&ontology;#width"/>
<daml:DatatypeProperty rdf:about="&ontology;#height"/>
<daml:DatatypeProperty rdf:about="&ontology;#unit"/>
<daml:DatatypeProperty rdf:about="&ontology;#color"/>
<daml:DatatypeProperty rdf:about="&ontology;#resolutionwidth"/>
<daml:DatatypeProperty rdf:about="&ontology;#resolutionheight"/>
<daml:DatatypeProperty rdf:about="&ontology;#resolutionunit"/>
<daml:DatatypeProperty rdf:about="&ontology;#resolutionbpp"/>
<daml:DatatypeProperty rdf:about="&ontology;#resolutiongraphics"/>
<daml:DatatypeProperty rdf:about="&ontology;#memoryamount"/>
<daml:DatatypeProperty rdf:about="&ontology;#cpuspeed"/>
<daml:DatatypeProperty rdf:about="&ontology;#cpuunit"/>
<daml:DatatypeProperty rdf:about="&ontology;#memoryunit"/>
<daml:DatatypeProperty rdf:about="&ontology;#memoryusagetype"/>
<daml:DatatypeProperty rdf:about="&ontology;#bandwidth"/>
<daml:DatatypeProperty rdf:about="&ontology;#bandwidthunit"/>
```

```
</rdf:RDF>
```

A.4 Descrizione di un PdfViewer

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://www.ubinet.it/Ontology.daml">
  <!ENTITY instance "http://www.ubinet.it/Ontology.daml#UbiPdfViewer">
  <!ENTITY DEFAULT "http://www.ubinet.it/Ontology.daml">]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd       =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">

  <daml:Ontology rdf:about="">
     <dc:title>&quot;Ontology&quot;</dc:title>
     <dc:date>07/03/2004</dc:date>
     <dc:creator>Esposito Massimo</dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
              definire una risorsa Web che ha funzioni di Viewer
             di file pdf in un ambiente pervasivo.
     </dc:description>
     <dc:subject>Un&apos;istanza di un PdfViewer.</dc:subject>
     <daml:versionInfo>5.1</daml:versionInfo>
  </daml:Ontology>

  <!--
########################################################################## --
>
  <!-- # Instance Definition of a PdfViewer # -->
  <!--
########################################################################## --
>

  <rdf:Description rdf:about="&instance;">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Resource"/>
     </rdf:type>
     <ontology:resourceName>
      <xsd:string xsd:value="UbiPdfViewer"/>
     </ontology:resourceName>
     <ontology:resourceUrl>
         <xsd:string
            xsd:value="http://www.ubinet.it/UbiPdfViewer"/>
     </ontology:resourceUrl>
     <ontology:resourcePageUrl>
         <xsd:string
            xsd:value="http://www.ubinet.it/ UbiPdfViewer/Page "/>
     </ontology:resourcePageUrl>
```

```
    <ontology:resourceTextDescription>
      <xsd:string xsd:value="Questa risorsa consente di eseguire un
            file pdf."/>
    </ontology:resourceTextDescription>
    <ontology:provides rdf:resource="&instance;_view"/>
    <ontology:provides rdf:resource="&instance;_close"/>
    <ontology:provides rdf:resource="&instance;_firstpage"/>
    <ontology:provides rdf:resource="&instance;_lastpage"/>
    <ontology:provides rdf:resource="&instance;_lineup"/>
    <ontology:provides rdf:resource="&instance;_linedown"/>
    <ontology:provides rdf:resource="&instance;_fitpage"/>
    <ontology:provides rdf:resource="&instance;_fitheight"/>
    <ontology:provides rdf:resource="&instance;_fitwidth"/>
    <ontology:provides rdf:resource="&instance;_singlepage"/>
    <ontology:provides rdf:resource="&instance;_fullscreen"/>
    <ontology:provides rdf:resource="&instance;_fromfullscreen"/>
    <ontology:provides rdf:resource="&instance;_nextpage"/>
    <ontology:provides rdf:resource="&instance;_prevpage"/>
    <ontology:provides rdf:resource="&instance;_pageup"/>
    <ontology:provides rdf:resource="&instance;_pagedown"/>
    <ontology:provides rdf:resource="&instance;_print"/>
    <ontology:offeredBy rdf:resource="&ontology;#UbiLaptop"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_view">
    <rdf:type>
       <daml:Class rdf:about="&ontology;#Service"/>
    </rdf:type>
    <ontology:presents rdf:resource="&instance;_view_Profile"/>
    <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_print">
    <rdf:type>
       <daml:Class rdf:about="&ontology;#Service"/>
    </rdf:type>
    <ontology:presents rdf:resource="&instance;_print_Profile"/>
    <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_firstpage">
    <rdf:type>
       <daml:Class rdf:about="&ontology;#Service"/>
    </rdf:type>
    <ontology:presents rdf:resource="&instance;_firstpage_Profile"/>
    <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_lastpage">
    <rdf:type>
       <daml:Class rdf:about="&ontology;#Service"/>
    </rdf:type>
    <ontology:presents rdf:resource="&instance;_lastpage_Profile"/>
    <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_lineup">
    <rdf:type>
       <daml:Class rdf:about="&ontology;#Service"/>
    </rdf:type>
    <ontology:presents rdf:resource="&instance;_lineup_Profile"/>
    <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="&instance;_linedown">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_linedown_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_fitpage">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_fitpage_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_fitheight">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_fitheight_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_fitwidth">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_fitwidth_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_singlepage">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_singlepage"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_fullscreen">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_fullscreen_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_fromfullscreen">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents
           rdf:resource="&instance;_fromfullscreen_Profile"/>
   <ontology:providedBy rdf:resource="&instance;"/>
</rdf:Description>

<rdf:Description rdf:about="&instance;_close">
   <rdf:type>
      <daml:Class rdf:about="&ontology;#Service"/>
   </rdf:type>
   <ontology:presents rdf:resource="&instance;_close_Profile"/>
```

```
      <ontology:providedBy rdf:resource="&instance;"/>
   </rdf:Description>

   <rdf:Description rdf:about="&instance;_nextpage">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Service"/>
      </rdf:type>
      <ontology:presents rdf:resource="&instance;_nextpage_Profile"/>
      <ontology:providedBy rdf:resource="&instance;"/>
   </rdf:Description>

   <rdf:Description rdf:about="&instance;_prevpage">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Service"/>
      </rdf:type>
      <ontology:presents rdf:resource="&instance;_prevpage_Profile"/>
      <ontology:providedBy rdf:resource="&instance;"/>
   </rdf:Description>

   <rdf:Description rdf:about="&instance;_pageup">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Service"/>
      </rdf:type>
      <ontology:presents rdf:resource="&instance;_pageup_Profile"/>
      <ontology:providedBy rdf:resource="&instance;"/>
   </rdf:Description>

   <rdf:Description rdf:about="&instance;_pagedown">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Service"/>
      </rdf:type>
      <ontology:presents rdf:resource="&instance;_pagedown_Profile"/>
      <ontology:providedBy rdf:resource="&instance;"/>
   </rdf:Description>

<!--  UbiPdfViewer_view_Profile  -->

   <rdf:Description rdf:about="&instance;_view_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_view"/>
      <ontology:serviceName>
       <xsd:string xsd:value="view"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di aprire file
          pdf."/>
      </ontology:textDescription>

      <!-- Descriptions of IOs -->

      <ontology:input>
         <rdf:Description rdf:about="&instance;_view_input_1">
           <rdf:type>
             <daml:Class rdf:about="&ontology;#ParameterDescription"/>
           </rdf:type>
           <ontology:parameterName>
              <xsd:string xsd:value="Filename"/>
           </ontology:parameterName>
        <ontology:restrictedTo rdf:resource="&ontology;#String"/>
       </rdf:Description>
        </ontology:input>
```

```
  </rdf:Description>

<!--  UbiPdfViewer_print_Profile  -->

  <rdf:Description rdf:about="&instance;_print_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_print"/>
     <ontology:serviceName>
      <xsd:string xsd:value="print"/>
     </ontology:serviceName>
     <ontology:textDescription>
      <xsd:string xsd:value="Questo servizio consente di stampare file
       pdf."/>
     </ontology:textDescription>

     <!-- Descriptions of IOs -->

     <ontology:input>
        <rdf:Description rdf:about="&instance;_print_input_1">
          <rdf:type>
            <daml:Class rdf:about="&ontology;#ParameterDescription"/>
          </rdf:type>
          <ontology:parameterName>
             <xsd:string xsd:value="Filename"/>
          </ontology:parameterName>
       <ontology:restrictedTo rdf:resource="&ontology;#String"/>
     </rdf:Description>
     </ontology:input>
  </rdf:Description>

<!--  UbiPdfViewer_close_Profile  -->

  <rdf:Description rdf:about="&instance;_close_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_close"/>
     <ontology:serviceName>
        <xsd:string xsd:value="close"/>
     </ontology:serviceName>
     <ontology:textDescription>
       <xsd:string xsd:value="Questo servizio consente la chiusura di
       un file pdf aperto."/>
     </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_nextpage_Profile  -->

  <rdf:Description rdf:about="&instance;_nextpage_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_nextpage"/>
     <ontology:serviceName>
        <xsd:string xsd:value="nextpage"/>
     </ontology:serviceName>
     <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di
             visualizzare la pagina successiva a quella corrente
             all'interno di un dcoumento pdf."/>
     </ontology:textDescription>
```

```
  </rdf:Description>

<!--  UbiPdfViewer_prevpage_Profile  -->

  <rdf:Description rdf:about="&instance;_prevpage_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_prevpage"/>
     <ontology:serviceName>
      <xsd:string xsd:value="prevpage"/>
     </ontology:serviceName>
     <ontology:textDescription>
       <xsd:string xsd:value="Questo servizio consente di visualizzare
             la pagina precedente a quella corrente all'interno di un
             dcoumento pdf."/>
     </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_pageup_Profile  -->

  <rdf:Description rdf:about="&instance;_pageup_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_pageup"/>
     <ontology:serviceName>
      <xsd:string xsd:value="pageup"/>
     </ontology:serviceName>
     <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di spostarsi
             di una schermata verso l'alto all'interno della pagina
             corrente di un documento pdf."/>
     </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_pagedown_Profile  -->

  <rdf:Description rdf:about="&instance;_pagedown_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_pagedown"/>
     <ontology:serviceName>
      <xsd:string xsd:value="pagedown"/>
     </ontology:serviceName>
     <ontology:textDescription>
       <xsd:string xsd:value="Questo servizio consente di spostarsi di
             una schermata verso il basso all'interno della pagina
             corrente di un documento pdf."/>
     </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_firstpage_Profile  -->

  <rdf:Description rdf:about="&instance;_firstpage_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_firstpage"/>
     <ontology:serviceName>
      <xsd:string xsd:value="firstpage"/>
     </ontology:serviceName>
```

```
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di ritornare
              alla prima pagina di un documento pdf."/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_lastpage_Profile  -->

  <rdf:Description rdf:about="&instance;_lastpage_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_lastpage"/>
      <ontology:serviceName>
       <xsd:string xsd:value="lastpage"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di andare
              all'ultima pagina di un documento pdf."/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_lineup_Profile  -->

  <rdf:Description rdf:about="&instance;_lineup_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_lineup"/>
      <ontology:serviceName>
       <xsd:string xsd:value="lineup"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di spostarsi di
              una riga verso l'alto all'interno della pagina corrente
              di un documento pdf."/>
      </ontology:textDescription>
   </rdf:Description>

<!--  UbiPdfViewer_linedown_Profile  -->

  <rdf:Description rdf:about="&instance;_linedown_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_linedown"/>
      <ontology:serviceName>
       <xsd:string xsd:value="linedown"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di spostarsi di
              una riga verso il basso all'interno della pagina corrente
              di un documento pdf."/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_fitpage_Profile  -->

  <rdf:Description rdf:about="&instance;_fitpage_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_fitpage"/>
```

```
      <ontology:serviceName>
       <xsd:string xsd:value="fitpage"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente ridimensionare
              la pagina in modo da visualizzarla interamente nella
              finestra"/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_fitheight_Profile  -->

  <rdf:Description rdf:about="&instance;_fitheight_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_fitheight"/>
      <ontology:serviceName>
       <xsd:string xsd:value="fitheight"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di
              ridimensionare la pagina in modo da adattarla all'altezza
              della finestra."/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_fitwidth_Profile  -->

  <rdf:Description rdf:about="&instance;_fitwidth_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_fitwidth"/>
      <ontology:serviceName>
       <xsd:string xsd:value="fitwidth"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente di
              ridimensionare la pagina in modo da adattarla alla
              larghezza della finestra."/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_singlepage_Profile  -->

  <rdf:Description rdf:about="&instance;_singlepage_Profile">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Profile"/>
      </rdf:type>
      <ontology:presentedBy rdf:resource="&instance;_singlepage"/>
      <ontology:serviceName>
       <xsd:string xsd:value="singlepage"/>
      </ontology:serviceName>
      <ontology:textDescription>
        <xsd:string xsd:value="Questo servizio consente impostare il
              layout di pagina in modo da visualizzare una sola pagina
              per volta"/>
      </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_fullscreen_Profile  -->
```

```
  <rdf:Description rdf:about="&instance;_fullscreen_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_fullscreen"/>
     <ontology:serviceName>
      <xsd:string xsd:value="fullscreen"/>
     </ontology:serviceName>
     <ontology:textDescription>
       <xsd:string xsd:value="Questo servizio consente di leggere i
             documenti a pieno schermo di un documento pdf."/>
     </ontology:textDescription>
  </rdf:Description>

<!--  UbiPdfViewer_fromfullscreen_Profile  -->

  <rdf:Description rdf:about="&instance;_fromfullscreen_Profile">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Profile"/>
     </rdf:type>
     <ontology:presentedBy rdf:resource="&instance;_fromfullscreen"/>
     <ontology:serviceName>
      <xsd:string xsd:value="fromfullscreen"/>
     </ontology:serviceName>
     <ontology:textDescription>
       <xsd:string xsd:value="Questo servizio consente di usicre dalla
             vista a pieno schermo di un documento pdf."/>
     </ontology:textDescription>
  </rdf:Description>

  <daml:DatatypeProperty rdf:about="&ontology;#parameterName"/>
  <daml:DatatypeProperty rdf:about="&ontology;#resourceName"/>
  <daml:DatatypeProperty rdf:about="&ontology;#resourceUrl"/>
  <daml:DatatypeProperty
             rdf:about="&ontology;#resourceTextDescription"/>
  <daml:DatatypeProperty rdf:about="&ontology;#serviceName"/>
  <daml:DatatypeProperty rdf:about="&ontology;#textDescription"/>

</rdf:RDF>
```

A.5 Descrizione di un utente

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://www.ubinet.it/Ontology.daml">
  <!ENTITY instance
       "http://www.ubinet.it/Ontology.daml#Jim ">
  <!ENTITY DEFAULT "http://www.ubinet.it/Ontology.daml">]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd      =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
```

```
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">

  <daml:Ontology rdf:about="">
     <dc:title>&quot;Ontology&quot;</dc:title>
     <dc:date>12/01/2004</dc:date>
     <dc:creator>Esposito Massimo</dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
              definire un utente di un ambiente pervasivo.
     </dc:description>
     <dc:subject>Un&apos;istanza di un utente.
     </dc:subject>
     <daml:versionInfo>1.0</daml:versionInfo>
  </daml:Ontology>

  <!--
##################################################################### --
>
  <!-- # Instance Definition of an User # -->
  <!--
##################################################################### --
>

  <rdf:Description rdf:about="&instance;">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#User"/>
     </rdf:type>
     <ontology:userName>
      <xsd:string xsd:value="Jim"/>
     </ontology:userName>
     <ontology:userSurname>
      <xsd:string xsd:value="Huck"/>
     </ontology:userSurname>
     <ontology:userEmail>
      <xsd:string xsd:value="jimhuck@ubinet.it "/>
     </ontology:userEmail>
  </rdf:Description>

  <daml:DatatypeProperty rdf:about="&ontology;#userName"/>
  <daml:DatatypeProperty rdf:about="&ontology;#userSurname"/>
  <daml:DatatypeProperty rdf:about="&ontology;#userEmail"/>

</rdf:RDF>
```

A.6 Descrizione di un esempio di contesto

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY daml "http://www.daml.org/2001/03/daml+oil">
  <!ENTITY xsd "http://www.w3.org/2000/10/XMLSchema">
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY ontology "http://www.ubinet.it/contextOntology.daml">
  <!ENTITY instance
       "http://www.ubinet.it/contextOntology.daml#UbiNet ">
  <!ENTITY DEFAULT "http://www.ubinet.it/contextOntology.daml">]>

<rdf:RDF
  xmlns:rdf      =    "&rdf;#"
```

```
  xmlns:rdfs     =    "&rdfs;#"
  xmlns:daml     =    "&daml;#"
  xmlns:xsd      =    "&xsd;#"
  xmlns:dc       =    "&dc;#"
  xmlns:ontology =    "&ontology;#"
  xmlns          =    "&DEFAULT;#">

  <daml:Ontology rdf:about="">
     <dc:title>&quot;Ontology&quot;</dc:title>
     <dc:date>12/01/2004</dc:date>
     <dc:creator>Esposito Massimo </dc:creator>
     <dc:description>Questa ontologia è stata realizzata per
              definire un esempio di contesto per un ambiente
              pervasivo.
     </dc:description>
     <dc:subject>Un&apos;istanza di un contesto.
     </dc:subject>
     <daml:versionInfo>1.0</daml:versionInfo>
  </daml:Ontology>

  <!--
########################################################################## --
>
  <!-- # Instance Definition of a Context # -->
  <!--
########################################################################## --
>

  <rdf:Description rdf:about="&instance;">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Building"/>
     </rdf:type>
     <ontology:placeName>
      <xsd:string xsd:value="UbiNet"/>
     </ontology:placeName>
     <ontology:buildingAddress>
      <xsd:string xsd:value="Via Mille n.24"/>
     </ontology:buildingAddress >
     <ontology:numberOfFloors>
      <xsd:integer xsd:value="1"/>
     </ontology:numberOfFloors>
     <ontology:contains rdf:resource="&instance;_room1"/>
     <ontology:contains rdf:resource="&instance;_room2"/>
  </rdf:Description>

<!--  UbiNet_room1_Descriptiom  -->

  <rdf:Description rdf:about="&instance;_room1">
     <rdf:type>
        <daml:Class rdf:about="&ontology;#Room"/>
     </rdf:type>
     <ontology:placeName>
      <xsd:string xsd:value="Room1"/>
     </ontology:placeName>
     <ontology:floorNumber>
      <xsd:integer xsd:value="1"/>
     </ontology:floorNumber>
     <ontology:isContainedIn>
      <xsd:string xsd:value="UbiNet"/>
     </ontology:isContainedIn >
     <ontology:availableResource
             rdf:resource="&instance;UbiPrintService"/>
     <ontology:availableResource
```

```
              rdf:resource="&instance;UbiVideoConferencingService"/>
      <ontology:placedDevice rdf:resource="&instance;Pc1"/>
      <ontology:placedDevice rdf:resource="&instance;Pc2"/>
      <ontology:locatedUser rdf:resource="&instance;Fabio"/>
      <ontology:locatedUser rdf:resource="&instance;Massimo"/>
      <ontology:locatedUser rdf:resource="&instance;Guido"/>
  </rdf:Description>

<!--  UbiNet_room2_Descriptiom  -->

   <rdf:Description rdf:about="&instance;_room2">
      <rdf:type>
         <daml:Class rdf:about="&ontology;#Room"/>
      </rdf:type>
      <ontology:placeName>
       <xsd:string xsd:value="Room2"/>
      </ontology:placeName>
      <ontology:floorNumber>
       <xsd:integer xsd:value="1"/>
      </ontology:floorNumber>
      <ontology:isContainedIn>
       <xsd:string xsd:value="UbiNet"/>
      </ontology:isContainedIn >
      <ontology:availableResource
              rdf:resource="&instance;UbiPrintService"/>
      <ontology:availableResource
              rdf:resource="&instance;UbiPdfViewer"/>
      <ontology:placedDevice rdf:resource="&instance;Pc3"/>
      <ontology:placedDevice rdf:resource="&instance;UbiLaptop"/>
      <ontology:locatedUser rdf:resource="&instance;Jim"/>
      <ontology:locatedUser rdf:resource="&instance;huck"/>
      <ontology:locatedUser rdf:resource="&instance;Pippo"/>
   </rdf:Description>

  <daml:DatatypeProperty rdf:about="&ontology;#placeName"/>
  <daml:DatatypeProperty rdf:about="&ontology;#floorNumber"/>
  <daml:DatatypeProperty rdf:about="&ontology;#numberOfFloors"/>
  <daml:DatatypeProperty rdf:about="&ontology;#buildingAddress"/>

</rdf:RDF>
```

Bibliografia

[1] Robert E. McGrath, Anand Ranganathan, Roy H. Campbell, M. Dennis Mickunas, *"Use of Ontologies in Pervasive Computing Environments"*.

[2] M. Satyanarayanan, Carnegie Mellon, *"Pervasive Computing: Vision and Challenger"*.

[3] M. Weiser, *"The Computer for the 21st Century," Scientific Am.*, Sept., 1991, pp. 94-104.

[4] L.Barbieri, *"Web Services: Protocolli e Strumenti, Interoperabilità ed evoluzioni future"*.

[5] T. Berners-Lee, J. Hendler, O. Lassila, *"The Semantic Web"* pp. 34-43.

[6] W3C, *"The Semantic Web"* http://www.w3.org/2001/sw.

[7] W3C, *"Extensible Markup Language XML"*, http://www.w3.org/XML.

[8] W3C, *"Namespaces in XML"* http://www.w3.org/TR/REC-xml-names/.

[9] W3C, *"XML Schema"*, http://www.w3.org/XML/Schema.html.

[10] W3C, *"XML Schema Part 2: Datatypes"*, http://www.w3c.org/TR/xml-schema-2.

[11] W3C, *"Resource Description Framework (RDF)"* , http://www.w3c.org/RDF.

[12] F.Vitali, *"Metadati RDf e RDFS"*.

[13] Horrocks, Ian, *"CORBA-FaCT"*, *http://www.cs.man.ac.uk/~horrocks/FaCT/CORBAFaCT.html*.

[14] Horrocks, Ian and Sattler, Ulrike, *"Ontology Reasoning with SHOQ(D) Description Logic", International Joint Conference on Artificial Intelligence, Seattle, 2001*.

[15] Pan, Jeff Z. and Horrocks, Ian, *"Reasoning in the SHOQ(D) Description Logic",Workshop on Description Logics (DL-2002), 2002. http://dlweb.man.ac.uk.~panz/Zhilin/download/Papers/Pan-Horrocks-shoqdn-2002.pdf* .

[16] W3C, *"Feature Synopsis for OWL Lite and OWL," W3C Working Draft 29 July 2002. http://www.w3.org/TR/2002/WD-owl-features-20020729/*.

[17] Horrocks,Ian, *"Reasoning with Expressive Description Logics: Theory and Practice"*.

[18] Harmelon, Frank van, Patel-Schneider, Peter F., and Horrocks, Ian, *"A Model-Theoretic Semantics for DAML+OIL"*, *http://www.daml.org/2001/03/modeltheoretic-semantics.html*.

[19] Harmelon, Frank, Pael-Schneider, Peter F., and Horrocks, Ian, *"Annotated DAML+OIL (March 2001) Ontology Markup"*, *http://www.daml.org/2001/03/daml+oil-walkthru.html*.

[20] Harmelon, Frank van, Patel-Schneider, Peter F., and Horrocks, Ian, *"Reference description of the DAML+OIL (March 2001) ontology markup language", http://www.daml.org/2001/03/reference.html*.

[21] T.R.Gruber (1995): *"Toward Principles for the Design of Ontologies Used for Knowledge Sharing"*, International Journal of Human-Computer Studies, pp. 907-928.

[22] R.Neches, R.E.Fikes, T.Finin, T.R.Gruber, T.Senator, W.R.Swartout (1991): *"Enabling technology for knowledge sharing"*.

[23] W.R.Swartout, R.Patil, K.Knight, T.Russ (1997): *"Toward distributed, use of large-scaleontologies"*, In Spring Synposium Series on Ontological Engineering, Stanford.

[24] A.Bernaras, I.Laresgoiti, J.Corera (1996): *"Building and reusing ontologies for elettrical network applications"*, In proceedings of the European Conference on Artificial Intelligence.

[25] T.Gruber (1994): *"Shared Re-usable Knowledge Bases"*.

[26] Foundation for Intelligent Physical Agent. *"FIPA Device Ontology Specification"*.

[27] DAML-S Home Page, http://www.daml.org/services/.

[28] Anind K. Dey,*"Providing Architectural Support for Building Context-Aware Applications"* PhD thesis, Georgia Institute of Technology, 2000.

[29] Michel Barbeau, *"Mobile, Distributed, and Pervasive Computing"*.

[30] Pervasive Computing Group, *"A Middleware Infrastructure for Active Surroundings"*, Information & Communications University, Korea.

[31] Gaia Home Page, http://choices.cs.uiuc.edu/2k/Gaia/.

[32] Davies N., Gellersen H.W., *"Beyond Prototypes: Challenges in Deploying Ubiquitous Systems"*, IEEE Pervasive Computing, Gennaio-Marzo 2002.

[33] Salil Pradhan, *"Semantic location"* White paper, HP Labs, cooltown.hp.com , 2000.

[34] Tim Kindberg and John Barton, *"A web-based nomadic computing system"* White paper, HP Labs, cooltown.hp.com, 2001.

[35] Kagal L., Korolev V., Avancha S., Joshi A., Finin T., Yesha Y., *"Centaurus: An Infrastructure for Service Management in Ubiquitous Computing Environments"*, Dept. of Computer Science and Electrical Engineering, University of Maryland Baltimore Conty, USA, 2002.

[36] Chen H., Finin T., *"An Ontology for Context Aware Pervasive Computing Environments"*, University of Mariland, Baltimore County, USA, 2003.

[37] Chen H., Finin T., *"Semantic Web in the Context Broker Architecture"*, University of Mariland, Baltimore County, USA, 2004.

[38] Wallbank N., *"A Requirements Analysis of Infrastructures for Ubiquitous Computing Environments"* Computing Department Lancaster University, 2002.

[39] Davies, N., and Gellersen, H.-W. Leaving the Lab: *"Issues in Deploying Ubiquitous Computing Systems and Applications"*. Pervasive Computing, 2002.

[40] D.Salber, A.K.Dey, G.D. Abowd, *"Defining an HCI Research Agenda for an Emerging Interaction Paradigm"* Technical report, GVU Center and College of Computin, Georgia Tech, 1999.

[41] S.Maffioletti, *"Requirements for an Ubiquitous Computing Infrastructure"*, Department of Informatics, University of Fribourg.

[42] Mascolo C., Capra L., Emmerich W., *"Mobile Computing Middleware"*, Dept. of Computer Science, University College London, 2002.

[43] Roman M., *"An Application Framework for active space applications"*, 2003.

[44] Banavar G., *"Challenges in Design and Software Infrastructure for Ubiquitous Computing Applications"*, IBM TJ Watson Research Center, University of Zurich.

[45] G.Coulouris, J.Dollimore, T.Kindberg, *"Distributed Systems-Concepts and Design"*,3rd Edition, Addison-Wesley,2000.

[46] Mattern F., Sturm P., *"From Distributed Systems to Ubiquitous Computing-The State of the Art, Trends, and Prospects of Future Networked Systems"*, Department of Computer Science.

[47] Virgilio A., Baldoni R., *"Un'introduzione alla architettura CORBA con elementi avanzati di interazione client/server in Java"*, Dip. di Informatica e Sistemistica, Università di Roma "La Sapienza".

[48] Magnanini P., *"Sistemi di Discovery: La soluzione Jini"*, Università degli Sudi di Bologna, 2001.

[49] Della Penna G.,*"Web Service Definition Language(WSDL)"*, Università degli Sudi di L'Aquila.

[50] W3C, *"SOAP Version 1.2 Part 1: Messaging Framework"*, W3C Candidate Recommendation 2002. http://www.w3.org/TR/soap12-part1/.

[51] W3C, *"Web Services Description Language (WSDL) Version 1.2"*, W3C Working Draft 9 July 2003. http://www.w3.org/TR/wsdl12/.

[52] uddi.org, *"UDDI Technical White Paper"*, 6 September 2002. http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf.

[53] uddi.org, *"UDDI Version 3.0"* 19 July 2002. http://uddi.org/pugs/uddi-v300-published-20020719.pdf.

[54] Apache Axis, *"Web Services"*, http://ws.apache.org/axis.

[55] Apache Jakarta Project, *http://jakarta.apache.org/*.