Indice

Indice 1

Introduzione 6

Capitolo 1 Il Pervasive Computing 8

1.1 Introduzione 8
1.2 L'evoluzione verso il Pervasive Computing 11
1.2.1 Sistemi distribuiti 11
1.2.2 Sistemi mobili 11
1.2.3 Sistemi pervasivi 12
1.3 Background e stato dell'arte 13
1.3.1 Aura 13
1.3.2 GAIA: un middleware per Pervasive Environments 15
1.3.3 Cooltown 16
1.3.4 Centaurus 17
1.3.5 Progetto CoBrA 19
1.4 Caratteristiche di un sistema pervasivo 20
1.4.1 Context Information e Run-time Adaptation 21

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

1.4.2 Task Recognition e Pro-activity 22
1.4.3 Resource Abstraction e Discovery 22
1.4.4 Eterogeneity e Service UI Adaptation 22
1.4.5 Security e Privacy 23
1.4.6 Fault-tolerance e Scalability 23

Capitolo 2 Tecnologie per il Pervasive Computing 25

2.1 La comunicazione nei sistemi distribuiti 25
2.2 Paradigmi di comunicazione 26
2.2.1 Remote Procedure Call 26
2.2.2 XML-based RPC 27
2.2.3 Remote Method Invocation 28
2.3 Middleware ed infrastrutture di comunicazione 29
2.3.1 CORBA 29
2.3.2 Jini 34
2.3.2.1 Discovery 35
2.3.2.2 Join 35
2.3.2.3 Lookup 36
2.3.3 JMS 37
2.3.3.1 Comunicazione Point-to-Point 37
2.3.3.2 Comunicazione Publish/Subscribe 38
2.3.3.3 JMS API Architecture 39
2.3.4 Web Services 40
2.3.4.1 WSDL 42
2.3.4.2 UDDI 43
2.4 La comunicazione in ambienti pervasivi 45
2.4.1 L'interoperabilità: esigenza di nuovi strumenti 46
2.5 Web Semantico 47
2.5.1 Architettura e linguaggi del Web Semantico 48
2.5.2 XML e XML Schema 49
2.5.3 RDF e RDF Schema 51

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

2.5.4 DAML+OIL 53
2.5.4.1 Sistemi basati su Frame 54
2.5.4.2 Description Logics 55
2.5.5 OWL 56

Capitolo 3 Modello architetturale 58

3.1 Introduzione 58
3.2 Requisiti del sistema 58
3.2.1 Lo scenario e gli attori 59
3.2.2 Perché i Web Services 60
3.3 Il modello architetturale 61
3.4 System Service Manager 62
3.4.1 System Services 63
3.4.2 DHCP Service 63
3.4.3 Ontology Service 63
3.4.4 Location Service 65
3.4.5 Context Service 66
3.4.6 Event Service 67
3.4.7 User Manager Service 69
3.5 Public Services Manager 69
3.5.1 PresenceService 70
3.5.2 DiscoveryService 71
3.5.3 ServiceDirectory 72
3.5.4 Integrazione e composizione dei Servizi Web 72
3.6 Ubiquitous Gateway 72
3.6.1 La descrizione del dispositivo client 73
3.6.2 Il discovery dei servizi 74

Capitolo 4 I servizi di sistema implementati 77

4.1 Introduzione 77

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

4.2 Event Service 79
4.3 Ontology Service 80
4.3.1 OntoKB 81
4.3.2 OntoServer 82
4.3.3 OntoWrapper 87
4.3.4 Modifiche apportate all'ontologia di base 88
4.4 Location Service 91
4.4.1 Comunicazione sincrona 94
4.4.2 Comunicazione asincrona 97
4.5 Context Service 99
4.5.1 Il Context Service come servizio CORBA 99
4.5.1.1 Comunicazione sincrona 104
4.5.1.2 Comunicazione asincrona 108
4.5.2 Il Context Service come Web Service. 112

Capitolo 5 Casi di Studio: due servizi dipendenti dal contesto 127

5.1 Introduzione 127
5.1.1 La descrizione del servizio 129
5.2 Il servizio di stampa locale 130
5.2.1 Implementazione CORBA 131
5.2.2 Conversione in Web Service 133
5.2.3 Registrazione e configurazione del servizio 139
5.3 Il servizio di musica di ambiente 141
5.3.1 Registrazione e configurazione del servizio 146
5.4 Lo scenario in cui operano i servizi 147
5.5 L'Interfaccia utente 148
5.5.1 Start-up dell'ambiente 149
5.5.2 Il servizio di musica di ambiente 152
5.5.3 IL servizio di stampa locale 153

Conclusioni e sviluppi futuri 161

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Bibliografia 163

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Introduzione

I recenti progressi nelle tecnologie hardware e di comunicazione stanno lentamente cambiando il modo di concepire l'elaborazione informatica e le modalità d'interazione fra l'uomo ed i computer. In breve, ci troveremo a vivere in ambienti popolati da una serie di dispositivi "intelligenti" che comunicano e cooperano tra di loro per assisterci nello svolgimento delle nostre comuni attività.
A questo tipo di scenari ci si riferisce quando si parla di *Ubiquitous* (o *Pervasive*) *Computing*: scenari in cui, capacità di calcolo e di elaborazione assumono caratteristiche di ubiquità, si diffondono nella nostra realtà e penetrano all'interno degli oggetti che ci circondano.

Piuttosto che un fenomeno tecnologico, l'Ubiquitous Computing è una visione che ci spinge a riconsiderare il ruolo dei computer e della computazione nel nostro mondo quotidiano. E' una visione alla ricerca di una qualità fondamentalmente nuova dell'utilizzo del calcolatore.

All'interno dei laboratori del CNR è stato definito un modello architetturale, ed implementato un prototipo battezzato UbiSystem, di sistema pervasivo, con il fine di realizzare un ambiente dinamico e scalabile per l'erogazione, la ricerca e la fruizione di servizi.
L'obiettivo di questa tesi è l'espansione del suddetto prototipo di UbiSystem affinché soddisfi i requisiti, di grande rilievo in un ambiente pervasivo, di sensibilità alla locazione e di sensibilità al contesto.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La tesi è così articolata: nel capitolo 1 vengono introdotti i concetti di Pervasive ed Ubiquitous Computing, discutendo problematiche, aspetti innovativi e stato dell'arte; nel capitolo 2 vengono descritte le principali tecnologie che sono alla base del Pervasive Computing, in particolar modo, quelle impiegate per lo sviluppo della tesi; nel capitolo 3 viene descritto il modello architetturale di sistema pervasivo proposto; nel capitolo 4 viene illustrato il prototipo implementato con i suoi servizi di sistema; nel capitolo 5 vengono descritti due casi d'uso che mettono in evidenza le nuove funzionalità offerte dal sistema di location-awareness e di context-awareness; nelle Conclusioni e sviluppi futuri, infine, vengono discussi i risultati ottenuti e prospettate, appunto, le possibili articolazioni future.

Il presente lavoro è stato sviluppato presso l'ICAR-CNR sezione Napoli nei laboratori di San Giorgio (NA).

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Capitolo 1

Il Pervasive Computing

1.1 Introduzione

Era il 1991 quando Mark Weiser, direttore scientifico delle ricerche tecnologiche allo Xerox PARK, in un suo articolo d'avanguardia, utilizzò per la prima volta il termine *Ubiquitous Computing* [36]. Nel suo articolo, Weiser annunciava un cambiamento nel modo di concepire l'elaborazione automatica e descriveva scenari in cui i computer, onnipresenti, entravano sempre più a far parte della vita di tutti i giorni.

I *computer* ed il *computing* in generale, infatti, stanno lentamente ed inesorabilmente navigando verso nuovi paradigmi. Negli anni sessanta, alla parola *"computer"* venivano associati grandi e costosi *mainframe*, caratterizzati da un grosso numero di utenti che ne condividevano le risorse. Si parlava di paradigma *"many people per computer"*: molti utenti per una sola macchina. Il progresso tecnologico ha poi consentito la realizzazione dei *personal computer* che hanno significativamente modificato il tipo di utilizzo dei sistemi di calcolo, trasformando il paradigma in *"one person per computer"*: ogni persona poteva disporre di un proprio calcolatore. Nell'ultimo decennio, la diffusione di *laptop*, *Personal Digital Assistant* (PDA), telefoni cellulari multifunzione, dispositivi portatili dotati di microprocessori e di capacità di immagazzinare dati, ha mutato ulteriormente il rapporto uomo-computer, aprendo le porte all'era dei *"many computers per person"*: tanti elaboratori per una singola persona [31].

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

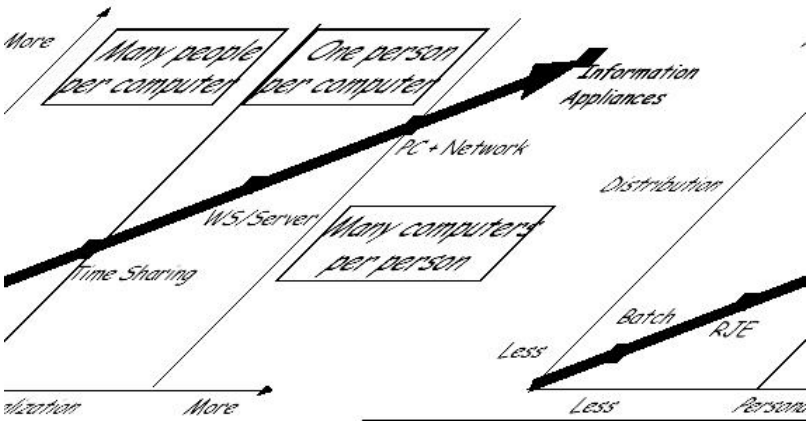

fig. 1.1: L'evoluzione del computing dai mainframes ad oggi.

Il proliferare di questi dispositivi intelligenti, sempre più piccoli e meno costosi, insieme con i progressi delle tecnologie di comunicazione, ha indotto Mark Weiser ad immaginare un futuro, non troppo lontano, nel quale i computer entrano a far parte integrante di ogni oggetto della realtà quotidiana, favorendo la comunicazione e l'elaborazione di informazioni in maniera naturale: *everywhere, all the time*.

L'essenza della sua idea era *"the creation of environments saturated with computing and computational capability, yet gracefully integrated with human users"*[32].

Un sistema di Ubiquitous Computing è caratterizzato da due attributi fondamentali [22]:

- *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 è possibile identificare due dimensioni che forniscono una più chiara definizione degli *ubiquitous system* ed esprimono le relazioni esistenti con le altre aree di ricerca emergenti:

- *Mobilità dell'utente*: esprime la libertà che l'utente ha di muoversi quando interagisce con il sistema;
- *Trasparenza di interfaccia*: 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

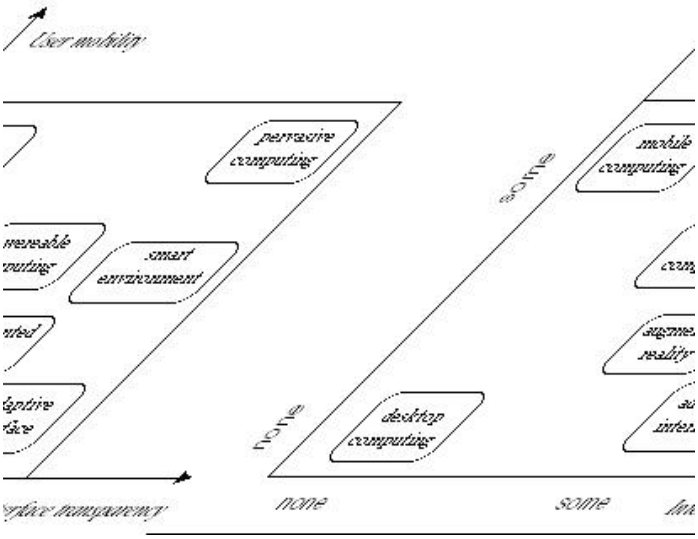

fig. 1.2: Attributi di un sistema pervasivo.

Quindi, nell'ottica di Weiser, l'*Ubiquitous Computing* mira alla realizzazione di un mondo non più vincolato alle scrivanie, ma composto da ambienti dotati di capacità computazionali e comunicative, talmente integrati con l'utente da diventare una "tecnologia che svanisce" [32], 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 a vivere nel mondo reale, insieme con le persone [36].

Tuttavia, gli scenari dipinti da Weiser 13 anni fa erano anacronistici; la tecnologia hardware necessaria per la loro realizzazione semplicemente non esisteva. E così, i tentativi compiuti allo Xerox PARC fallirono.

Dopo diversi anni, i recenti sviluppi tecnologici hanno dato nuovo impulso alle ricerche sull'*Ubiquitous Computing*, di recente ribattezzato anche col nome di *Pervasive Computing*, per suggerire il carattere pervasivo con cui l'"intelligenza elaborativa" si diffonde e si manifesta negli oggetti che ci circondano.

Probabilmente, i tempi non sono ancora maturi e la visione di Weiser resta ancora futuristica: *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it"*. Di sicuro, però, oggi ci troviamo in una posizione migliore, rispetto a quella dei ricercatori della Xerox del 1991, per affrontare le problematiche relative all'*Ubiquitous*, o *Pervasive*, *Computing*.

Da qualche anno, di fatto, in diverse Università e Centri di Ricerca del mondo sono sorte numerose iniziative che seguono la direzione tracciata da Weiser; ciascun progetto affronta i vari

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

problemi da differenti punti di vista, talvolta contrastanti, ma tutti sono impegnati nello sforzo comune di rendere il *Pervasive Computing* una realtà.

1.2 L'evoluzione verso il Pervasive Computing

Il *Pervasive Computing* può essere visto come una nuova forma di computazione altamente dinamica e disaggregata: gli utenti sono mobili e i servizi sono forniti da una serie di componenti distribuiti che collaborano tra loro [5]. Le applicazioni necessarie per supportare queste nuove esigenze sono, da un punto di vista architetturale, non monolitiche bensì costituite da moduli allocati in numerosi nodi della rete.
In tal senso, il *Pervasive Computing* costituisce una nuova tappa nel percorso evolutivo dell'elaborazione e del calcolo distribuito [32].

1.2.1 Sistemi distribuiti

Il *Distributed Computing* mira a ripartire dati e capacità computazionali su componenti indipendenti, residenti su macchine diverse, che comunicano tra loro attraverso reti di interconnessione.
Le metodologie attraverso le quali comunicare ed accedere ai dati e agli strumenti di calcolo da postazioni remote, le tecniche sulla replicazione e ridondanza dei dati, che favoriscono la disponibilità e la reperibilità delle informazioni e aumentano l'affidabilità del sistema nel complesso, rappresentano l'oggetto di studio per questa forma di computing.

1.2.2 Sistemi mobili

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* [24].
In questi nuovi scenari di calcolo distribuito, non si hanno più nodi di rete fissi, con connessioni stabili e veloci, ma nodi costituiti da dispositivi mobili che accedono alla rete e la abbandonano continuamente ed in maniera del tutto imprevedibile, dotati di connessioni precarie e contraddistinte da forti cambiamenti sulle caratteristiche di banda.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Le limitate capacità di calcolo e di memoria dei dispositivi mobili, le esigenze di risparmio energetico, rappresentano ulteriori aspetti di cui il *Mobile Computing* si sta occupando.

1.2.3 Sistemi pervasivi

I sistemi di *Pervasive Computing* sono a loro volta anche sistemi distribuiti e mobili. Pertanto, le problematiche inerenti il *mobile* ed il *distributed computing* vengono riprese in questo nuovo paradigma ma, in certo senso, amplificate oltremodo a causa dei requisiti stringenti e dei particolari contesti definiti in questi nuovi ambienti.

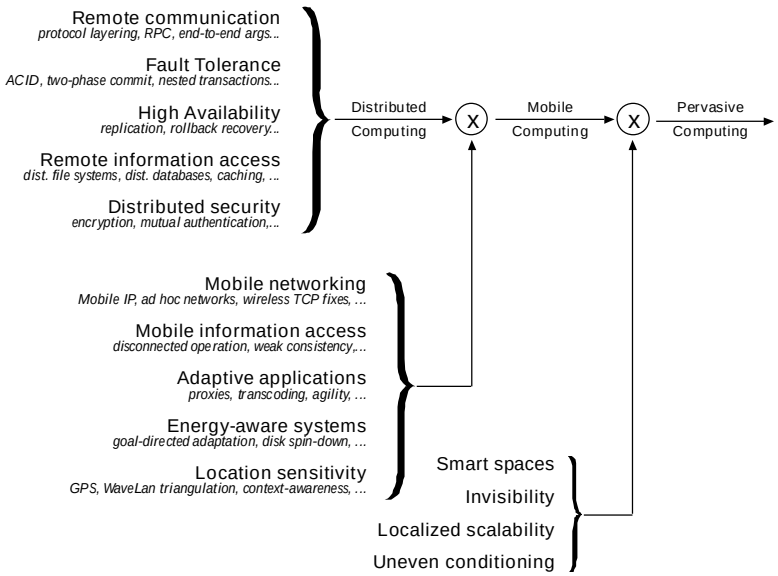

fig. 1.3: Dal Distributed Computing al Pervasive Computing.

La figura 1.3 mostra come i problemi di ricerca nel *Pervasive Computing* sono relazionati a quelli del *Distributed* e del *Mobile Computing*. Muovendoci da sinistra verso destra, incontriamo nuove incognite. Inoltre, le soluzioni di problematiche precedentemente affrontate diventano ancora più complesse: come suggeriscono i simboli in figura, l'aumento di complessità è moltiplicativo piuttosto che additivo. E' molto più difficile progettare e realizzare un sistema pervasivo che un sistema distribuito di comparabile robustezza e maturità [32].

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

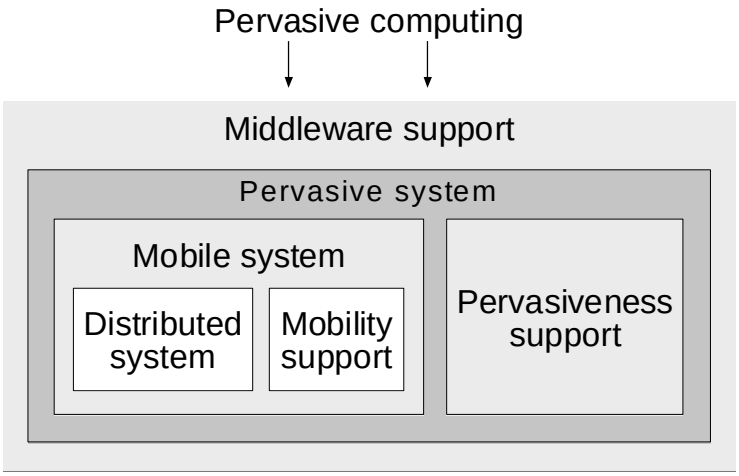

fig. 1.4: Un modello architetturale per sistemi pervasivi.

Concludiamo questo paragrafo con la figura 1.4 che riassume quanto finora detto, illustrando come si compone l'architettura di un sistema pervasivo a partire da sistemi mobili e distribuiti [30].

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 *ubiquitous computing* evidenziano chiaramente l'ingente impiego di tempo e di risorse che è stato profuso negli ultimi anni.
Ognuna di queste iniziative ha focalizzato la propria attenzione su uno specifico contesto con determinati applicazioni e servizi (ad esempio abitazioni, uffici, *meeting-room*), 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 analizzati, singolarmente, gli esempi più significativi di sistemi pervasivi implementati, evidenziando, per ciascuno di essi, le caratteristiche principali e i modelli architetturali.

1.3.1 Aura

Aura [27], il progetto seguito alla Carnieg Mellon University in Pittsburgh, mira a fornire all'utente un ambiente di *computing* "libero da distrazioni" (*distraction free ubiquitous*

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

computing), dove le persone possono accedere ai servizi o svolgere le proprie attività senza interventi sul sistema o sull'ambiente.
Aura utilizza essenzialmente due concetti per perseguire i suoi obiettivi:

- *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.

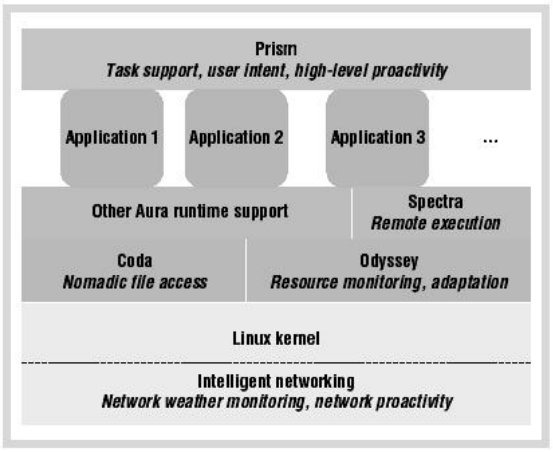

fig. 1.5: L'architettura di AURA.

Aura è composta da cinque componenti principali:

- *Odyssey* - è un'estensione del sistema operativo, che consente 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 della rete o dei server;
- *Spectra* - è un meccanismo di esecuzione remota ed adattativa che utilizza il contesto per decidere come meglio eseguire le invocazioni remote;
- *Prism* - che si occupa di catturare e gestire le intenzioni dell'utente ed in più fornisce un supporto di alto livello per affrontare gli aspetti di *proactivity* e di *self-tuning.*

Aura ricicla, in pratica, tecnologie esistenti per creare nuovi sistemi adatti ad essere inseriti in ambienti pervasivi.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

1.3.2 GAIA: un middleware per Pervasive Environments

GAIA ([14], [28],[29]) 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 [29].

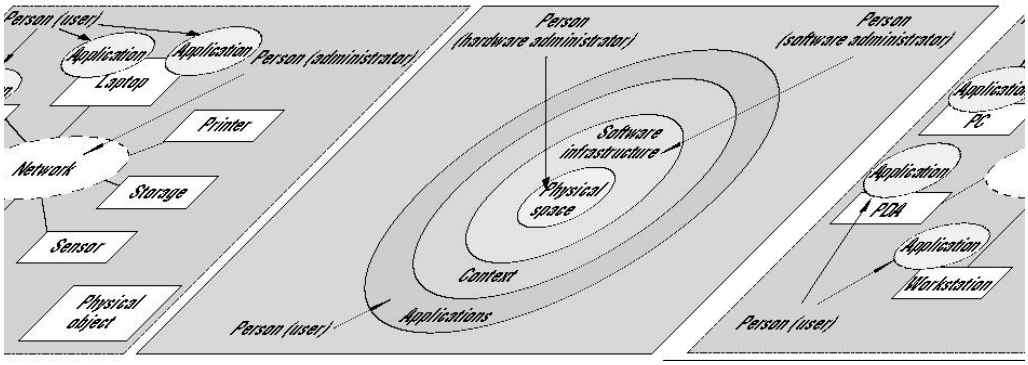

fig. 1.6: Physical Space (sinstra) and Active Space (destra) in GAIA.

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* e *component repository*;
- *Gaia Application Framework*: modella le applicazioni come collezioni di componenti distribuiti, analizzando 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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

appropriati per la loro istanza, 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 controllarle attraverso i servizi del Kernel di Gaia.

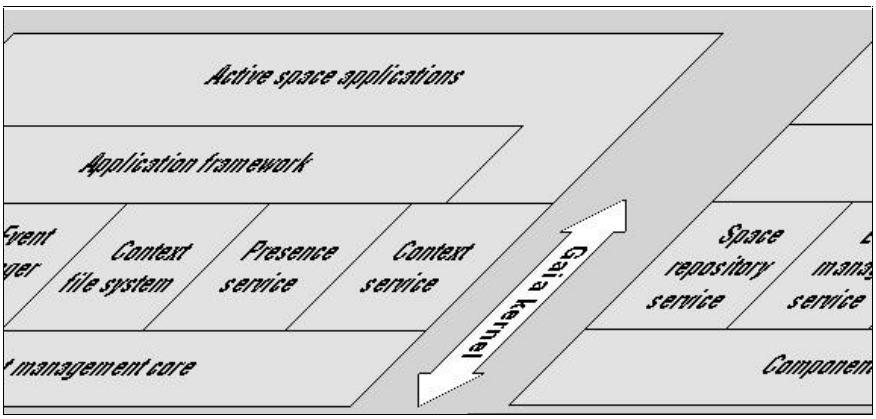

fig. 1.7: L'infrastruttura di GAIA.

In definitiva, GAIA cerca di portare le funzionalità di un sistema operativo in uno spazio attivo (come gli eventi, i segnali, il file system, al sicurezza, i processi, gruppi di processi), naturalmente estendendo questi concetti ed aggiungendo il nuovo concetto di contesto.

1.3.3 Cooltown

Il progetto realizzato presso i laboratori della Hewlett-Packard, Cooltown [10], 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 e entità fisiche e servizi elettronici [20].
Nella visione di Cooltown, ogni dispositivo fisico deve possedere un accesso tramite il web attraverso il quale può essere controllato dall'utente. L'URL può essere acquisito usando segnali radio (Bluetooth), infrarossi, codici a barre oppure può essere fornito direttamente in formato testuale.
Nel caso in cui solo un ID è ottenuto (come per i codici a barre), appositi server convertono l'ID in una URL, favorendo la possibilità di avere più resolver a seconda del contesto.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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

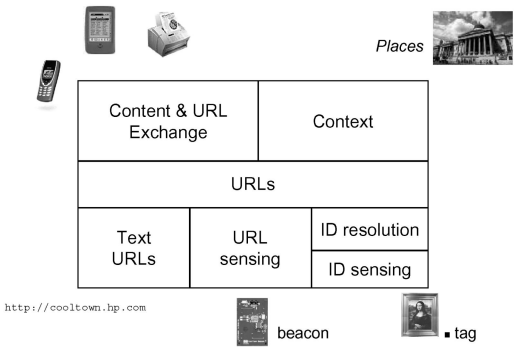

fig. 1.8: La visione di Cooltown.

In Cooltown, dove il protocollo di comunicazione è HTTP, è possibile comunicare con i dispositivi anche se non si è direttamente agganciati ad una rete globale adoperando ad esempio tecnologie come Bluetooth o IrDA, che risultano sufficienti per individuare, ad esempio, una stampante ed avviare il processo di stampa di un documento.
Le caratteristiche dei dispositivi dell'ambiente HP sono descritte in una web-form indipendente dal tipo di dispositivo stesso; ciò consente una semplice interazione tra dispositivi, mediante semplici 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-interface sono essenzialmente rivolte all'uomo e non sono adatte ad una elaborazione automatica.

1.3.4 Centaurus

Centaurus [19] definisce un'infrastruttura ed un protocollo di comunicazione per servizi software ed hardware eterogenei che devono essere resi disponibili agli utenti ovunque ne abbiano bisogno. Tali servizi intelligenti devono essere capaci di comprendere le necessità dell'utente per fornirgli migliore supporto ed agevolare le sue attività.
L'architettura di Centaurus consiste di cinque componenti:

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *CentaurusComm Trasport Protocol*: è un protocollo di trasporto affidabile, *message-based*, che può essere utilizzato indipendentemente dal tipo di mezzo trasmissivo;
- *Communication Managers:* gestisce tutta la comunicazione con i *client* usando differenti moduli del *CentaurusComm protocol*;
- *Service Managers:* controlla l'accesso ai servizi ed agisce da *gateway* tra i servizi ed i *client*;
- *Services*: sono oggetti che offrono funzionalità ai *client* Centaurus. I servizi contengono informazioni necessarie a localizzare il più vicino *Service Manager* ed a registrarsi ad esso. Una volta registrato, un servizio può essere acceduto da qualsiasi *client* attraverso il *Communication Manager*;
- *Clients*: forniscono un'interfaccia 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*.

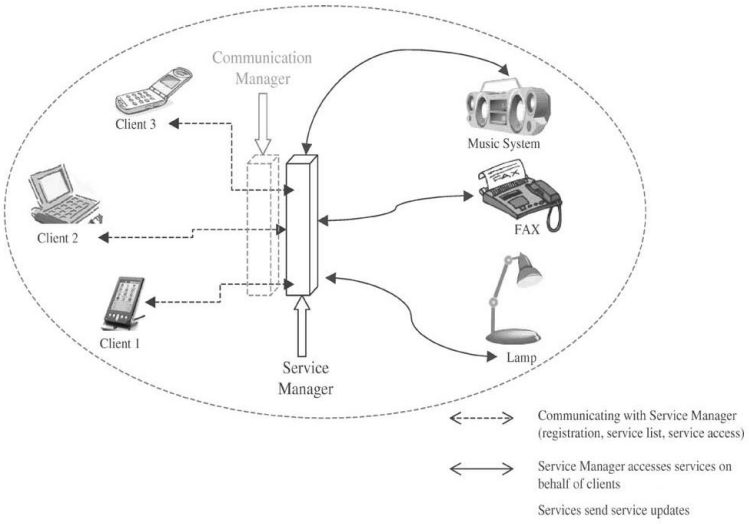

fig. 1.9: Architettura di Centaurus.

Tutti i componenti del modello usano un linguaggio proprietario, basato su XML, il Centaurus Communication Markup Language (CCML), per la comunicazione, offrendo al sistema un'interfaccia uniforme ed adattabile.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

1.3.5 Progetto CoBrA

All'Università del Maryland in Baltimora è stato sviluppato un modello architetturale per *pervasive environments* con particolare attenzione alle problematiche di context-aware computing.

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, 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, attraverso le quali agevolare la comunicazione fra i diversi agenti e rappresentare lo stato del sistema.

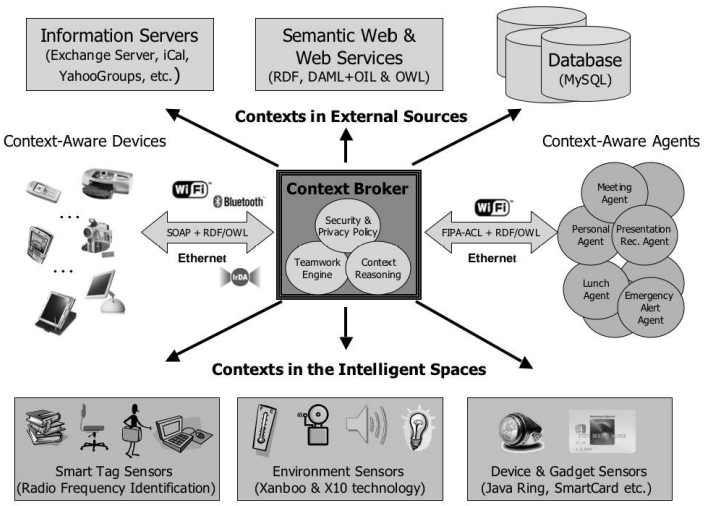

fig. 1.10: Architettura di COBRA.

L'architettura del context broker è costituita da quattro componenti funzionali ([8],[9]):

- *Context Knowledge Base*: immagazzina in maniera persistente la conoscenza del contesto. Fornisce un set di API per consentire agli altri componenti di accedere alle informazioni memorizzate. Contiene anche l'ontologia dello specifico ambiente e le regole euristiche associate ad esso;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *Context Reasoning Engine*: è un motore che applica algoritmi di inferenza sulle informazioni di contesto immagazzinate. 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 valutare i privilegi di accesso, associati alle differenti entità, sia per condividere particolari frammenti delle informazioni di contesto e ricevere notifiche dei cambi di contesto.

1.4 Caratteristiche di un sistema pervasivo

Nei paradigmi di *ubiquitous computing*, servizi ed informazioni sono virtualmente accessibili dovunque, in ogni istante attraverso qualsiasi dispositivo.
Ma considerazioni di carattere amministrativo, territoriale e culturale ci inducono ad analizzare l'*ubiquitous computing* in ambienti discreti, dai confini ben definiti, come per esempio case, uffici, sale convegno, aeroporti, stazioni, musei, etc. In altre parole, è bene considerare il mondo suddiviso in tanti domini pervasivi, piuttosto che vederlo come un unico enorme sistema [21].

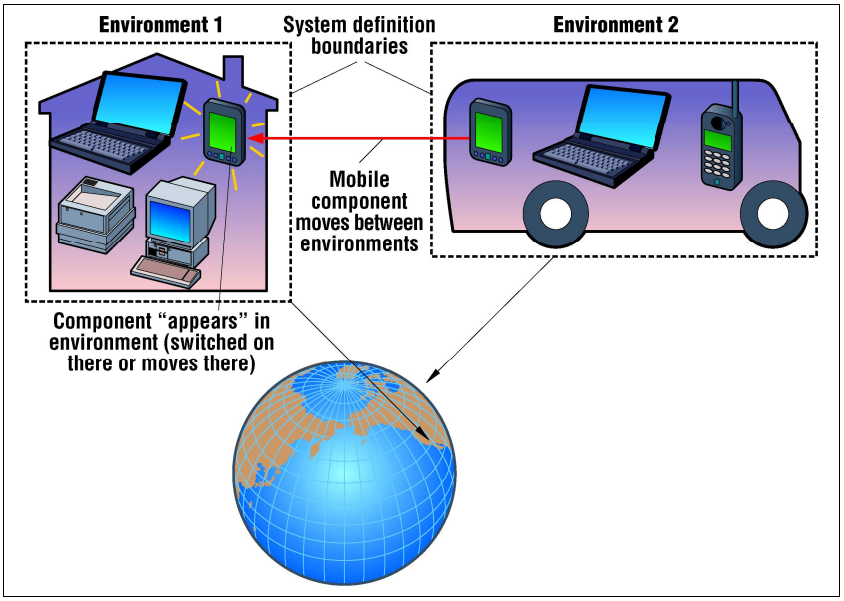

fig. 1.11: Il mondo suddiviso in ambienti pervasivi con confini ben definiti.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Ogni singolo ambiente è contraddistinto da componenti o unità software che implementano astrazioni di servizi, clienti, risorse o applicazioni ed, in generale, ogni ambiente è costituito da un'infrastruttura fissa e da una serie di elementi mobili che, in maniera del tutto impredicibile, entrano a far parte del sistema e lo abbandonano continuamente, talvolta migrando fra i diversi domini: tutti i componenti dell'ambiente devono essere in grado di interagire fra di loro.
A partire da queste considerazioni e dai progetti presentati nei paragrafi precedenti è possibile individuare una serie di aspetti di carattere generale e indicazioni di fondamentale importanza per l'analisi, lo sviluppo e l'implementazione di un sistema di pervasive computing [35].

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, trasformarli in informazioni di contesto e condividere le informazioni tra le diverse applicazioni in esecuzione sui vari dispositivi, 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, le applicazioni possono adattare il loro comportamento in maniera diversa a seconda dei casi. Possiamo definire due tipi di 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 necessariamente avere una percezione dell'ambiente ragionevolmente accurata, e disporre di meccanismi per il rilevamento e la correzione di informazioni di contesto inattendibili o contrastanti.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

1.4.2 Task Recognition e Pro-activity

Un sistema di *pervasive computing* dovrebbe essere capace, a partire dalle informazioni di contesto collezionate, di elaborare sullo stato corrente dell'ambiente e sulle intenzioni dell'utente e modificare dinamicamente il proprio comportamento per assisterlo nelle sue attività.

Diversamente dai sistemi di *computing* convenzionali in cui il comportamento del computer è principalmente composto di risposte all'interazione con l'utente, il *pervasive computing* mira alla realizzazione di un modello in cui i dispositivi sono la parte attiva nell'interazione con l'utente. Quindi se la tecnologia corrente si basa su persone che indicano ai computer ciò che devono 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.

I modelli che tengono conto dell'esperienza, del passato, rappresentano un importante strumento per la caratterizzazione di 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 determinate situazioni.

1.4.3 Resource Abstraction e Discovery

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

Dovrebbero essere previsti anche dei meccanismi per scoprire, interrogare ed interagire con le risorse nell'ambiente, per consentire l'introduzione di nuovi componenti senza onerose operazioni di configurazione e di riconfigurazione dei componenti esistenti.

1.4.4 Eterogeneity e Service UI Adaptation

Un ambiente pervasivo è caratterizzato da una moltitudine di dispositivi eterogenei, in numero variabile, come laptop, PDA, telefoni mobili.

Alla riduzione delle dimensione dei dispositivi dell'ambiente corrisponde una maggiore crescita del numero di dispositivi connessi ed una intensificazione delle interazioni 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,

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

portano alla conclusione che distribuire ed installare servizi per ogni classe e famiglia diventa ingestibile, specialmente in un'area geografica molto estesa.
E' importante, quindi, che i servizi forniscano agli utenti interfacce che possano adattarsi alle caratteristiche del dispositivo *client*, senza stravolgere le funzionalità del servizio stesso [13]. 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 Security e Privacy

Un sistema pervasivo generalmente gestisce grosse quantità di informazioni, molte delle quali acquisite tramite sensori e riguardanti gli utenti.
Dal punto di vista dell'utente, è desiderabile che vengano rispettati i principi di privacy e che sia garantita la sicurezza di queste informazioni.
Ad esempio, risulta fastidioso sapere che la stazione di polizia più vicina possa conoscere in quale stanza ci si trovi nella propria casa, attraverso i rilevatori di moto del sistema d'allarme, o quanto alcool si sta consumando, deducendo questa informazione dal sistema che gestisce l'inventario degli alimenti.
D'altra parte, in molte situazioni, una certa perdita di privacy può essere tollerata, come ad esempio in situazioni di pericolo.
Per aggiungere sicurezza alle informazioni, i servizi nell'ambiente non dovrebbero consentire accessi non autorizzati: ad esempio, a casa propria, non dovrebbe essere possibile per un ospite, aumentare il riscaldamento o controllare il funzionamento del forno [4].

1.4.6 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Capitolo 2

Tecnologie per il Pervasive Computing

2.1 La comunicazione nei sistemi distribuiti

Un sistema distribuito consiste in un insieme di computer che comunicano su di una rete per coordinare le azioni e i processi di un'applicazione [25]. Le tecnologie per la realizzazione di sistemi distribuiti hanno suscitato molto interesse negli ultimi anni, anche 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 e la crittografia, i *file system* distribuiti, la replicazione dei dati e i meccanismi di transazione distribuita, forniscono una solida infrastruttura di *run-time* che supporta le applicazioni di rete odierne.

Il modello dominante è ancora la tradizionale architettura *client-server*, ma lo sviluppo di applicazioni in ambito distribuito si sta basando sempre di più sia sull'impiego di supporti *middleware*, che forniscono astrazioni di alto livello, come oggetti distribuiti condivisi, sia su altri generi di servizi, come 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 ha luogo l'elaborazione informatica. Di conseguenza l'attenzione si focalizza su nuovi standard e

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

protocolli di alto livello, come XML, mentre passano in secondo piano aspetti di basso livello, come ad esempio le peculiarità di un sistema operativo.

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 complessi problemi 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 [25].

2.2.1 Remote Procedure Call

Un classico schema di comunicazione, che ben si addice al modello *client-server*, è la chiamata a procedure remote (RPC). In questo modello, un componente agisce da client quando richiede un servizio ad un altro componente, da server quando, invece, è lui a rispondere alle richieste. 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 spediti 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 ed inviarli 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 rappresentano i meccanismi di trasporto su cui si basano le più generali piattaforme *middleware* di cui si parlerà in seguito.

2.2.2 XML-based RPC

Sebbene i sistemi RPC trattano 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. La ragione principale è che ciascun *framework* definisce la propria tecnica di codifica per le strutture dati. Nonostante 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 scambiare 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 tipi primitivi in tipi composti per ottenere nuove categorie di dati.
Uno dei primi *framework* RPC basati su XML è stato SOAP (Simple Object Access Protocol) [40] 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 World *Wide Web Consortium* (W3C).

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

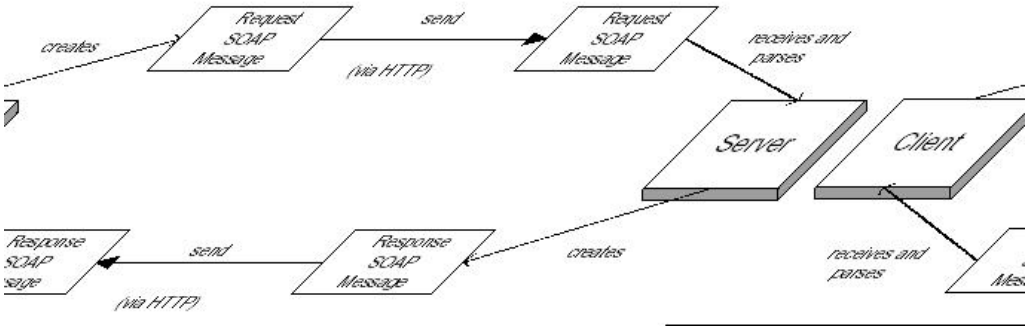

fig. 2.1: Interazione client-server usando SOAP.

SOAP [40] definisce solo la struttura del messaggio ed alcune regole per elaborarlo, 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.

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.

In questo contesto fa la sua comparsa RMI, una nuova tecnica di comunicazione remota per sistemi *Java-based*. 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 mobilità del codice Java, gli stub necessari per l'invocazione possono essere scaricati a tempo di compilazione 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 operare dei controlli di attendibilità.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 fra le varie piattaforme, dovuta a differente hardware, software, sistema operativo e linguaggio di programmazione.
Inoltre forniscono un set di servizi standard che in genere sono indispensabili per lo sviluppo di applicazioni distribuite, come servizi di directory, sicurezza e crittografia.

2.3.1 CORBA

Una delle più usate infrastrutture per 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.

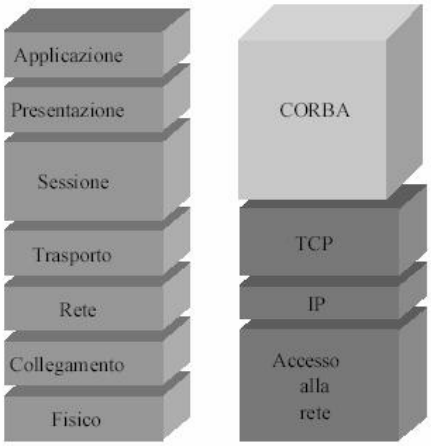

fig. 2.2: CORBA rispetto alla pila OSI.

Una parte delle specifiche descrive l'*Inteface Definition Language* (IDL) che deve essere supportato da tutte le implementazioni di CORBA. L'IDL di CORBA si rifà al C++ ed è usata dalle applicazioni per definire i metodi di un oggetto che è possibile utilizzare esternamente.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

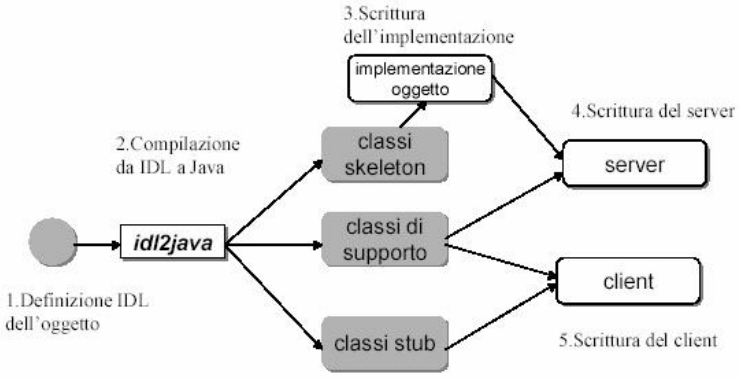

fig. 2.3: Ciclo di sviluppo di un'applicazione CORBA attraverso Java.

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* e *skeleton* realizzano le funzionalità di *marshalling* e di *unmarshalling* dei dati. Lo *skeleton*, inoltre, dal lato server, coopera con un *Object Adapter* per le operazioni che riguardano l'attivazione dell'oggetto.

Il componente centrale di un sistema CORBA è l'O*bject 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.

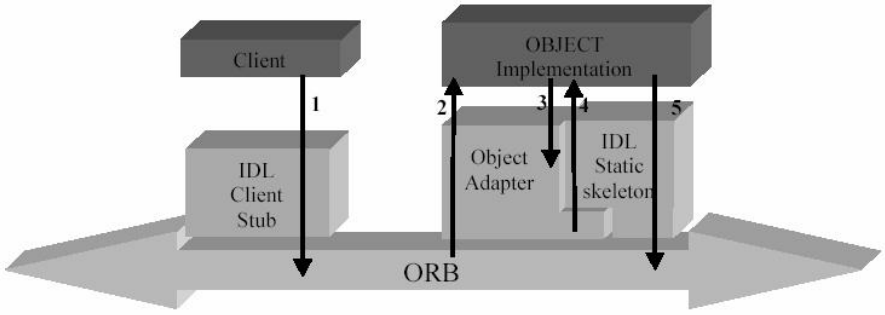

fig. 2.4: Interazione client-server in CORBA.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 l'interfaccia dell'oggetto server deve essere conosciuta 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.
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.
La comunicazione tra ORB nella rete avviene tramite i protocolli GIOP e IIOP che implementano il marshalling, l'unmarshalling e la rappresentazione esterna dei dati.

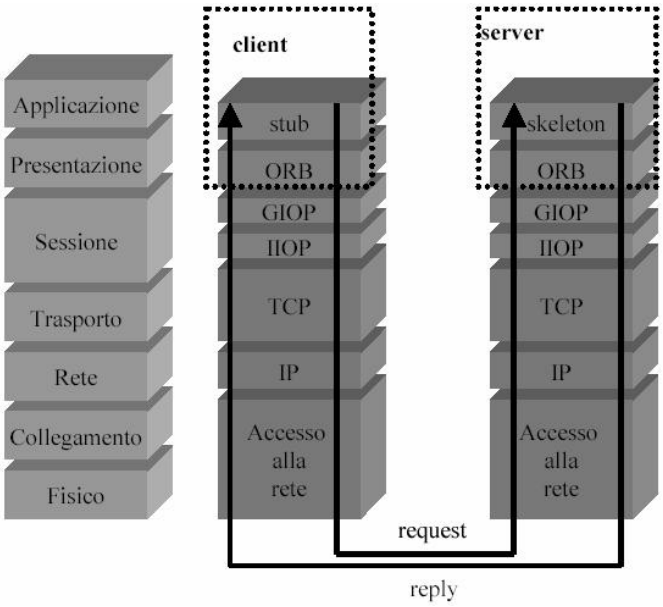

fig. 2.5: Collocazione dei protocolli GIOP-IIOP rispetto alla pila OSI.

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.
In aggiunta all'ORB, CORBA definisce un framework di oggetti supplementari, 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;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- 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 e-mail;
- 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 alcuni *object service* particolarmente importanti per i sistemi distribuiti: il *Naming Service*, il *Trading Service*, l' *Event Service* ed il *Notification 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 dandogli un nome.

Il *Trading Service* è un servizio di locazione di oggetti, così come il *Naming Service*, ma lavora ad un livello più astratto. Nel *Naming Service* gli oggetti vengono ricercati attraverso un nome. Nel *Trading Service* si fanno delle richieste per ricevere liste di oggetti che soddisfano certe caratteristiche o proprietà (inserite nella richiesta) [34].

L' *Event Service* è il servizio che si occupa della gestione di eventi CORBA, dove un evento è definito come una occorrenza di oggetto che è di interesse per uno o più oggetti [44], fornendo in tal modo un supporto per la comunicazione asincrona ad accoppiamento lasco tra oggetti. Per disaccoppiare la comunicazione tra un produttore di eventi (supplier) ed un consumatore (consumer) l' *Event Service* prevede l'astrazione di un canale di eventi (*Event Channel*) che funge da consumer per i supplier e da supplier per i consumer. In tal modo un supplier notifica l'evento senza conoscere i consumers interessati.

La figura 2.6 illustra i componenti dell' *Event Service* sopra descritti.

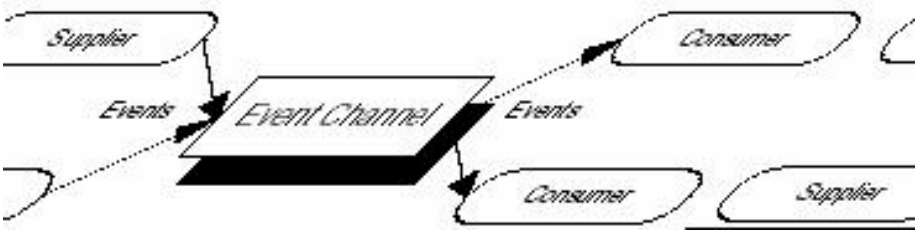

fig. 2.6: Componenti di CORBA Event Service

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Il *Notification Service* è una estensione dell' *Event Service* ed affronta le seguenti problematiche:

- strategie di filtraggio eventi (filtering);
- mantenimento di informazioni sullo stato dell' *Event Channel*;
- definizione di parametri di qualità del servizio di consegna degli eventi;

La figura 2.7 [45] illustra i componenti del *Notification Service* e mette in evidenza gli elementi che fanno del *Notification Service* una estensione dell'*Event Service*.

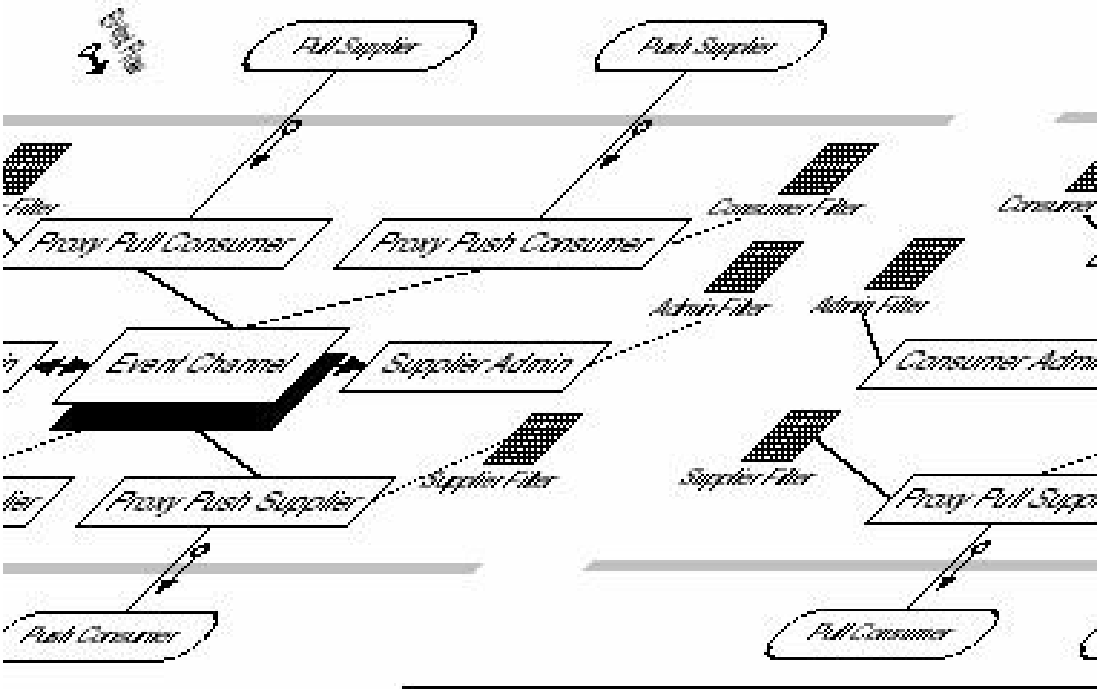

fig. 2.7: componenti del CORBA Notification Service

CORBA ha riscontrato molto successo nelle industrie 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

2.3.2 Jini

Jini è un'infrastruttura basata sul top di Java ed RMI per creare una federazione fra dispositivi e componenti software che implementano servizi. Jini rende possibile per qualsiasi dispositivo in grado di eseguire una Java Virtual Machine di interoperare con gli altri, offrire ed utilizzare 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 è possibile considerare servizi anche dispositivi con particolari funzionalità hardware.

I client possono usare un particolare servizio senza avere una conoscenza a priori della sua implementazione, scaricando quando necessario un oggetto proxy.

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 attraverso il quale 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 ricercare il servizio richiesto.

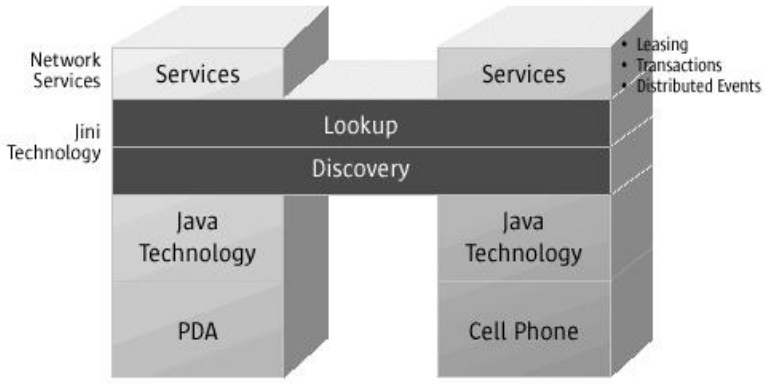

fig. 2.8: Architettura di Jini.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

2.3.2.1 Discovery

I servizi usano un meccanismo standard per registrarsi ed entrare a far parte della federazione: innanzitutto, interrogano la rete locale per localizzare il *Lookup Service*.

Il Lookup Service 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 locale comunità 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 l'obbligo di registrasi presso il Lookup Service stesso. 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.

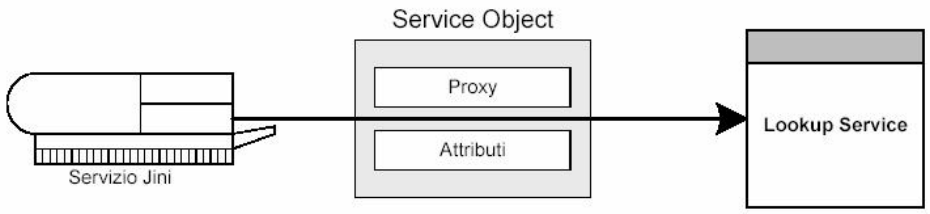

fig. 2.9: Registrazione del servizio.

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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 determinate caratteristiche. 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.

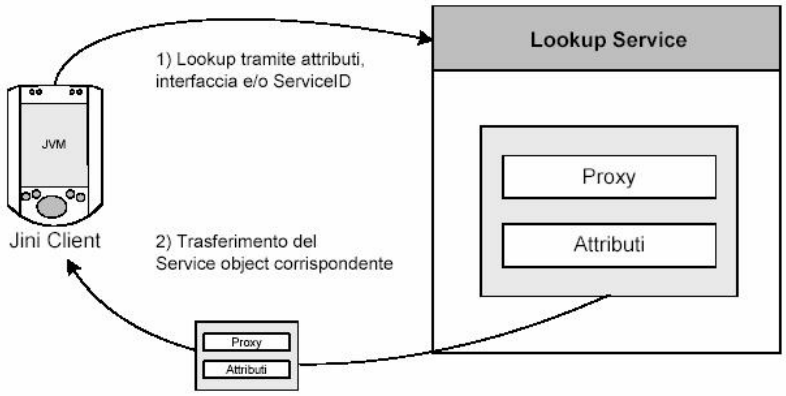

fig. 2.10: Lookup di un servizio.

Se gli attributi specificati dal client corrispondono ad un servizio registrato nel Lookup Service, quest'ultimo restituisce il service object corrispondente che verrà usato per invocare il servizio [23].

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: vedi Microsoft contro SUN.

Per questi motivi, sono stati ricercati nuovi modelli basati su tecnologie aperte e popolari, come ad esempio i *Web Services*.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

2.3.3 JMS

Java Message Service è una API Java che consente ad una applicazione di creare, inviare, ricevere e leggere messaggi. JMS è un progetto della Sun e le sue specifiche vennero pubblicate per la prima volta nell' Agosto 1998. L'ultima release delle specifiche è avvenuta nell'Agosto 2001 con la versione 1.0.2b.

Le specifiche definiscono un set comune di interfacce e di relative semantiche che consentono ad un programma scritto in Java di comunicare con altre implementazioni di messaging.

JMS minimizza il set di concetti che un programmatore deve conoscere per imparare ad usare prodotti di messaging ma nello stesso tempo mette a disposizione sufficienti features per supportare sofisticate applicazioni di messaging.

JMS mette e disposizione non solo una comunicazione strettamente accoppiata ma anche

- Asincrona. Un JMS provider può inviare messaggi ad un client, in modo ordinato, quando quest'ultimo sopraggiunge; un client non deve fare una esplicita richiesta di messaggi per riceverli. La comunicazione può essere di tipo *Point-to-Point* o di tipo *Publish/Subscribe*.
- Affidabile. JMS assicura che il messaggio venga spedito soltanto una volta. Livelli inferiori di affidabilità sono disponibili per applicazioni a cui sia consentita la perdita di messaggi o di duplicazione di questi ultimi.

2.3.3.1 Comunicazione Point-to-Point

Una applicazione point-to-point (PTP) si basa sui concetti di message queues, senders, e receivers. Ciascun messaggio è indirizzato ad una specifica coda dalla quale i messaggi vengono prelevati dai receivers. Le code di messaggi conservano tutti i messaggi inviati finché questi ultimi non vengano consumati e finché non scada la loro validità.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La figura 2.11 illustra la comunicazione PTP.

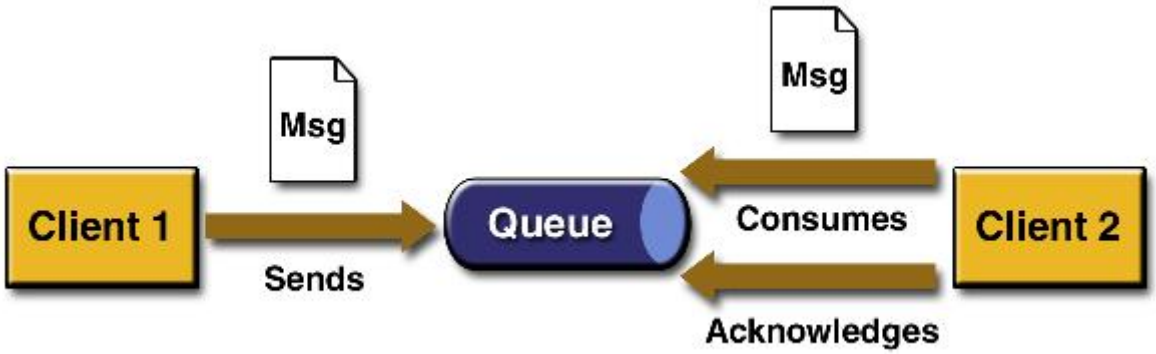

fig. 2.11: Point-to-Point Messaging

- Ciascun messaggio ha un solo consumatore.
- Un sender ed un receiver di un messaggio non hanno dipendenza temporale.
- Il receiver segnala quando il messaggio è stato correttamente processato.

L'uso della comunicazione PTP avviene quando si vuole che un messaggio inviato venga correttamente processato da un solo consumatore.

2.3.3.2 Comunicazione Publish/Subscribe

In una applicazione di tipo publish/subscribe (pub/sub) i clients fanno riferimento a messaggi pubblicati in un topic. I topics conservano i messaggi solo il tempo necessario affinché questi ultimi vengano distribuiti tra i subscribers correnti.

Pub/sub messaging ha le seguenti caratteristiche.

- Ciascun messaggio potrebbe avere più consumatori.
- Publishers e subscribers hanno dipendenza temporale.Un client che si iscrive ad un topic può consumare solo i messaggi pubblicati dopo la sua iscrizione al topic e deve essere continuamente attivo affinché non perda alcun messaggio.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

JMS rilassa le dipendenze temporali consentendo ai clients di creare *durable subscriptions*. Durable subscriptions consentono ad un client di non perdere alcun messaggio qualora quest'ultimo sia temporaneamente inattivo.

L'uso della Comunicazione di tipo pub/sub avviene quando ciascun messaggio possa essere processato da zero, uno o più consumatori. La figura 2.12 illustra questo tipo di comunicazione.

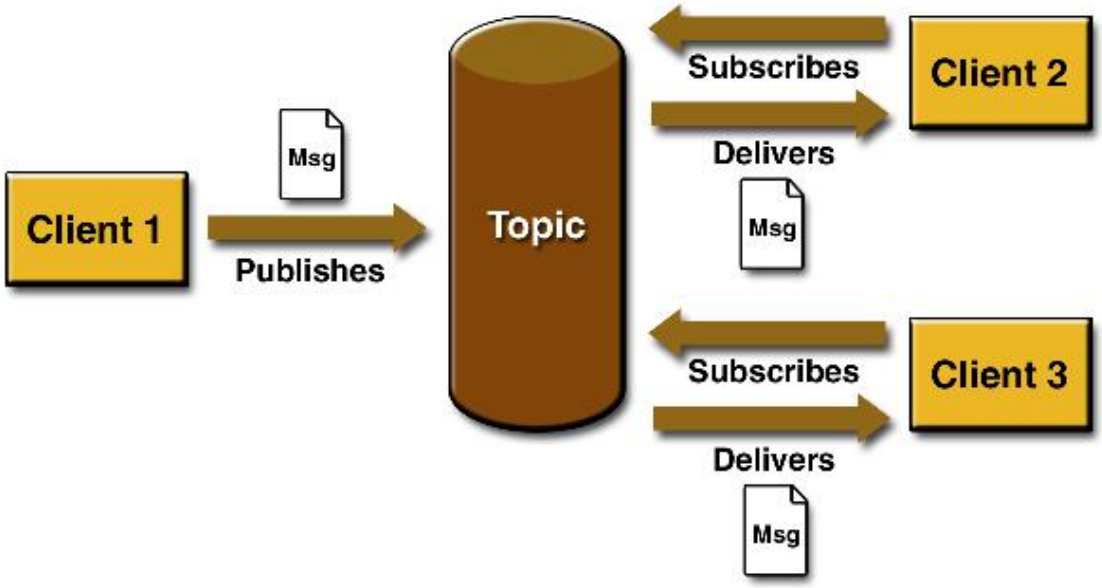

fig. 2.12: Publish/Subscribe Messaging

2.3.3.3 JMS API Architecture

Una applicazione JMS è costituita dalle seguenti parti.

- Un *JMS provider* che implementa le interface JMS e mette a disposizione le features di amministrazione e controllo.
- *JMS clients* sono programmi o componenti, scritti in linguaggio Java, che producono e consumano messaggi.
- *Messages* sono oggetti che contengono la informazioni che vengono scambiate tra i clients JMS.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *Administered objects* sono oggetti JMS preconfigurati e creati da un amministratore per l'uso dei clients. Sono di tipo *destinations* e *connection factories*. Una *destination* è l'oggetto che un client usa per specificare dove inviare il messaggio prodotto o da dove prelevare un messaggio da consumare. Nella comunicazione PTP le destinations sono le queues mentre in quella pub/sub le destinations sono identificabili nei topics. Una *connection factory* è un oggetto che un client usa per creare una connessione con il provider. Una connection factory incapsula un set di parametri necessari per configurare la connessione. Tali parametri sono definiti da un amministratore.

La figura 2.13 illustra come le suddette parti interagiscono. L' Administrative tool consente di associare le destinations e le connection factories all' interno del Java Naming and Directory Interface™ (JNDI) API namespace. Un client JMS può in seguito cercare gli administered objects nel namespace e così stabilire una connessione logica con questi ultimi attraverso il provider JMS.

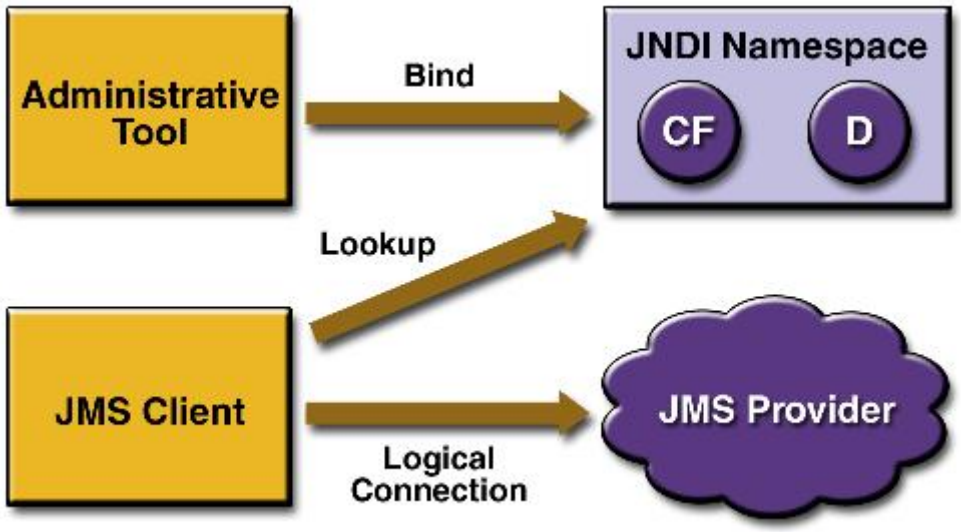

fig. 2.13: JMS API Architecture

2.3.4 Web Services

Il modello dei *Web Services* [41] è un insieme di standard per la pubblicazione, il discovery e la composizione di servizi indipendenti in una rete aperta [6].

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Il concetto di web service è molto simile a quello di oggetto, nel significato con cui lo si usa nella programmazione Object-Oriented: 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 una 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.

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:

- 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.

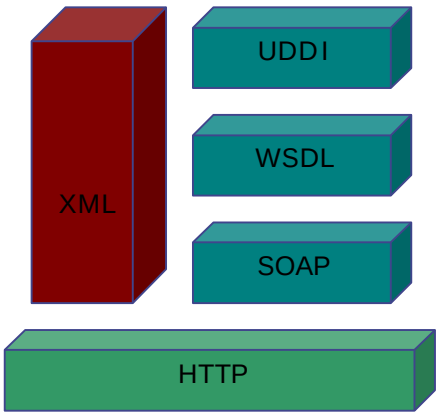

fig. 2.14: Protocolli standard dei Web Services.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

2.3.4.1 WSDL

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

Il Web Service Description Language [42] 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.

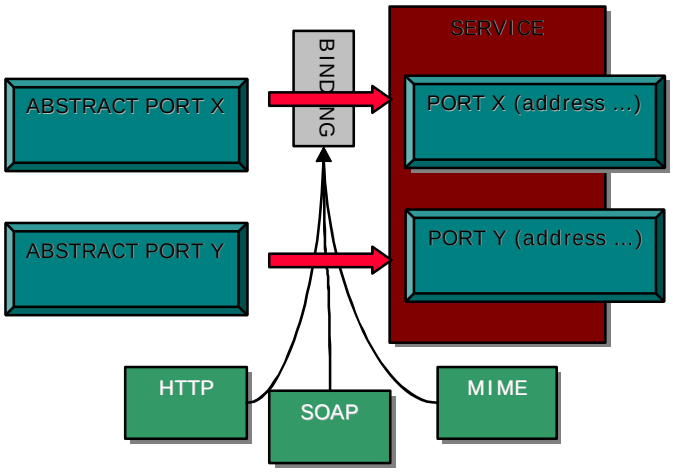

fig. 2.15: Dalla descrizione astratta a quella concreta.

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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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, 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.

2.3.4.2 UDDI

L'Universal Description, Discovery and Integration (UDDI) [33] 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. Con UDDI, è possibile pubblicare e individuare informazioni su un'azienda e i servizi Web da essa proposti.

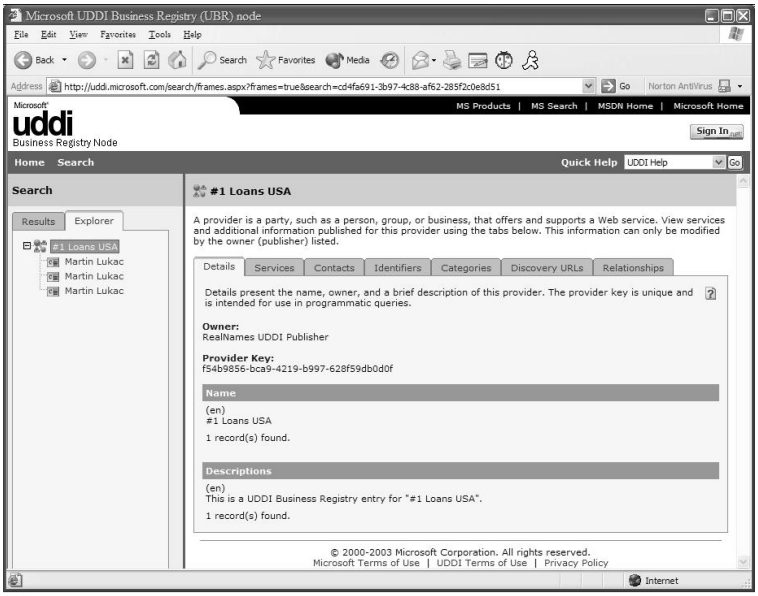

fig. 2.16: Esempio di registro UDDI della Microsoft.

Questi dati possono essere classificati mediante tassonomie standard, in modo da facilitare la ricerca delle informazioni per categoria.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Particolare di primaria importanza, in UDDI sono disponibili informazioni sulle interfacce tecniche dei servizi delle aziende. Mediante un gruppo di chiamate di API XML basate su SOAP, è possibile interagire con UDDI, sia in fase di progettazione che in fase di esecuzione, per rilevare dati tecnici che consentono l'attivazione e l'utilizzo di tali servizi.
Utilizzato in questo modo, UDDI funge da infrastruttura per una configurazione software basata su servizi Web.
Lo scenario nel quale trova pieno utilizzo l'UDDI 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.

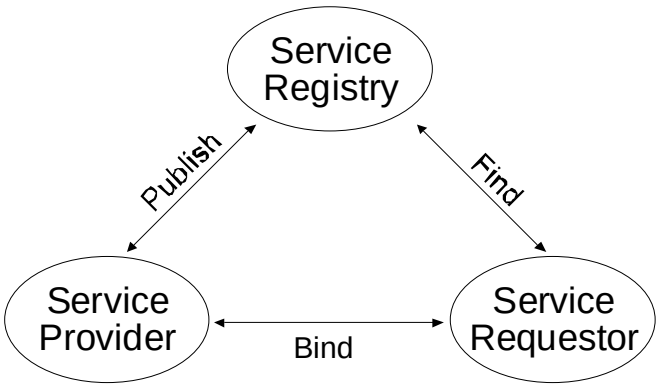

fig. 2.17: Scenario di utilizzo di UDDI.

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;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *yellow pages*, che classificano azienda e servizi attraverso tassonomie 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 esposti 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 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 services, 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. E' 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.

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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 prevede meccanismi per supportare l'interoperabilità semantica tra essi [26].

Nei sistemi distribuiti, lo scambio di messaggi rappresenta, come si è visto, un 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.

2.4.1 L'interoperabilità: 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 solo agli attributi definiti

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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 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.
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 [7] 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.
Il Web Semantico rappresenta un'estensione del Web attualmente esistente che, in aggiunta, mira alla cooperazione tra uomo e calcolatori.
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:

- 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;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- 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 così 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).

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

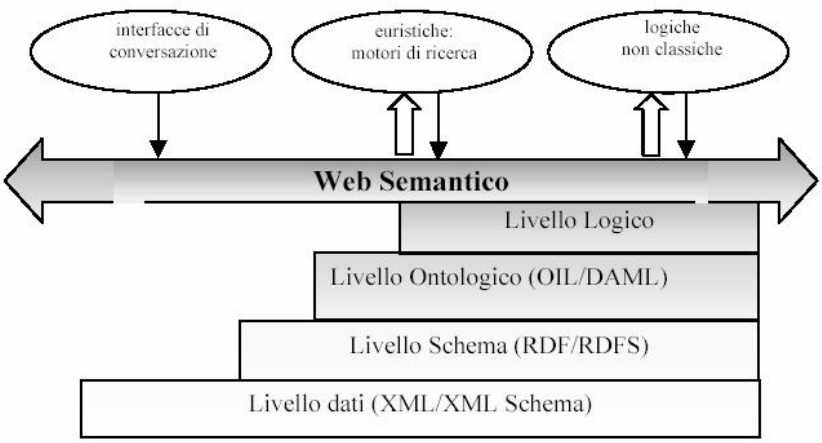

fig. 2.18: Architettura del Web Semantico.

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) [37] 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 version="1.0" ?>
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
<person>
        <name>Marco</name>
        <personal_info>
                <office_number>18</office_number>
                <phone_number>1024</phone_number>
        </personal_info>
</person>
```

fig. 2.19: Esempio di file XML.

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*).

E' 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: 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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

2.5.3 RDF e RDF Schema

RDF (Resource Description Framework) [39] è lo standard che consente l'aggiunta di semantica a un documento e quindi si pone ad un livello direttamente superiore rispetto ad XML. RDF 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*: un'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 è stato graficamente schematizzato in figura 2.20.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

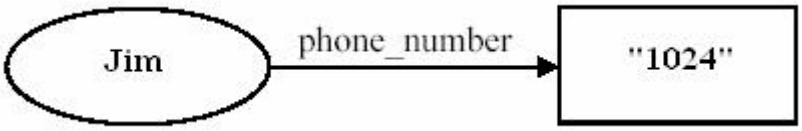

fig. 2.20: Semplice esempio di un'asserzione RDF.

La precedente asserzione è formalizzata in RDF/XML come segue:

```
<?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>
```

fig. 2.21: Esempio di 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 (Java-like). Una risorsa può, per esempio, essere definita come istanza di una classe (o

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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.

Inoltre, il loro utilizzo nei sistemi di rappresentazione della conoscenza è limitato 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 [12] 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).

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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.

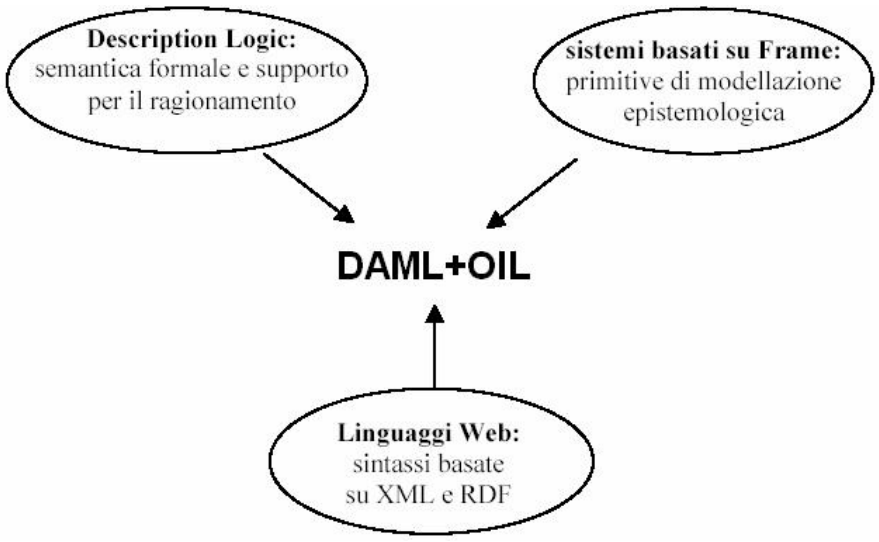

fig. 2.22: Il modello di DAML+OIL.

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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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* 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 formata da 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*).

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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

- la *satisfiability* di un concetto: se un concetto esiste;
- la *subsumption*: 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 subsumption 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) è un'estensione di DAML+OIL ottenuta attingendo da esso gli insegnamenti tratti dal suo uso applicativo e, in virtù di ciò, molto simile ad esso. Come DAML+OIL, anche OWL estende RDF ed RDF Schema e usa la sintassi XML. 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 adottano tecniche di reasoning. E' stato realizzato per supportare le Description Logics esistenti.
- *OWL Lite*: è un sottoinsieme di OWL DL che è facilmente utilizzabile ed implementabile. Aggiunge ad RDF Schema molte funzionalità utili per supportare le applicazioni web. Comprende molte delle caratteristiche più usate di OWL ed è destinato

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

agli sviluppatori che vogliono utilizzare OWL ma vogliono iniziare con un set relativamente semplice di caratteristiche del linguaggio.

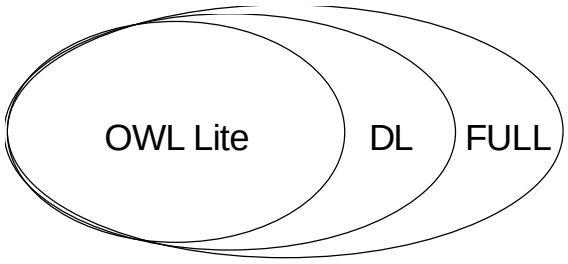

fig. 2.23: Sottolinguaggi di OWL

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Capitolo 3

Modello architetturale

3.1 Introduzione

Dopo l'analisi effettuata nei primi capitoli siamo pronti per presentare la nostra idea di modello architetturale per ambienti di pervasive computing.
Le principali scelte di progetto sono state prese coerentemente con lo stato dell'arte e si è tentato di non introdurre nuova complessità al panorama esistente, ricorrendo a tecnologie, protocolli e standard già esistenti ed affermati, senza definirne di nuovi. D'ora in avanti il sistema verrà anche chiamato col nome di UbiSystem.

3.2 Requisiti del sistema

L'architettura dovrà essere sufficientemente generale da poter garantire nuovi sviluppi ed evoluzioni future e da non comportare l'esclusione a priori di determinate soluzioni e tecnologie.
Per quanto visto nei capitoli precedenti, il sistema dovrà garantire i seguenti requisiti funzionali:

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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Inoltre, non vanno trascurati tutta una serie di requisiti non funzionali, alcuni dei quali strettamente legati agli aspetti di distributed computing, quali: scalabilità, tolleranza ai guasti, sicurezza ed attendibilità, semplicità, etc.

3.2.1 Lo scenario e gli attori

La figura 3.1 dipinge lo scenario all'interno del quale è stato sviluppato il nostro prototipo. L'ambiente è costituito da una rete LAN con estensioni di tipo wireless: ad una serie di nodi fissi se ne affiancano altri di natura completamente mobile.

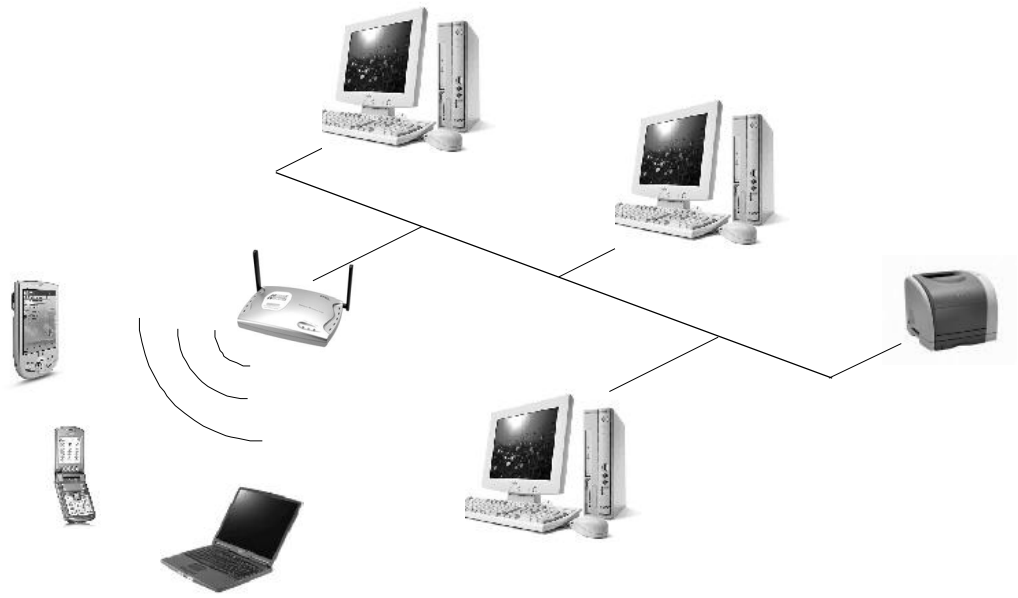

fig. 3.1: Scenario dell'ambiente pervasivo.

Gli attori che interagiscono con UbiSystem sono principalmente due:

- Gli utenti *(client)*- che accedono al sistema mediante dispositivi portatili (palmari, cellulari di ultima generazione, laptop) e che sono interessati ad utilizzare i servizi offerti;
- I fornitori di servizi *(service provider)* - costituiti da moduli software che offrono servizi e compiono azioni per conto degli utenti.

Dal punto di vista dell'utente è desiderabile poter interagire con il sistema mediante il proprio dispositivo senza dover procedere all'installazione di software aggiuntivo né di dover eseguire procedure di configurazione di alcuna sorta. Per questo motivo l'entry-point degli utenti sarà un'applicazione web e quindi accessibile mediante un comune web browser.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Per quanto riguarda i service provider, è stato stabilito che i servizi offerti all'interno dell'ambiente devono essere necessariamente dei Web Services.
Questa scelta, adottata coerentemente con la nostra linea progettuale di utilizzare tecnologie e standard web-based, non rappresenta una forte limitazione: di fatti, per riutilizzare moduli già esistenti (pensiamo, ad esempio, a servizi CORBA) è sempre possibile creare appositi wrappers che espongano i servizi come Web Services.

3.2.2 Perché i Web Services

La scelta di modellare i servizi all'interno del nostro ambiente come Web Services è stata motivata da una serie di considerazioni che proponiamo di seguito.
Un Web Service costituisce un sistema software disegnato per supportare l'interoperabilità e l'interazione fra macchine attraverso la rete. E' caratterizzato da un'interfaccia descritta mediante formati machine-processable. E' possibile interagire con un servizio Web mediante linguaggi e protocolli standardizzati o in via di standardizzazione quali WSDL, SOAP, HTTP, SMTP, FTP, etc. e quindi secondo tecnologie web-based (cfr. § 2.3.4).
Anche CORBA si propone di garantire interoperabilità multipiattaforma e multilinguaggio ma, trattandosi soltanto di specifiche, spesso si riscontrano problemi di incompatibilità fra ORB fra le reali implementazioni. Inoltre, la forte limitazione di CORBA è quella di non poter usare Internet come infrastruttura di comunicazione poiché il suo traffico viene sistematicamente bloccato dai firewall presenti sulla rete.
Per quanto riguarda i problemi di Service Discovery e di integrazione di servizi, esistono in letteratura diverse soluzioni. Probabilmente la più famosa è la soluzione Jini proposta dalla SUN (cfr. § 2.3.2). Ma come abbiamo visto precedentemente, la tecnologia Jini è strettamente legata alla piattaforma Java e si basa su RMI come middleware.
Tornado ai Web Services, c'è da sottolineare che accanto alla grande opera di standardizzazione che sta compiendo il World Wide Web Consortium, sono anche ingenti gli investimenti in ricerca e sviluppo in tal senso da parte delle maggiori industrie e aziende dell'Information Technology, che scommettono su questa nuova ed emergente tecnologia.
Inoltre, insieme a DAML-OIL e OWL (cfr. § 2.5.4 e 2.5.5) stanno nascendo ontologie come DAML-S ed OWL-S pensate appositamente per arricchire le descrizioni formali dei Web Services con contenuti di natura semantica.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

3.3 Il modello architetturale

Nel definire il nostro framework, una prima distinzione che abbiamo fatta è stata fra i servizi di supporto, finalizzati alla gestione generale del sistema, ed i cosiddetti servizi pubblici di utilità o di utente, ovvero i *servizi pervasivi*. La figura 3.2 dipinge l'architettura generale dell'intero sistema.

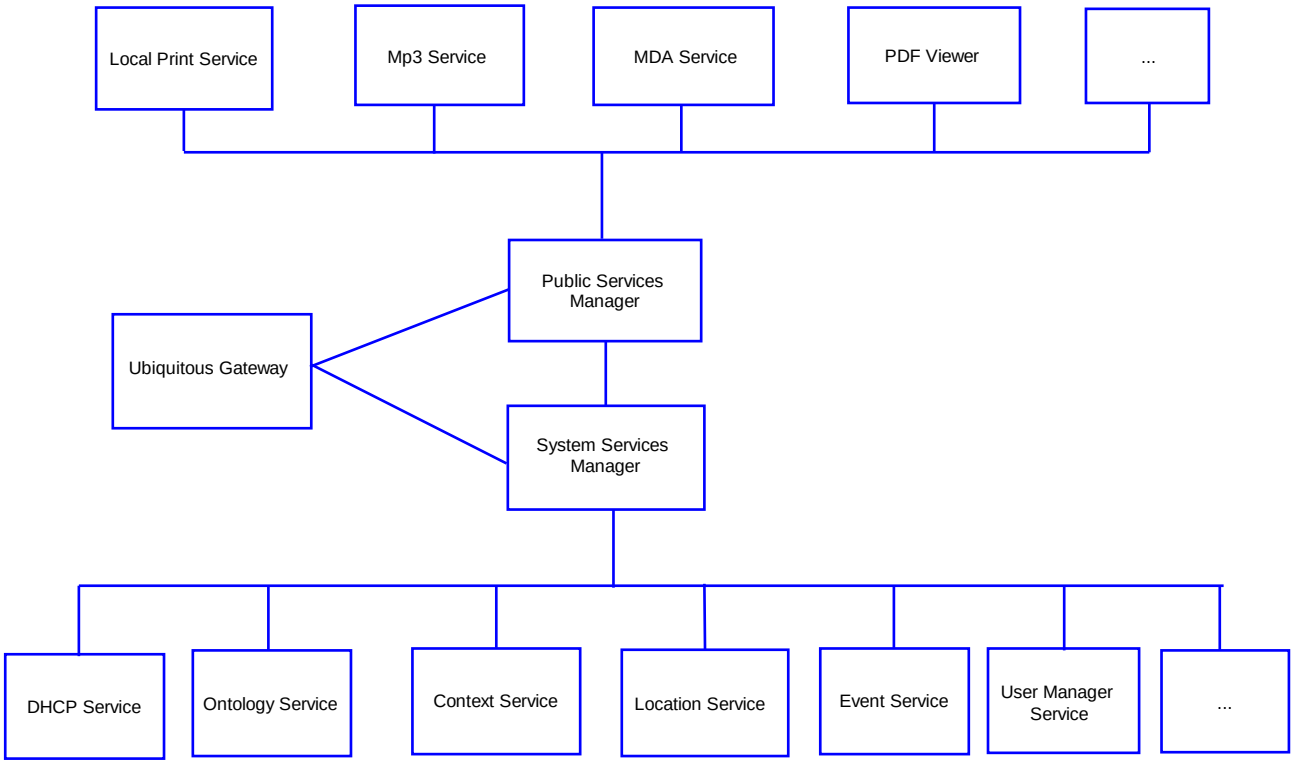

fig. 3.2: Architettura logica del sistema

L'architettura si sviluppa su due piani. Al primo abbiamo:

- l'Ubiquitous Gateway - che rappresenta l'entry point per gli utenti del sistema pervasivo;
- il System Services Manager - è il componente che gestisce i servizi di supporto del sistema;
- il Public Services Manager - che ha il compito di amministrare, invece, i servizi pervasivi di utilità;

Al secondo, da un lato abbiamo i servizi di sistema, quali:

- l'User Manager Service - che svolge servizi per la gestione degli utenti.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- il Context Service - che definisce un modello di contesto e lo condivide con il resto dell'ambiente.
- l'Ontology Service - per la gestione delle ontologie dell'ambiente
- il Location Service – per il rilevamento delle informazioni sulla posizione dei dispositivi mobili o anche sulla posizione degli utenti.
- l'Event Service – per il supporto alla comunicazione basata su paradigmi asincroni.
- il DHCP Service – per l'assegnazione dinamica degli indirizzi della rete;

Dall'altro lato, invece, abbiamo i servizi pubblici, che fanno riferimento al Public Services Manager:

- Servizi di stampa;
- Servizi di Mp3 JukeBox;
- Servizi di videoproiezione, etc.;

L'architettura così disegnata consente di disaccoppiare le funzionalità del sistema ed è facilmente scalabile.

Nei paragrafi successivi e nei seguenti capitoli verranno approfondite nel dettaglio tutte le funzionalità e le responsabilità di ogni singolo componente.

3.4 System Service Manager

E' il componente presso il quale vengono registrati i servizi di sistema, ovvero tutti quei servizi che sono di supporto per l'ambiente pervasivo. Tali servizi possono essere acceduti da parte dell'Ubiquitous Gateway e del Public Service Manager soltanto attraverso questo componente (ad esempio, immaginiamo un servizio di stampa che prima di avviare il processo necessita dell'autenticazione dell'utente che ne faccia la richiesta).

La scelta di introdurre il System Service Manager e di distinguerlo dal gestore dei servizi di utilità (Pervasive Manager) è stata dettata dall'esigenza di avere a disposizione un meccanismo semplice per aggiungere nuovi servizi di supporto per l'ambiente (ad esempio servizi di policy e sicurezza, o magari di QoS) ed, allo stesso tempo, di mantenere distinte le due tipologie di servizi offerte per ragioni pratiche ed amministrative.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

3.4.1 System Services

I tipi di servizi necessari per coordinare e gestire l'intero sistema pervasivo possono essere molteplici.

Ne sono stati individuati alcuni che riportiamo di seguito e che si propogono di soddisfare i requisiti elencati nel paragrafo 3.2.

3.4.2 DHCP Service

Si potrebbe obiettare sulla posizione assegnata al servizio DHCP in figura 3.2. In realtà, le funzioni del componente illustrato vanno al di là della semplice assegnazione automatica degli indirizzi IP ai nuovi elementi che entrano a far parte della rete. Infatti, il DHCP server ha la peculiarità aggiuntiva di notificare al sistema il momento e il punto di accesso in cui un dispositivo si aggancia o si sgancia dalla rete.

Questa caratteristica è molto importante per garantire la proprietà di *location-awareness* e *context-awareness* all'interno dell'ambiente pervasivo.

3.4.3 Ontology Service

L'Ontology Service è il componente il cui compito è quello di gestire le ontologie utilizzate all'interno dell'ambiente.

Una Ontologia è la descrizione formale di concetti appartenenti ad un determinato dominio applicativo.

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 il sistema.

In particolare, attraverso l'uso di ontologie:

- *vengono definiti i termini usati all'interno dell'ambiente*, per stabilirne in maniera univoca sintassi e semantica;
- *vengono verificate le descrizioni* di ciascuna entità affinché rispettino gli schemi definiti;
- *è possibile intraprendere discovery semantico*, per rendere più efficiente la ricerca di oggetti cercando di scoprire tutte e sole le entità rilevanti;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- è *possibile garantire interoperabilità fra i componenti,* fissando definizioni formali ed universali che in particolare descrivono i singoli servizi offerti da ogni risorsa.

Nel progetto, vengono definite ontologie per descrivere i vari aspetti dell'ambiente. Attraverso di esse, viene fissato uno schema logico ed una tassonomia standard, interpretabile oltre che dall'uomo anche da software automatici.

In particolare, sono state sviluppate ontologie di:

- Contesto - stabilisce concetti inerenti lo stato dell'ambiente: persone presenti, luminosità, temperature, attività in esecuzione, etc.;
- Servizi - stabilisce concetti e caratteristiche che le descrizioni dei servizi disponibili dell'ambiente devono rispettare;
- Device - stabilisce un modello per la caratterizzazione e classificazione dei dispositivi presenti nell'ambiente, comprese peculiarità hardware e software;
- Utenti - stabilisce lo schema concettuale caratterizzante gli utenti degli ambienti pervasivi.

Tutte le ontologie sono sviluppate in linguaggio DAML+OIL (cfr. § 2.5.4).

L'Ontology Service mette a disposizione un insieme di API (*Application Programming Interface*) per caricare le ontologie e costruire una Knowledge Base, verificare le descrizioni delle entità, interrogare la Knowledge Base e ricercare definizioni di concetti e relazioni.

In figura 3.3 è mostrato il bootstrap dell'Ontology Service: allo start-up le ontologie di base sopraelencate vengono lette ed interpretate dall'Ontology Service che poi le traduce per inviarle al FaCT Reasoner che costruisce la Knowledge Base.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

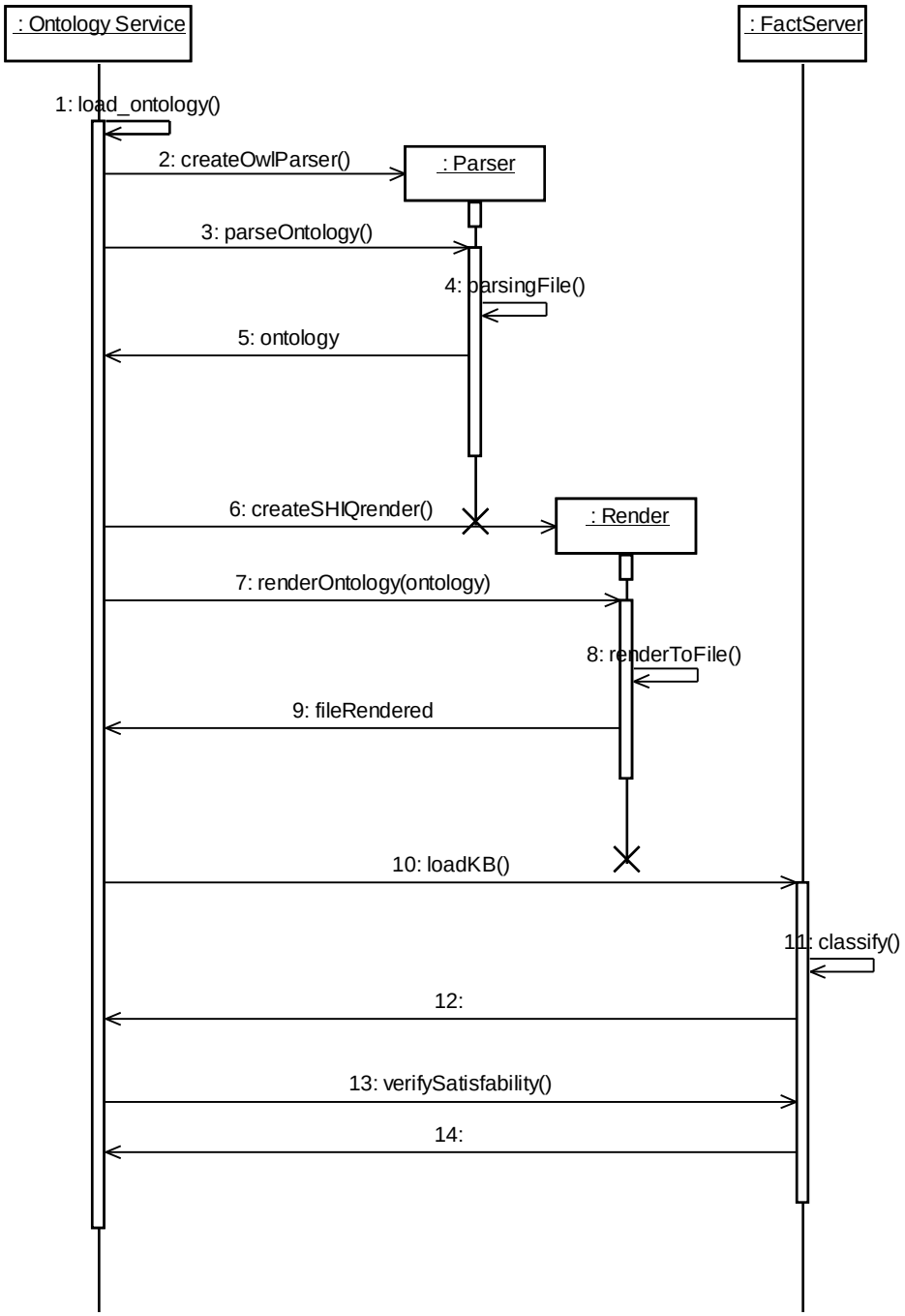

fig. 3.3: Bootstrap dell'Ontology Service.

3.4.4 Location Service

Il Location Service si occupa della localizzazione degli utenti del sistema e della localizzazione dei dispositivi mobili all'interno dell'ambiente pervasivo realizzato. Gli altri servizi di sistema possono ad esempio interrogare il Location Service sulla locazione di un utente richiedendo

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

l'edificio, la stanza o la locazione interna alla stanza in cui quest'ultimo si trova: il Location Service è in grado di operare a diversi livelli di dettaglio nell'estrazione delle informazioni di localizzazione.

Il Location Service garantisce la proprietà di *location-awareness*. Esso si occupa, infatti, anche della diffusione di eventi, generati quando un utente o dispositivo mobile cambia locazione all'interno del sistema pervasivo. In questo modo le altre entità di sistema possono essere avvertite di eventuali cambiamenti di posizione di utenti o di dispositivi mobili, e viene così realizzata, per l'UbiSystem, la sensibilità alla locazione a cui si è fatta sopra riferimento con il termine location-awareness.

3.4.5 Context Service

Il contesto e la sensibilità al 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 0.

Il Context Service ha il compito di costruire e condividere con gli agenti del sistema le informazioni di contesto dell'ambiente.

E' possibile immaginare diverse tipologie di contesto che possono essere usate dalle applicazioni:

- contesti fisici (posizione e tempo);
- contesti ambientali (aspetti meteorologici, livelli di luce e sonoro);
- contesti informazionali (quote azionarie, punteggi sportivi);
- contesti applicativi (e-mail, 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 dall'ambiente e di entità software che inferiscono nuove relazioni a partire dai dati percepiti.

Immaginiamo, ad esempio, il seguente scenario: un sensore segnala l'ingresso di una persona nell'ambiente; nel momento in cui la persona entra, è in atto una presentazione PDF; se in un'ontologia è stato definito che durante una presentazione tutte le persone presenti sono dei

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

“partecipanti alla presentazione”, si potrebbe concludere che la persona appena entrata è un nuovo partecipante alla presentazione.
Il Context Service definisce il contesto dell’ambiente sia a partire dai dati raccolti da appositi sensori sia da quelli provenienti dagli altri componenti del sistema alcuni dei quali, ad esempio, sono l’Ubiquitous Gateway , il Service Manager ed il Location Service.
In questo modo, il Context Service è informato sul numero e sull’identità degli utenti correntemente presenti all’interno dell’ambiente, dei servizi disponibili, dello stato delle applicazioni.

Le applicazioni ed i servizi dell’ambiente, a partire dalle informazioni di contesto ricevute dal Context Service, possono adattare il proprio comportamento secondo le diverse situazioni. Per esempio, un’applicazione di gestione dell’illuminazione di un ambiente potrebbe automaticamente accendere le luci regolandone l’intensità secondo le preferenze delle persone presenti all’interno dell’ambiente; oppure, potrebbe regolare l’intensità della illuminazione in dipendenza dall’ora del giorno.
Anche in questo caso, l’impiego di ontologie rende più semplice per gli sviluppatori specificare il comportamento dei propri applicativi secondo le diverse esigenze.

3.4.6 Event Service

In ambienti pervasivi non è sempre sufficiente disporre di meccanismi di comunicazione basati solo su paradigmi sincroni. Infatti, la mobilità degli utenti, la dinamicità dei servizi, i problemi legati alla momentanea perdita di connessione (ad esempio per l’esaurimento della batteria del dispositivo che si utilizzando) inducono a prendere in considerazione anche modelli di comunicazione asincrona.
L’Event Service offre il supporto per questo tipo di comunicazione adottando un modello del tipo Publisher-Server-Consumer.
Il Server, cioè l’Event Service, si interpone fra i publishers ed i consumers ed agisce come una sorta di middleware per smistare i messaggi.
La comunicazione fra publishers e consumers può essere divisa in due fasi. Il consumers comunica la sua sottoscrizione all’Event Service per un determinato canale d’interesse e l’Event Service colleziona i messaggi per conto del consumer.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Dall'altro lato, i publishers possono inviare messaggi di notifica all'Event Service che si occuperà, poi, di smistarli ai diretti interessati.
La scelta di formare dei canali d'interesse permette di disaccoppiare la comunicazione fra i produttori ed i consumatori. E', inoltre, possibile pensare ad una gerarchizzazione dei canali: ad esempio si potrebbe pensare ad un PresenceChannel e a dei sottonodi possibili quali, PresenceServiceChannel, PresenceUserChannel, PresenceDeviceChannel.
Un messaggio inerente un nodo foglia, comporterà la notifica dell'evento anche a tutti i sottoscrittori dei nodi padri e così via.
Alcuni canali predefiniti riguardano:

- l'ingresso di una nuova persona all'interno dell'ambiente;
- la fruibilità di un nuovo tipo di servizio;
- l'installazione di nuovi dispositivi;
- il cambiamento del contesto dell'ambiente.

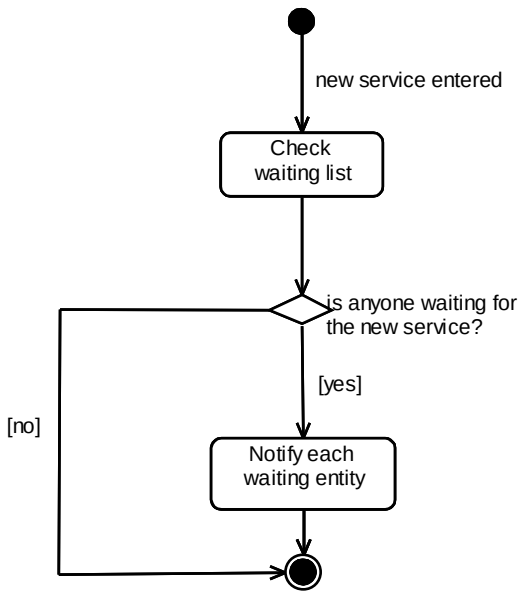

fig. 3.4: Diagramma delle attività intraprese dall'Event Service al deploy di un nuovo servizio all'interno dell'ambiente.

In figura 3.4 è mostrato il diagramma delle attività intraprese dell'Event Service nel momento in cui è disponibile un nuovo servizio: viene controllato se ci sono dei consumatori in attesa di questo determinato evento ed un messaggio di notifica è inoltrato ad ognuno di essi.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La gestione della cancellazione delle sottoscrizione dei consumatori può essere gestita con un meccanismo di leasing: il lease è il periodo di tempo durante il quale viene ritenuta valida la sottoscrizione.
Scaduto il periodo di lease, l'Event Service ritiene decaduta la sottoscrizione se nel frattempo non è stata rinnovata da parte del consumer.

3.4.7 User Manager Service

Lo User Manager Service ha il compito di svolgere tutte le attività legate ad una corretta gestione degli utenti che interagiscono con il sistema. In particolare il servizio deve:

- Provvedere all'autenticazione di ogni nuovo utente.
- Costruire e condividere con gli agenti di sistema le informazioni relative al profilo di ciascun utente.
- Gestire il mapping tra l'utente ed i vari dispositivi attraverso i quali l'utente si aggancia alla rete.
- Gestire i privilegi di accesso dell'utente ai servizi offerti.
- Monitorare ciascun utente presente nell'ambiente.

Compito dell'User Manager Service è, quindi, anche tener traccia degli spostamenti e delle azioni effettuate da ciascun utente. In tal modo, l'UbiSystem può notificare all'utente i servizi a cui egli può avere accesso trovandosi nella locazione in cui è stato rilevato, dando rilievo ai servizi verso i quali l'utente ha mostrato maggiore interesse.

3.5 Public Services Manager

Rappresenta il componente referente per i *service providers* che intendono pubblicare un nuovo servizio.
Le funzionalità e le responsabilità del Public Services Manager sono:

- Gestire la pubblicazione di nuovi servizi e la rimozione di servizi esistenti;
- Mantenere e rendere disponibili le descrizioni di tutti i servizi registrati;
- Offrire meccanismi per il discovery dei servizi disponibili.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

In figura 3.5 è riportato il diagramma delle classi che implementano questo componente.

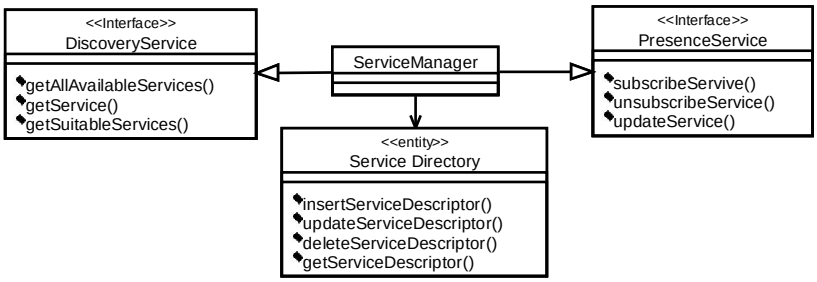

fig. 3.5: Diagramma delle classi che implementano il Public Services Manager.

Il diagramma si compone di due interfacce, della classe ServiceManager che le implementa e di una ServiceDirectory. Analizziamo le responsabilità di ogni singola classe.

3.5.1 PresenceService

E' l'interfaccia attraverso la quale un *service provider* agisce per registrare e cancellare i propri servizi.

I metodi messi a disposizione sono *subscribe()* per effettuare l'operazione di registrazione di un servizio ed *unsubscribe()* che invece può essere invocato per rimuovere un servizio.

In fase di registrazione, il *service provider* deve fornire al ServiceManager la descrizione del servizio che intende pubblicare; il ServiceManager controlla mediante l'Ontology Service che la descrizione appena pervenuta sia consistente ed aggiorna la ServiceDirectory. Quanto appena detto è riportato nel sequence diagram di figura 3.6.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

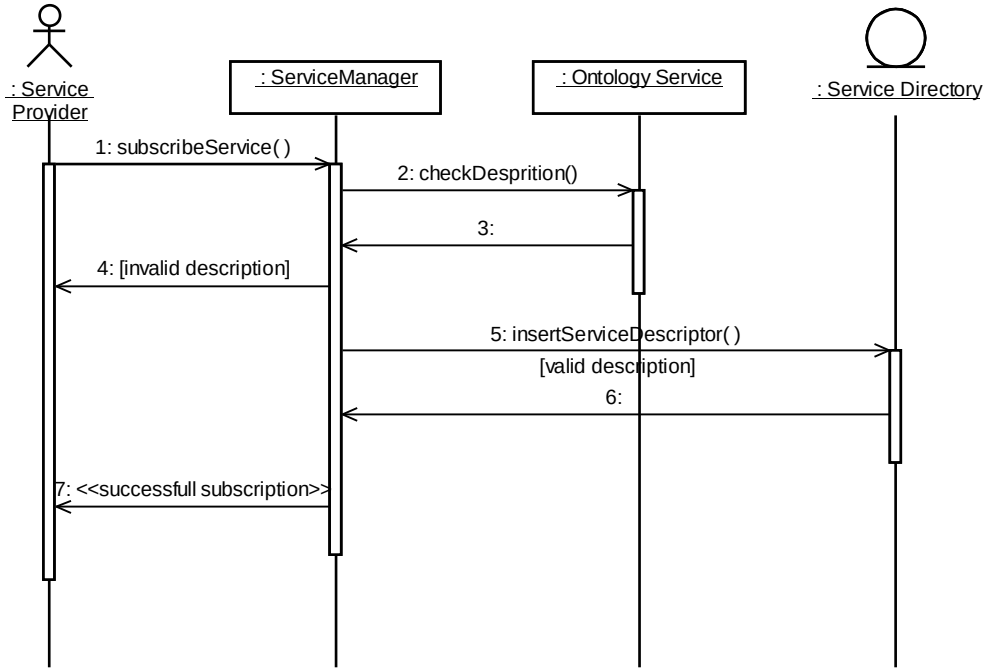

fig. 3.6: Sottoscrizione di un nuovo servizio.

Una volta effettuata la registrazione, il servizio può cambiare la descrizione delle sue caratteristiche (*updateService()*), ad esempio se nuove funzionalità vengono aggiunte o se magari le stesse funzionalità vengono svolte con una differente implementazione.

3.5.2 DiscoveryService

E' l'interfaccia per intraprendere attività di service discovery. Attraverso il DiscoveryService è possibile reperire la lista di tutti i servizi disponibili nell'ambiente (metodo *getAllAvailableServices()*) oppure selezionarne uno in particolare (*getService()*).

A differenza dei comuni protocolli di discovery, forse l'aspetto più interessante è quello di poter intraprendere processi di discovery semantico.

Come abbiamo visto in precedenza, ad ogni entità presente nell'ambiente viene associata una descrizione ontologica; ciò resta valido anche per i servizi pubblicati.

In architetture quali CORBA e Jini, i servizi sono completamente descritti attraverso i loro requisiti funzionali e cioè tramite le loro interfacce: IDL se parliamo di CORBA, interfacce java se parliamo di Jini.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La WSDL, al pari della IDL o delle interfacce java, definisce i requisiti funzionali di un Web Service, ma ad essa sono affiancate nuove informazioni di natura semantica.
Queste informazioni possono essere utilizzate per esprimere requisiti non funzionali quali: affidabilità, prestazioni, banda disponibile, latenza di canale, etc.

3.5.3 ServiceDirectory

Infine, l'ultimo componente del class diagram di figura 3.5 è la ServiceDirectory, dove vengono mantenute le descrizioni dei servizi pubblicati. Ogni qualvolta se ne renderà necessario, il ServiceManager effettuerà un accesso nella ServiceDirectory per prelevare la descrizione di un determinato servizio (ad esempio per verificare se soddisfa le caratteristiche di una particolare richiesta).

3.5.4 Integrazione e composizione dei Servizi Web

Un altro vantaggio nell'impiego dei Web Services è quello di poter facilmente aggregare e comporre servizi esistenti per crearne di nuovi.
La composizione dei servizi è ancora una volta agevolata dalla disponibilità di descrizioni semantiche. Per adempiere a questi compiti, il ServiceManager interagisce con l'Ontology Service per stabilire relazioni fra concetti ed i collegamenti possibili fra i diversi servizi.

3.6 Ubiquitous Gateway

L'Ubiquitous Gateway rappresenta l'entry point per gli utenti del sistema verso l'ambiente pervasivo.
Da un punto di vista funzionale, l'Ubiquitous Gateway dovrà:

- Registrare e autenticare gli utenti che effettuano l'accesso;
- Produrre la descrizione del dispositivo mediante il quale l'utente interagisce con l'ambiente;
- Offrire meccanismi per ricercare ed utilizzare le risorse disponibili nell'ambiente.

In figura 3.7, è mostrato il diagramma delle classi dell'Ubiquitous Gateway.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

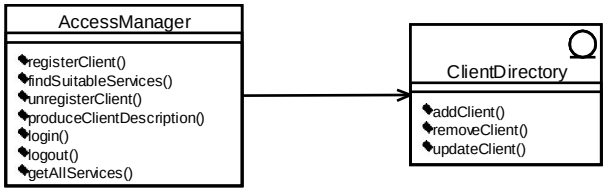

fig. 3.7: Diagramma delle classi dell'Ubiquitous Gateway.

3.6.1 La descrizione del dispositivo client

Caratterizzare il dispositivo mediante il quale un utente accede alle risorse dell'ambiente è un aspetto molto importante. Ad esempio, mentre è ragionevole supporre uno streaming audio-video su di un dispositivo PDA, non lo è altrettanto se il client è costituito da un telefono cellulare; o ancora, mentre per controllare l'impianto di riscaldamento di un locale, per un PDA potrebbe essere fornita un'interfaccia di tipo grafica, per un cellulare si dovrebbe pensare ad un'interfaccia a caratteri.

Ma più semplicemente, di uno stesso servizio potrebbero essere fornite più implementazioni dai requisiti non funzionali diversi (parametri di qualità, banda, ritardi, etc.) ed è utile sapere le caratteristiche del dispositivo che richiede il servizio, per poter scegliere l'implementazione che meglio gli si adatta.

Le caratteristiche del device, sia di carattere hardware che software, vengono descritte in un file DAML+OIL prodotto dall'Ubiquitous Gateway.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

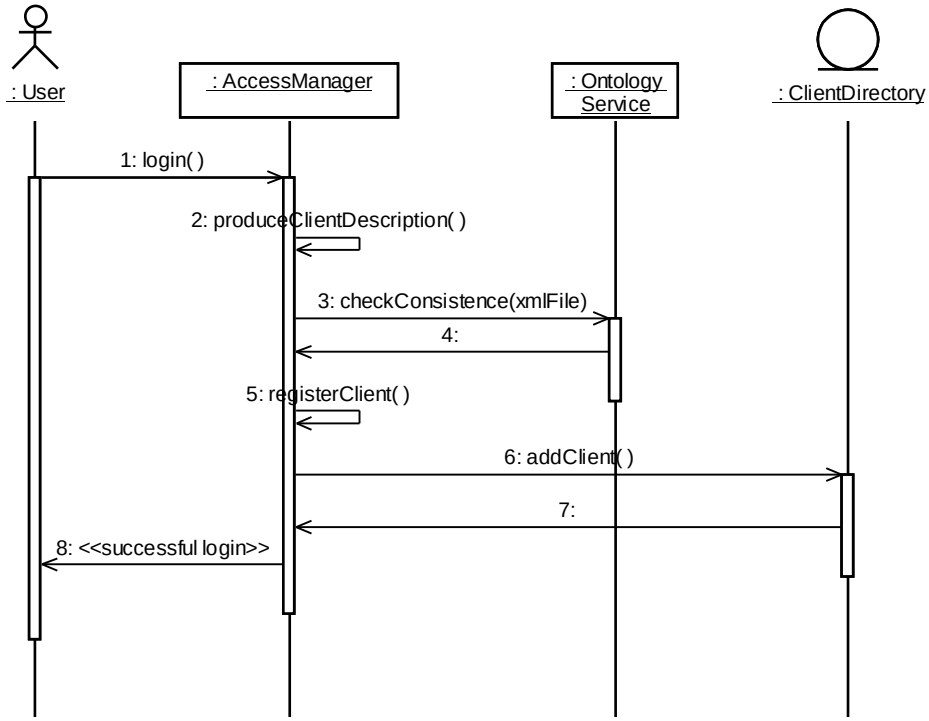

fig. 3.8: Ingresso di un nuovo utente nel sistema.

Come mostra lo scenario di figura 3.8, l'Ubiquitous Gateway, una volta definita la descrizione del dispositivo, ne verifica la consistenza tramite l'Ontology Service, dopodiché provvede a registrare il nuovo utente aggiornando il ClientDirectory.

Soltanto, dopo questa fase l'utente potrà effettivamente interagire con l'ambiente richiedendone i servizi.

3.6.2 Il discovery dei servizi

In figura 3.9, è rappresentato lo scenario d'interazione intrapreso dall'utente nel momento in cui, tramite l'Ubiquitous Gateway, intraprende il discovery di tutte le risorse disponibili nell'ambiente.

In questo scenario, non sono prese in considerazione le caratteristiche del dispositivo.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

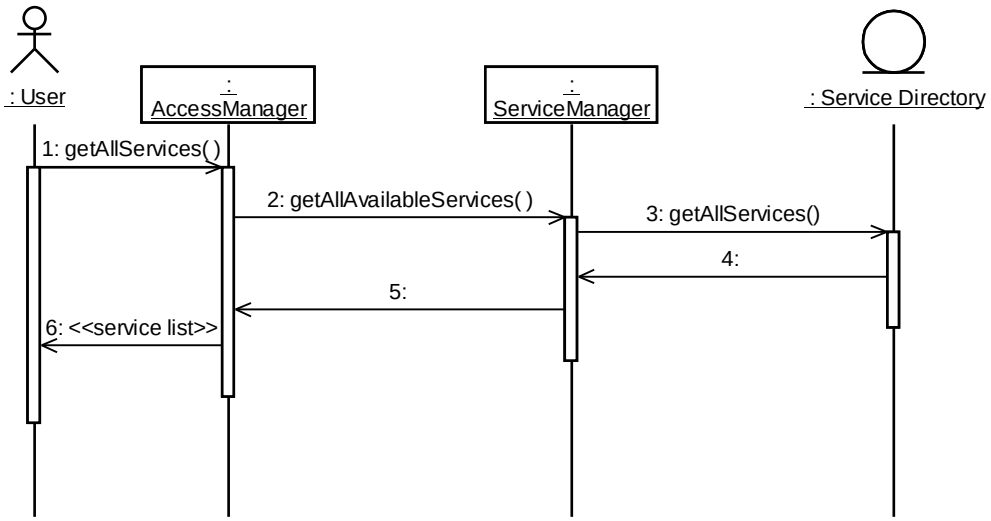

fig. 3.9: L'utente richiede la lista di tutti i servizi disponibili.

L'interazione, invece, si complica leggermente se l'utente è interessato a scoprire tutti e soli quei servizi compatibili con il proprio device.

Nel diagramma di figura 3.10, l'AccessManager prima di avviare il processo di discovery, ritira dalla ClientDirectory la descrizione del dispositivo dell'utente e la inoltra al ServiceManager; questi, dopo aver prelevato dalla Service Directory tutti i servizi disponibili, produce una lista con tutti e soli i servizi adattabili al dispositivo.

Laddove necessario, il ServiceManager interroga l'Ontology Service per stabilire le relazioni fra i concetti definiti dalle descrizioni.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

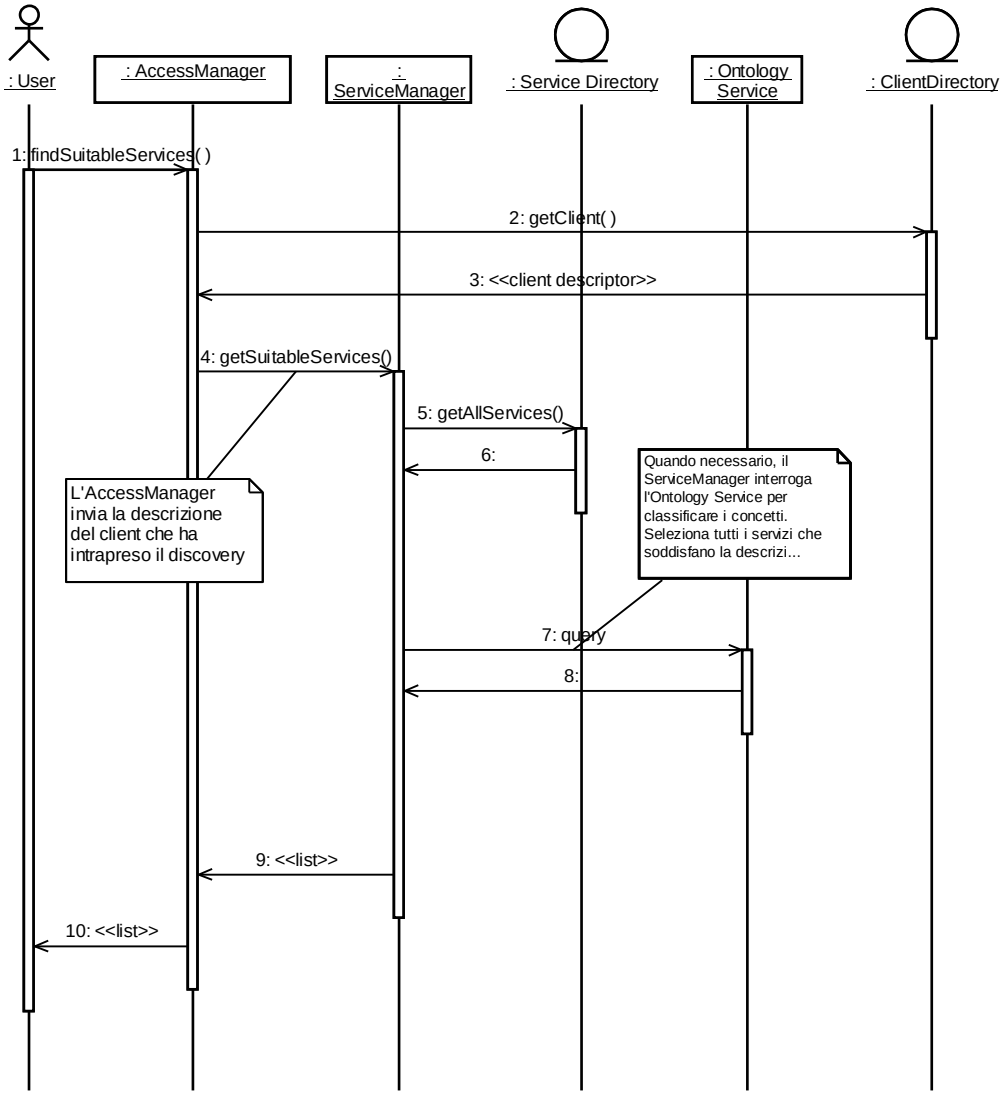

fig. 3.10: Discovery dei servizi secondo le caratteristiche del dispositivo.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Capitolo 4

I servizi di sistema implementati

4.1 Introduzione

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.

Come esaminato nel capitolo precedente 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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Dal punto di vista delle risorse e dei servizi pubblici, 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.

I servizi di sistema sono realizzati come servizi CORBA e solo se necessario, ovvero quando espongono funzionalità che devono essere rese accessibili dall'esterno, sono esposti come Web Services.
CORBA è l'infrastruttura di comunicazione tra i servizi di sistema. Tale scelta è legata alle seguenti considerazioni:

- CORBA è una tecnologia di comunicazione più sicura ed efficiente di quella su cui si basano i Web Services
- Allo stato attuale anche implementazioni gratuite di piattaforme CORBA offrono i servizi, ad esempio l'Event ed il Notification Service, descritti nelle specifiche di CORBA, in grado di offrire meccanismi per la realizzazione di tipologie di comunicazione, tra oggetti, diversa da quella sincrona.

In dettaglio, una comunicazione ad eventi, e quindi una interazione tra oggetti non più sincrona ma asincrona, è molto importante: UbiSystem, come precedentemente osservato, deve poter essere capace di gestire eventi che possono verificarsi in momenti casuali come, ad esempio:

- l'ingresso di una nuova persona all'interno dell'ambiente;
- la fruibilità di un nuovo tipo di servizio;
- l'installazione di nuovi dispositivi;
- il cambiamento del contesto dell'ambiente.

Nella progettazione dei servizi di sistema descritti nel prosieguo del capitolo, si è scelto di utilizzare, come implementazione della infrastruttura CORBA, JacORB.
JacORB è un Java object request broker open source, molto popolare (usato da enti di ricerca e in campo aziendale) ed efficiente che, progettato nel 1995 in un dipartimento della Freie Universit¨at Berlin (FUB), continua ad essere usato per ricerche alla FUB, specialmente nel campo della sicurezza di oggetti distribuiti.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La versione utilizzata è JacORB 2.2, comprensiva di alcuni servizi tra i quali il Name Service ed il Notification Service.

Nei successivi paragrafi si descriveranno in dettaglio i servizi di sistema che caratterizzano l'attuale implementazione del prototipo UbiSystem:

- l'Event Service
- l'Ontology Service
- il Location Service
- il Context Service

4.2 Event Service

L'Event Service è il servizio di sistema che offre il supporto per la comunicazione asincrona tra le diverse entità dell'ambiente. L'Event Service, si interpone fra i publishers ed i consumers ed agisce come una sorta di middleware per smistare i messaggi. La comunicazione fra publishers e consumers può essere divisa in due fasi. Il consumers comunica la sua sottoscrizione all'Event Service per un determinato canale d'interesse e l'Event Service colleziona i messaggi per conto del consumer.
Dall'altro lato, i publishers possono inviare messaggi di notifica all'Event Service che si occuperà, poi, di smistarli ai diretti interessati utilizzando l'apposito canale di interesse.
Nel prototipo di UbiSystem implementato l'Event Service è costituito dal Notification Service CORBA e da un provider JMS.
Per i servizi di sistema l'Event Service è il Notification Service: quando un servizio di sistema è publisher di eventi ricorre al Notification Service per la creazione e l'utilizzo di un apposito canale; analogamente un servizio di sistema che deve svolgere un ruolo di consumer di eventi deve ricorrere alle funzionalità del Notification Service per poter esporsi come consumatore degli eventi che viaggiano su un determinato canale.
I servizi esterni invece percepiscono L'Event Service come provider JMS su protocollo http: un servizio pubblico percepirà sempre gli eventi come messaggi JMS che viaggiano in una coda JMS.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La scelta del protocollo http come protocollo di trasporto per i messaggi è legata al fatto che i servizi presenti nell'ambiente sono Web Services o comunque applicazioni web-based.
Mediante utilizzo di appositi oggetti wrapper avviene la comunicazione asincrona tra servizi esterni, normalmente consumatori di eventi, e servizi di sistema, normalmente produttori di eventi. Come vedremo in dettaglio nel paragrafo relativo al Context Service, i suddetti oggetti wrapper hanno prevalentemente il compito di trasformare un canale ad eventi CORBA in una coda di messaggi JMS. In questo modo un servizio di sistema continuerà a inviare eventi attraverso un canale CORBA che gli viene fornito dal Notification Service ed un servizio pubblico continuerà a prelevare eventi da una coda JMS fornitagli dal provider JMS.

4.3 Ontology Service

L'Ontology Service è un servizio CORBA, che tramite wrapper si espone come Web Service. Realizzato in linguaggio Java, fornisce un'interfaccia generica per gestire ontologie espresse in DAML+OIL.
In particolare l'Ontology Service gestisce un'unica ontologia, nel proseguo del capitolo spesso denominata ontologia base, 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) [46], un classificatore di concetti open source molto popolare ed efficiente.

In questo lavoro di tesi sono state aggiunte delle nuove funzionalità all'Ontology Service, legate soprattutto alla estrazione, dalla ontologia di base, di informazioni sui dispositivi installati nell'ambiente.

Il modello architetturale dell'Ontology Service, invariato rispetto l'implementazione precedente, è illustrato di seguito nella figura 4.1.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

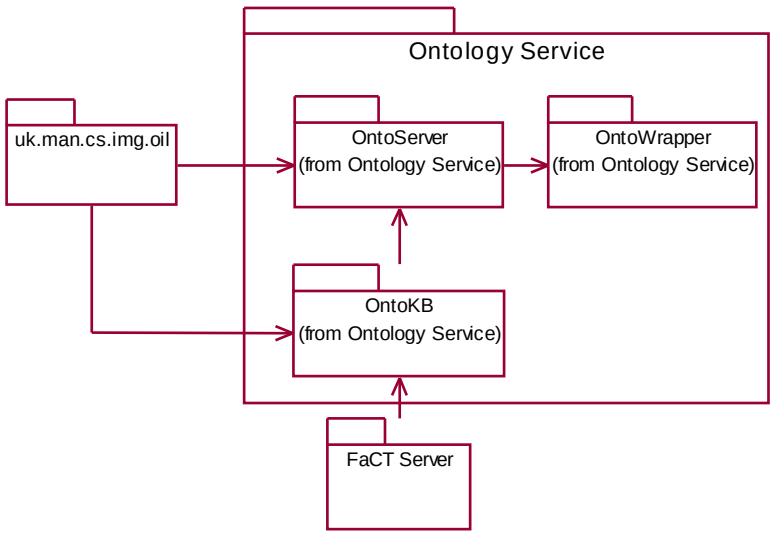

fig. 4.1: Modello architetturale dell'Ontology Service.

Esso presenta tre componenti fondamentali: l'*OntoKB*, l'*OntoServer* e l'*OntoWrapper*.

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.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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.

Nel riquadro sottostante è riportato uno stralcio dell'interfaccia IDL del servizio. In neretto sono evidenziati i metodi aggiunti.

```
module sGn {

module Ontology {

   /* Eccezioni generate */
   exception InvalidOntology {};
   exception OntologyNotFound {};
   exception OntologyNotLoaded {};
   exception ClassNotFound {};
   exception IndividualNotFound {};
   exception FactServerNotFound {};
   exception PropertyNotFound {};

   /* Definizione di lista di stringhe */
   typedef sequence<string> stringList;
   typedef sequence<string> stringListOfDevices;

   interface OntologyServer {

      void initFromFile (in string uri)
         raises (OntologyNotFound, InvalidOntology);

      void clearOntology();

      boolean entityConnection (in string xmlDescription)
         raises (InvalidOntology, OntologyNotLoaded);
      /* Quando una nuova Entity entra nell'ambiente, invia la propria
        descrizione XML. L'Ontology server converte la descrizione, la invia
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
   al FaCT Server e ne controlla la validità. */

stringList getListOfClasses (in string searchString)
   raises (OntologyNotLoaded);
/* Effettua una ricerca nell'Ontologia del parametro searchString */

stringList getAllIndividualOfClass (in string className)
   raises (OntologyNotLoaded, ClassNotFound);
/* Ritorna una lista di tutti gli individuals di una classe */

stringList getAllPropertiesOfClass (in string className)
   raises (OntologyNotLoaded, ClassNotFound);
/* Ritorna una lista con le proprietà di una classe */

      stringList getAllPropertiesOfIndividual (in string indName)
            raises (OntologyNotLoaded, IndividualNotFound);
/* Ritorna una lista con le proprietà di un individual */

stringList getAllSupers (in string className)
   raises (OntologyNotLoaded, ClassNotFound);
/* Ritorna una lista con le classi delle quali className è figlia */

stringList getAllSubs (in string className)
   raises (OntologyNotLoaded, ClassNotFound);
/* Ritorna una lista con le classi figlie di className */

stringList getPropertyValuesOfIndividual (in string individualName, in string propName)
   raises (OntologyNotLoaded, IndividualNotFound, PropertyNotFound);
/* Ritorna i valori delle proprietà di un individual dell'ontologia */

    …
    …

    stringList getDevices() raises (IndividualNotFound);
    stringList getDevicesinLocation(in string locationID) raises (IndividualNotFound);
    stringList getDevicesinRoom(in string roomID) raises (IndividualNotFound);
    stringList getDevicesinBuilding(in string buildingID) raises (IndividualNotFound);
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
        stringList getFiltredDevicesFromDevices(in stringListOfDevices arrayOfdevices,in string propName,in string propValue) raises (IndividualNotFound);
        stringList getDevicesFiltredbypropertyvalue(in string propName,in string propValue) raises (IndividualNotFound);
        stringList getDevicesFiltredbypropertyvalueinLocation(in string locationID,in string propName,in string propValue) raises (IndividualNotFound);
        stringList getDevicesFiltredbypropertyvalueinRoom(in string roomID,in string propName,in string propValue) raises (IndividualNotFound);
        stringList getDevicesFiltredbypropertyvalueinBuilding(in string buildingID,in string propName,in string propValue) raises (IndividualNotFound);

        string getLocationofDevice(in string deviceID);
        string getRoomofDevice(in string deviceID);
        string getBuildingofDevice(in string deviceID);

    }; // interface OntologyServer

}; // module Ontology

}; // module sGn
```

fig. 4.2: Interfaccia IDL dell' Ontology Service.

L'OntoServer 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.

L'OntoServer fornisce funzionalità per operare su ontologie e descrizioni di entità:

- *initFromFile*. Il parametro uri contiene l'indirizzo del file daml dal quale leggere l'ontologia. Legge il file, con l'ausilio del parser del package oil.jar converte il file in formato shiq, lo invia al FaCT Server e controlla la soddisfabilità dell'ontologia;
- *entityConnection*. Quando una nuova Entity, ad esempio un servizio pubblico, entra nell'ambiente, invia la propria descrizione XML. L'Ontology server converte la descrizione, la invia al FaCT Server e ne controlla la validità;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *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;

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

- *getAllIndividualOfClass* consente di ricavare dalla KB tutte le istanze relative ad una data classe. Viene prima verificata l'appartenenza 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'appartenenza 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'appartenenza 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.

Alle funzioni sopra esposte sono stati aggiunti i seguenti metodi per estrarre, dall'ontologia di base, le informazioni relative ai dispositivi installati nell'ambiente pervasivo: è previsto, in una implementazione futura di UbiSystem, un servizio di sistema che guiderà l'installazione dei dispositivi che dovranno far parte dell'ambiente pervasivo, ad esempio sensori, stampanti, proiettori, telecamere etc. , e che dovrà fornire per ciascun dispositivo una sua descrizione

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

ontologica. Quest'ultima, inserita nella ontologia di base fin quando il dispositivo non verrà disinstallato dall'ambiente, conterrà, e quindi anche consentirà di estrarre, ad esempio, informazioni riguardanti specifiche hardware e software del dispositivo, servizi offerti dal dispositivo e posizione di quest'ultimo all'interno dell'ambiente pervasivo.

- *getDevices* consente di recuperare tutte le istanze della classe Devices presenti nella ontologia. Restituisce, in particolare, gli identificativi, con cui sono gestiti nella ontologia di base, dei dispositivi installati nell'ambiente;
- *getDevicesinLocation* restituisce i dispositivi installati all'interno della *location* specificata. Una *location* è un termine usato all'interno della ontologia ed è da interpretare come un ambiente atomico. Una *location* , a seconda del dettaglio che si vuole raggiungere, può essere ad esempio una scrivania, una bacheca, un muro o sezione di muro, un punto geometrico rappresentato da proprie coordinate spaziali;
- *getDevicesinRoom* ritorna i dispositivi installati all'interno della stanza specificata;
- *getDevicesinBuilding* ritorna i dispositivi installati all'interno dell'edificio indicato come parametro di ingresso;
- *getFiltredDevicesFromDevices* restituisce, da un insieme di dispositivi, i dispositivi che soddisfano il valore di una specificata *property*. Nell'ambito di una ontologia una *property* è una proprietà associata ad un oggetto caratterizzata da un nome e da un insieme di valori che essa può assumere. Tra parametri di ingresso il metodo riceve i parametri denominati *propName* e *propValue* che indicano rispettivamente il nome della *property* ed il valore che la *property* deve assumere. Se si settasse ad esempio *propName* a *devicetype* e *propValue* a *printer*, il metodo estrarrebbe tutte le stampanti dall'insieme dei dispositivi specificato.
- *getDevicesFiltredbypropertyvalue* restituisce, da tutti dispositivi presenti nell'ambiente, quelli che soddisfano il valore di una specificata *property*;
- *getDevicesFiltredbypropertyvalueinLocation* restituisce, da tutti dispositivi presenti nella location specificata, quelli che soddisfano il valore di una specificata *property*;
- *getDevicesFiltredbypropertyvalueinRoom* restituisce, da tutti dispositivi presenti nella stanza specificata, quelli che soddisfano il valore di una specificata *property*;;
- *getDevicesFiltredbypropertyvalueinBuilding* restituisce, da tutti dispositivi presenti nell'edificio specificato, quelli che soddisfano il valore di una specificata *property*;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *getLocationofDevice* estrae l'informazione sulla location ove il dispositivo è stato installato;
- *getRoomofDevice* estrae la stanza nella quale il dispositivo è installato;
- *getBuildingofDevice* estrae l'edificio in cui il dispositivo è installato.

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 come un servizio CORBA, attivo ed esistente di per sé, utilizzabile dagli altri servizi di sistema 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:

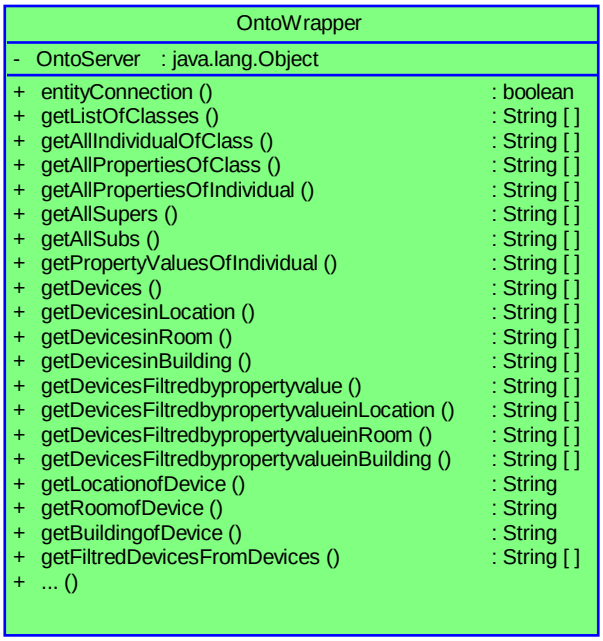

fig. 4.3: La Classe OntoWrapper

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Essa, in pratica, funge da client CORBA per l'OntoServer: il suo costruttore adopera il Naming Service per ottenere il riferimento all'oggetto OntoServer 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 [3] 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 [2], che mettono a disposizione una serie di toolkit che rendono estremamente semplice la realizzazione di Web Service a partire da applicazioni Java.

Nel paragrafo relativo all'esposizione del Context Service come Web Service comunque verrà descritta in dettaglio la procedura per la pubblicazione di un servizio CORBA in Web Service.

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

4.3.4 Modifiche apportate all'ontologia di base

Rispetto alla precedente implementazione dell'Ontology Service, sono state apportate modifiche riguardanti l'ontologia di base che il servizio carica al suo avvio e sulla quale avvengono tutti i ragionamenti. Per essere context-aware le applicazioni necessitano di un'infrastruttura composta da sensori che percepiscono le informazioni di contesto e da 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. È stata quindi proposta e sviluppata una nuova ontologia, che arricchisce quella di base, che definisce informazioni fisiche di contesto, andando a descrivere ambienti con confini geografici ben identificabili come edifici e stanze. Non esistendo standard o

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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. L'attuale ontologia di contesto proposta è comunque stata gerarchicamente progettata per eventuali estensioni future.

Di seguito sono analizzate le singole classi dell'ontologia sviluppata:

- *Place* rappresenta la classe di tutti i possibili ambienti rappresentati nell'ontologia.
- *AtomicPlace* è la classe generale che descrive gli ambienti atomici, cioè ambienti che non contengono altri ambienti ma che sono contenuti in ambienti composti.
- *CompoundPlace* è la classe generale che descrive tutti gli ambienti composti, ossia ambienti che ne contengono altri e che possono essere a loro volta contenuti.
- *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.

- *InnerLocation* è la classe rappresentante le *location* sopra descritte: una *location* è un generico ambiente atomico identificabile all'interno di una stanza o comunque all'interno di un *CompoundPlace*.
- *Building* rappresenta un edificio con un proprio indirizzo, un certo numero di piani e contenente stanze.
- *Room* rappresenta una stanza sempre contenuta in un edificio e contenente sempre locazioni interne.
- *Floor* rappresenta un piano di un edificio. È caratterizzato da un numero, il *FloorNumber*, e deve contenere almeno una locazione interna.

La rappresentazione gerarchica delle classi sopra descritte è illustrata di seguito nella figura 4.4. Per quanto affermato sopra, l'ontologia di contesto proposta è stata gerarchicamente progettata per eventuali estensioni future. In tal senso va giustificata la classe generale *Envirorment* presente in figura .

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

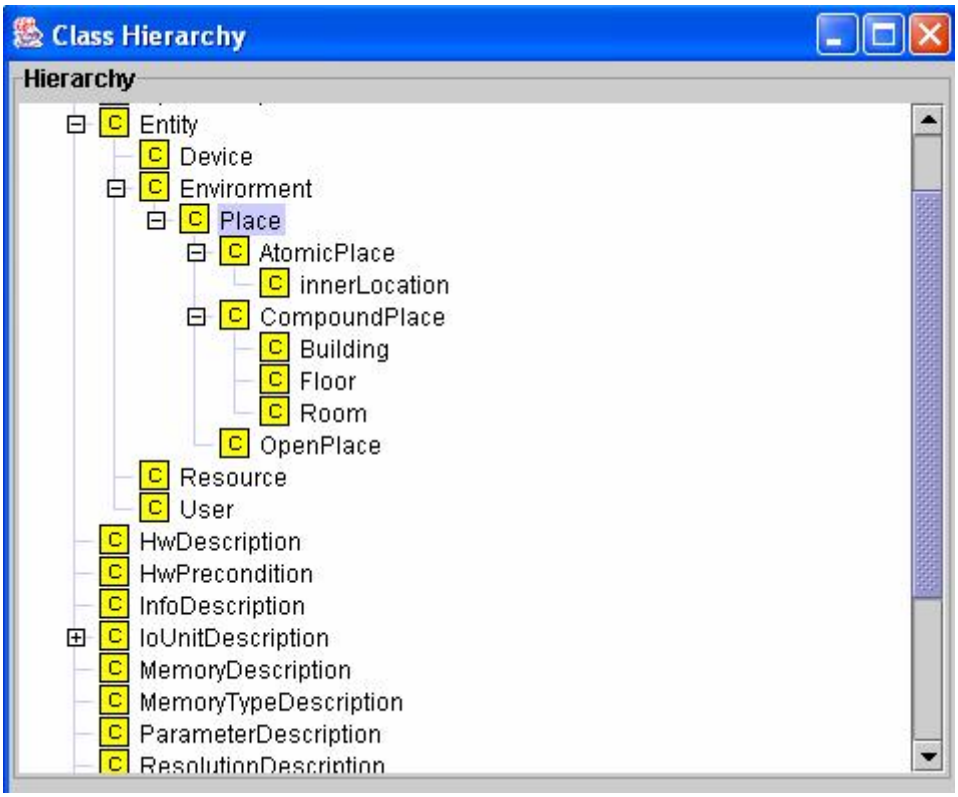

fig. 4.4: Rappresentazione gerarchica, all'interno dell'ontologia di base, delle classi costituenti l'ontologia di contesto.

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 locazioni interne poiché esse sono *AtomicPlace* e non possono contenersi fra loro.

Inoltre attraverso le informazioni di contesto aggiunte, è possibile mostrare, ad esempio ad un utente, non tutte le risorse che offre l'edificio ma solo le risorse a cui può accedere trovandosi nella stanza in cui è stato rilevato dai sensori, condizionando quindi il suo comportamento e le sue possibili scelte in base al contesto in cui è in quel momento inserito.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

4.4 Location Service

Il Location Service ha il compito di estrarre le informazioni relative alla posizione di un determinato dispositivo mobile, oppure di un utente (anche se attualmente rappresentato da un dispositivo), all'interno dell'ambiente. Tali informazioni vengono estratte quando:

- un servizio di sistema invoca uno dei metodi esposti dal Location Service attraverso il suo file IDL;
- un dispositivo si aggancia alla rete o si sgancia da quest'ultima.

Il Location Service è quindi in grado di interagire con gli altri servizi di sistema attraverso una modalità di comunicazione sincrona ma anche attraverso una modalità di comunicazione ad eventi, a cui, di seguito, si farà riferimento come modalità di comunicazione asincrona.

Il Location Service soddisfa la proprietà, necessaria in un ambiente pervasivo, di location-awareness.

Il Location Service è un servizio CORBA e non si espone anche come Web Service.
Le funzionalità offerte dal Location Service sono progettate per essere utilizzate solo dai servizi di sistema, nel prototipo implementato di UbiSystem il ruolo di client è generalmente coperto dal Context Service, e quindi le funzionalità del servizio non devono essere rese accessibili anche ai servizi pubblici.

La figura 4.5 illustra il modello architetturale del Location Service.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

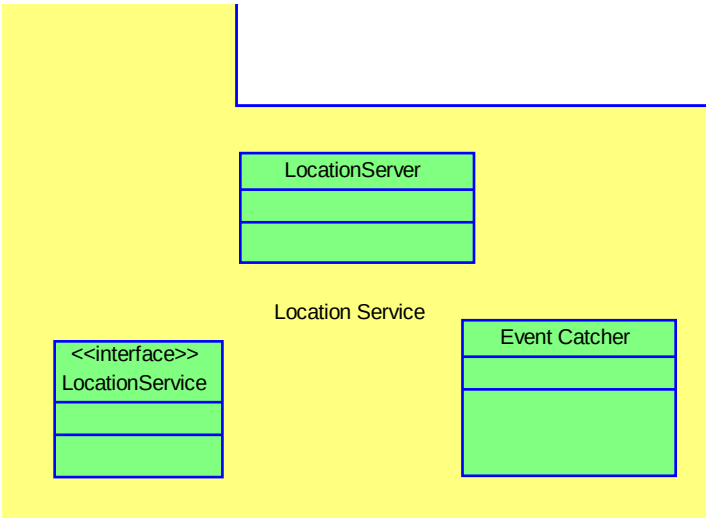

fig. 4.5 :modello architetturale del Location Service

Il *LocationServer* è il componente che si occupa della procedura di inizializzazione del Location Service.

Il *LocationService* è l'interfaccia java corrispondente all'interfaccia IDL del servizio. In particolare, i metodi esposti sono quelli attraverso i quali il servizio stabilirà con gli altri servizi di sistema una comunicazione di tipo sincrono.

L'*Event Catcher* è l'oggetto java che si occupa della ricezione di eventi e della loro elaborazione. Nel Location Service gli aspetti legati alla modalità di comunicazione asincrona sono a cura di quest'ultimo componente.

La procedura di inizializzazione del servizio da parte del *LocationServer* è descritta in dettaglio mediante la figura 4.6.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

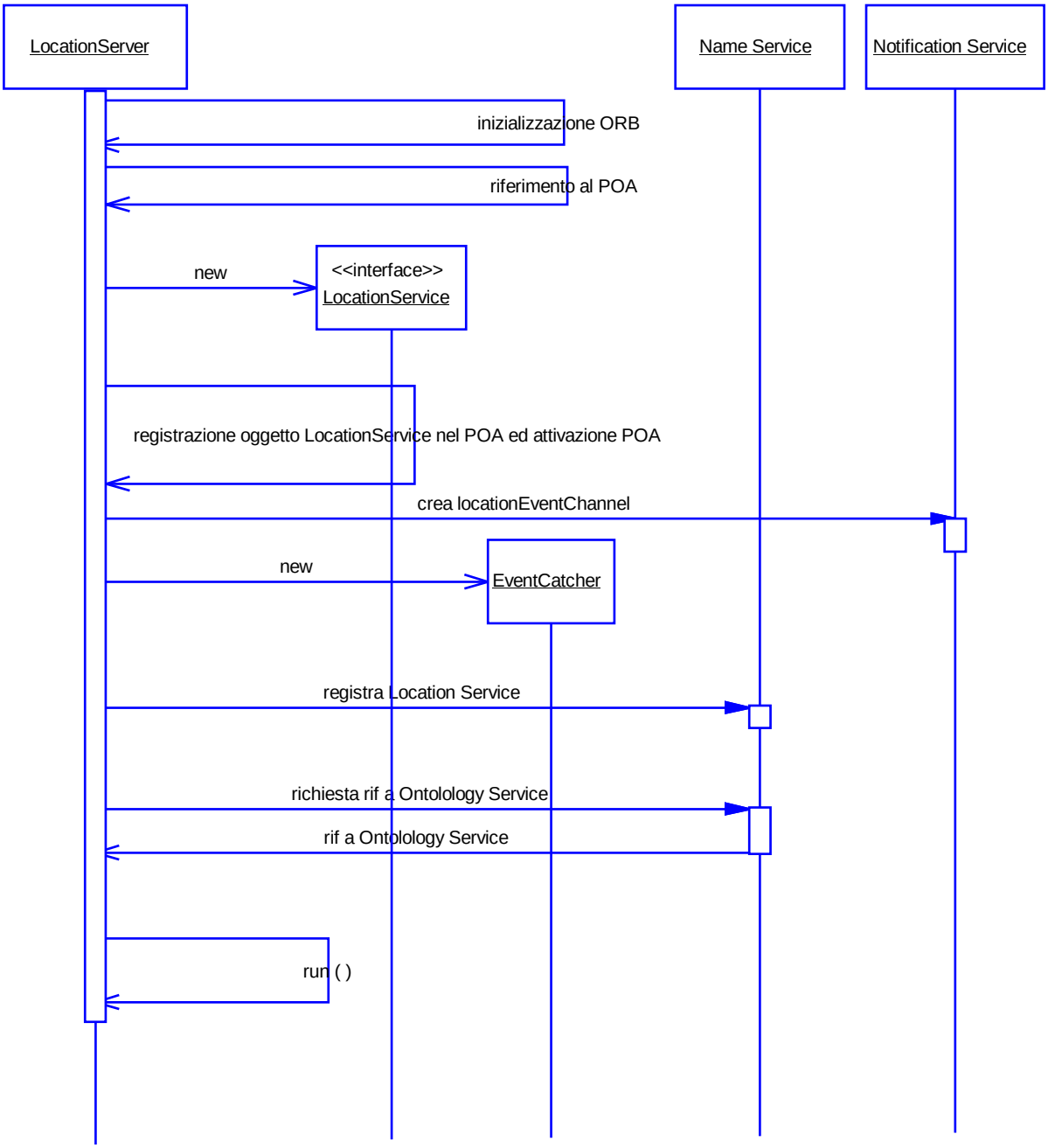

fig. 4.6:diagramma di sequenza associato alla inizializzazione del Location Service.

In particolare appartengono alla fase di inizializzazione:

- le operazioni mediante le quali il servizio si espone come servizio CORBA;
- la creazione dell'*LocationService* a cui spetta la gestione della richieste sincrone;
- la creazione dell'*Event Catcher* per la rilevazione e l'inoltro degli eventi;
- la risoluzione del riferimento all'Ontology Service attraverso il Name Service.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Affinché il servizio operi correttamente, infine, è necessario che includa alcune librerie, viste come black box, denominate, in questa sede, *librerie di gestione dei sensori*. Tali librerie rappresentano una soluzione alternativa e temporanea al DHCP Service essendo quest'ultimo ancora assente nella attuale implementazione di UbiSystem. Attraverso l'utilizzo di tali librerie il Location Service è in grado di capire quale sensore ha rilevato uno specifico dispositivo mobile oppure quando e dove un dispositivo mobile si aggancia alla rete.

4.4.1 Comunicazione sincrona

I metodi attraverso i quali il Location Service realizza una comunicazione sincrona con gli altri servizi sono tutti quelli esposti tramite interfaccia IDL del servizio.
L'interfaccia IDL del servizio è illustrata di seguito nella figura 4.7.

```
/*IDL Location Service */

module LS {

        exception IndividualNotFound {};

        interface LocationService {

                string getLocation (in string IP) raises (IndividualNotFound);
                string getRoom (in string IP) raises (IndividualNotFound);
                string getBuilding (in string IP) raises (IndividualNotFound);
        };
};
```

fig. 4.7: interfaccia IDL del Location Service.

I metodi esposti consentono l'estrazione di informazioni riguardanti la posizione di uno specifico dispositivo:

- *getLocation* restituisce la *location* di un dispositivo a cui è associato un determinato IP.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *getRoom* restituisce la stanza in cui il dispositivo si trova.
- *getBuilding* restituisce l'edificio in cui il dispositivo è stato rilevato.

Il Location Service comunica solo con altri servizi di sistema. Nel prototipo implementato di UbiSystem il ruolo di client del servizio è interpretato dal ContextService: la figura 4.8 illustra l'interazione tra il Location Service e gli altri servizi di sistema implementati all'invocazione del metodo *getRoom*.

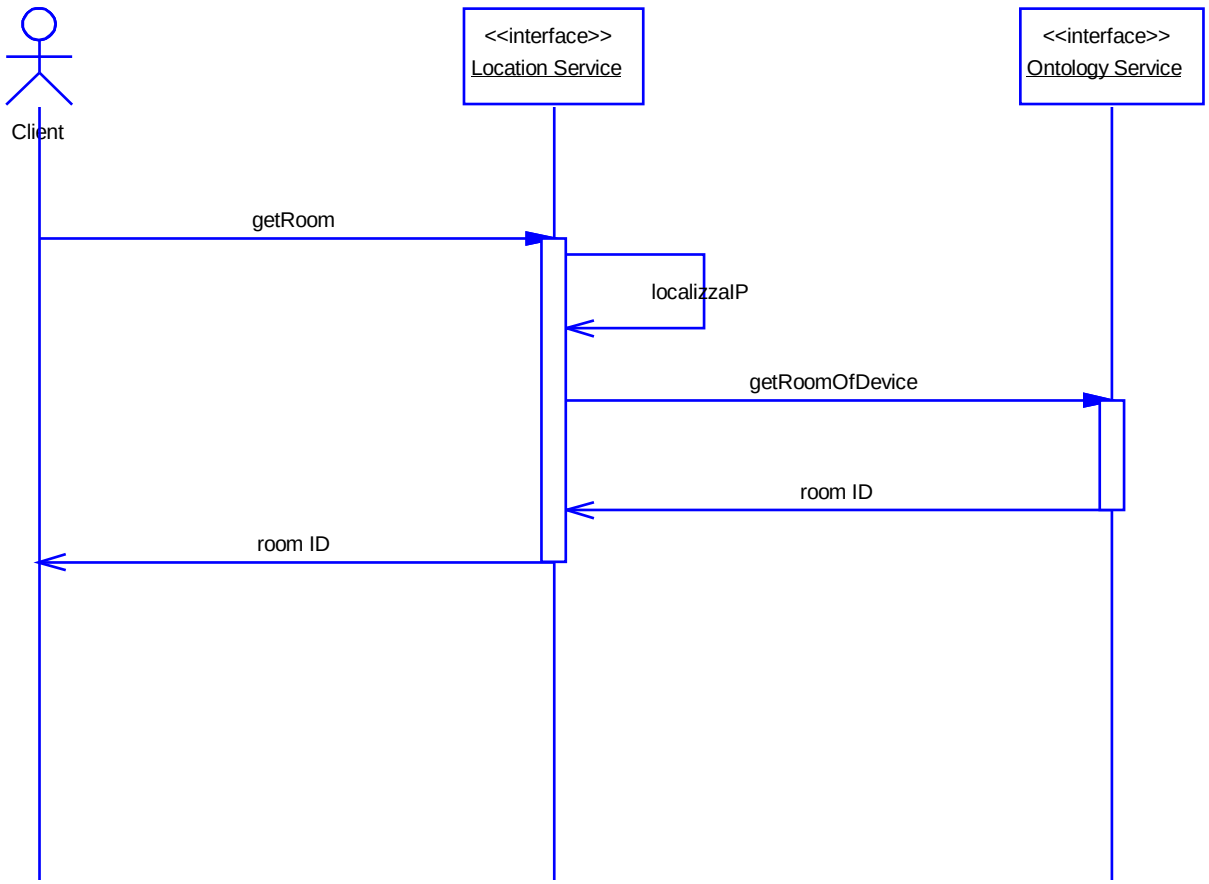

fig. 4.8: modello di interazione per la funzione getRoom.

Come si evince dalla figura il Location Service, ricevuta la richiesta dal client, ovvero dal Context Service, esegue alcuni passi elaborativi al termine dei quali restituisce l'identificativo della stanza in cui è stato rilevato il dispositivo specificato attraverso il suo IP.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Più in dettaglio il Location Service, ricevuto l'IP del dispositivo, invoca il metodo *localizzaIP* che, attraverso le librerie per la gestione dei sensori presenti nell'ambiente, ritorna l'identificativo del sensore che ha rilevato il dispositivo richiesto.
Ottenuto in tal modo l'ID del sensore, viene recuperata la stanza in cui è stato installato il sensore, e di conseguenza la stanza in cui si trova il dispositivo richiesto, mediante una opportuna query all'Ontology Service.

Lo scenario sopra esposto è simile per le funzioni *getLocation* e *getBuilding*. Di seguito sono riportati i modelli di interazione per questi due metodi ed è stato messo in rilievo il ruolo di client principale, del servizio in esame, che il Context Service assume.

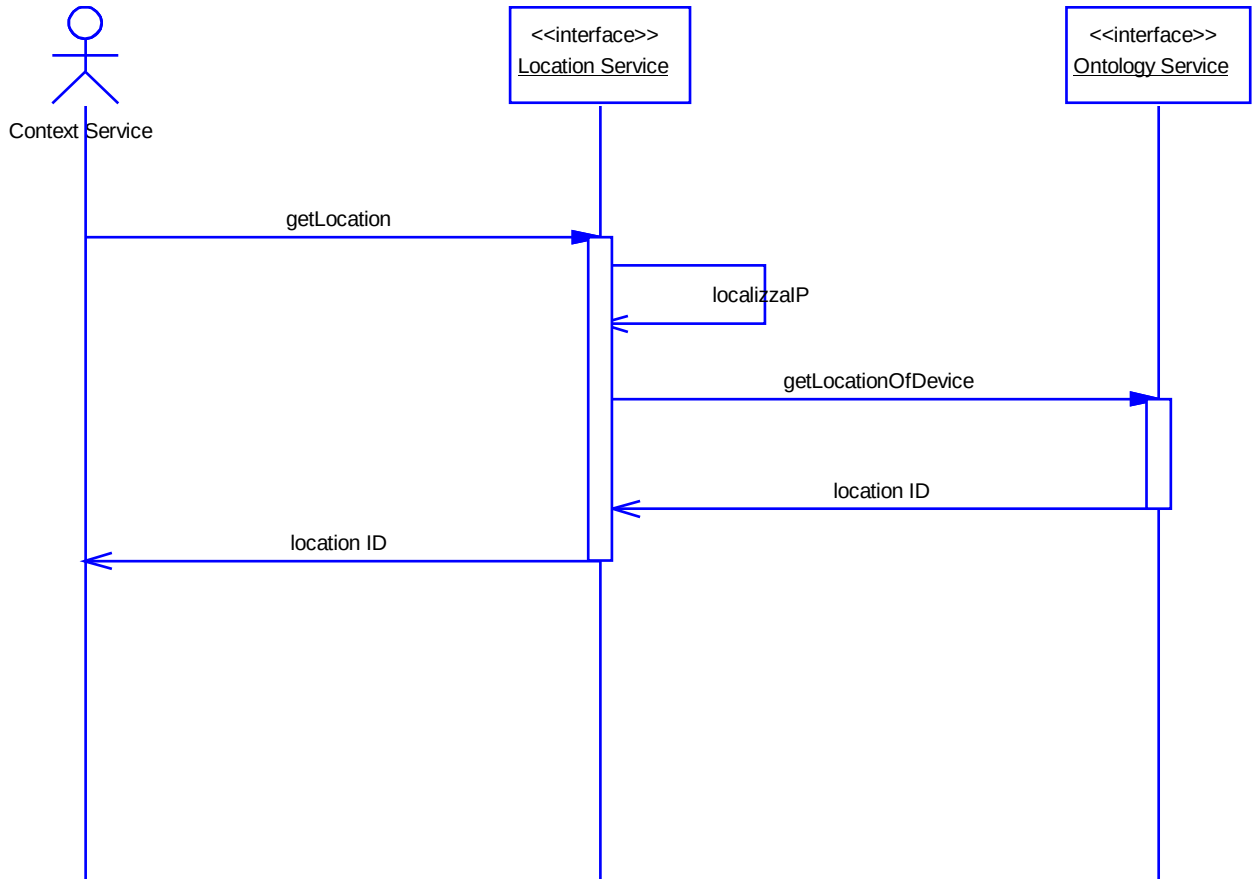

fig. 4.9: modello di interazione per la funzione getLocation.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

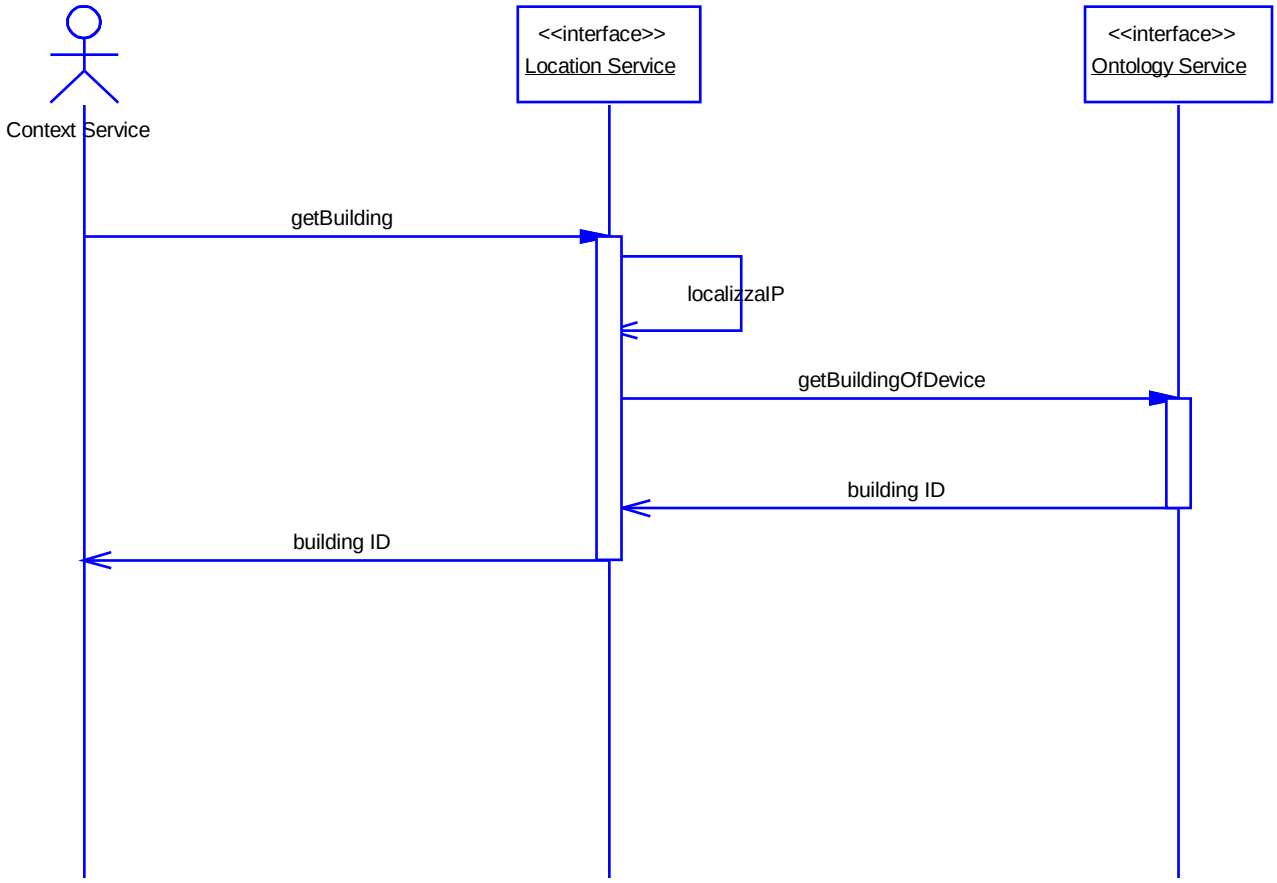

fig. 4.10: modello di interazione per la funzione getBuilding.

4.4.2 Comunicazione asincrona

Il servizio notifica, sotto forma di eventi, gli spostamenti dell'utente, oppure di un dispositivo mobile, e comunica tali eventi ai servizi di sistema che lo richiedono, tra i quali, in particolare, il Context Service. Tra il Location Service ed il Context Service è stata quindi implementato un meccanismo di comunicazione asincrona mediante un apposito canale ad eventi messo a disposizione dal Notification Service di JacORB.

Quando un sensore rileva un dispositivo, le suddette librerie di gestione dei sensori generano un evento memorizzandolo in un apposito buffer. Tale evento è un oggetto java costituito dalla coppia *IP del dispositivo rilevato* ed *ID del sensore* che ha rilevato il dispositivo. L'Event

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Catcher, mediante un metodo denominato *getEvent*, richiamato in modo ciclico, preleva l'evento dal buffer e lo trasforma nella coppia *IP del dispositivo rilevato* e *ID della location* in cui si trova il sensore e quindi il dispositivo.
L'evento viene poi inserito nel canale di comunicazione, denominato *locationEventChannel* e creato nella fase di inizializzazione del servizio, tra il Location Service ed il Context Service. Anche questa operazione è a cura dell'Event Catcher.

La figura 4.11 descrive le operazioni suddette svolte dall'Event Catcher.

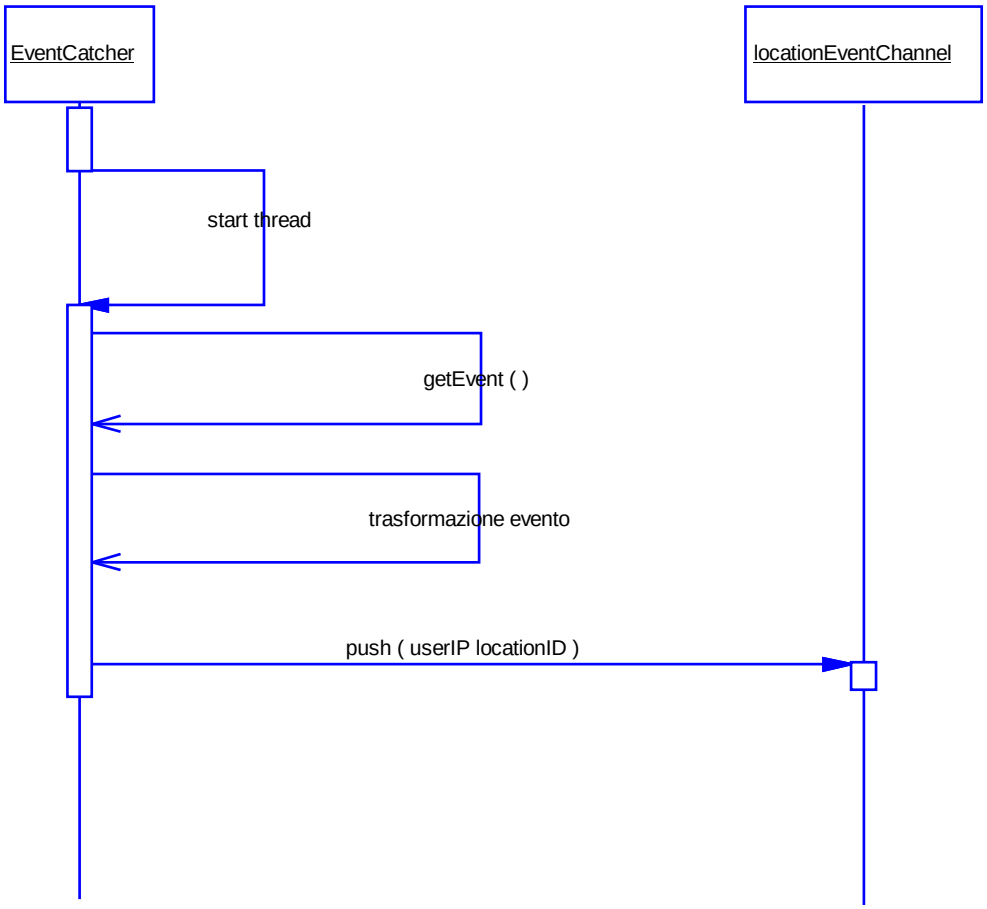

fig. 4.11: gestione eventi da parte dell'Event Catcher nella ipotesi che vi sia un evento nel buffer.

Come si evince dal diagramma di sequenza sopra illustrato, l'Event Catcher non effettua alcun controllo, prima dell'operazione di push sul canale, per capire se ci sia o meno un consumatore,

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

nello specifico il Context Service, effettivamente interessato all'evento. In altre parole il Location Service effettua sempre un push sul canale ad ogni evento rilevato.
La gestione degli eventi inutili quindi non è ad opera del Location Service ma ad opera del Notification Service. Quest'ultimo elimina gli eventi dal canale qualora non vi sia alcun consumatore interessato. La gestione degli eventi presenti nel canale da parte del Notification Service è efficiente e tale da non giustificare un appesantimento del Location Service in termini di gestione degli eventi inutili.

4.5 Context Service

Mentre il Location Service garantisce la proprietà di *location-awareness*, è compito del Context Service garantire la proprietà di *context-awareness*.

Il Context Service offre funzionalità per la rilevazione ed elaborazione di informazioni sul contesto nell'ambiente UbiSystem.

I Client del Context Service possono essere sia i servizi di sistema che i servizi pubblici.
Pertanto, coerentemente a quanto descritto nella introduzione di questo capitolo, il Context Service è un servizio CORBA che si espone mediante wrapper come Web Service.

4.5.1 Il Context Service come servizio CORBA

Il Context Service è visto dagli altri servizi di sistema come servizio CORBA. In figura 4.12 viene illustrato il suo modello architetturale.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

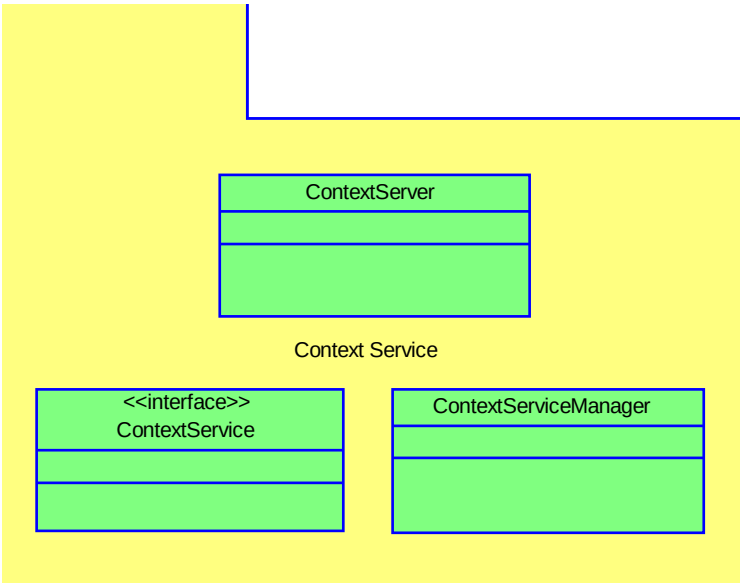

fig. 4.12: modello architetturale del Context Service.

Il *ContextServer* si occupa della esposizione del servizio come servizio CORBA e della risoluzione dei riferimenti agli altri servizi di sistema.
Il *ContextService* è l'interfaccia java corrispondente all'interfaccia IDL del servizio. Si occupa della gestione di tutte le richieste che giungono al servizio di contesto da eventuali client.
Il *ContextServiceManager* si occupa degli aspetti di comunicazione asincrona che il servizio deve coprire. Nel paragrafo relativo alla comunicazione asincrona verranno presentate ed illustrate le funzionalità offerte dal *ContextServiceManager* ed il ruolo che esso effettivamente copre.

Il seguente diagramma di sequenza, figura 4.13, illustra l'inizializzazione del Context Service.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

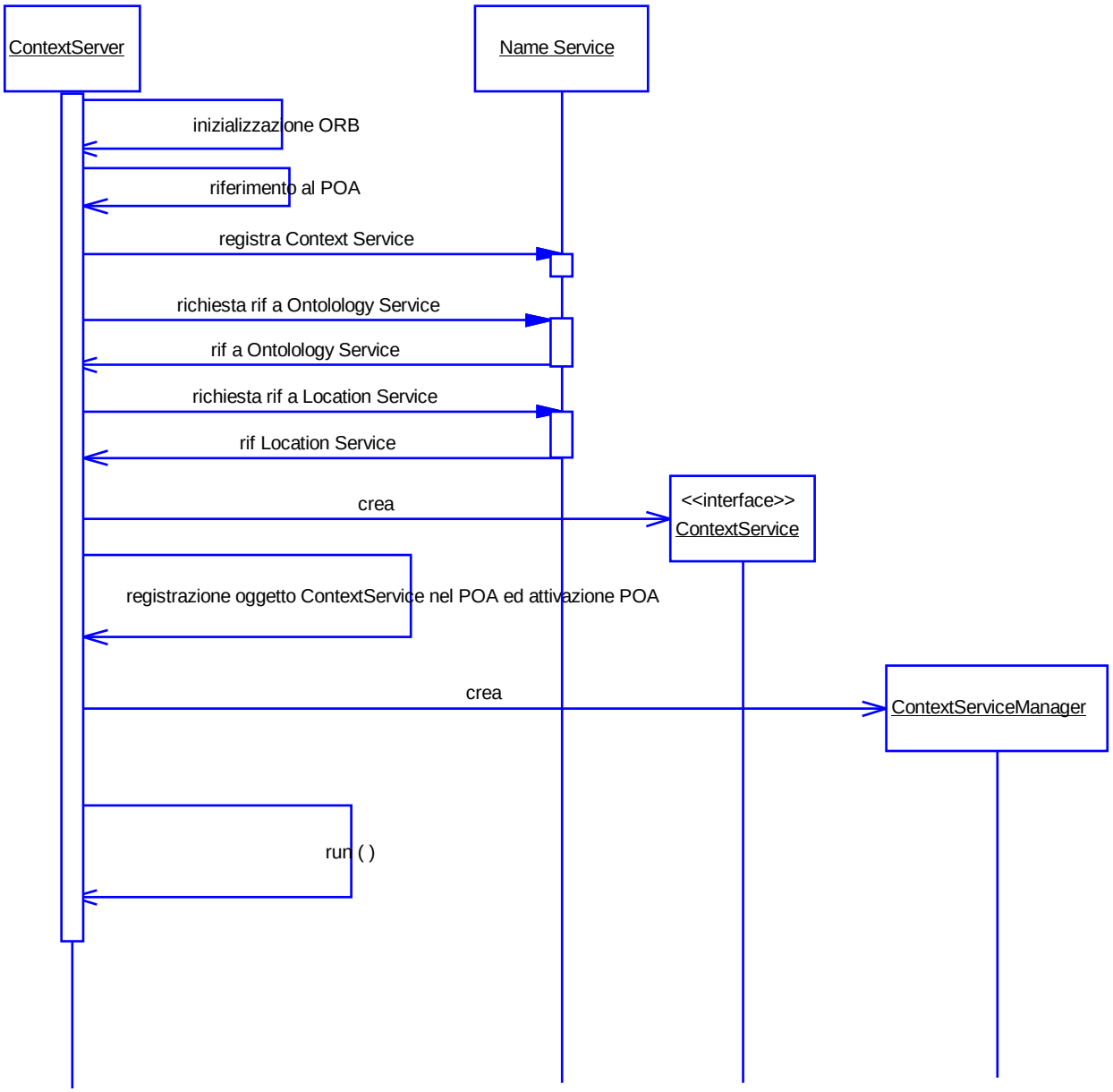

fig. 4.13: inizializzazione del Context Service.

Si osservi in particolare che nella fase di inizializzazione il Context Service non è ancora a conoscenza del *locationEventChannel* attraverso il quale il Context Service comunica indirettamente con il Location Service. Questo perché il Context Service dovrà interessarsi agli eventi contenuti nel suddetto canale solo quando sarà effettivamente richiesto da parte di un client.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Il Context Service espone metodi che consentono il recupero di informazioni legate a diverse tipologie di contesto.

In sede di progetto si è stabilito che il Context Service mettesse a disposizione metodi attraverso l'invocazione dei quali un client fosse facilmente in grado di reperire le informazioni di contesto a cui fosse interessato.
Il Context Service non definisce e costruisce un contesto specifico, ad esempio un contesto di utente, a cui i client devono uniformarsi, ma espone metodi affinché ciascun client possa estrarre le informazioni di contesto secondo la propria visione ed interpretazione di contesto: come definito nel precedente capitolo per "contesto" si intende qualsiasi informazione che possa essere usata per caratterizzare il comportamento di una persona o di entità software e quindi è chiaro che a client diversi possano corrispondere interpretazioni di contesto anche molto diverse tra loro.
Da questa considerazione si intuisce anche che i metodi attualmente implementati dal servizio di contesto potrebbero essere comunque non sufficienti a soddisfare particolari richieste.
Il Context Service è stato pertanto progettato ed implementato in modo da poter soddisfare una certa gamma di probabili richieste ed in modo da semplificare l'aggiunta di nuovi metodi.

La figura 4.14 illustra l'interfaccia IDL che il Context Service espone ed implementa.

```
/* IDL Context Service */

module CS {

exception NoSuchElement {};

typedef sequence<string> devices;
typedef string identificativo_canale;

interface ContextService {

/*metodi per recuperare informazioni sulla locazione e sulla stanza dell'utente                    */
/*FLAG =0 => viene restituito l'ID della location o della room così come memorizzato nella ontologia */
/*FLAG =1 => viene restituito il NOME della location o della room                                   */
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
string get_location_of_user_IdentifiedByHisDevice(in string userIP,in boolean Flag);

 string get_room_of_user_IdentifiedByHisDevice(in string userIP,in boolean Flag);

 string get_building_of_user_IdentifiedByHisDevice(in string userIP,in boolean Flag);

/*rilevazione dispositivi senza alcun filtraggio effettuata a livello di Location,di Room e di Building */

devices getDevicesInLocationOfUserIdentifiedByHisDevice(in string userIP) raises (NoSuchElement);

devices getDevicesInRoomOfUserIdentifiedByHisDevice(in string userIP) raises (NoSuchElement);

devices getDevicesInBuildingOfUserIdentifiedByHisDevice(in string userIP) raises (NoSuchElement);

/*rilevazione dispositivi con filtraggio effettuata a livello di Location,di Room e di Building*/
/*il filtraggio si basa sullo specifico  valore di una proprietà*/

devices getFiltredDevicesInLocationOfUserIdentifiedByHisDevice(in string userIP,in string propName,in string propValue) raises (NoSuchElement);

devices getFiltredDevicesInRoomOfUserIdentifiedByHisDevice(in string userIP,in string propName,in string propValue) raises (NoSuchElement);

devices getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice(in string userIP,in string propName,in string propValue) raises (NoSuchElement);

/* metodi per consentire un monitoraggio continuato */
/* NB:i metodi con monitoraggio continuato restituiscono tutti un identificativo di canale su cui i client dovranno mettersi in ascolto*/

void close_connection(in string id_canale);

identificativo_canale getDevicesInLocationOfUserIdentifiedByHisDevice_monitored (in  string userIP);
identificativo_canale getDevicesInRoomOfUserIdentifiedByHisDevice_monitored (in string userIP);
identificativo_canale getDevicesInBuildingOfUserIdentifiedByHisDevice_monitored (in string userIP);

identificativo_canale getFiltredDevicesInLocationOfUserIdentifiedByHisDevice_monitored (in string userIP,in string propName,in string propValue);
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
identificativo_canale getFiltredDevicesInRoomOfUserIdentifiedByHisDevice_monitored (in string userIP,in string propName,in string propValue);
identificativo_canale getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice_monitored (in string userIP,in string propName,in string propValue);

/* i seguenti metodi servono per tracciare gli spostamenti del singolo utente identificato dall'IP del dispositivo in uso*/
identificativo_canale get_location_of_user_IdentifiedByHisDevice_monitored(in string userIP);
identificativo_canale get_room_of_user_IdentifiedByHisDevice_monitored(in string userIP);



/*i seguenti metodi effettuano un monitoraggio continuo degli utenti identificati da un IP in una specifica location o in una specifica room*/

identificativo_canale tracking_users_in_location(in string locationID);
identificativo_canale tracking_users_in_room(in string roomID);
};
};
```

fig. 4.14: interfaccia IDL del Context Service.

I metodi sopra illustrati si divino nelle seguenti due categorie:

I. Categoria contenente i metodi a cui un client attinge per ottenere informazioni a cui è interessato solo al momento dell'invocazione del metodo.
II. Categoria contenente i metodi che effettuano un monitoraggio continuo delle informazioni alle quali il client è interessato.

4.5.1.1 Comunicazione sincrona

Il Context Service stabilisce una comunicazione di tipo sincrona quando un client richiama uno dei metodi appartenente alla prima categoria:

- *get_location_of_user_IdentifiedByHisDevice* restituisce il nome o l'identificativo della location in cui si trova l'utente a seconda che il valore della flag passata come argomento

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

sia true oppure false. Come parametro in ingresso riceve l'IP del dispositivo che l'utente sta usando e con cui si è agganciato alla rete. Il metodo invoca il metodo opportuno esposto dal Location Service e descritto sopra. Il Location Service non si espone come Web Service pertanto un servizio pubblico interessato alla location di un utente deve richiederla al Context Service. Ciò è coerente con l'osservazione che la posizione di un utente è una informazione associata al contesto di un utente;

- *get_room_of_user_IdentifiedByHisDevice* restituisce il nome o l'identificativo della stanza, a seconda che il valore della flag passata come argomento sia true o false, all'interno della quale è stato rilevato l'utente. Come parametro in ingresso richiede l'IP del dispositivo che identifica l'utente;
- *get_building_of_user_IdentifiedByHisDevice* restituisce l'edificio all'interno del quale l'utente si trova. Come parametro in ingresso richiede l'IP del dispositivo che identifica l'utente;
- *getDevicesInLocationOfUserIdentifiedByHisDevice* riceve in ingresso l'IP del dispositivo associato ad un utente, ovvero quello utilizzato per agganciarsi alla rete, e restituisce come output una lista di tutti i dispositivi presenti all'interno della location in cui si trova l'utente;
- *getDevicesInRoomOfUserIdentifiedByHisDevice* restituisce l'identificativo di tutti i dispositivi presenti nella stanza dell'utente per il quale viene invocato il metodo;
- *getDevicesInBuildingOfUserIdentifiedByHisDevice* ritorna una lista con gli identificativi di tutti i dispositivi presenti nell'edificio in cui si trova l'utente specificato;
- *getFiltredDevicesInLocationOfUserIdentifiedByHisDevice* restituisce una lista contenente i dispositivi presenti nella location dell'utente filtrati in base al valore che una determinata *property* deve assumere. Tra i parametri d'ingresso *propName* e *propValue* specificano rispettivamente il nome ed il valore della *property* sulla base della quale avviene il filtraggio dei dispositivi.
- *getFiltredDevicesInRoomOfUserIdentifiedByHisDevice* ritorna i dispositivi nella stanza dell'utente filtrandoli sulla base del valore assunto dalla proprietà specificata in modo analogo a quanto descritto per il metodo precedente.
- *getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice* ha un funzionamento analogo a quello degli ultimi due metodi sopra descritti: restituisce i dispositivi all'interno

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

dell'edificio in cui si trova l'utente filtrandoli in base al determinato valore della proprietà da soddisfare.

Un tipico scenario di interazione mediante scambio sincrono di messaggi è quello illustrato nella figura 4.15 che, nel caso particolare fa riferimento al metodo *getDevicesInRoomOfUserIdentifiedByHisDevice*.

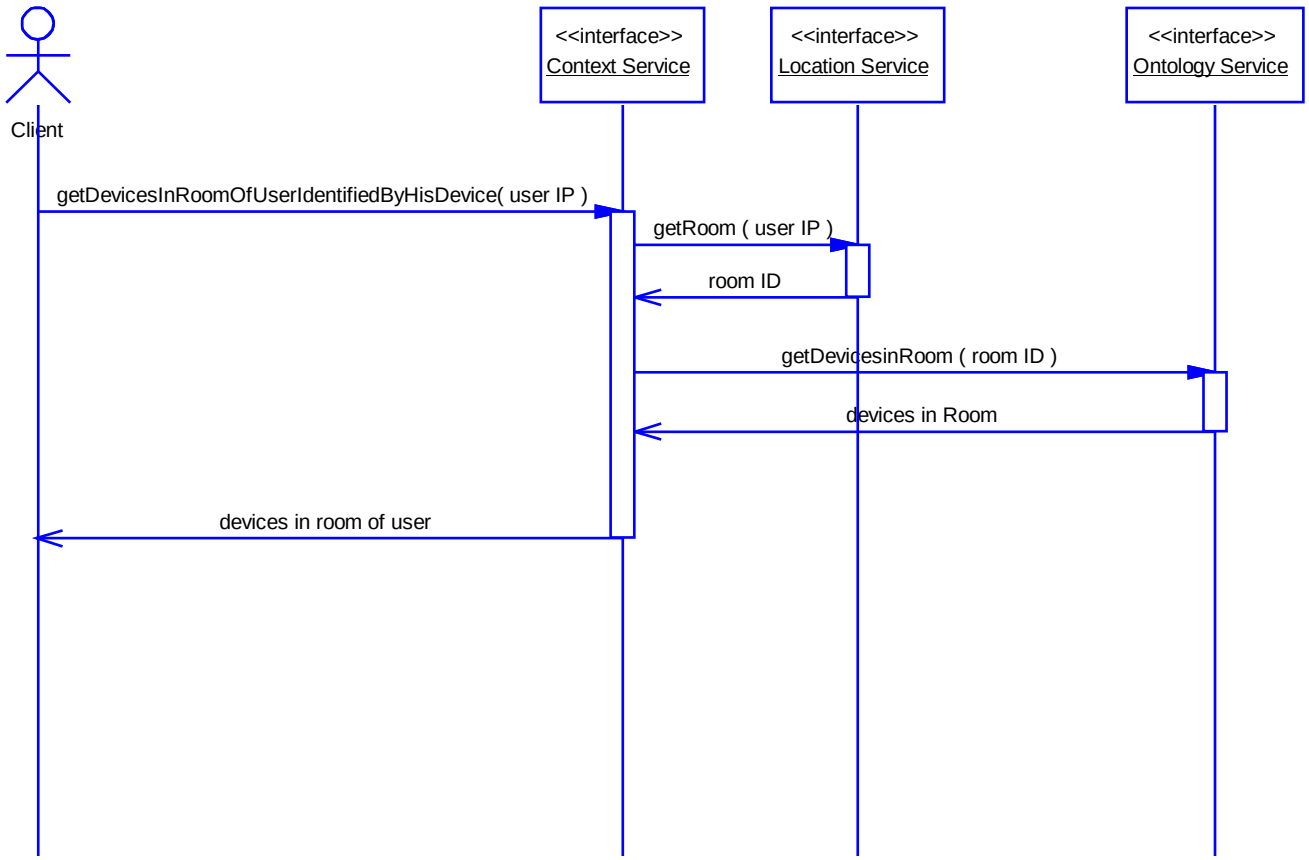

fig. 4.15: modello di interazione per la funzione getDevicesInRoomOfUserIdentifiedByHisDevice.

La figura 4.15 descrive l'interazione che il Context Service stabilisce con gli altri servizi di sistema nel momento in cui un client invoca il metodo *getDevicesInRoomOfUserIdentifiedByHisDevice*.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Il servizio invoca prima l'intervento del Location Service per conoscere la stanza dello specifico utente quindi richiede l'intervento dell'Ontology Service affinché quest'ultimo estragga dall'ontologia di base l'informazione riguardante i dispositivi presenti nella stanza.

Un client del Context Service può essere un servizio pubblico o un servizio di sistema. Nel prototipo implementato di UbiSystem sono solo servizi pubblici a richiamare le funzionalità offerte dal servizio di contesto.
Nello scenario illustrato non ha granché importanza specificare la natura del client in quanto il modello di interazione non varia da quello illustrato. A variare sarà il modo in cui il client vede il Context Service: se il client è un servizio di sistema percepirà il Context Service come servizio CORBA, se il client invece è un servizio pubblico, vedrà il Context Service come Web Service. Non varierà invece il modo con cui il Context Service interagisce con gli altri servizi di sistema implementati per soddisfare la richiesta giuntagli.

Come descritto in precedenza ciascun client del Context Service è interessato alla costruzione di un proprio contesto.
Supponiamo ad esempio l'esistenza di un servizio il cui compito è ricevere le richieste di stampa da parte di un utente e di inoltrare tali richieste alla stampante presente nella stanza di quest'ultimo.
Tale servizio, all'arrivo di una richiesta, sarà interessato alla costruzione di un contesto d'utente che, secondo la proprio visione, può essere completamente descritto dalle informazioni riguardanti gli identificativi delle stampanti presenti nella stanza dell'utente da cui è partita la richiesta.
La costruzione di tale tipo di contesto è immediata: è sufficiente invocare il metodo *getFiltredDevicesInRoomOfUserIdentifiedByHisDevice* esposto dal Context Service passando come parametri di ingressi l'IP del dispositivo associato all' utente in esame, specificare *devicetype* come nome della proprietà e *printer* come valore della proprietà affinché il filtraggio dei dispositivi nella stanza avvenga proprio sulle stampanti.

La figura 4.16 illustra il modello di interazione per il metodo appena menzionato.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

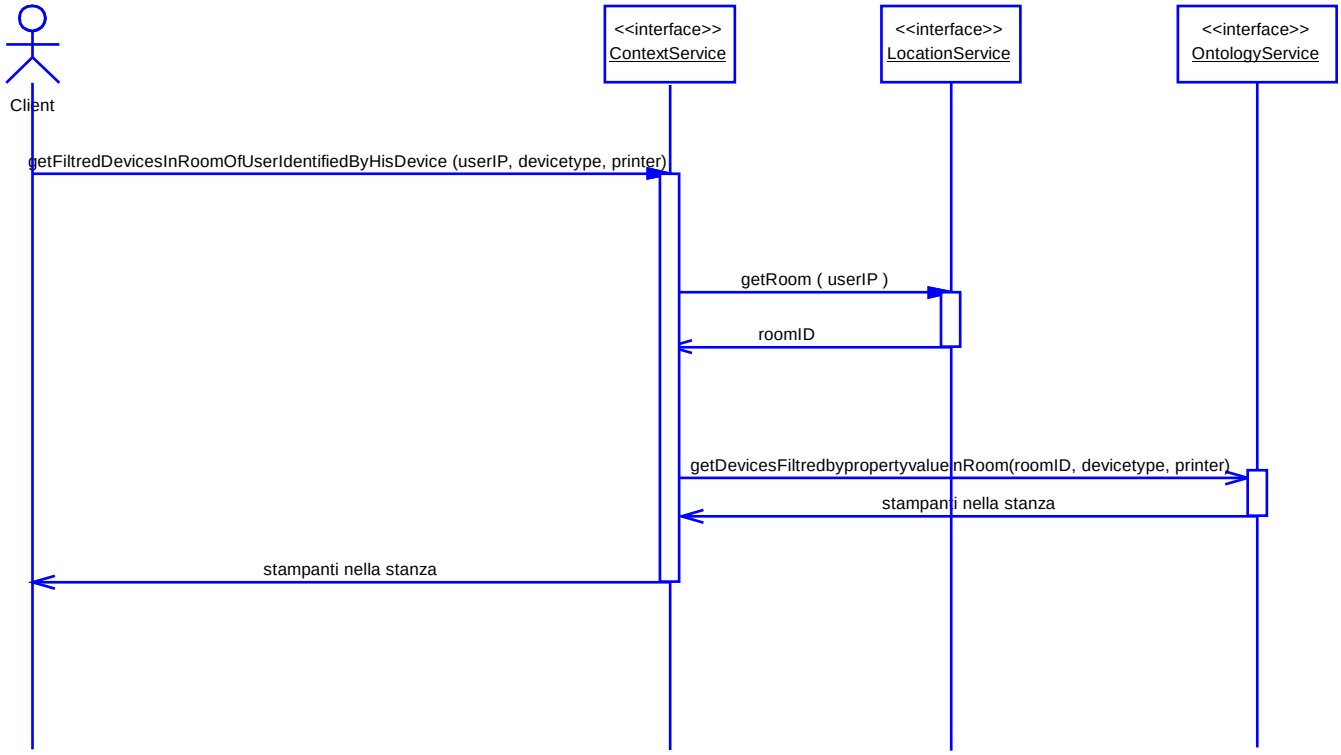

fig. 4.16: modello di interazione per la funzione getFiltredDevicesInRoomOfUserIdentifiedByHisDevice.

Altri servizi possono avere una visione di contesto più elaborata rispetto a quella appena descritta. Esempi di servizi che necessitano di informazioni per la costruzione di contesti più elaborati di quello sopra esposto sono descritti nel capitolo successivo dove sono trattati in dettaglio i servizi pubblici implementati per evidenziare le funzionalità offerte da UbiSystem.

4.5.1.2 Comunicazione asincrona

I metodi che in precedenza abbiamo definito come facenti parte della seconda categoria sono i metodi attraverso i quali il Context Service meglio soddisfa il requisito di *context-awareness*, inteso, in questo caso, soprattutto come sensibilità alle variazioni di contesto, che un sistema pervasivo deve soddisfare. Tali metodi infatti, implementando un monitoraggio continuo su uno specifico set di informazioni, fanno si che il Context Service sia effettivamente sensibile ai cambiamenti di contesto.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- *getDevicesInLocationOfUserIdentifiedByHisDevice_monitored* genera un evento CORBA, contenente tutti i dispositivi presenti nella location di uno specifico utente, ogni qualvolta che quest'ultimo cambia location;
- *getDevicesInRoomOfUserIdentifiedByHisDevice_monitored*, quando un utente viene rilevato in una nuova stanza, produce un evento CORBA contenente tutti i dispositivi interni alla stanza dell'utente monitorato;
- *getDevicesInBuildingOfUserIdentifiedByHisDevice_monitored* ha un comportamento analogo a quanto descritto per i due metodi precedenti: ad ogni spostamento di uno specifico utente da un edificio ad un altro genera un evento CORBA che racchiude tutti i dispositivi a cui l'utente può accedere all'interno dell'edificio in cui è stato rilevato;
- *getFiltredDevicesInLocationOfUserIdentifiedByHisDevice_monitored* estrae i dispositivi presenti nella location di un determinato utente, filtrandoli sulla base di un valore di una specificata proprietà e generando un evento CORBA contenente le informazioni estratte ogni qualvolta l'utente cambi location;
- *getFiltredDevicesInRoomOfUserIdentifiedByHisDevice_monitored* genera un evento CORBA con i dispositivi rilevati nella stanza di un specificato utente e filtrati sulla base di una valore che deve assumere una determinata proprietà associata a tali dispositivi, ogni qualvolta che l'utente cambia location;
- *getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice_monitored* genera un evento CORBA quando un utente, per il quale avviene il monitoraggio, si sposta da un edificio ad un altro. Le informazioni contenute nell'evento generato sono costituite dai dispositivi per i quali è soddisfatto il valore di una determinata proprietà ed installati nell'edificio in cui l'utente è stato appena rilevato;
- *get_location_of_user_IdentifiedByHisDevice_monitored* traccia gli spostamenti, da una location ad un'altra, effettuati dall'utente per il quale avviene il monitoraggio;
- *get_room_of_user_IdentifiedByHisDevice_monitored* effettua un monitoraggio sugli spostamenti di uno specifico utente generando un evento contenente l'identificativo della stanza in cui l'utente è stato appena rilevato;
- *tracking_users_in_location* rileva ed identifica, attraverso l'IP dei dispositivi ad essi associati, gli utenti presenti nella location in esame;
- *tracking_users_in_room* tiene traccia degli utenti rilevati all'interno nella stanza per la avviene il monitoraggio.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Quando un client è interessato al monitoraggio di determinate informazioni di contesto invocherà uno dei metodi sopra. Ciascun metodo restituisce sempre una stringa rappresentante l'identificativo del canale. Quest'ultimo è il canale sul quale il client dovrà mettersi poi in ascolto degli eventi, contenenti le informazioni alle quali il client è interessato, e che di fatto realizza la comunicazione asincrona tra client e Context Service.

All'invocazione di uno dei metodi appartenente alla seconda categoria non vi è solo comunicazione asincrona tra client e servizio di contesto ma viene stabilita una comunicazione asincrona anche tra quest'ultimo ed il Location Service attraverso il *locationEventChannel* di cui sopra: il Context Service estrae un evento dal *locationEventChannel*, lo elabora trasformandolo nell'evento CORBA a cui il client è interessato, ed infine effettua il push dell'evento ottenuto sul canale di nome *canaleID* ed al quale è sicuramente associato un consumatore: il client.

Affinché l'estrazione dal *locationEventChannel*, la trasformazione, ed il push su *canaleID* dell'evento avvenga in modo corretto è necessario associare a ciascun metodo un oggetto java capace di gestire la corretta esecuzione di queste operazioni. Ad ogni metodo sopraesposto è associato pertanto un oggetto denominato *gestore*. A sua volta, la gestione, quindi la creazione, il controllo e la distruzione, di ciascun oggetto *gestore* è affidata al *ContextServiceManager*.

In figura 4.17 è illustrata l'interazione tra gli oggetti sopra descritti all'atto dell'invocazione di un metodo della seconda categoria.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

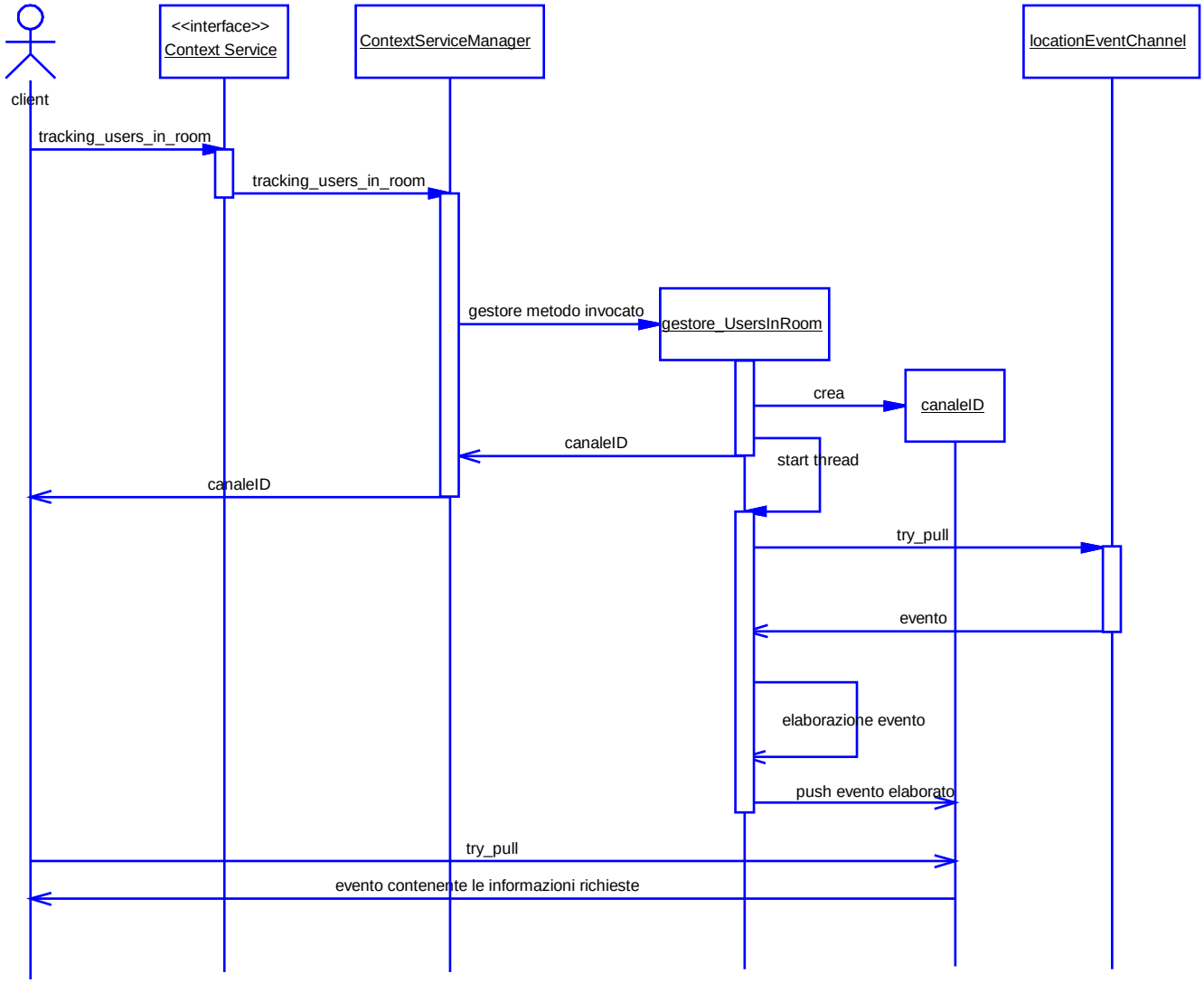

fig. 4.17: modello di interazione per il metodo tracking_users_in_room

Si osservi, in particolare, il ruolo svolto da *ContextService*. Esso da una parte, attraverso il *ContextServiceManager*, richiede la creazione di un aggetto *gestore* affinché possa poi avvenire, in seguito all'invocazione del metodo, la comunicazione asincrona tra client e servizio di contesto, dall'altra, in modo sincrono, comunica al client il canale sul quale dovrà porsi in ascolto. Quest'ultimo aspetto, ovvero la costruzione di un canale dedicato, è di rilievo sia per aspetti di information hiding, nel senso che solo il client sa come interpretare le informazioni che viaggiano sul canale ad esso associato, sia per aspetti legati alla gestione risorse: il client, quando non più interessato al monitoraggio, deve comunicare esplicitamente il rilascio delle risorse occupate mediante il metodo *close_connection* la cui invocazione produce, appunto, la distruzione del relativo oggetto *gestore* secondo lo scenario illustrato in figura 4.18 e, in seguito,

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

la chiusura del canale dedicato da parte del Notification Service. Quest'ultima operazione è prevista per default dal Notification Service di JacORB nel momento in cui il canale non presenta più produttori ne consumatori.

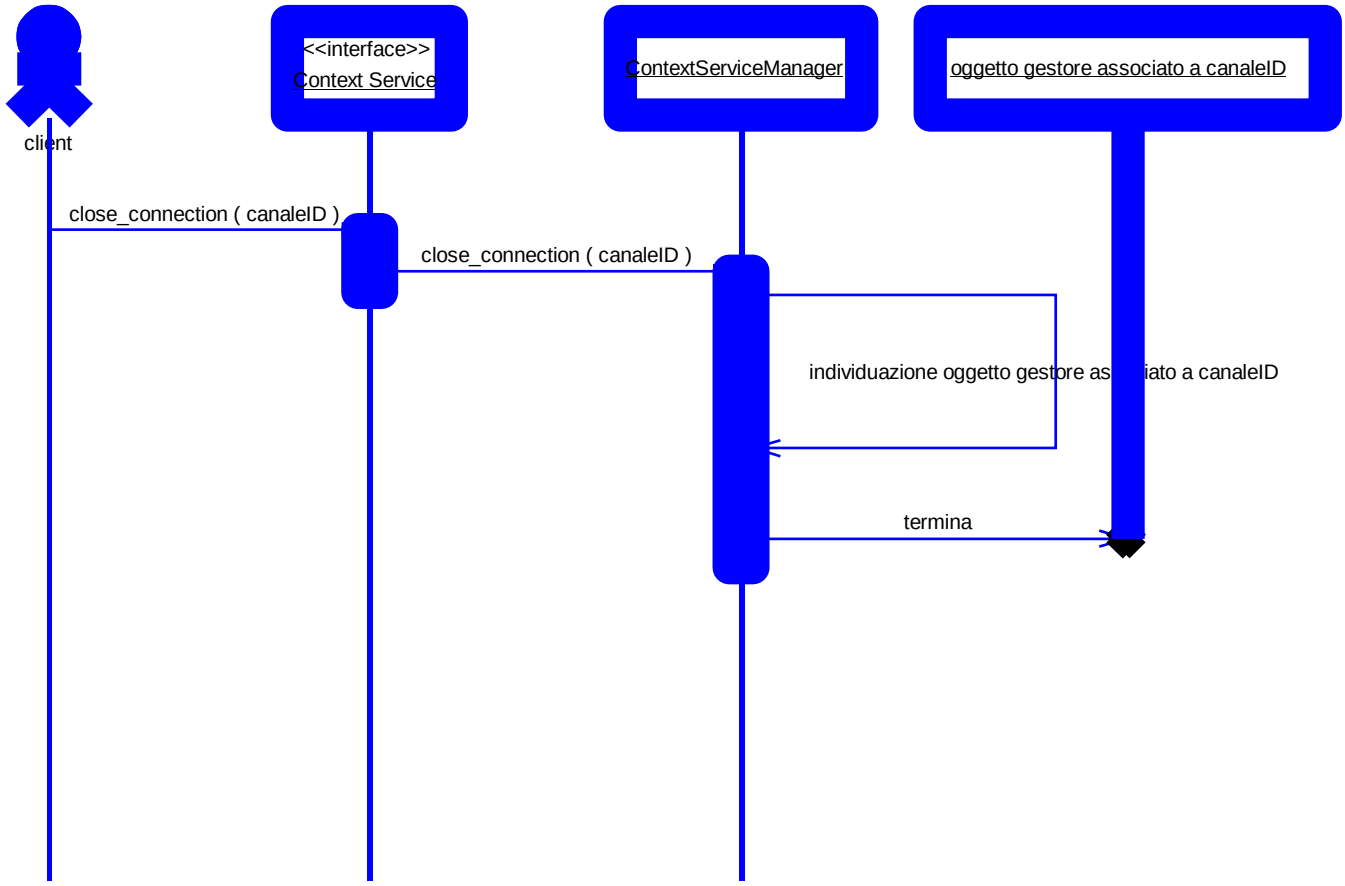

fig. 4.18: modello di interazione per il metodo close_connection.

4.5.2 Il Context Service come Web Service.

A questo punto, il servizio di contesto va ristrutturato in maniera tale da poter essere esposto nell'ambiente come un servizio web. Solo in questo modo infatti le funzionalità che espone potranno essere accessibili da parte dei servizi pubblici.

Per lo scopo sono state utilizzate le librerie di Apache Axis [2] che mettono a disposizione una serie di toolkit che consentono la realizzazione di Web Service a partire da applicazioni Java.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Innanzitutto vanno create un'interfaccia ed una classe java che implementino il servizio: ContextWS.ContextServerInterface ed ContextWS.ContextServerWS.

```
package ContextWS;

public interface ContextServerInterface {

  public String get_location_of_user_IdentifiedByHisDevice(String userIP, boolean Flag);

  public String get_room_of_user_IdentifiedByHisDevice(String userIP, boolean Flag);

  public String get_building_of_user_IdentifiedByHisDevice(String userIP, boolean Flag);

  public String get_location_of_user_IdentifiedByHisDevice_monitored(String userIP);

  public String get_room_of_user_IdentifiedByHisDevice_monitored(String userIP);

  public String tracking_users_in_location(String locationID);

  public String tracking_users_in_room(String roomID);

  public String[] getDevicesInLocationOfUserIdentifiedByHisDevice(String userIP) throws Exception;

  public String[] getDevicesInRoomOfUserIdentifiedByHisDevice(String userIP) throws Exception;

  public String[] getDevicesInBuildingOfUserIdentifiedByHisDevice(String userIP) throws Exception;

  public String[] getFiltredDevicesInLocationOfUserIdentifiedByHisDevice(String userIP,String propName,String
propValue) throws Exception;

  public String[] getFiltredDevicesInRoomOfUserIdentifiedByHisDevice(String userIP,String propName,String
propValue) throws Exception;

  public String[] getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice(String userIP,String propName,String
propValue) throws Exception;

  public void close_connection(String id_canale);
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
  public String getDevicesInLocationOfUserIdentifiedByHisDevice_monitored (String userIP);

  public String getDevicesInRoomOfUserIdentifiedByHisDevice_monitored  (String userIP);

  public String getDevicesInBuildingOfUserIdentifiedByHisDevice_monitored   (String userIP);

  public String getFiltredDevicesInLocationOfUserIdentifiedByHisDevice_monitored (String userIP,String
propName,String propValue);

  public String getFiltredDevicesInRoomOfUserIdentifiedByHisDevice_monitored (String userIP,String
propName,String propValue);

  public String getFiltredDevicesInBuildingOfUserIdentifiedByHisDevice_monitored (String userIP,String
propName,String propValue);

}
```

fig. 4.19: Interfaccia ContextServerInterface per il Web Service.

La classe ContextWS.ContextServerWS sarà client del servizio Corba descritto nel precedente paragrafo.

```
package ContextWS;

import CS.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.io.*;
import java.net.*;

public class ContextServerWS implements ContextServerInterface {

  private ContextService server = null;
  ORB orb;

  public ContextServerWS() {
    init();
  }
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
 public void init() {

  try {
    // Crea ed inizializza l'ORB
    orb = ORB.init((String[])null, null);

    // Root naming context
    NamingContextExt nc = NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));

    // Utilizzo il Naming per ottenere il riferimento all'oggetto
    server = ContextServiceHelper.narrow(nc.resolve(nc.to_name("ContextService")));
    System.out.println("Access to NameServer");

   } catch (Exception e) {
    e.printStackTrace();
    System.exit(1);
   }
 }

/*************************************************************************************************
*/
/*          Metodi pubblici dell'interfaccia ContextServerInterface che non richiedono un monitoraggio continuo
*/
/*************************************************************************************************
*/


public String[] getDevicesInLocationOfUserIdentifiedByHisDevice(String userIP) throws Exception {
          return server.getDevicesInLocationOfUserIdentifiedByHisDevice(userIP);
 }


public String get_room_of_user_IdentifiedByHisDevice(String userIP, boolean Flag) {
          return server.get_room_of_user_IdentifiedByHisDevice(userIP,Flag);
 }

          …
          …
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
/*****************************************************************************************
*/
/*          Metodi pubblici dell'interfaccia ContextServerInterface che richiedono un monitoraggio continuo
*/
/*****************************************************************************************
*/

public String get_room_of_user_IdentifiedByHisDevice_monitored(String userIP){
   ContextWS.wrapper_ntfy_channel_RoomOfUser wrapper_forRoomOfIP=new
ContextWS.wrapper_ntfy_channel_RoomOfUser();
            …
   return wrapper_ forRoomOfIP.get_room_of_user_IdentifiedByHisDevice_monitored(userIP);
  }

public String tracking_users_in_room(String roomID){
   ContextWS.wrapper_ntfy_channel_perIPinRoom wrapper_forUsersInRoom=new
ContextWS.wrapper_ntfy_channel_perIPinRoom();
            …
   return wrapper_forUsersInRoom.tracking_users_in_room(roomID);
  }
            …
            …
}//fine classe
```

fig. 4.20: Classe ContextWS.ContextServerWS.

Quando un client invoca uno dei metodi esposti dal servizio di contesto che effettua un monitoraggio continuo riceverà, come valore di ritorno del metodo invocato, l'identificativo della coda di messaggi sulla quale il client dovrà porsi in ascolto per la ricezione degli eventi. Infatti, la comunicazione asincrona che viene stabilita tra il Context Service ed il servizio pubblico avviene non più tramite canale CORBA ma tramite coda di messaggi JMS.

La classe ContextWS.ContextServerWS, all'invocazione di un metodo con monitoraggio continuo, istanzia uno oggetto java denominato *wrapper_ntfy_channel* il cui compito è estrarre le informazioni dal canale CORBA e trasformarle in messaggi da inviare sulla coda specifica. Di

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

fatto un oggetto *wrapper_ntfy_channel* realizza il wrapping tra un canale CORBA ed una messages queue JMS.
A ciascun metodo con monitoraggio continuo è associato uno specifico oggetto *wrapper_ntfy_channel* in quanto, in base alle specifiche del Notification Service, le informazioni contenute nel canale CORBA possono essere estratte solo da un consumatore che sappia come farlo. Ad esempio, al metodo *get_room_of_user_IdentifiedByHisDevice_monitored* è associata una istanza di *wrapper_ntfy_channel_RoomOfUser* mentre per il metodo *tracking_users_in_room* è associata una istanza di *wrapper_ntfy_channel_perIPinRoom*. Gli oggetti *wrapper_ntfy_channel* differiscono tra loro solo nel modo di estrarre le informazioni dal canale CORBA.
Il client estrae i messaggi JMS nella coda dedicata. Tale operazione non è onerosa in quanto si è convenuti in sede di progetto che ciascun oggetto *wrapper_ntfy_channel* impacchetti l'evento prelevato dal canale CORBA in un messaggio JMS avente sempre la struttura di un array di stringhe.

Di seguito è riportato uno stralcio del codice associato alla classe java che implementa l'oggetto wrapper_ntfy_channel_perIPinRoom: per l'implementazione delle specifiche JMS si è ricorso ad OpenJMS.
OpenJMS è una implementazione open source, molto popolare ed efficiente, della Sun Microsystems's Java Message Service API 1.0.2 Specification. Espone le seguenti features:

- Modelli di messaging Point-to-Point e publish-subscribe.
- Consegna garantita dei messaggi.
- Consegna sincrona ed asincrona dei messaggi.
- Persistenza mediante JDBC.
- Transazioni locali.
- Filtraggio dei messaggi mediante l'utilizzo di SQL92-like selectors.
- Authentication.
- Administration GUI.
- File di configurazione XML-Based.
- In-memory e database garbage collection.
- Supporto Applet.
- Integrazione con Servlet containers quali Jakarta Tomcat.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- Supporto per i protocolli RMI, TCP, http e SSL.
- Supporto per un grande numero di *destinations* e *subscribers*.

```
package ContextWS;

//librerie per la gestione del Notification Service
import org.omg.CosNotification.*;
import org.omg.CosNotifyComm.*;
import org.omg.CosNotifyFilter.*;
import org.omg.CosNotifyChannelAdmin.*;
import org.omg.CORBA.Any;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextExtPackage.*;
import java.io.*;
import org.omg.PortableServer.*;
import org.omg.CORBA.NO_IMPLEMENT;

//librerie jms
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.jms.QueueConnectionFactory;
import javax.jms.QueueConnection;
import javax.jms.Session;
import javax.jms.QueueSession;
import javax.jms.Queue;
import javax.jms.QueueSender;
import javax.jms.TextMessage;
import javax.jms.ObjectMessage;
…
…

public class wrapper_ntfy_channel_perIPinRoom extends StructuredPullConsumerPOA implements Runnable{

  //variabili associate a CORBA
  private EventChannel channel = null;
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
  private FilterFactory filterFactory = null;
  private Filter filter = null;
  private ConsumerAdmin consumerAdmin;
  private StructuredProxyPullSupplier  proxyPullSupplier = null;
  private StructuredPullConsumer structuredPullConsumer;
  private String userid = "MeMyselfAndI";
  private String channelId=null;
  private ContextService contextservice=null;
  org.omg.CORBA.ORB orb;
  //variabili associate illa interfaccia Runnable
  private Thread thread;
  private boolean fineThread=false;
  //variabili associate a JMS
  Queue queue;
  QueueSession session;

…
…

  public String tracking_users_in_room( String identificativostanza) {

    POA poa = null;
    try
    {
      // inizializza POA
      …
      // ottiene riferimento al Name Service
      …

      /*ottiene riferimento al  ContextService CORBA*/
      …
      //chiamata alla procedura CORBA
      channelId=contextservice.tracking_users_in_room (identificativostanza);

      // si collega al canaleCORBA dedicato
      channel =
          EventChannelHelper.narrow(nc.resolve(nc.to_name(channelId)));

      poa.the_POAManager().activate();
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
// creo ed attivo il client
structuredPullConsumer =
    (StructuredPullConsumer)new wrapper_ntfy_channel_perIPinRoom()._this(orb);

// ottengo il consumer admin ed il proxy supplier
consumerAdmin  = channel.default_consumer_admin();

proxyPullSupplier =
    StructuredProxyPullSupplierHelper.narrow(
    consumerAdmin.obtain_notification_pull_supplier(
    ClientType.STRUCTURED_EVENT,
    new org.omg.CORBA.IntHolder() ) );

// collego il consumatore al suo produttore
proxyPullSupplier.connect_structured_pull_consumer( structuredPullConsumer );

// ottengo la default filter factory
filterFactory = channel.default_filter_factory();
if( filterFactory == null )
{
  System.err.println("No default filter Factory!");
}
else
{
  filter = filterFactory.create_filter("EXTENDED_TCL");
  EventType [] eventTypes =
      new EventType[] { new EventType("IP", "in room"),
      new EventType("IP", "Canceled") };

  ConstraintExp constraint =
      new ConstraintExp ( eventTypes, "TRUE" );

  filter.add_constraints( new ConstraintExp[]{ constraint } );
  proxyPullSupplier.add_filter(filter);
}

Property[] qos = new Property[1];
org.omg.CORBA.Any data = org.omg.CORBA.ORB.init().create_any();
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
data.insert_short( PriorityOrder.value );
qos[0] = new Property( OrderPolicy.value, data);

try
{
  consumerAdmin.set_qos(qos);
}
catch (UnsupportedQoS ex)
{
  System.err.println("Unsupported QoS");
}
catch (NO_IMPLEMENT e) {
  …
}
/****************************************/
/* INIZIALIZZAZIONE PRODUTTORE JMS  */
/****************************************/
//creazione di un JNDI context che operi con protocollo HTTP
Hashtable properties = new Hashtable();
properties.put(Context.INITIAL_CONTEXT_FACTORY,
            "org.exolab.jms.jndi.InitialContextFactory");
properties.put(Context.PROVIDER_URL, "http://localhost/");
Context context = new InitialContext(properties);

//recupero di una QueueConnectionFactory
QueueConnectionFactory factory =
   (QueueConnectionFactory) context.lookup("JmsQueueConnectionFactory");

//creazione di una connessione alla coda:
QueueConnection connection = factory.createQueueConnection();
connection.start();

//creazione di una sessione:
session =
   connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);

//recupero della coda vera e propria:
//Queue queue = (Queue) context.lookup("queue1");
queue = (Queue) session.createQueue(channelId);
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
  }
  catch (Exception e)
  {
   e.printStackTrace();
   System.exit(1);
  }

  …
}

//inizio thread che si pone in ascolto continuo di eventi
 public void run(){
  …
  while (true)
  {
   org.omg.CORBA.BooleanHolder bh =
      new org.omg.CORBA.BooleanHolder();
   try
   {
    StructuredEvent event =
       proxyPullSupplier.try_pull_structured_event(bh);

    //ricezione MSG dal canale CORBA
    if( bh.value )
    {
     System.out.println("Ricevuto evento dal canale"+channelId);

     Property properties [] = event.filterable_data;
     for (int y=0;y<properties.length;y++){
      System.out.println("\t" + properties[y].name +" : " + properties[y].value.extract_string() );
     }

     //INVIO MESSAGGIO JMS:
     QueueSender sender = session.createSender(queue);

     String[] stringhe=new String[properties.length];
     for (int i=0;i<properties.length;i++){
      stringhe[i]=properties[i].value.extract_string();
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
        ObjectMessage message = session.createObjectMessage(stringhe);

        sender.send(message);
        System.out.println("invio MSG");
        …
       }
     …
     …
     }
     catch( Exception e )
     {
      e.printStackTrace();
     }
    }

  }


}
```

fig. 4.21: Implementazione in una classe java dell'oggetto wrapper_ntfy_channel_perIPinRoom.

Una volta definite le classi ContextWS.ContextServerInterface ed ContextWS.ContextServerWS viene generata l'interfaccia WSDL del servizio Web, mediante il tool Java2WSDL di axis:

```
java org.apache.axis.wsdl.Java2WSDL -o context.wsdl  -l"http://localhost/axis/services/contextserver"
-n urn:contextserver -p"ContextWS" urn:contextserver ContextWS.ContextServerInterface
```

I parametri necessari sono il nome del file di ouput (context.wsdl), l'URL del Web Service (http://localhost/axis/services/contextserver), il target namespace per la WSDL (urn:contextserver), il mapping fra il package java ed il namespace (ContextWS = urn: contextserver) ed infine l'interfaccia java del servizio (ContextWS.ContextServerInterface).
Una volta eseguito il programma verrà generato il file context.wsdl, che descrive il formato dei messaggi da inviare al Web Service, quali sono i metodi esposti, quali sono i parametri ed i valori di ritorno.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Il prossimo passo è quello di generare a partire dalla WSDL del servizio il codice necessario per effettuarne il deploy. Questa volta verrà adoperato il programma WSDL2Java:

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p ContextWS.ws context.wsdl
```

I parametri indicano: la directory base di output (.); lo scope del servizio (Session, Application, Request); la generazione anche del codice server-side (-s); il package dove includere il codice generato (ContextWS.ws) ed infine il file dove reperire la wsdl (context.wsdl, ma potrebbe esse anche una URL).
Il programma genera i seguenti file nella directory Mp3WS.ws:

- *ContextserverSoapBindingImpl.java*: la classe il cui codice implementa il Web Service (andrà leggermente modificata);
- *ContextServerInterface.java*: l'interfaccia remota per il ContextService;
- *ContextServerInterfaceService.java*: interfaccia di servizio per il Web Service;
- *ContextServerInterfaceServiceLocator.java*: implementa ContextServerInterfaceService; è una Helper Factory che fornisce l'accesso al servizio;
- *ContextserverSoapBindingStub.java:* client-side stub;
- *Deploy.wsdd*: file contenente le informazioni necessarie per effettuare il deploy del Web Service sul sistema Axis;
- *Undeploy.wsdd:* file necessario per effettuare l'undeploy del servizio da Axis.

Prima di poter effettuare il deploy del servizio, bisogna apportare qualche modifica al file ContextserverSoapBindingImpl.java.
In Figura 4.22, è riportato un stralcio del codice della classe; in neretto sono evidenziate le linee di codice aggiunte.

```
/**
 * ContextserverSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package ContextWS.ws;
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
public class ContextserverSoapBindingImpl implements ContextWS.ws.ContextServerInterface{

 private ContextWS.ContextServerInterface cont = new ContextWS.ContextServerWS();

 public java.lang.String get_location_of_user(java.lang.String in0, boolean in1) throws java.rmi.RemoteException {
  try{
   return cont.get_location_of_user(in0,in1);
  }catch(Exception e){return null;}
 }

 public java.lang.String get_location_of_user_IdentifiedByHisDevice(java.lang.String in0, boolean in1) throws
java.rmi.RemoteException {
  try{
   return cont.get_location_of_user_IdentifiedByHisDevice(in0,in1);
  }catch(Exception e){return null;}

 }

        …
        …

}//fine classe
```

fig. 4.22: Modifiche apportate alla classe ContextserverSoapBindingImpl.

A questo punto non resta che includere in un package le classi finora sviluppate, copiare il package all'interno della cartella axis/WEB-INF/lib situata all'interno di Tomcat, ed infine procedere con il deploy tramite il comando:

```
java org.apache.axis.client.AdminClient -p 80 ContextWS/ws/deploy.wsdd
```

Quando viene eseguito quest'ultimo comando, Tomcat dovrà essere in esecuzione ed in ascolto sulla porta 80 (-p 80). Il servizio web è finalmente attivo e disponibile.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

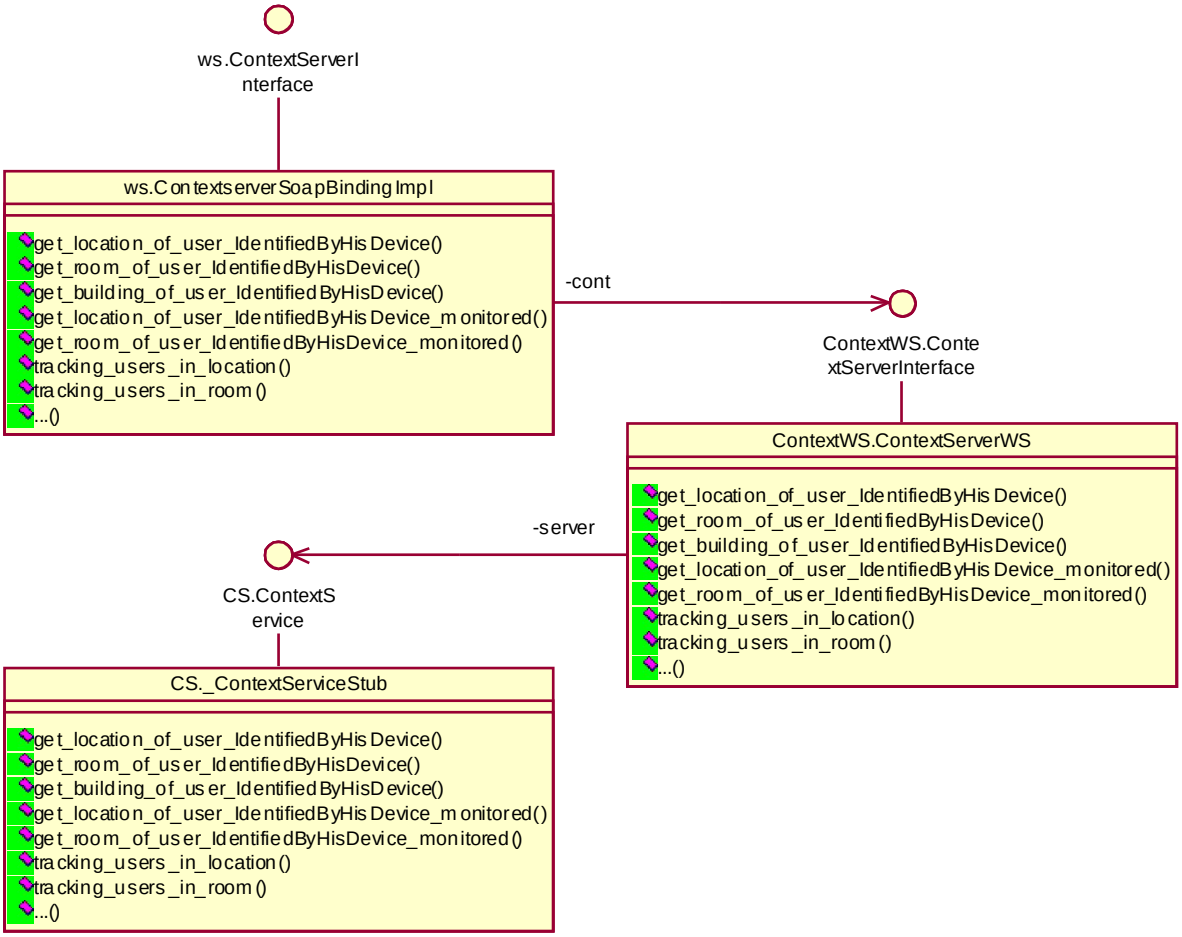

fig. 4.23: creazione del servizio web a partire dal servizio CORBA

In figura 4.23 viene riportato il diagramma con le classi principali del Servizio Web. In particolare, la classe ContextserverSoapBindingImpl implementa l'interfaccia ContextServerInterface (package ContextWS.ws) attraverso la quale i client remoti accedono al servizio.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Capitolo 5

Casi di Studio: due servizi dipendenti dal contesto

5.1 Introduzione

Il sistema UbiSystem implementato è context-aware. Per la sperimentazione sono stati realizzati due servizi che basano il loro funzionamento sull'utilizzo delle informazioni di contesto che il sistema UbiSystem fornisce: un servizio di stampa locale ed un servizio di gestione della musica d'ambiente.

Nel prosieguo del capitolo verranno descritti i suddetti servizi in dettaglio. Prima però è utile evidenziare alcuni aspetti comuni.

Un utente, munito di palmare, potrà accedere all'home page dell'ambiente pervasivo in cui gli saranno forniti gli strumenti necessari per il discovery dei servizi disponibili. Affinché sia possibile il discovery, un servizio dovrà prima registrarsi presso il Service Manager. In fase di registrazione presso il Service Manager il servizio deve fornire la sua descrizione che andrà convalidata attraverso l'Ontology Service per verificarne la consistenza.
Queste attività vengono svolte automaticamente dalla classe ServiceProvider, contenuta nel package omonimo, messo a disposizione degli sviluppatori di servizi pervasivi.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Un servizio deve estendere la classe ServiceProvider per usufruire delle sue funzionalità: quando si crea un oggetto rappresentante il servizio viene richiamato il costruttore della classe padre, il quale avvia un nuovo thread che:

- ricerca l'Ontology Service nell'ambiente;
- recupera da file la descrizione del servizio;
- registra il servizio inviando la descrizione all'Ontology Service.

Se l'Ontology Service al momento della registrazione non è attivo, il thread, periodicamente, opererà nuovi tentativi finché la registrazione non sarà effettuata.

Lo sviluppatore di un servizio dovrà fornire solamente alcuni parametri al ServiceProvider attraverso il file ServiceConfig.xml.

```
<?xml version="1.0" ?>
- <config>
        <OntologyServerUrl>
                http://www.ubinet.it/axis/services/ontologyserver
        </OntologyServerUrl>
        <description>
                serviceDescription.daml
        </description>
        <delay>
                120000
        </delay>
</config>
```

fig. 5.1: Esempio di file di configurazione per il ServiceProvider.

In figura 5.1 è riportato il file di configurazione per un generico servizio. Nel file sono indicati: l'URL presso il quale contattare l'Ontology Service (*http://www.ubinet.it/axis/services/ontologyserver* ; il nome del file dove è contenuta la descrizione del servizio (*serviceDescription.daml*); l'intervallo di tempo, espresso in millisecondi, ogni quanto riavviare un nuovo tentativo di registrazione del servizio (*120 secondi*).

Il package ServiceProvider fa uso del package ontoWS che contiene le classi e le interfacce necessarie lato client per interagire con l'Ontology Service. A sua volta ontoWS necessita delle librerie di axis.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

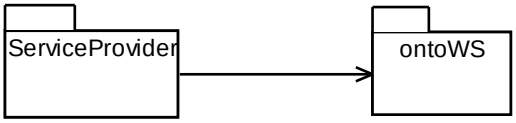

Figura 5.2: Dipendenza fra il package ServiceProvider ed il package ontoWS.

5.1.1 La descrizione del servizio

Un servizio Web è descritto mediante la propria WSDL che ne esprime i requisiti funzionali. Accanto alla WSDL viene fornita anche una descrizione in DAML-S.

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. 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.

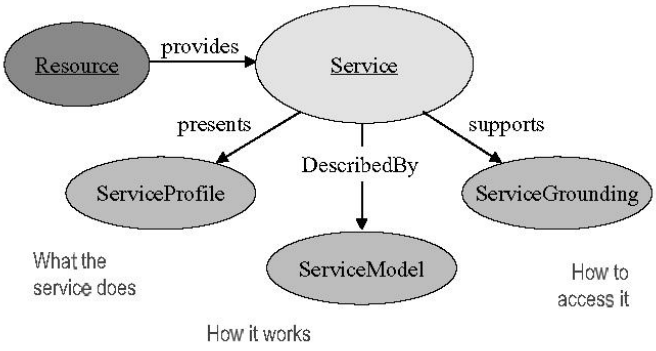

fig. 5.3: Visione di alto livello dell'ontologia fornita DAML-S.

Attraverso il Service Profile vengono fornite le informazioni necessarie 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. 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; il secondo specifica il protocollo di

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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.

5.2 Il servizio di stampa locale

Il servizio di stampa locale, in esecuzione nell'ambiente, consente agli utenti, che si sono agganciati alla rete, di scegliere i documenti da stampare, settare alcune preferenze di stampa, in particolare stampa a colori o monocromatica, e ritirare la stampa presso la stampante che il servizio ritiene di più agevole accesso per l'utente: il servizio avvia in automatico la stampa presso la stampante nella stanza dell'utente oppure chiede all'utente di scegliere esplicitamente la stanza ove gli sia più comodo prelevare i documenti. Quest'ultimo scenario si verifica quando nessuna stampante è presente nella stanza ove si trova l'utente o quando le stampanti presenti nella stanza dell'utente non sono in grado di soddisfare le preferenze di stampa dell'utente.

Il servizio necessita di informazioni di contesto per stabilire quali sono le stampanti a cui l'utente possa accedere più agevolmente.
Il servizio non necessita di un monitoraggio continuo delle informazioni di contesto. La richiesta delle informazioni di contesto avviene solo quando necessario, ovvero quando l'utente fa esplicita richiesta di stampa. Il servizio comunica sempre in modo sincrono con l'UbiSystem attraverso le apposite funzionalità offerte dal Context Service.

Il servizio è stato implementato con tecnologia CORBA e poi, generato l'apposito wrapper, giacché il servizio espone funzionalità che devono essere rese accessibili dall'esterno, è stato esposto all'interno dell'ambiente sottoforma di servizio WEB.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

In figura 5.4 è mostrata l'interfaccia IDL del servizio. Il metodo *print* è il metodo attraverso il quale il servizio avvia la stampa per l'utente, identificato dall'IP del dispositivo con cui si è agganciato alla rete, che ne fa richiesta. Attraverso l'invocazione del metodo *printwith* il servizio invia invece la stampa su una specifica stampante.
I metodi *setFile* e *setColor* sono metodi attraverso i quali l'utente specifica il documento da stampare ed il tipo di stampa inteso come a colori o monocromatica. Il metodo *getStatus* infine consente di ricavare informazioni di controllo sul servizio.

```
/* IDL localPrintService */

module Print {

        interface StampaService {

        void print (in string userIP);

        void printwith (in string printerName);

        long getStatus();

        void setFile (in string filename);

        void setColor (in boolean colorFlag);

        };
};
```

fig. 5.4: interfaccia IDL del servizio di stampa locale.

5.2.1 Implementazione CORBA

In figura 5.5 è rappresentato il diagramma delle classi del servizio di stampa locale. La classe principale (con il metodo *main()*) è localPrintService che estende la classe ServiceProvider sopra descritta.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

La classe localPrintService costituisce il processo server e, attraverso il metodo *initOrb()*, svolge le seguenti attività:

- Inizializza l'ORB, attraverso l'operazione statica init() della classe ORB, ottenendo in questo modo un riferimento necessario per il successivo utilizzo.
- Crea un riferimento al POA. Tale riferimento viene restituito dall'ORB stesso attraverso il metodo *resolve_initial_reference()*.
- Attiva l'oggetto nel POA. Viene dapprima istanziato un oggetto Stampalocale e successivamente attraverso il POA viene ricavato l'Object Reference, attraverso il quale i client possono effettuare invocazioni.
- Pubblica l'Object Reference. L'IOR viene reso pubblico registrandolo nel Naming Service.
- Attiva l'ORB. L'ORB viene esplicitamente attivato e posto in uno stato di ascolto delle richieste.

L'interfaccia IDL del servizio di figura 5.4 viene implementata dalla classe Stampalocale.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

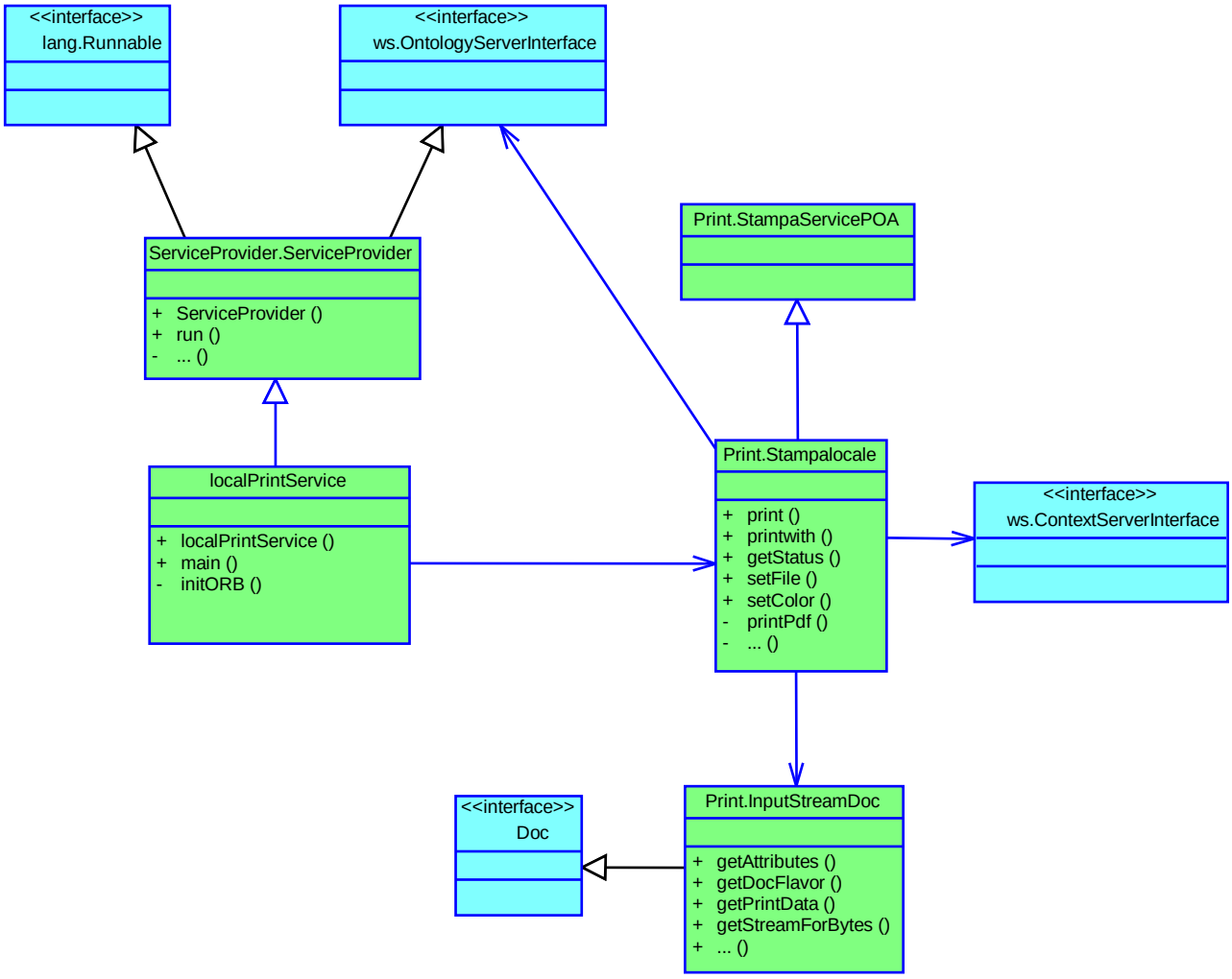

fig. 5.5: Diagramma delle classi del servizio di stampa locale.

La classe Stampalocale, implementando i metodi della interfaccia IDL, comunica con il Context Service e con l'Ontology Service per poter estrarre informazioni circa le stampanti presenti nella stanza dell'utente ed in grado di soddisfare le preferenze di stampa richieste.

5.2.2 Conversione in Web Service

A questo punto, il servizio CORBA va ristrutturato in maniera tale da poter essere esposto nell'ambiente come un servizio web affinché le funzionalità che offre siano accessibili dall'esterno.

Il procedimento utilizzato per l'esposizione del servizio di stampa locale è analogo a quanto descritto nel capitolo precedente relativamente all'esposizione del Context Service come Web

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Service: innanzitutto vanno create un'interfaccia ed una classe java che implementino il servizio: localPrintWS.PrintServerInterface ed localPrintWS.PrintServer.

```
package localPrintWS;

public interface local_PrintServerInterface {

  //avvio stampa
  void print (String userIP);
  void printwith (String printerName);

  //stato stampante
  long getStatus();

  //alcuni metodi per settare le preferenze di stampa
  void setFile (String filename);
  void setColor (boolean colorFlag);

} // interface
```

fig. 5.6: Interfaccia PrintServerInterface per il Web Service.

La classe localPrintWS.PrintServer sarà client del servizio Corba realizzato nel precedente paragrafo.

```
package localPrintWS;

import localPrintWS.local_PrintServerInterface;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import Print.*;

public class local_PrintServer implements local_PrintServerInterface {

  private StampaService server = null;
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
  public local_PrintServer() {
    init();
  }

  private void init() {
    try {
      // Crea ed inizializza l'ORB
      ORB orb = ORB.init((String[])null, null);

      // Root naming context
      NamingContextExt nc =
NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));

      // Utilizzo il Naming per ottenere il riferimento all'oggetto
      server = StampaServiceHelper.narrow(nc.resolve(nc.to_name("localPrintService")));
      System.out.println("Access to NameServer");

    } catch (Exception e) {
      e.printStackTrace();
      System.exit(1);
    }
  }

  /* Metodi dell'interfaccia localPrintServer */

  //avvio stampa
  public void print (String userIP){
    server.print(userIP);
  }

  public void printwith (String printerName){
    server.printwith(printerName);
  }
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
  //stato stampante
  public long getStatus(){
    return server.getStatus();
  }

  //alcuni metodi per settare le preferenze di stampa
  public void setFile (String filename){
    server.setFile(filename);
  }
  public void setColor (boolean colorFlag){
    server.setColor(colorFlag);
  }
}
```

fig. 5.7: Classe localPrintWS.PrintServer

Una volta definite queste due classi viene generata l'interfaccia WSDL del servizio Web, mediante il tool Java2WSDL di axis:

```
java org.apache.axis.wsdl.Java2WSDL -o localPrintWS.wsdl
-l"http://localhost/axis/services/localprintservice"
-n urn:localprintservice -p"localPrintWS" urn:localprintservice
localPrintWS.local_PrintServerInterface
```

Analogamente a quanto descritto per il Context Service i parametri necessari sono il nome del file di ouput (localPrintWS.wsdl), l'URL del Web Service (http://localhost/axis/services/localprintservice), il target namespace per la WSDL (urn: localprintservice), il mapping fra il package java ed il namespace (localPrintWS = urn:localprintservice) ed infine l'interfaccia java del servizio (localPrintWS.local_PrintServerInterface). Una volta eseguito il programma verrà generato il file localPrintWS.wsdl.

Il prossimo passo è quello di generare a partire dalla WSDL del servizio il codice necessario per effettuarne il deploy. Questa volta verrà adoperato il programma WSDL2Java:

```
java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p localPrintWS.ws localPrintWS.wsdl
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

I parametri indicano: la directory base di output (.); lo scope del servizio (Session, Application, Request); la generazione anche del codice server-side (-s); il package dove includere il codice generato (localPrintWS.ws) ed infine il file dove reperire la wsdl (localPrintWS.wsdl, ma potrebbe esse anche una URL).

Il programma genera i seguenti file nella directory localPrintWS.ws:

- *LocalprintserviceSoapBindingImpl.java*: la classe il cui codice implementa il Web Service (andrà leggermente modificata);
- *Local_PrintServerInterface.java*: l'interfaccia remota;
- *Local_PrintServerInterfaceService.java*: interfaccia di servizio per il Web Service;
- *Local_PrintServerInterfaceServiceLocator.java*: implementa Local_PrintServerInterfaceService; è una Helper Factory che fornisce l'accesso al servizio;
- *LocalprintserviceSoapBindingStub.java:* client-side stub;
- *Deploy.wsdd*: file contenente le informazioni necessarie per effettuare il deploy del Web Service sul sistema Axis;
- *Undeploy.wsdd:* file necessario per effettuare l'undeploy del servizio da Axis.

Le modifiche necessarie da apportate al file LocalprintserviceSoapBindingImpl.java sono quelle evidenziate in neretto nel riquadro sotto.

```
/**
 * LocalprintserviceSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis WSDL2Java emitter.
 */

package localPrintWS.ws;

public class LocalprintserviceSoapBindingImpl implements
localPrintWS.ws.Local_PrintServerInterface{

  private localPrintWS.local_PrintServerInterface servente_stampa=new
localPrintWS.local_PrintServer();
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
   public void print(java.lang.String in0) throws java.rmi.RemoteException {
     servente_stampa.print(in0);
   }

   public void setColor(boolean in0) throws java.rmi.RemoteException {
     servente_stampa.setColor(in0);
   }
   …
}
```

fig. 5.8: Modifiche apportate alla classe LocalprintserviceSoapBindingImpl.

A questo punto non resta che includere in un package le classi finora sviluppate, copiare il package all'interno della cartella axis/WEB-INF/lib situata all'interno di Tomcat, ed infine procedere con il deploy tramite il comando:

```
java org.apache.axis.client.AdminClient -p 80 localPrintWS/ws/deploy.wsdd
```

Quando viene eseguito quest'ultimo comando, Tomcat dovrà essere in esecuzione ed in ascolto sulla porta 80 (-p 80). Il servizio web è finalmente attivo e disponibile.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

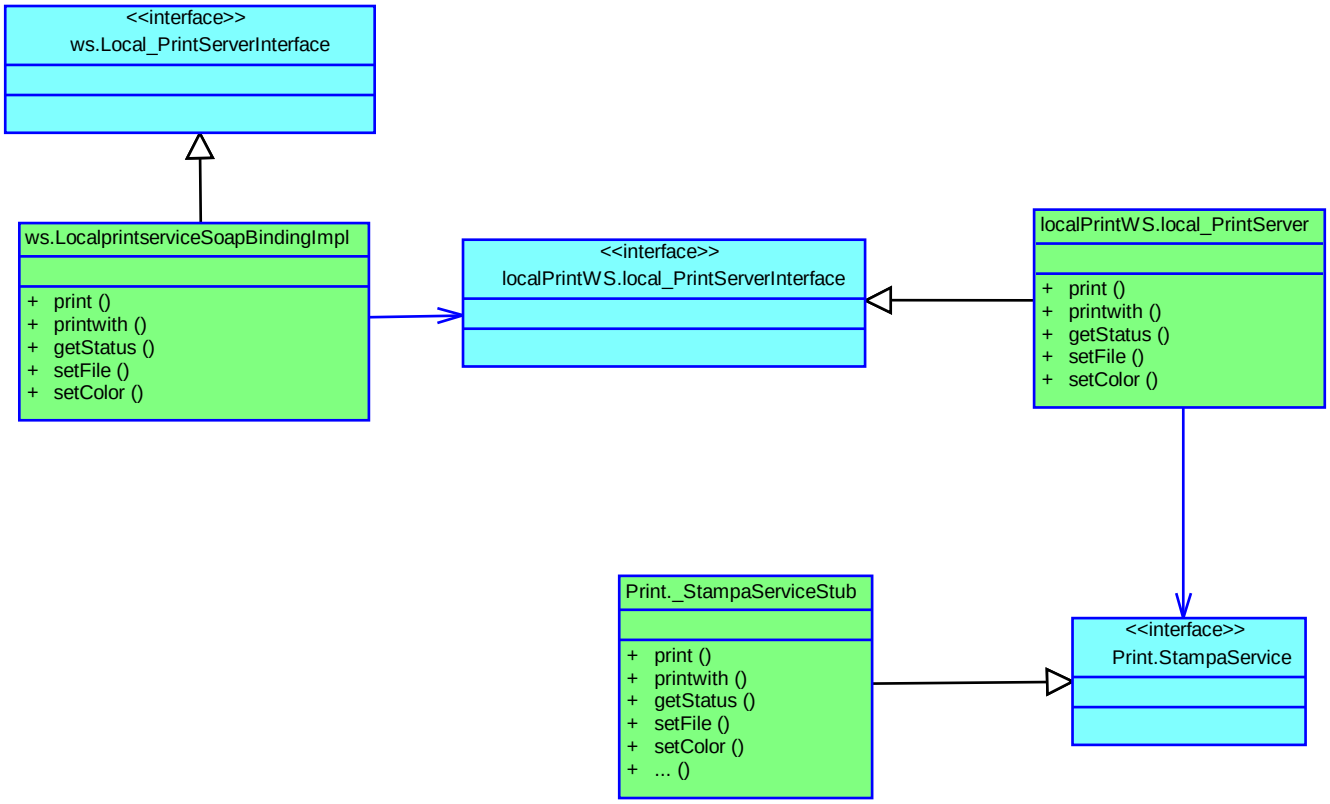

fig. 5.9: Creazione del servizio web a partire dal servizio Corba.

In figura 5.9 viene riportato il diagramma con le classi principali del Servizio Web. In particolare, la classe LocalprintserviceSoapBindingImpl implementa l'interfaccia Local_ PrintServerInterface (package localPrintWS.ws) attraverso la quale i client remoti accedono al servizio.

5.2.3 Registrazione e configurazione del servizio

La registrazione del servizio presso il Service Manager, come descritto nella introduzione al capitolo, è svolta in automatico dalla classe Service Provider.

Il file di configurazione ServiceConfig.xml necessario è il seguente:

```
<?xml version="1.0"?>
<config>
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
        <OntologyServerUrl>http://localhost/axis/services/ontologyserver</OntologyServerUrl>
        <description>localPrintService.daml</description>
        <delay>80000</delay>
</config>
```

fig. 5.10: file di configurazione del servizio di stampa locale.

In figura 5.11 è riportato uno schema che riassume la descrizione del servizio di stampa presente nel file localPrintService.daml. Vengono indicati il nome della risorsa e del servizio offerto ed i tipi di file supportati per la stampa.

Descrizione della risorsa Web LocalPrintService				
ResourceName				LocalPrintService
ResourceTextDescription				LocalPrintService è una risorsa Web per la stampa dei file.
ResourceUrl				http://ubinet.it/localPrintService
Prints				PdfFile, JPEGFile, PNGFile, GIFFile
Service	Profile	ServiceName		localPrintService
		TextDescription		Questo servizio consente di stampare file Pdf.
		Input	ParameterName	Filename
			RestrictedTO	String
		Prints		PdfFile, PEGFile, PNGFile, GIFFile

Fig. 5.11: Schema della descrizione del servizio di stampa locale.

Il servizio necessita, inoltre, di un ulteriore file di configurazione, per la stampa di documenti in formato pdf, contenente il percorso completo dove reperire l'Acrobat Reader.

```
<?xml version="1.0"?>
<config>
        <acropath>C:\Programmi\Adobe\Acrobat 5.0\Reader\AcroRd32.exe</acropath>
</config>
```

fig. 5.12: file di configurazione proprio del servizio di stampa.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

5.3 Il servizio di musica di ambiente

Il servizio di musica di ambiente è un servizio sviluppato con lo scopo di mettere in evidenza come UbiSystem sia effettivamente sensibile ai cambiamenti di contesto.
Il servizio provvede alla gestione della musica di sottofondo di una determinata stanza a tema.
Si immagini a riguardo, in un museo, una stanza dedicata alla cultura egiziana, all'interno della quale, se presenti visitatori, viene eseguita una specifica musica a tema.

Nella concreta implementazione del servizio: se nella stanza è presente almeno un visitatore il servizio provvede alla esecuzione della musica di sottofondo mentre provvede alla sua interruzione quando la stanza si svuota. La musica di sottofondo verrà poi ripresa quando verrà notificato al servizio la presenza di un nuovo visitatore. Il servizio inoltre ha una funzionalità aggiuntiva: ad ogni nuovo individuo che giunge nella stanza da il benvenuto tramite un apposito messaggio vocale.
Affinché il servizio possa offrire tali funzionalità è necessario che invochi l'apposito metodo del Context Service affinché quest'ultimo lo tenga continuamente aggiornato sugli individui presenti nella stanza.

Il servizio di musica di ambiente, pertanto, a differenza di quanto avviene con il servizio di stampa in locale, utilizza e sperimenta le funzionalità che richiedono un monitoraggio continuo sulle informazioni di contesto che UbiSystem fornisce mediante il Context Service.

Il servizio di musica di ambiente si registra normalmente all'interno dell'ambiente pervasivo realizzato tramite la classe Service Provider. Utilizza le funzionalità offerte dal Context Service come web services ma non è esso stesso un Web Service in quanto non espone metodi che devono essere resi accessibili dall'esterno. Il servizio di musica di ambiente è un servizio java che comunica in remoto tramite tecnologia JMS su protocollo http.

La figura seguente illustra il diagramma delle classi, di alto livello, del servizio in esame.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

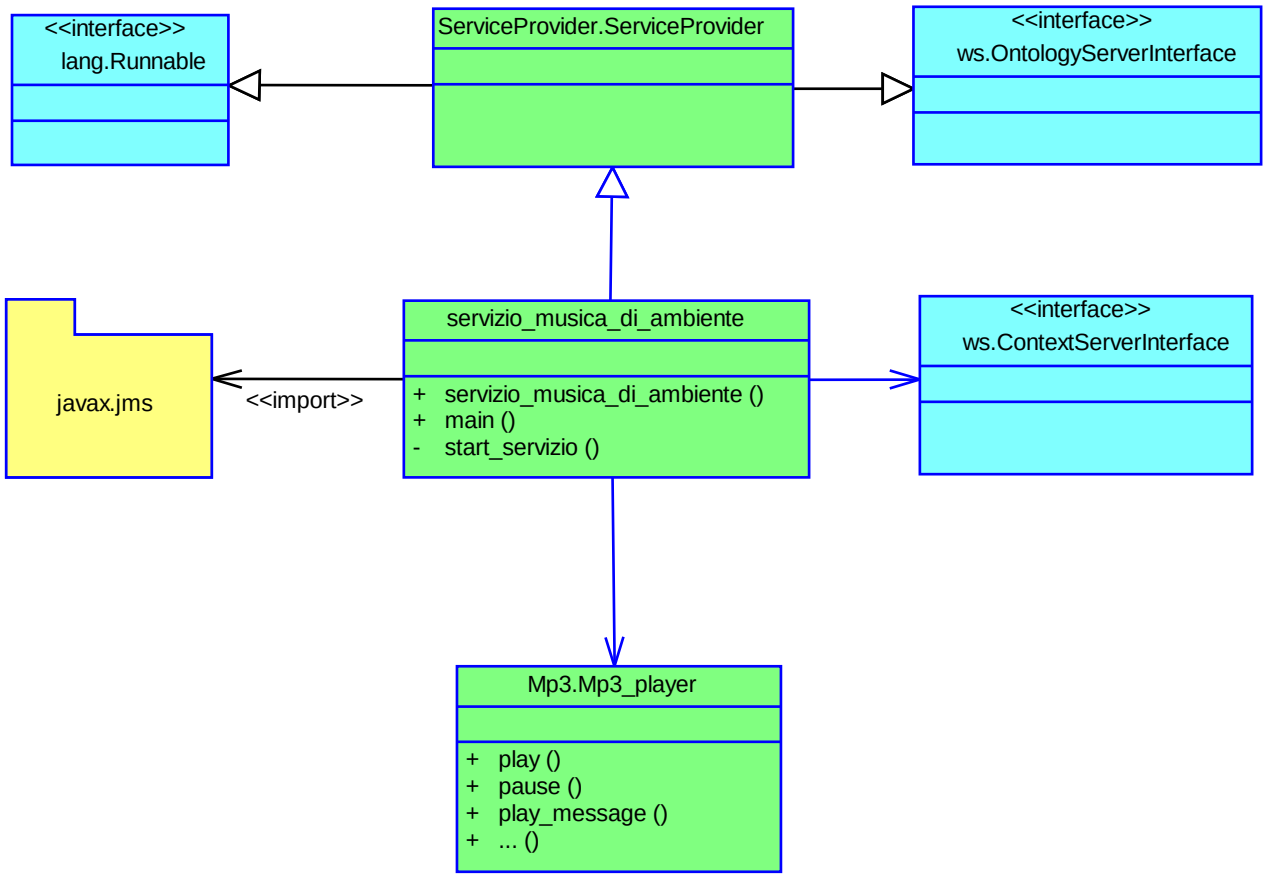

fig. 5.13: Diagramma delle classi del servizio di musica di ambiente

la classe principale, contenente il metodo *main()*, del servizio è la classe servizio_musica_di_ambiente.java.

```
import Mp3.*;

import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.jms.QueueConnectionFactory;
import javax.jms.QueueConnection;
import javax.jms.Session;
import javax.jms.QueueSession;
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
import javax.jms.Queue;
import javax.jms.QueueSender;

import javax.jms.JMSException;
import javax.jms.MessageListener;
import javax.jms.QueueReceiver;
import javax.jms.TextMessage;
import javax.jms.Message;
import javax.jms.ObjectMessage;

public class servizio_musica_di_ambiente extends ServiceProvider.ServiceProvider {

  //definizione variabili
            ...

  public servizio_musica_di_ambiente() throws Exception {
     super();//invocazione al ServiceProvider
     start_servizio();
  }



  public static void main(String [] args) throws Exception {
    new servizio_musica_di_ambiente();
  }

  private void start_servizio() throws Exception{

    // riferimento al Context Service
    ContextWS.ws.ContextServerInterfaceServiceLocator service = new
ContextWS.ws.ContextServerInterfaceServiceLocator();
    // Now use the service to get a stub to the service
    ContextWS.ws.ContextServerInterface servant = service.getcontextserver();

   //oggetto Mp3_player
    final Mp3.Mp3_player mp3servant=new Mp3_player();
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
…
//iniz. play list...
        …
        …
try{

  //creazione di un JNDI context
  Hashtable properties = new Hashtable();
  properties.put(Context.INITIAL_CONTEXT_FACTORY,
                "org.exolab.jms.jndi.InitialContextFactory");
  properties.put(Context.PROVIDER_URL, "http://localhost/");
  Context context = new InitialContext(properties);
  System.out.println("[INFO] JNDI context è OK");



  //recupero di una QueueConnectionFactory
  QueueConnectionFactory factory =
     (QueueConnectionFactory) context.lookup("JmsQueueConnectionFactory");
  System.out.println("[INFO] QueueConnectionFactory OK");
  //creazione di una connessione alla coda:
  QueueConnection connection = factory.createQueueConnection();
  connection.start();
  System.out.println("[INFO] connessione alla coda OK");
  //creazione di una sessione:
  QueueSession session =
     connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
  System.out.println("[INFO] sessione OK");
  //recupero della coda vera e propria:

  //Queue queue = (Queue) context.lookup("queue1");
  String nomecoda=null;
  try{
    nomecoda = servant.tracking_users_in_room("stanzaMotionCapture");
  }catch(Exception e){
```


PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

```
      System.out.println("[INFO] CHIAMATA AL METODO CON MONITORAGGIO CONTINUO FALLITA");
    }
    Queue queue = (Queue) session.createQueue(nomecoda);
    System.out.println("ottenuta coda");

    //recupero messaggi:

    QueueReceiver receiver = session.createReceiver(queue);
    receiver.setMessageListener(new MessageListener() {
      public void onMessage(Message message) {
          //implementazione della logica del servizio
          …
          …
    });
…
…
  }
}
```

fig. 5.14: Classe servizio_musica_di_ambiente.java

la figura 5.14 illustra un stralcio della classe principale del sistema ed in neretto evidenzia la chiamata al metodo opportuno del Context Service alla connessione tra il servizio e la coda dedicata a quest'ultimo.

Tramite la funzione *start_servizio()*, la classe servizio_musica_di_ambiente.java, svolge le seguenti attività:

- Ottiene un riferimento al Context Service.
- Istanzia un oggetto della classe Mp3.Mp3_Player.java che, sfruttando le librerie di Java Media Framework (JMF 2.1.1.e), ha il compito di creare i player per i brani musicali, che rappresentano la musica di sottofondo, e per i messaggi vocali.
- Effettua le operazioni necessarie per la creazione di un JNDI Context e per una sua esposizione come consumatore di messaggi in una coda JMS.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- Invoca il metodo esposto dal Context Service *tracking_users_in_room* per essere aggiornato, tramite una coda JMS, continuamente sulla identità degli utenti presenti nella stanza monitorata
- Preleva gli eventi generati dall'UbiSystem sottoforma di messaggi JMS ed in base al contenuto informativo prelevato implementa la logica del servizio.

5.3.1 Registrazione e configurazione del servizio

Il servizio si registra normalmente al Service Manager attraverso la classe Service Provider che fornisce la sua descrizione, al Service Manager, che andrà convalidata attraverso l'Ontology Service per verificarne la consistenza.
Il contenuto del file di configurazione del servizio in esame è il seguente:

```
<?xml version="1.0"?>
<config>
        <OntologyServerUrl>http://localhost/axis/services/ontologyserver</OntologyServerUrl>
        <description>mda.daml</description>
        <delay>80000</delay>
</config>
```

fig. 5.15: Contenuto del file ServiceConfig.xml associato al servizio di musica di ambiente.

Il file mda.daml contiene le informazioni riguardanti la descrizione del servizio. In figura 5.16 è riportato uno schema che riassume la descrizione del servizio.

Descrizione del servizio di musica di ambiente	
ResourceName	MDAService
ResourceTextDescription	MDAService è un servizio per la gestione della musica di sottofondo e di specifici messaggi vocali all'interno di una stanza
ResourceUrl	http://ubinet.it/MDAService

fig. 5.16: Schema della descrizione del servizio di musica di ambiente.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

5.4 Lo scenario in cui operano i servizi

Lo scenario in cui i servizi realmente operano è costituito da due stanze: Stanza della Segreteria e Stanza di Motion Capture.

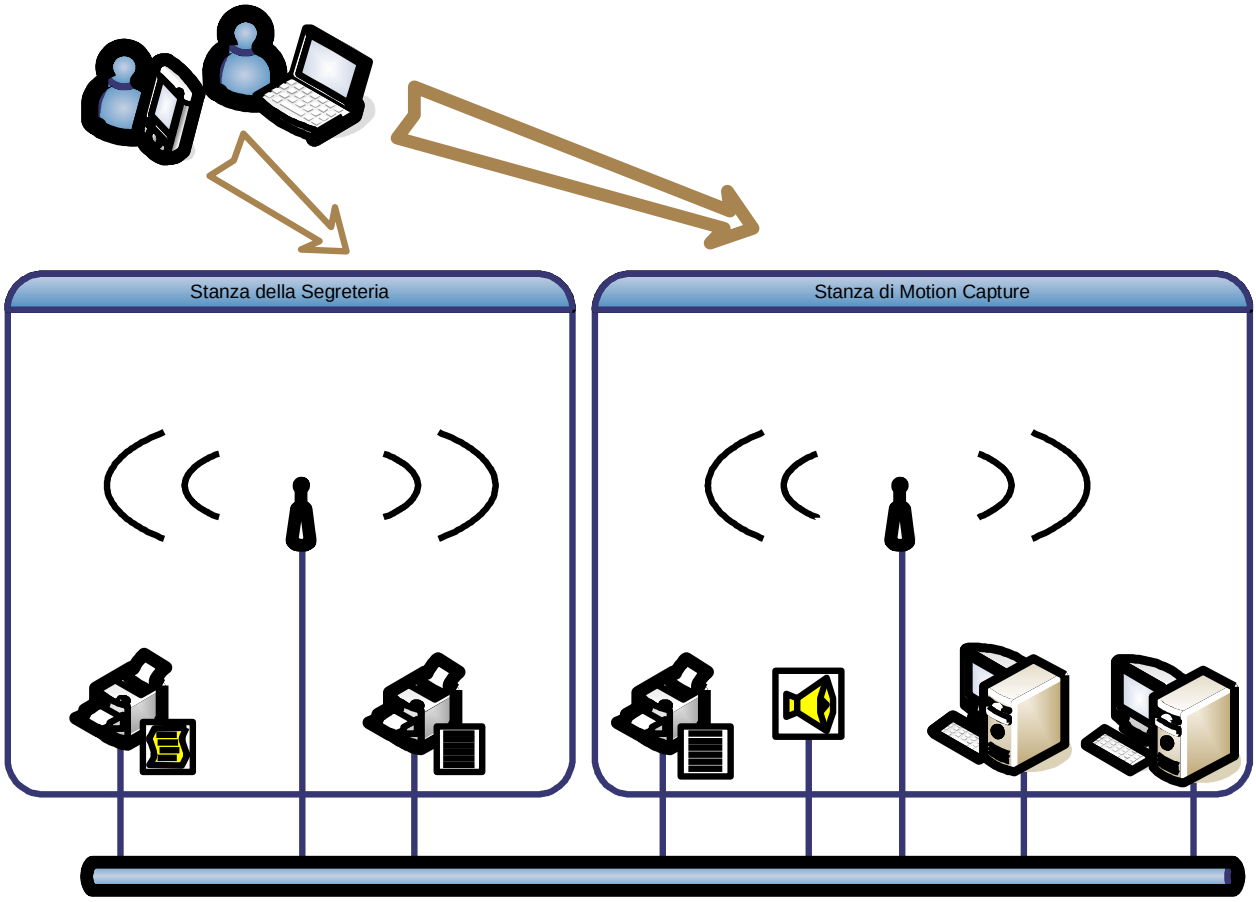

fig. 5.17: Scenario relativo all'ambiente pervasivo implementato.

Ciascuna stanza comprende un sensore, un access-point D-link, per il rilevamento dei dispositivi mobili che entrano a far parte dell'ambiente.

Nella Stanza della Segreteria sono installate due stampanti, una per la stampa a colori, l'altra per quella in bianco e nero. La stanza di Motion Capture comprende invece una stampante per la stampa in bianco e nero ed un impianto sonoro per la riproduzione, ad esempio, di brani musicali. L'intero ambiente è coperto da una rete LAN con access-point wireless.

I componenti software di UbiSystem sono poi dislocati su due specifici nodi della rete, fisicamente presenti nella stanza di Motion Capture, nel modo illustrato nella figura 5.18.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

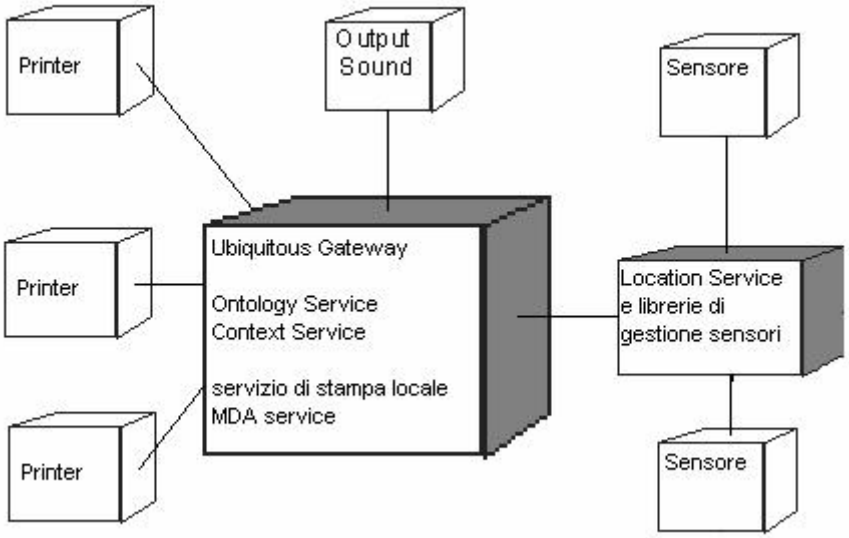

fig. 5.18: Dislocazione dei Programmi sui nodi della rete.

La disposizione dei programmi sui nodi della rete potrebbe essere ovviamente differente: ad esempio, si potrebbero associare alcuni nodi specifici ai servizi di sistema ed altri nodi specifici ai servizi esterni.

5.5 L'Interfaccia utente

I servizi realizzati sono operativi e disponibili presso determinati URL. Adesso occorre analizzare il problema dell'interazione con l'utente che, per nostra ipotesi, dovrà avvenire tramite browser.

Attualmente è lo stesso sviluppatore che mette a disposizione una serie di pagine Web attraverso le quali interagire per sfruttare le funzionalità offerte dal proprio servizio.

In alternativa, sarebbero possibili altri due tipi di scenari:

- nel momento in cui il servizio effettua la registrazione fornisce, oltre alla propria descrizione, anche l'interfaccia Web mediante la quale consumare il servizio; l'interfaccia verrebbe ospitata all'interno di un Web Application Server del sistema;

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

- lo sviluppo di un modulo capace di generare dinamicamente l'interfaccia Web per un servizio a partire dalla descrizione offerta.

Le due soluzioni proposte non sono esclusive ma complementari: si potrebbe adottare la seconda quando non è disponibile anche l'interfaccia utente.

Per ciascuno dei servizi pubblici è stata sviluppata un'applicazione web che offre una semplice interfaccia grafica per richiamare le funzionalità del relativo Web Service o semplicemente per mostrare la sua Home Page. Per tale scopo è stata adoperata la tecnologia delle Java Server Page e delle Servlet Java, che costituiscono una soluzione multipiattaforma per la realizzazione di pagine HTML dinamiche. JSP e Servlet permettono, infatti, di svincolarsi sia dal tipo di sistema operativo che dal tipo di browser, in quanto il linguaggio Java viene interpretato solamente dal lato server sollevando il browser da questo compito, limitando la sua funzione a semplice interprete di pagine HTML.

5.5.1 Start-up dell'ambiente

Affinché i servizi possano effettuare la registrazione i principali componenti dell'ambiente devono essere attivi ed in esecuzione. In particolare, è necessario avviare l'Ontology Service e caricare l'ontologia di base dell'ambiente, lanciare il Location Service ed in seguito il Context Service. È necessario caricare l'ontologia di base prima degli altri servizi perché attraverso l'ontologia di base l'Ontology Service può verificare le descrizioni dei servizi per validare la correttezza sintattica e semantica secondo quanto asserito nelle ontologie.

Fatto ciò è possibile avviare un qualunque servizio implementato: la classe Service Provider provvederà in automatico alla registrazione del servizio presso il Service Manager.

Ad esempio, allo start-up del servizio di stampa locale il seguente output viene generato al termine della corretta registrazione del servizio:

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

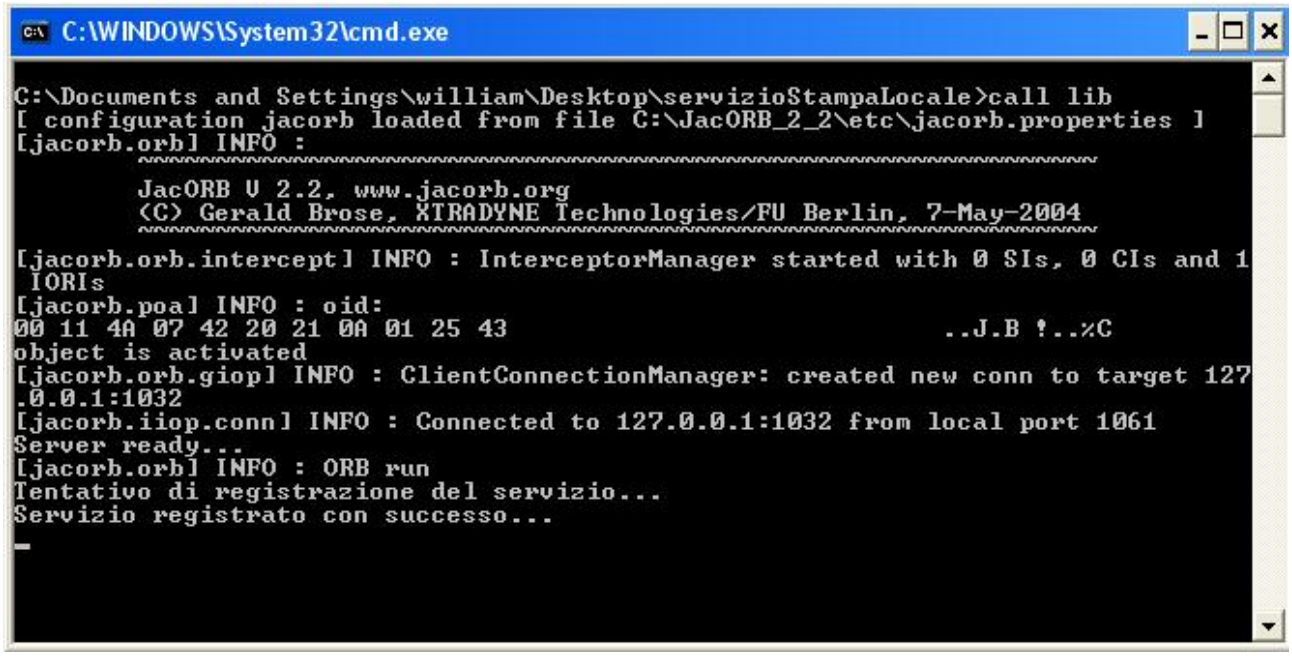

fig. 5.19: Attivazione e registrazione del servizio di stampa locale.

Quando i servizi sono in esecuzione è possibile accedervi tramite l'Ubiquitous Gateway.

L'entry point per l'ambiente è l'indirizzo http://www.ubinet.it/. La relativa pagina è mostrata in figura 5.20.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

fig. 5.20: Home Page dell'ambiente pervasivo.

Scegliendo la voce Servizi del Menù a sinistra, viene visualizzata la pagina in cui vengono elencati i servizi correntemente disponibili nell'ambiente insieme con una breve descrizione ed i relativi link.

Cliccando sui link presenti sarà possibile accedere alle funzionalità che ciascun servizio espone.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

fig. 5.21: Elenco dei servizi disponibili nell'ambiente.

5.5.2 Il servizio di musica di ambiente

Il link denominato mda, acronimo di Musica Di Ambiente, di figura 5.21 porta alla corrispondente home page del servizio. Quest'ultima è costituita dalla sola descrizione testuale del servizio. Non presenta alcun pulsante di interazione con l'utente: il servizio di musica di ambiente non prevede alcun controllo dall'esterno. Se il servizio è in esecuzione non è più possibile, ad esempio, cambiare la stanza su cui avviene il monitoraggio, la musica di sottofondo o il messaggio vocale di benvenuto sebbene ciò potrebbe essere molto utile per il personale addetto alla gestione del servizio.

La scelta di evitare alcun controllo dall'esterno e quindi l'esposizione di opportuni metodi come Web Services da parte del servizio di musica di ambiente, è legata al fatto che chiunque potrebbe avere accesso a tali funzionalità. Anche personale non addetto alla manutenzione del servizio potrebbe modificarne le impostazioni: UbiSystem infatti, benché lo preveda, non implementa

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

ancora lo User Manager Service a cui spetta, tra l'altro, l'assegnazione ed il controllo dei privilegi di accesso che ciascun utente ha verso uno specifico servizio dell'ambiente.

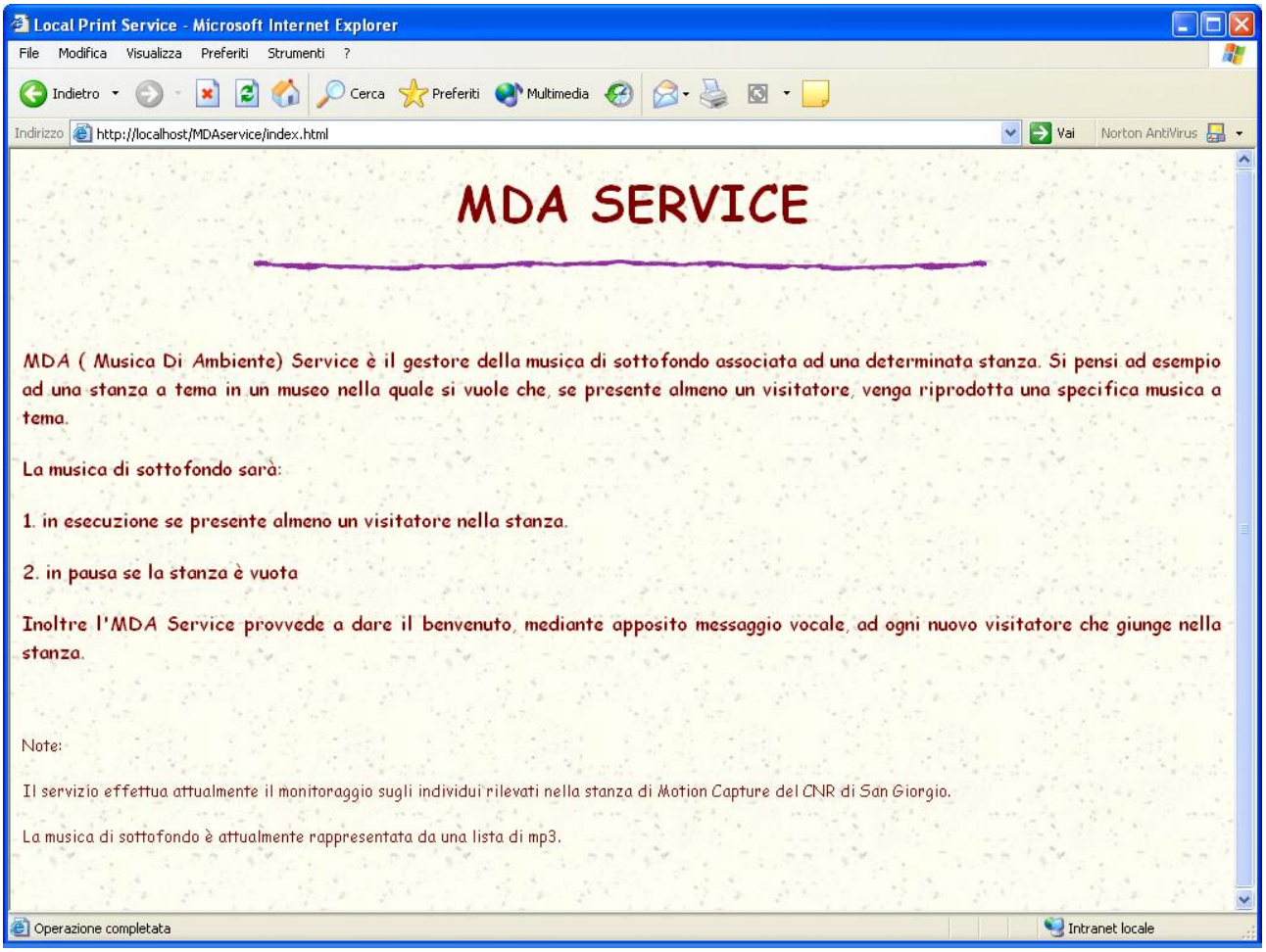

fig. 5.22: Home page del servizio di musica di ambiente.

5.5.3 IL servizio di stampa locale

Dalla pagina web contenente l'elenco dei servizi disponibili è possibile accedere alla home page del servizio di stampa locale mediante il link localPrintService. Eseguendo tale operazione viene visualizzato dal browser la pagina illustrata in figura 5.23.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

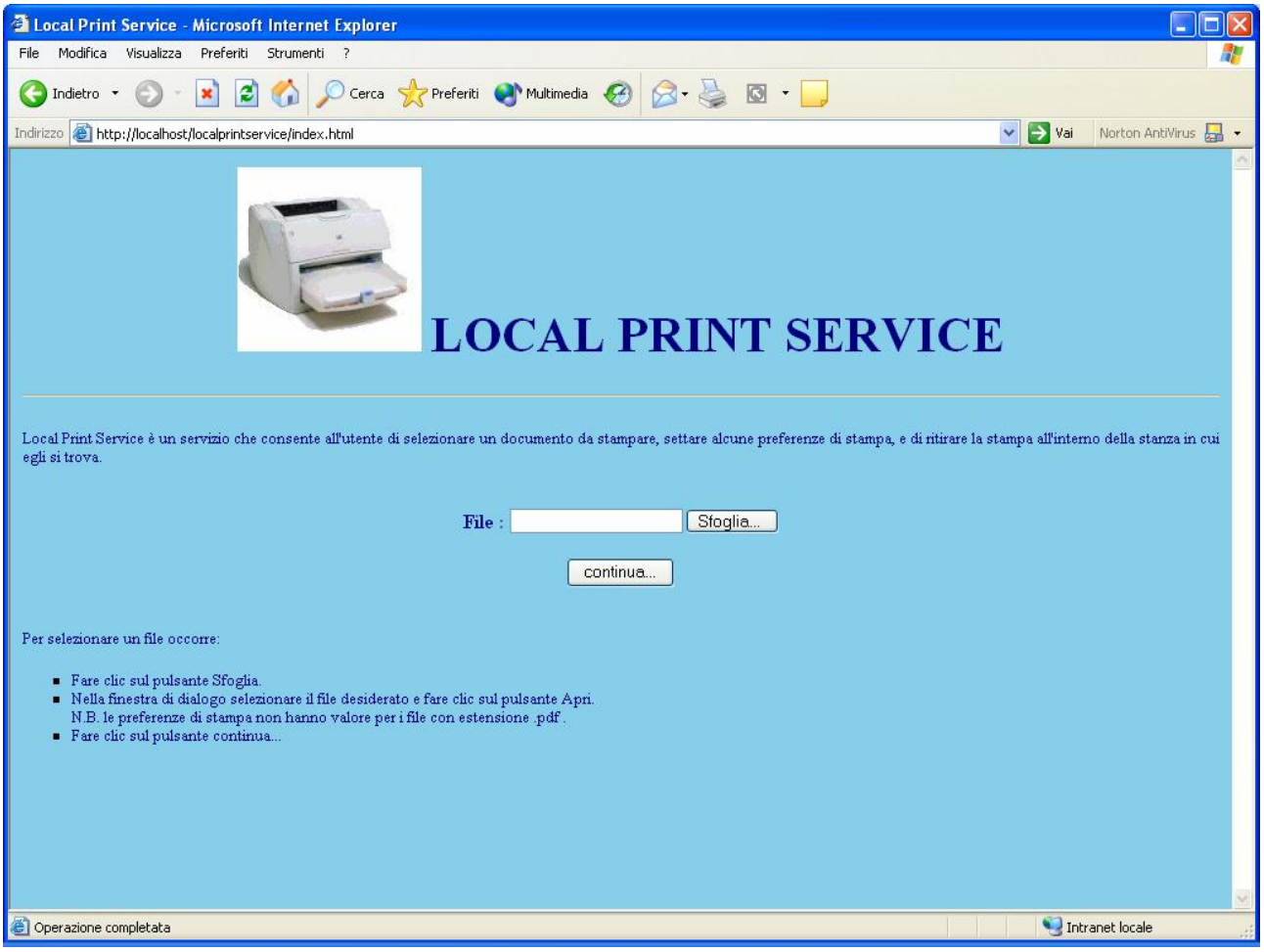

fig. 5.23: Home page del servizio di stampa locale.

La pagina contiene una breve descrizione del servizio ed una breve guida per eseguire l'upload del documento che si intende stampare. L'utente può selezionare il documento che intende stampare attraverso il pulsante *Sfoglia*.

In seguito alla pressione del pulsante *Sfoglia* viene visualizzata la finestra che guida l'utente nella selezione del documento da stampare.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

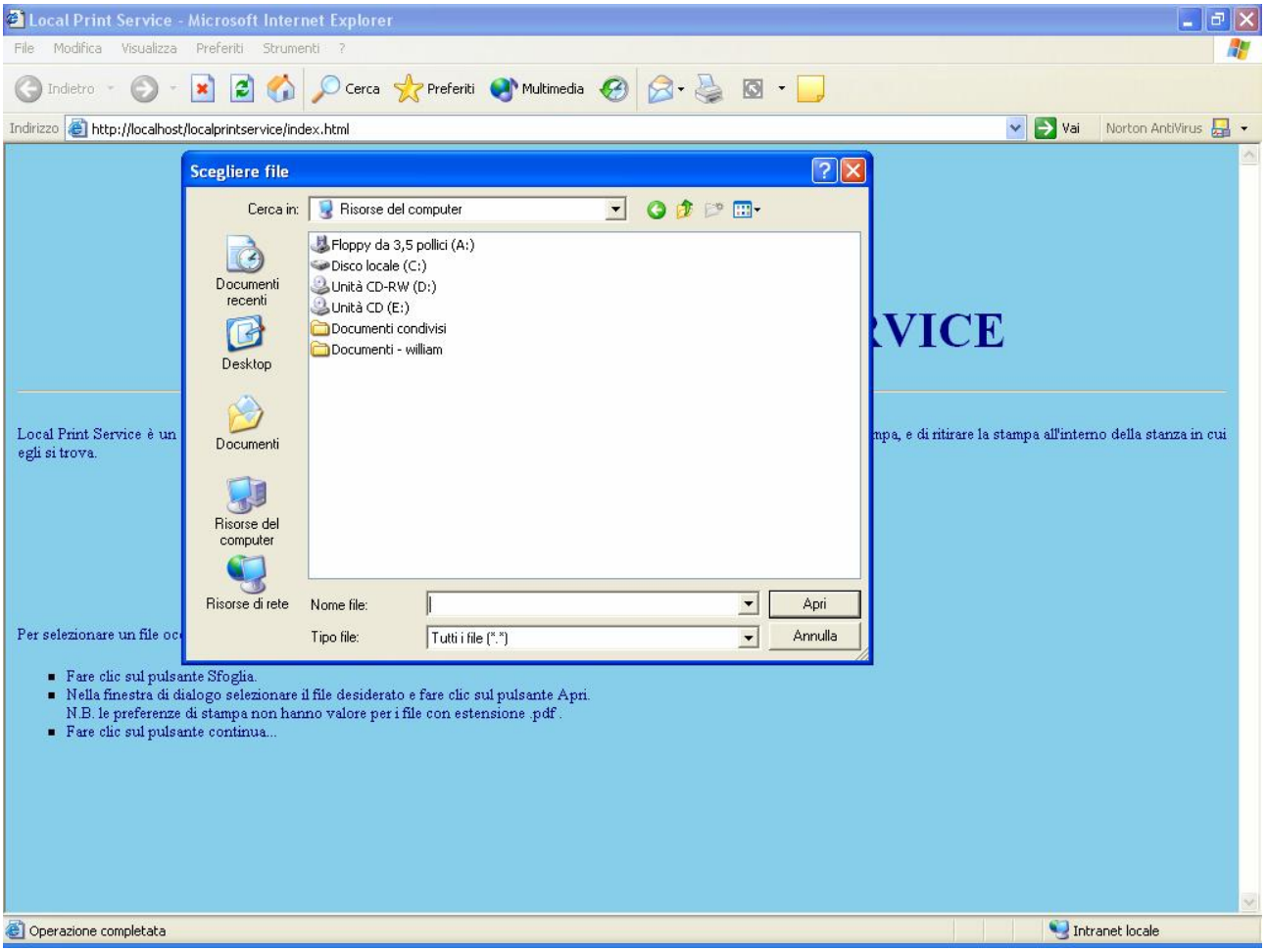

fig. 5.24: Finestra che guida l'utente nella scelta del file da stampare.

Selezionato il documento da stampare, l'utente può cliccare sul pulsante *continua*, presente nella home page del servizio, affinché il servizio possa poi guidarlo nella selezione delle preferenze di stampa attraverso la pagina illustrata in figura 5.26. Qualora l'utente avesse utilizzato il pulsante *continua* prima di selezionare alcun documento da stampare, il servizio di stampa locale l'avrebbe avvertito con il messaggio illustrato in figura 5.25.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

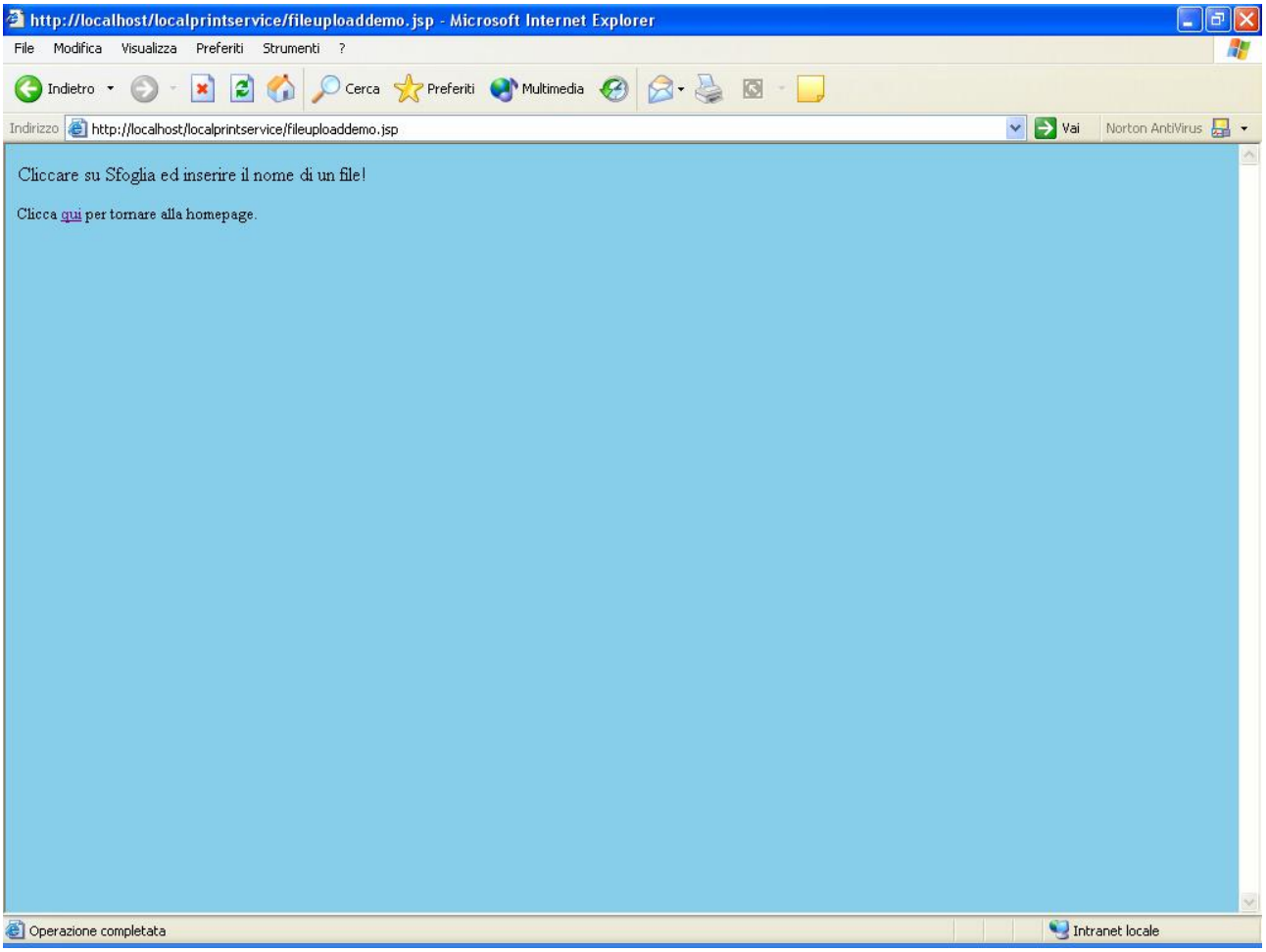

fig. 5.25: Messaggio di errore visualizzato dal servizio quando l'utente intende settare le preferenze di stampa senza aver selezionato alcun documento da stampare.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

fig. 5.26: Pagina Web per la selezione delle preferenze di stampa.

A questo punto l'utente deve settare le proprie preferenze di stampa. All'interno della pagine illustrata in figura 5.26, l'utente può selezionare la stampa a colori o in bianco e nero, mediante gli apposito bottoni radio, può ritornare alla home page per selezionare un documento da stampare diverso da quello corrente, mediante il pulsante *ritorna*, oppure può avviare la stampa del documento, con le preferenze settate, attraverso il pulsante *stampa*.

A questo punto il servizio effettua il discovery delle stampanti presenti nella stanza dell'utente che possono soddisfare le richieste di stampa. Il discovery è possibile perché, ricordiamo , a ciascuna stampante installata nell'ambiente corrisponde un sua descrizione all'interno della ontologia di base.
Se presenti, il servizio avvia la stampa e ne comunica il buon esito attraverso il messaggio illustrato in figura 5.27.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

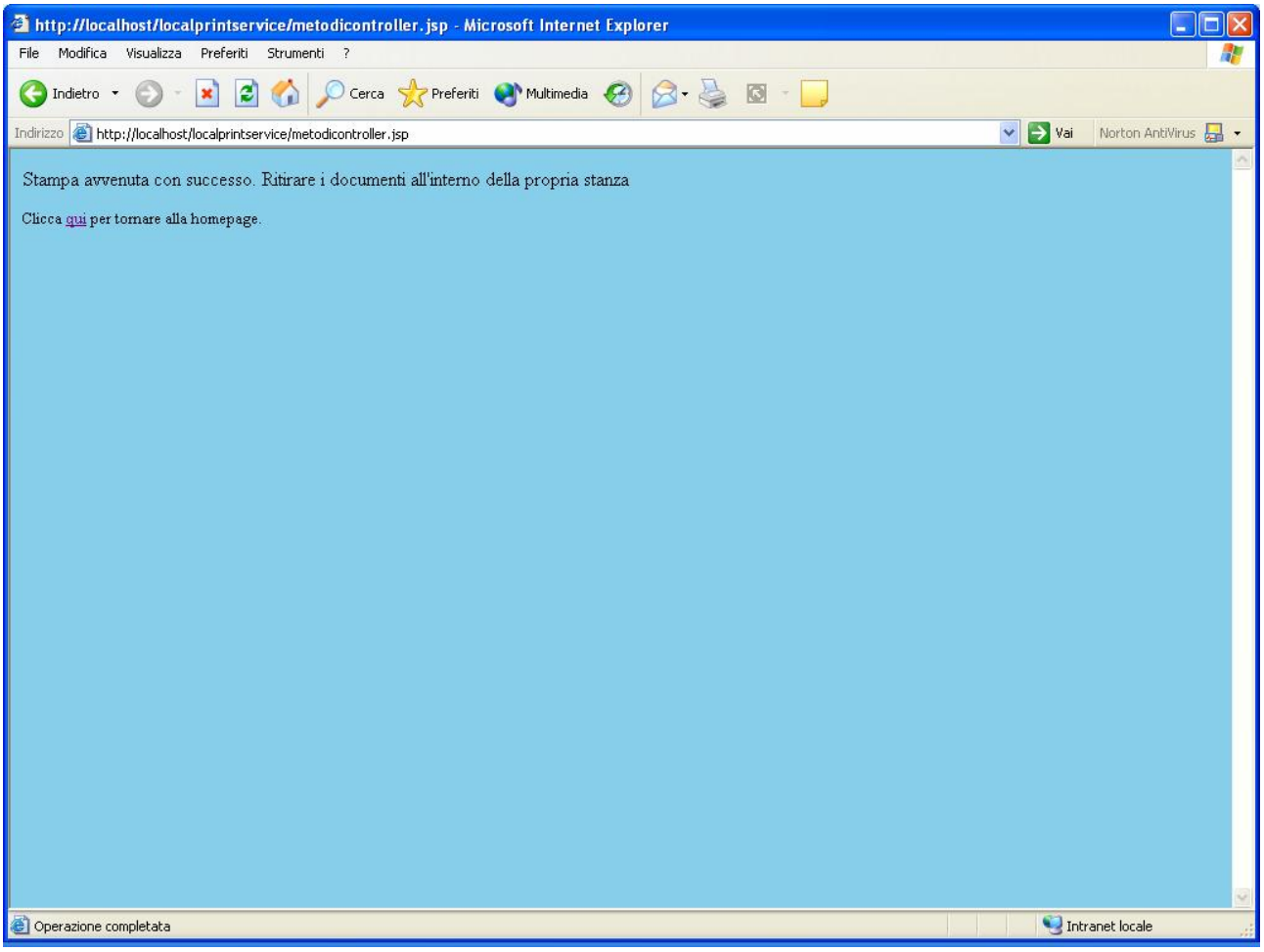

fig. 5.27: Messaggio di stampa avvenuta con successo.

Qualora il servizio non trovi alcuna stampante che soddisfi le preferenze di stampa, all'interno della stanza in cui si trova dell'utente, comunicherà a quest'ultimo la possibilità di ritirare la stampa all'interno di un'altra stanza mediante la pagina web illustrata in figura 5.28.

La pagina web di figura 5.28 contiene un elenco con tutte le stampanti in grado di soddisfare le richieste di stampa presenti nell'edificio e indica, per ciascuna stampante, la stanza in cui si trova.

Basandosi su quest'ultima informazione l'utente sceglie in quale stanza prelevare la stampa.

L'utente comunica al servizio la sua scelta mediante l'apposito pulsante. Il servizio avvia prima la stampa sulla stampante nella stanza scelta dall'utente e poi mostra a quest'ultimo una pagina web che, tramite messaggio di testo, gli ricorda la stanza dove ha scelto di ritirare i documenti stampati.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

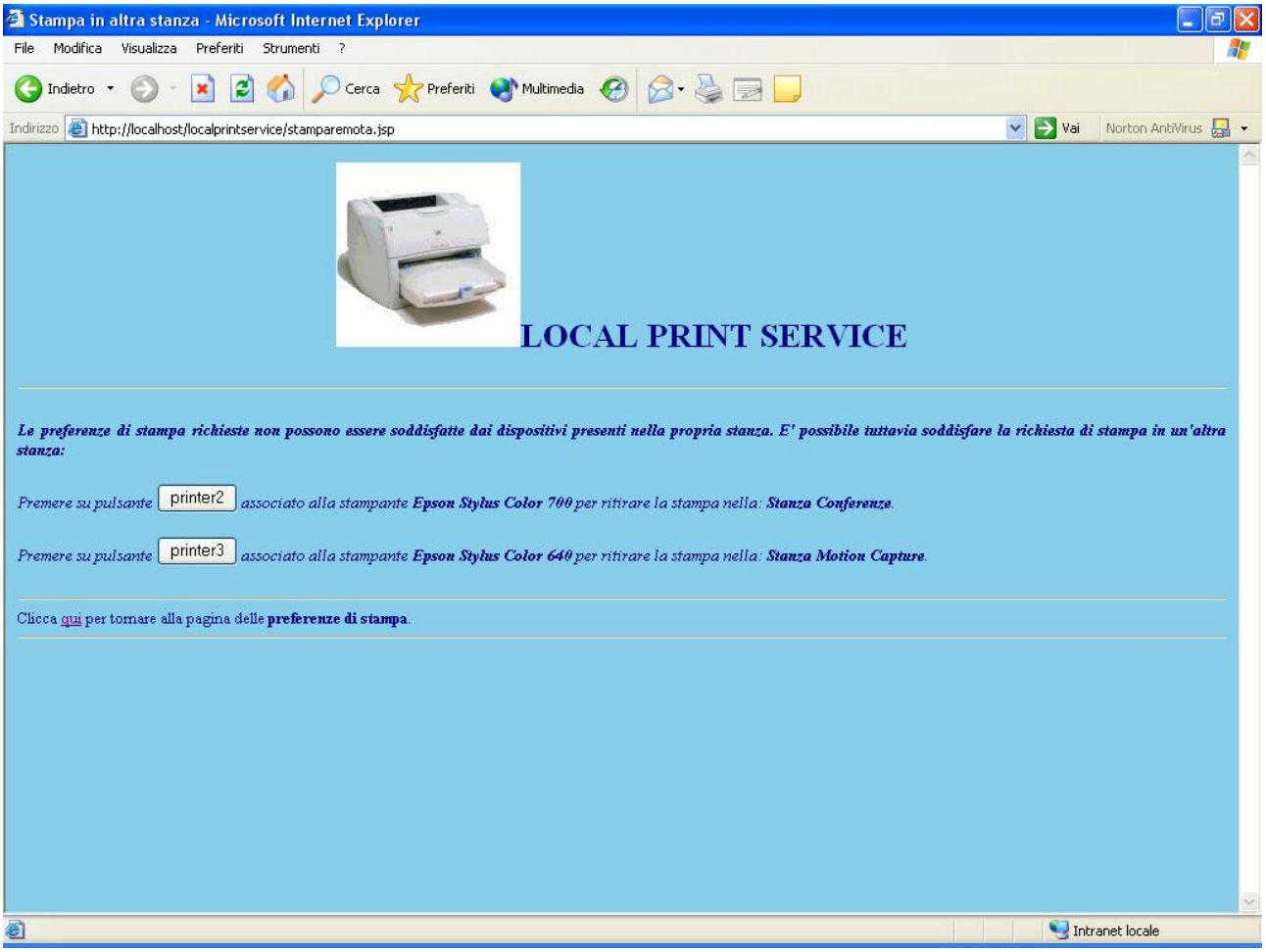

fig. 5.28: Pagina web attraverso la quale l'utente può avviare la stampa in una stanza a sua scelta.

Quando non esiste alcuna stampante in grado di soddisfare le richieste di stampa il servizio invia all'utente l'apposito messaggio.

Le informazioni relative al nome ed alla posizione delle stampanti, elencate nella pagina web sopra illustrata, sono contenute nella ontologia di base.

Come accennato nel precedente capitolo, in una successiva implementazione di UbiSystem, verrà realizzato un servizio di sistema, un Device Manager Service, che provvederà alla installazione, al monitoraggio ed alla disinstallazione automatica dei dispositivi che devono costituire parte integrante dell'ambiente pervasivo.

Compito del Device Manager Service sarà, in fase di installazione, la generazione automatica della descrizione ontologica per ciascun dispositivo e l'inserimento di quest'ultima all'interno della ontologia di base. La descrizione ontologica di uno specifico dispositivo sarà parte

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

integrante della ontologia di base fintanto che il dispositivo risulti presente e correttamente funzionante nell'ambiente.

Il servizio di stampa locale, pertanto, non prevede la gestione dei malfunzionamenti delle stampanti installate o eventuali errori di comunicazione, tra server, su cui il servizio è in esecuzione, e stampante, tali da inibire la stampa.

Poiché le informazioni sulle stampanti installate nell'ambiente vengono attualmente inserite a mano, nella ontologia di base, può facilmente accadere, semplicemente non digitando correttamente il nome di una stampante, che risultino, come esistenti e funzionanti, stampanti che di fatto non esistono nell'ambiente pervasivo implementato. In questo caso il servizio prevede un apposito messaggio di errore che è illustrato nella figura 5.29.

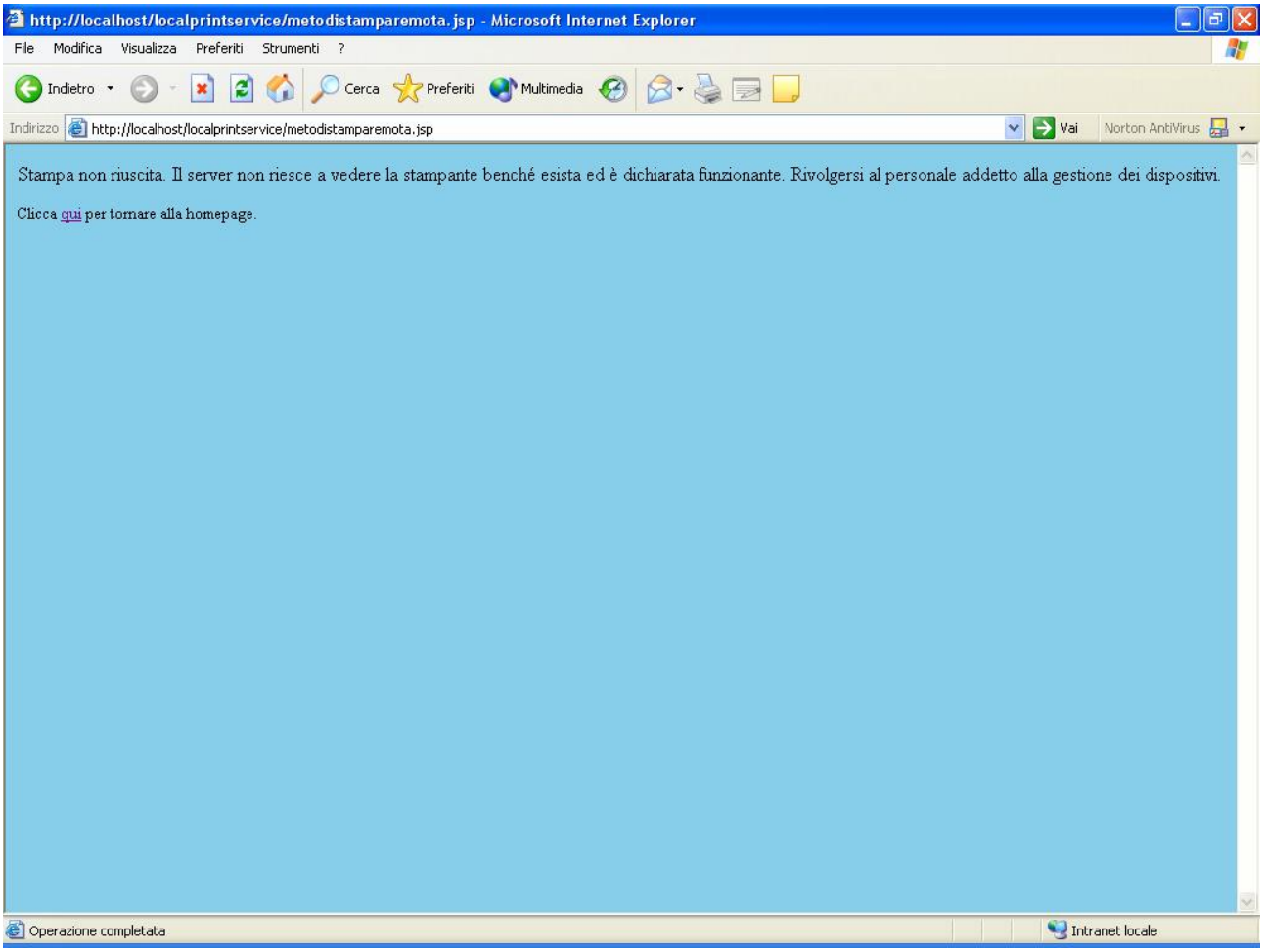

fig. 5.29: Messaggio di errore generato quando una stampante non è installata correttamente nell'ambiente pervasivo implementato.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Conclusioni e sviluppi futuri

Il nuovo prototipo implementato adotta tecnologie web-based: l'interazione con l'ambiente e la ricerca delle risorse disponibili al suo interno avviene per mezzo di un semplice browser.
Per la realizzazione dei servizi sono state indicate le emergenti tecnologie dei Web Services, il cui punto di forza è di utilizzare un set base di protocolli disponibili ovunque, garantendo l'interoperabilità tra piattaforme diverse e mantenendo, comunque, la possibilità di utilizzare protocolli più avanzati e specializzati per effettuare compiti specifici.
Inoltre, l'approccio seguito ha visto l'impiego di elementi del Web Semantico. In particolar modo, si è fatto ricorso all'uso di ontologie per garantire l'interoperabilità sintattica e semantica fra le diverse entità presenti nell'ambiente e per fornire dei meccanismi avanzati di condivisione delle informazioni e di discovery dei servizi.
Il nuovo prototipo di UbiSystem soddisfa ora i requisiti di sensibilità al contesto ed alla locazione che sono necessari in un ambiente pervasivo: un utente, munito di dispositivo mobile, può muoversi in maniera facile ed intuitiva all' interno dell'ambiente per scoprire ed utilizzare le risorse disponibili in relazione anche alla propria posizione o ad altre informazioni di contesto.
Sono stati sviluppati quindi due servizi esterni, il cui funzionamento si basa proprio sulla sensibilità di UbiSystem al contesto ed alla locazione: il servizio di stampa locale, che, per grosse linee, consente ad un utente di stampare all'interno della propria stanza, ed il servizio di gestione della musica di ambiente, per la gestione, appunto, della musica di sottofondo all'interno di una determinata stanza in relazione ai visitatori presenti in quest'ultima.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Ci sono problemi a cui il sistema implementato ancora non fornisce una soluzione.
Ad esempio non è stato ancora risolto il problema di come un dispositivo possa fornire la sua descrizione in maniera automatica, quando entra a far parte dell'ambiente. Tale descrizione è infatti indispensabile per determinare il set di servizi compatibili con il dispositivo che l'ambiente può offrire oppure per determinare i servizi che il dispositivo stesso può offrire nel momento in cui entra a far parte dell'ambiente, considerazione, quest'ultima, di rilievo se si pensa anche al caso in cui è l'utente che entra nell'ambiente e vuole esporre un proprio servizio.
Una possibile soluzione potrebbe essere quella di inviare le informazioni necessarie attraverso l'header HTTP nel momento in cui viene effettuata una richiesta. Su questa strada si sta muovendo il World Wide Web Consortium che sta già affrontando questo genere di problematiche.

Vi sono poi altre problematiche da approfondire o da affrontare come quelle legate alla gestione degli utenti del sistema. UbiSystem, benché lo contempli, non implementa infatti alcun servizio per la gestione degli utenti e quindi, ad esempio, della gestione dei loro privilegi di accesso ai servizi o alle risorse presenti nell'ambiente: chiunque può utilizzare qualunque servizio o qualunque risorsa presente nell'ambiente.
Oppure, altro aspetto da approfondire legato alla gestione degli utenti, è quello relativo alla identificazione di quest'ultimi da parte dell'ambiente: ciascun utente è identificato da un suo unico dispositivo e quest'ultimo deve essere in grado di comunicare con l'ambiente mediante protocollo TCP/IP affinché possa essere rilevato; di fatto, normalmente, a ciascun utente potrebbero corrispondere più dispositivi o nessun dispositivo e a ciascun dispositivo potrebbe corrispondere una tecnologia verso la quale il sistema non è in grado di interfacciarsi.

Non sono state toccate, inoltre, questioni riguardanti la sicurezza e la privacy delle informazioni scambiate: aspetti che nei contesti di pervasive computing diventano di fondamentale importanza.

Concludendo, l'architettura proposta rappresenta soltanto un ulteriore passo avanti verso la realizzazione di un ambiente pervasivo completo; molti ancora gli sforzi che si devono compiere e numerosi restano gli ambiti in cui la ricerca dovrà muoversi affinché l'*Ubiquitous Computing* diventi una realtà.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

Bibliografia

[1] Anind K.D., *"Providing Architectural Support for Building Context-Aware Application"*, PhD thesis, Georgia Institute of Technology, 2000.

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

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

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

[5] Barbeau M., *"Mobile, Distributed and Pervasive Computing"*, Handbook of Wireless Networks and Mobile Computing, John Wiley and Sons, Inc., Febbraio 2002.

[6] Barbieri L., *"Web Services: Protocolli e Strumenti, Interoperabilità ed evoluzioni future"*, 2003.

[7] Berners-Lee T., *"Sematic Web road map"*, http://www.w3.org/DesignIssues/Semantic.html.

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

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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

[10] Cooltown Homepage, *http://www.cooltown.com*.

[11] DAML-S, http://www.daml.org/services

[12] DARPA Agent Markup Language Homepage, *http://www.daml.org/*

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

[14] GAIA Homepage, *http://choices.cs.uiuc.edu/gaia/*.

[15] Grimm R. et al., *"System Directions for Pervasive Computing"*, University of Washington, 2000.

[16] Harms D., *"JSP Servlet e MySql"*, Mc Graw-Hill Editore, Milano, 2002.

[17] Rakotonirainy A., Henricksen K., Indulska J., *"Infrastructure for Pervasive Computing: Challenges"*, School of Computer Science and Electrical Engineering, The University of Queensland, 2000.

[18] IEEE Pervasive Computing Homepage, *http://www.computer.org/pervasive/*.

[19] Joshi A., Kagal L., Korolev V., Avancha S., 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.

[20] Barton J., Kindberg T., *"A web-based nomadic computing system"*, White paper, HP Labs, cooltown.hp.com, 2001.

[21] Fox A., Kindberg T., *"System Software for Ubiquitous Computing"*, IEEE Pervasive Computing, Gennaio-Marzo 2002.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

[22] Maffioletti F., *"Requirements for an Ubiquitous Computing Infrastructure"*, Cultura Narodov Prichernomoria Journal vol.3. Simferopol, Ukraine, Settembre 2001.

[23] Magnanini P., *"Sistemi di discovery: La soluzione Jini"*, Università degli Studi di Bologna, 2001.

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

[25] Sturm P., Mattern F., *"From Distributed Systems to Ubiquitous Computing"*, Department of Computer Science, Switzerland - Germany, 2002.

[26] McGrath R.E., Ranganathan A., Campbell R.H., Mickus M.D., *"Use of Ontologies in Pervasive Computing Environments"*, Dept. of Computer Science, University of Illinois, Urbana-Champaign, USA, Aprile 2003.

[27] Project Aura Homepage, *http://www-2.cs.cmu.edu/~aura/*.

[28] Ranganathan A., Campbell R.H., *"A Middleware for Context-Aware Agents in Ubiquitous Computing Environments"*, Department of Computer Science, University of Illinois at Urbana-Champaign, USA, 2002.

[29] Hess C., Roman M., Cerqueira R., Ranganathan A., Campbell R.H., Nahrsted K., *"A Middleware Infrastructure for Active Spaces"*, IEEE Pervasive Computing, Ottobre-Dicembre 2002.

[30] Mukherjee A., Saha D., "*Pervasive Computing: A Paradigm for the 21st Century"*, IEEE Computer Society, Marzo 2003.

[31] Saracco R., *"Ubiquitous Computing"*, Mondo digitale, n.3, Settembre 2003.

[32] Satyanarayanan M., *"Pervasive Computing: Vision and Challenges"*, IEEE Personal Communication, Agosto 2001.

[33] UDDI.org, *Universal Description, Discovery and Integration of Web Services*, http://uddi.org/.

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

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

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

[36] Weiser M., *"The Computer for the 21st Century"*, Scientific Am., Settembre 1991; riedito dall'IEEE Pervasive Computing, Gennaio-Marzo 2002.

[37] World Wide Web Consortium, *Extensible Markup Language (XML) 1.0 (Third Edition)*, http://www.w3.org/TR/2004/REC-xml-20040204/, Febbraio 2004.

[38] World Wide Web Consortium, *OWL Web Ontology Language Semantics and Abstract Syntax*, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/, Febbraio 2004.

[39] World Wide Web Consortium, *Resource Description Framework*, http://www.w3.org/RDF/.

[40] World Wide Web Consortium, *SOAP Version 1.2 Part 1: Messaging Framework*, http://www.w3.org/TR/soap12-part1/, Giugno 2004.

[41] World Wide Web Consortium, *Web Services Architecture*, *http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/*, Febbraio 2004.

[42] World Wide Web Consortium, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/wsdl, Marzo 2001.

[43] Zhu F., Mutka M., Ni L., *"Classification of Service Discovery in Pervasive Computing Environments"*, Michigan State University, East Lansing, 2002.

[44] Stefano Russo, Carlo Savy, Domenico Cotroneo, Antonio Sergio, *"Introduzione a CORBA"*, McGraw-Hill, 2002

[45] Pradeep Gore, Ron Cytron Douglas Schmidt, Carlos O'Ryan, "*Designing and Optimizing a Scalable CORBA Notification Service"*

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com

[46] Horrocks, Ian and Sattler, “CORBA-FaCT”, http://www.cs.man.ac.uk/~horrocks/FaCT/CORBAFsCT.html

[47] OpenJMS, http://openjms.sourceforge.net

[48] JacORB, http://www.jacorb.org

PDF Creator - PDF4Free v2.0 http://www.pdf4free.com