**UNIVERSITÀ DEGLI STUDI DI BOLOGNA**
FACOLTÀ DI INGEGNERIA

Corso di laurea specialistica in Ingegneria Informatica

Reti di Calcolatori LS

# INFRASTRUTTURE PER LA PROFILAZIONE E L'ADATTAMENTO DI SERVIZI MULTIMEDIALI

**Candidato:** Luca Monaco

**Relatore:** Chiar.mo Prof. Ing. ANTONIO CORRADI
**Correlatori:** Dott. Ing. LUCA FOSCHINI
Chiar.mo Prof. Ing. PAOLO BELLAVISTA

**Anno accademico 2003 – 2004**

***Parole chiave***

Sistemi multimediali

Eterogeneità dei terminali

Adattamento di contenuti multimediali

Profilazione degli utenti

Agenti mobili

*Alla mia famiglia*

# INDICE

# INTRODUZIONE

La tecnologia dell'informazione si sta evolvendo verso uno scenario in cui l'accesso alle informazioni e ai servizi avverrà sempre più attraverso una molteplicità di terminali dalle caratteristiche molto diverse tra loro (e.g. computer, portatili, telefoni cellulari, smartphone, ecc. ).

D'altra parte stiamo assistendo ad una sempre maggior diffusione di applicazioni multimediali che cercano di coinvolgere un pubblico sempre più vasto offrendo servizi sempre più evoluti. Tuttavia un grosso limite alla diffusione di tali applicazioni è rappresentato dall'ingente quantità di risorse necessaria al loro funzionamento, non sempre a disposizione dei dispositivi utilizzati dagli utenti per accedere ai servizi offerti.

Per risolvere questo problema, la direzione intrapresa negli ultimi anni dalla comunità scientifica è quella di studiare approcci che permettano all'utente finale di raggiungere informazioni e servizi di interesse attraverso l'utilizzo di un qualsiasi terminale. Tali approcci affrontano il problema della personalizzazione dei contenuti, che può essere correlata non solo alle caratteristiche fisiche dei dispositivi di accesso, quali per esempio dimensione delle schermo e della memoria, ma anche ad altre "coordinate", molto spesso ortogonali fra loro. Tra queste si possono citare le specifiche esigenze dell'utente in termini ad esempio di preferenze individuali o di categoria, e di caratteristiche culturali ed ambientali legate alla sua locazione geografica quali per esempio la lingua utilizzata.

Obiettivo di questa tesi è lo studio di un sistema di adattamento di servizi multimediali orientato in particolare alle caratteristiche fisiche dei dispositivi che ne fanno richiesta. Il risultato di tale studio sarà la proposta e lo sviluppo di un'architettura per la realizzazione di tale sistema che verrà progettato nell'ottica di arricchire un'infrastruttura di supporto allo sviluppo di applicazioni multimediali, realizzata presso il Dipartimento di Elettronica Informatica e Sistemistica (DEIS) dell'Università di Bologna.

La tesi sarà organizzata come segue. Nel primo capitolo verrà presentato lo scenario in cui operano le applicazioni multimediali evidenziando la forte

eterogeneità che lo caratterizza e la necessità di studiare servizi per l'adattamento di contenuti di tipo multimediale. Il capitolo 2 sarà dedicato allo studio delle possibili strategie di adattamento adottabili e alla discussione di alcune delle soluzioni più rappresentative presentate in alcuni recenti lavori di ricerca. Nel capitolo 3 verranno presentate le tecnologie attualmente più utilizzate per la rappresentazione delle informazioni riguardanti le caratteristiche del dispositivo utilizzato dall'utente e di quelle relative ai contenuti multimediali. Relativamente agli aspetti legati alla tematica dell'adattamento, nel capitolo 4 verrà presentato MUM, il middleware che verrà arricchito del sistema sviluppato nel progetto di tesi che verrà presentato nei capitoli 5,6 e 7. In particolare il capitolo 5 sarà dedicato all'analisi del sistema di adattamento e alla sviluppo della sua architettura. Nel capitolo 6 verranno illustrate le scelte progettuali che hanno guidato l'implementazione del sistema, presentata nel capitolo 7 unitamente ai risultati raccolti nella fase di sperimentazione.

# *Capitolo 1*

# SUPPORTO DELL' ETEROGENEITÀ NELLE APPLICAZIONI MULTIMEDIALI

Comunicare sempre e ovunque è diventato un obiettivo chiave dello sviluppo tecnologico degli ultimi anni, in cui dispositivi mobili per la comunicazione come cellulari, smartphone, PDA, notebook hanno acquisito una sempre maggiore capacità computazionale diventando il target economico, attuale e del prossimo futuro, dell'industria ICT (Information and Comunication Tecnology). Fino a qualche anno fa, le applicazioni multimediali, richiedendo una disponibilità di risorse abbastanza elevata, erano supportate in modo limitato e solo da una parte di questo tipo di terminali. Oggi al contrario, anche grazie allo sviluppo di nuove tecnologie di comunicazione wireless e a banda larga (e.g. GPRS, UMTS, 802.11g), sono tornate ad essere un argomento centrale nei centri di ricerca del settore.

La prospettiva che si sta delineando è la creazione di una rete di comunicazione in grado sostenere *ubiquitous computing*: scenario di elaborazione distribuita delle risorse in cui gli utenti hanno la capacità di muoversi e la possibilità di accedere, da qualunque luogo e in qualsiasi momento, a servizi che tengono in considerazione l'ampia eterogeneità di dispositivi, utenti e mezzi di trasmissione che caratterizza l'attuale panorama delle tecnologie di telecomunicazione.

Uno dei maggiori problemi nella realizzazione di questa prospettiva è la gestione di queste eterogeneità che saranno argomento di analisi in questo capitolo, in cui si individuerà nell'adattamento un potente strumento per la distribuzione di contenuti multimediali personalizzati e si illustreranno, in modo molto introduttivo, i servizi che tale sistema dovrebbe realizzare per svolgere i propri compiti e alcuni dei suoi aspetti caratteristici più interessanti.

## 1.1 Eterogeneità e applicazioni multimediali

Uno dei problemi più importanti da affrontare nell'ottica di progettare applicazioni multimediali distribuite è la gestione dell'enorme eterogeneità di piattaforme e dispositivi esistenti nell'attuale panorama tecnologico. D'altre parte l'eterogeneità è una caratteristica che non interessa solo i destinatari delle informazioni, bensì un'ampia gamma di entità coinvolte nell'intero scenario di distribuzione (Provider, canale di comunicazione e informazione stessa).

### 1.1.1 Analisi e classificazione delle Eterogeneità

Lo scenario base di distribuzione di contenuti multimediali, schematizzato in Figura **1.1**, può essere presentato attraverso un normale schema di comunicazione cliente servitore. Le classi di oggetti che lo compongono sono: l'utente (il Client), il mezzo trasmissivo ed il fornitore dei contenuti (il Server).

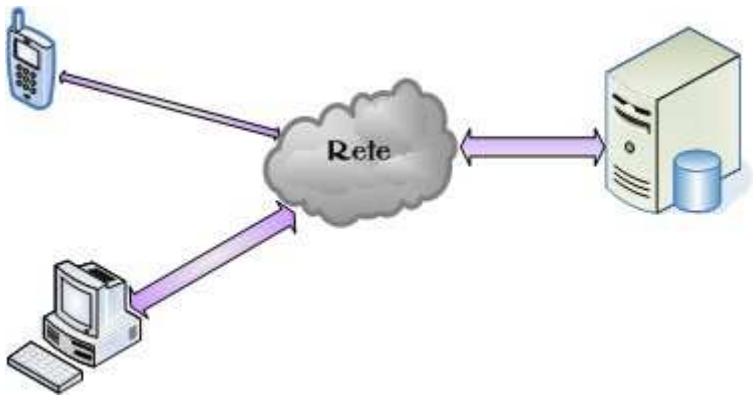



**Figura 1.1**: Modello di comunicazione Client/Server

In modo dipendente dall'applicazione possono esistere aspetti che differenziano notevolmente entità della stessa classe.

#### 1.1.1.1 Eterogeneità dell'Utente

In accordo a [Torl03], un utente può essere caratterizzato da aspetti legati a tre aree fondamentali:

1. La piattaforma Hardware/Software utilizzata per accedere ai servizi;
2. L'insieme dei suoi interessi e delle sue preferenze;
3. Il contesto geografico:

- Posizione geografica del luogo da cui invia le richieste;
- Risorse disponibili nell'ambiente circostante.

Mentre è facilmente comprensibile il tipo di informazione espressa nella prima e terza area, probabilmente è meno intuitivo capire che cosa si vuole comunicare nella seconda: le preferenze possono essere utilizzate per attribuire un peso a oggetti con un particolare valore semantico, in modo da indicare la parte dei contenuti e/o delle loro rappresentazioni che l'utente considera di maggior significato.

La rilevanza di una area rispetto alle altre e le informazioni che ognuna di esse deve contenere sono strettamente dipendenti dal tipo di applicazione che si sta considerando. Ad esempio, nell'ambito delle applicazioni multimediali le informazioni sulla piattaforma utilizzata rappresentano sicuramente quelle di maggior importanza: non ha senso distribuire una certa presentazione multimediale che rispetta perfettamente le preferenze del cliente se poi il dispositivo che egli utilizza non è in grado di renderla correttamente sul proprio schermo.

#### 1.1.1.2 Eterogeneità del mezzo trasmissivo

Il mezzo di trasmissivo utilizzato dalle parti per la comunicazione è di fondamentale importanza: il suo stato può pesantemente condizionare l'efficienza della trasmissione di informazioni in qualunque contesto applicativo.

Alcuni dei parametri più significativi per la caratterizzazione di un mezzo di trasmissione digitale sono:

- la ***"larghezza di banda"***, che misura la sua portata, espressa solitamente in *bps* (numero di bit per secondo), in termini di dati che possono essere trasmessi nella sezione del mezzo trasmissivo nell'unità di tempo.

  Questo parametro, molto importante nelle applicazioni multimediali distribuite, rappresenta un vincolo molto stringente: un flusso multimediale, infatti, richiede solitamente una banda molto ampia, di conseguenza, affinché possa essere trasmesso, è necessario utilizzare un mezzo di trasmissione dalla portata adeguata. Lo studio di questo problema è avviato ormai da diversi anni e ha prodotto notevoli risultati attraverso la definizione di algoritmi di compressione in grado di ridurre la banda necessaria per la trasmissione di

oggetti di tipo multimediale, a scapito di un aumento dei costi di gestione della CPU dovuti alla loro elaborazione.

- la ***latenza***, che misura il ritardo introdotto dalla linea di trasmissione, cioè il tempo impiegato dai dati discretizzati e impacchettati per giungere dalla sorgente al ricevente.
- Il ***jitter***, che misura la varianza della latenza, cioè lo scostamento della stessa dal suo valor medio.

Come mostra l'articolo [Fugg04], nel mercato dei sistemi di comunicazione convivono molti Standard dalle caratteristiche di trasmissione molto differenziate, ad esempio considerando solo il panorama delle tecnologie wireless si possono trovare:

- Il ***Bluetooth***, che consente la creazione di tre canali per la voce e un canale asimmetrico per la scambio dei dati (con una portata di 720 Kbit/s in download e 57 Kbit/s in upload), in un area di dieci metri e con un mobilità limitata.
- I ***sistemi WLAN*** (formalmente denominati IEEE 802.11 a/b/g), che permettono la creazione di reti wireless in edifici o aree definite (nell'ordine delle centinaia di metri) e che raggiungono una bit-rate dichiarata di 54 Mb/s.
- I **sistemi cellulari** che garantiscono servizi voce e dati su vaste aree geografiche ed in grado di trasmettere da una media di 28,8 Kbit/s (del GSM) a un massimo di 348 Kbit/s (massimo dell'UMTS in condizioni favorevoli).

Inoltre, presto, oltre a queste soluzioni già disponibili, si affiancheranno nuovi standard in fase di avanzata definizione, come i sistemi UWB (Ultra Wide Band) già in fase di sperimentazione in alcuni laboratori universitari europei che potrebbero garantire ben oltre i 100 Mbit/s con ridotte emissioni di potenza.

#### 1.1.1.3 Eterogeneità del fornitore dei contenuti multimediali

Il Server, in generale, è molto legato al tipo di risorse che gestisce: ad esempio una prima semplicistica distinzione potrebbe vedere da un parte i provider dedicati alla distribuzione di servizi (tipo quello di posta elettronica) dall'altra quelli che offrono informazioni e contenuti. Ovviamente a seconda della categoria a cui

appartiene, il Server può aver bisogno maggiormente di un certo tipo risorse piuttosto che di altre. Un applicazione per i contenuti multimediali ha bisogno di molta memoria per il mantenimento dei dati e di altrettante risorse di elaborazione per la loro gestione e per l'impacchettamento dei dati da trasmettere.

La gestione dei dati multimediali non è troppo onerosa se essi sono presenti già in formato compresso, in caso contrario i costi di gestione, sia in termini di spazio fisico necessario che di risorse di elaborazione potrebbero raggiungere livelli insostenibili. Come si accennava in precedenza esiste una ampia varietà di scelta anche sui formati utilizzabili per la compressione del oggetto multimediale, dunque, un altro parametro sul quale si potrebbe costruire una nuova classificazione dei provider è il formato di compressione gestito. In realtà la maggior parte dei Server dedicati all'archiviazione di contenuti multimediali gestisce indifferentemente i più comuni formati in cui essi vengono distribuiti, concentrandosi a limite su un particolare tipo di media piuttosto che su di un altro. Tuttavia, per avere un quadro completo sulla complessità dello scenario analizzato in questo capitolo è sembrato opportuno esplorare anche i formati video più comuni.

#### 1.1.1.4 Eterogeneità dei contenuti multimediali

Una prima classificazione dei dati multimediali può essere fatta in base al loro tipo, in particolare si possono distinguere i seguenti di media:

1. testo
2. immagini;
3. solo audio;
4. solo video;
5. audio/video;

Il loro contenuto informativo e il conseguente livello quantitativo di risorse necessario per la loro riproduzione e trasmissione aumenta nell'ordine in cui sono stati elencati (dall'altro verso il basso).

Un parametro molto importante per differenziare oggetti multimediali dello stesso tipo è la qualità. Essa è una caratteristica abbastanza difficile da esprimere e rappresentare, soprattutto perché fa riferimento alle variabili capacità percettive degli individui. Tuttavia, nei contenuti di tipo multimediale, esistono parametri oggettivi

che riescono a dare una rappresentazione della qualità abbastanza precisa. Consideriamo ad esempio un flusso video: le caratteristiche oggettive di qualità che possono essere individuate sono:

- La risoluzione (espressa in pixel);
- La presenza di artefatti (deterioramenti delle immagini);
- La fluidità del video nelle varie situazioni (espressa in *fps*, frame per secondo);
- La quantità di informazioni da trasmettere/riprodurre in un secondo (bit-rate).

Affinché un oggetto multimediale possa essere gestito in modo efficiente, bisogna comprimerlo, ovvero apportare delle trasformazioni su di esso che permettano di eliminare le informazioni superflue in modo da non sconvolgere il contenuto percepito dall'utente. Il gran numero di algoritmi per la compressione e la conseguente forte eterogeneità dei formati in cui un oggetto multimediale può essere rappresentato è motivata dal fatto che le proprietà statistiche di ciascuna configurazione di bit entro la sequenza da comprimere incidono fortemente sul risultato complessivo della riduzione. Questa considerazione ha favorito il proliferare di metodologie per la compattazione orientate all' "importanza" delle informazioni che hanno superato di gran lunga le soluzioni general purpose impiegate inizialmente.

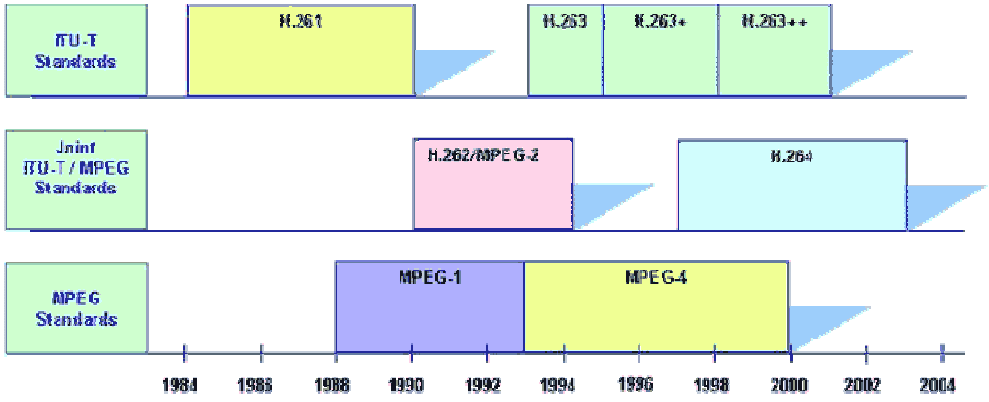



**Figura 1.2:** Standard sviluppati da ITU e ISO/IEC da 1984 ad oggi

La Figura **1.2**, tratta da [ITU-H264], mostra il lavoro svolto negli ultimi venti anni dalle due principali organizzazioni per lo sviluppo e la standardizzazione di formati

per la codifica del video digitale: l'ITU (International Comunication Union) con la gamma di standard H.26x ed ISO/IEC con gli standard MPEG. Le raccomandazioni ITU hanno un carattere più generico e nella maggior parte dei casi sono orientate alle applicazioni di comunicazione real-time come la video conferenza. Gli standard MPEG vengono definiti invece per necessità più ampie come lo storage (VideoCD, DVD), broadcast (cable, DSL, satellite e digitale terrestre) e streaming multimediale.

Il punto centrale intorno a cui ruotano tutti questi standard è il rapporto fra bit-rate e qualità, ovvero la quantità di dati necessaria per ogni secondo di video con una certa qualità. Non si analizzeranno in dettaglio il funzionamento delle tecniche di codifica per le quali si rimanda a [VideoCompression], ma nella tabella vengono forniti a scopo comparativo, alcuni dati che permetto di capire le loro capacità di compressione. I dati esprimono i parametri di compressione di un filmato da 90 minuti in qualità DVD (risoluzione 720x567 pixel, frame-rate 25 fps).

| Formato | Parametri di compressione | |
|---|---|---|
| | **Bit Rate (Mbit/s)** | **Spazio occupato (MB)** |
| **DigitalVideo** | 25 | 16850 |
| **Mpeg2** | 3 | 2025 |
| **Mpeg4** | 1,8 | 1215 |
| **H.263** | 2 | 1350 |
| **H.264** | 1,1 | 700 |

## 1.2 Soluzioni per il supporto dell'Eterogeneità

Nonostante l'esponenziale crescita tecnologica tenti di assottigliare le distanze tra terminali di categorie diverse, rimangono e continueranno a rimanere, differenze strutturali molto forti fra le varie tipologie di dispositivi che non permetteranno mai ad esempio a un cellulare di compiere esattamente le stesse funzionalità di un computer. Pensando ad esempio allo streaming di un video, su un cellulare sarà sempre necessario adattare il flusso alle dimensione del suo schermo o alla quantità di dati al secondo che il canale di comunicazione utilizzato riesce a trasferire. La

considerazione dell'eterogeneità anche solo al livello dei dispositivi dunque, complica abbastanza pesantemente la realizzazione delle applicazioni multimediali.

### 1.2.1 Aspetti architetturali

L'adozione di uno strato software (middleware) che astrae i dettagli della piattaforma utente potrebbe essere un ottima soluzione per facilitare lo sviluppo di applicazioni multimediali. Tuttavia, è sempre di fondamentale importanza tener conto del principio di compatibilità verso il basso: la maggior parte dell'utenza mobile, oggi, non dispone di apparecchi in grado di gestire dati multimediali e la copertura delle reti a larga banda è ancora minima. In questi casi, in cui le risorse del dispositivo utilizzato sono molto limitate, è difficile raggiungere l'utente con dati dal ricco contenuto informativo ed inoltre è possibile che le capacità computazionali del dispositivo siano così limitate da non permettergli di ospitare un middleware.

Le strategie architetturali utilizzabili per risolvere questo problema sono due: progettare un middleware modulare e configurabile in grado di scegliere automaticamente a tempo di esecuzione il set minimo di moduli da caricare, necessari per lo svolgimento di una certa attività, o in alternativa progettare soluzioni ad hoc per ogni singolo terminale. Questa seconda via abilita l'uso, dal lato cliente, di software proprietari, con relativi formati per la codifica del contenuto multimediale e introduce nel modello di comunicazione una figura intermedia tra l'utente e il fornitore del servizio: il proxy. Questa entità ospita il middleware multimediale (o parte di esso) e si pone come interfaccia del dispositivo cliente, infatti, conoscendo in dettaglio la sua piattaforma trasforma i contenuti multimediali nel formato supportato dal cliente.

Il proxy, oltre che per l'elaborazione dei contenuti, può essere usato anche per offrire altre funzionalità: ad esempio quella di accedere a insiemi di risorse di cui il cliente non può avere diretta visibilità o per negoziare il protocollo da utilizzare per la comunicazione in modo da sfruttare al meglio le caratteristiche del dispositivo e dell'ambiente circostante.

Nel lavoro di tesi è stata adottata un'architettura che va nella direzione della seconda presentata. La Figura **1.3** mostra lo scenario di distribuzione ed elaborazione

dei contenuti multimediali preso in considerazione, evidenziando l'aspetto collaborativo dei proxy.

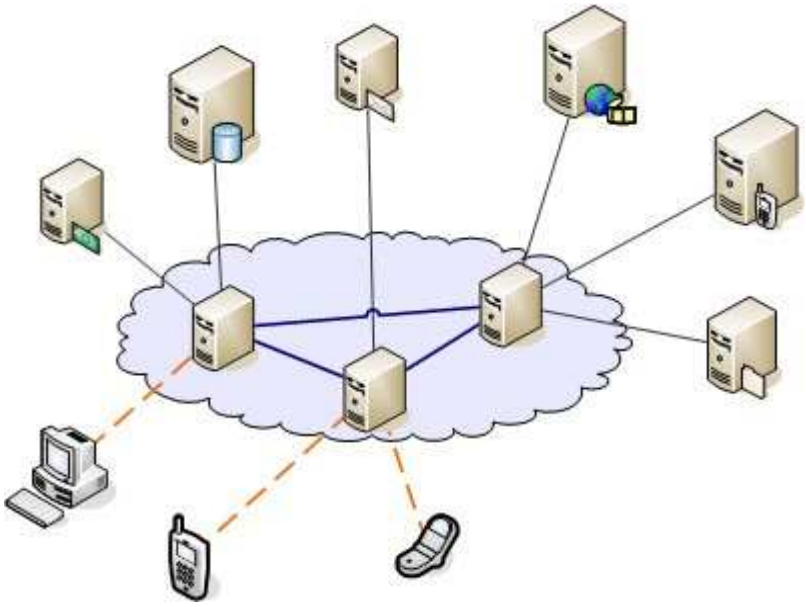


**Figura 1.3:** Scenario di elaborazione e distribuzione dei contenuti

### 1.2.2 Servizi essenziali per il supporto dell'Eterogeneità

I sistemi che offrono supporto all'eterogeneità vengono chiamati "Sistemi adattativi". L'adattatività è la caratteristica del sistema che esprime la sua capacità di modificare l'erogazione dei propri servizi sulla base dell'ambiente in cui essi verranno consumati. È dunque un proprietà assai desiderabile in uno scenario ampiamente eterogeneo, come quello illustrato nella prima parte del capitolo, in cui gli utenti possono usare i loro terminali per accedere a informazioni e contenuti in qualsiasi momento e in qualunque luogo si trovino.

Un middleware per facilitare lo sviluppo di sistemi di questo genere deve fornire un servizio di adattamento, che astraendo dalle peculiarità applicative, definisca componenti e servizi di base direttamente utilizzabili per:

- La rilevazione e la descrizione delle informazioni relative all'ambiente in cui il servizio verrà consumato.
- La descrizione e modellazione dei servizi e dei contenuti offerti dal sistema.
- La realizzazione di infrastrutture distribuite per l'adattamento dei contenuti all'ambiente utente.

Nei paragrafi successivi verranno discusse queste problematiche cercando di introdurre degli approcci utilizzabili per la loro soluzione.

#### 1.2.2.1 Profilazione dell'utente

Il primo passo verso la realizzazione di un sistema di adattamento è la costruzione di un sistema in grado di riconoscere le qualità degli utenti che richiedono i servizi. Le informazioni necessarie alla definizione del tipo di utente con il quale il sistema ha a che fare possono essere di varia natura ed in generale possono coinvolgere non solo caratteristiche dell'utente stesso, ma anche le risorse dell'ambiente in cui esso si trova. In letteratura questo ampio insieme di informazioni viene identificato con il termine *"contesto"*, la cui definizione più generale, data in [Anind00], lo definisce come un *qualunque insieme di informazioni che può essere usato per caratterizzare situazioni, persone, luoghi e oggetti considerati rilevanti ai fini dell'interazione tra utente e applicazione*.

Quando l'insieme delle informazioni rilevanti si limita al coinvolgimento delle sole caratteristiche dell'utente e non ne comprende altre riguardo l'ambiente esterno in cui esso si trova si preferisce parlare di *"profilo"* .

Molto spesso il profilo viene visto come entità multidimensionale, nella quale ogni dimensione corrisponde a una possibile coordinata di adattamento indipendente dalle altre. Quindi uno specifico profilo corrisponde a un punto, o più in generale a un insieme di punti, di questo spazio multidimensionale condiviso dai vari profili.

Alcune coordinate di interesse possono essere:

- ***Il dispositivo*** utilizzato per l'accesso ai servizi, con tutte le sue specifiche hardware e software in termini, per esempio, di dimensione dello schermo, dimensione della memoria, sistema operativo supportato e così via.
- ***Il canale di trasmissione*** usato per il collegamento del terminale utente con il sistema, caratterizzato in termini di capacità, affidabilità, qualità di servizio, sicurezza e costi.
- ***Caratteristiche e preferenze dell'utente***, in termini ad esempio di abilità di percezione della qualità di servizio, interessi, locazione geografica, fattori culturali e sociali.

Il profilo dunque è il mezzo attraverso il quale un sistema può descrivere le proprietà che caratterizzano un certo utente. Tuttavia, per adattare dei contenuti in base ad esso si ha il bisogno di rappresentare le informazioni che ne fanno parte in un linguaggio comprensibile al sistema di adattamento.

In generale, in letteratura le applicazioni che fanno uso di informazioni di contesto vengono chiamate *"Context-aware applications"*. Dal 1992, anno in cui la Olivetti nel progetto Active Badge [Want92] condusse una prima ricerca investigativa su questa tematica, la definizione delle problematiche legate al context-aware computing si sono evolute e oggi, come affermato in [Fugg04], la ricerca sembra convergere nel definire contex-aware *i sistemi che usano il contesto per fornire informazioni e/o servizi rilevanti agli utenti, dove il concetto di rilevanza dipende dal contesto applicativo in discussione.*

Secondo questa definizione, un'applicazione che semplicemente visualizza il contesto dell'utente senza modificare il proprio comportamento è di tipo contex-aware. Dunque un applicativo che monitora i cambiamenti del contesto e adatta i propri servizi in accordo ad esso è qualcosa di più. A questo proposito, in base al modo e allo scopo per il quale le informazioni di contesto vengono utilizzate, può essere creata una tassonomia, proposta da Shilit in [Shilit], che classifica le applicazioni context-awarness analizzando il loro comportamento secondo queste due dimensioni ortogonali.

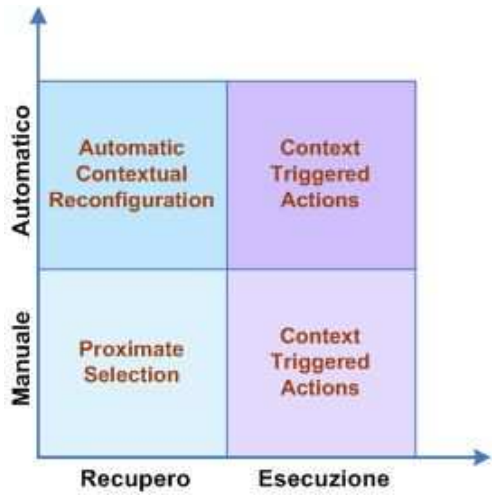



**Figura 1.4:** Tassonomia di Schilit

Come mostra la Figura **1.4**, nella tassonomia di Shilit si possono distinguere quattro tipi di applicazioni context-aware:

1. ***Proximate Selection***: Tecnica di interazione in cui viene presentata all'utente una lista di oggetti dove le entità rilevanti, in considerazione delle informazioni di contesto, sono enfatizzate o più facili da scegliere.
2. ***Automatic Contextual Reconfiguration***: Tecnica di sistema che, analizzando in modo automatico le informazioni presenti nel contesto, permette di creare un collegamento dinamico alle risorse disponibili.
3. ***Contextual Command***: Applicazioni che presentano all'utente i servizi disponibili per il proprio contesto e che vengono in messi in esecuzione al suo comando.
4. ***Context Triggered Actions***: Applicazioni che eseguono servizi selezionati per lo specifico contesto utente in modo automatico.

Poiché un profilo contiene solo un sottoinsieme delle informazioni rappresentanti un contesto, un sistema che lo utilizza per offrire un servizio come ad esempio quello per la personalizzazione dei contenuti, non può essere considerato context-aware. Tuttavia, restringendo la tassonomia di Shilit alle sole informazioni di profilo, tale sistema di adattamento ricadrebbe nell'ultima categoria visto che effettua delle operazioni di trasformazione sui contenuti richiesti in modo del tutto trasparente all'utente.

#### 1.2.2.2 Modello per la rappresentazione dei contenuti multimediali

Un altro elemento importante per un sistema di adattamento è la ricerca e, nel caso di più risultati, la selezione dell'oggetto multimediale sul quale effettuare l'adattamento. La soluzione del problema richiede:

1. L'individuazione dei contenuti che rispondono alla richiesta fatta dall'utente.
2. Una loro analisi per individuare il contenuto multimediale sul quale è più conveniente intervenire.

Per poter lavorare sugli oggetti multimediali in modo efficace non è opportuno considerarli nella loro totalità in quanto ingombranti e difficili da gestire. Al contrario sarebbe molto comodo poter sintetizzare le informazioni rilevanti che cateterizzano ogni dato in una struttura comprensibile e semplice che lo rappresenti:

il *"metadato"*. Esso può descrivere diversi aspetti caratteristici della risorsa: da quelli bibliografici (autore, titolo, produttore, ecc. ), a quelli tecnici riguardanti la codifica video e il formato, a quelli spazio-temporali quali la durata, la dimensione e la locazione.

Lo sviluppo di un modello per la gestione dei contenuti multimediali e dei relativi metadati è molto utile al fine di separare la logica di gestione e recupero delle informazioni da quella di elaborazione dell'adattamento. La ricerca delle informazioni è sicuramente una attività di grosso impatto sulle prestazioni dell'intero sistema. Il servizio di gestione dei contenuti, quindi, deve garantire la massima efficienza nell'esecuzione dei propri compiti. A questo proposito, si potrebbe sfruttare lo scenario distribuito per introdurre sistemi di caching dei contenuti che permetterebbero di migliorare la qualità di servizio percepita dall'utente influenzando parametri di qualità oggettivi come:

- Il ***tempo di trasmissione*** dei contenuti, che diminuisce notevolmente in quanto la memorizzazione di loro copie in vari nodi della rete permette soprattutto ai documenti più popolari, di compiere un percorso più breve per il raggiungimento del client.
- L'utilizzo della ***banda di trasmissione*** che si riduce conseguentemente alla riduzione del tempo di trasmissione.

Tuttavia l'introduzione del caching viene pagata in termini di spazio di memoria utilizzato nei vari nodi. Questo problema è in generale considerato irrilevante, dati i benefici che si ottengo dal miglioramento dei due parametri di qualità precedentemente discussi e il basso rapporto costo/capienza degli attuali dispositivi di memorizzazione.

## 1.3 Adattamento: aspetti caratteristici

Le informazioni rappresentate nei metadati insieme a quelle contenute nel profilo utente vengono elaborate dal sistema di adattamento al fine di garantire la distribuzione del contenuto multimediale richiesto più adeguato.

Il processo di adattamento può essere visto come il risultato di due fasi di elaborazione distinte:

1. ***Fase di configurazione***, in cui si caratterizza il processo prendendo decisioni importanti riguardo:
   - Quale presentazione adattare;
   - Quali mezzi adottare per effettuare il processo di trasformazione;
   - Chi dovrà essere il responsabile dell'adattamento.
2. ***Fase di traformazione***, in cui si procede con l'elaborazione del contenuto e la sua trasmissione verso il cliente che ne ha fatto richiesta.

La prima fase è sicuramente la più interessante poiché incapsula la "intelligenza" per poter effettuare le scelte che poi verranno applicate all'interno di un modulo di codifica appositamente configurato. La Figura **1.5** mostra attraverso uno schema concettuale come le attività di codifica vengano separate da quelle di decisione, evidenziando la forte modularità di questa struttura. Uno dei vantaggi più rilevanti di un approccio di questo genere è la possibilità eseguire la fase di configurazione dell'adattamento e le fase di trasformazione su nodi diversi del sistema, distribuendo il carico computazionale dell'intero processo di adattamento.

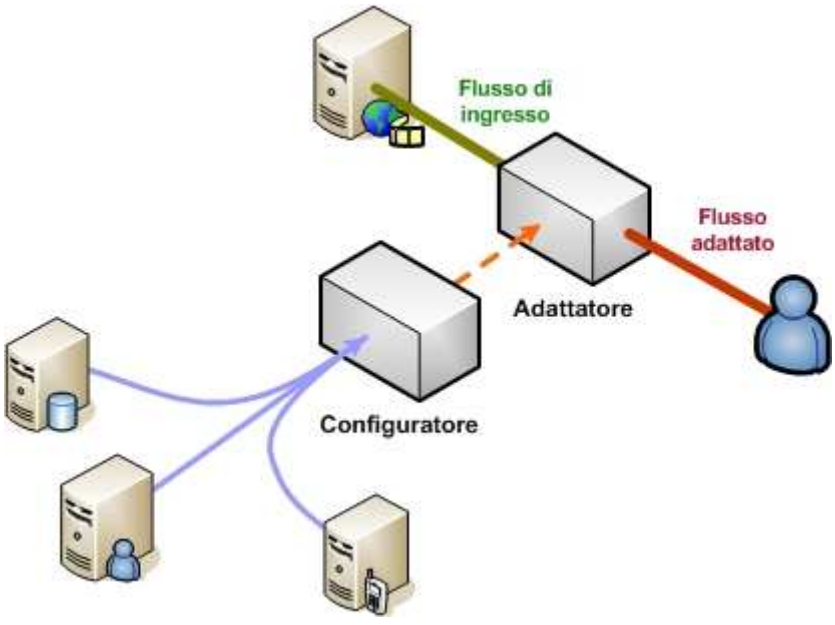



**Figura 1.5:** Struttura modulare del processo di adattamento.

La modularità dei servizi middleware, in generale, è una caratteristica molto apprezzata dagli sviluppatori di applicativi poiché permette loro di distribuire le

responsabilità del servizio sulle entità in gioco che ritengono più opportune, rendendole meno vincolate dai comportamenti derivanti dalle scelte di progetto fatte in fase di sviluppo del middleware.

### 1.3.1 Aspetti di configurazione del Servizio

Come anticipato precedentemente, le decisioni prese nella fase di configurazione scaturiscono dall'elaborazione di informazioni di due tipi: quelle riguardanti l'ambiente in cui verrà consumato il servizio richiesto dall'utente e quelle che descrivono i contenuti messi a disposizione del Server. Le scelte provenienti da questa elaborazione possono seguire due direzioni:

1. La minimizzazione del tempi necessari per la trasformazione del contenuto nel formato personalizzato per l'utente.
2. La minimizzazione del tempo necessario per il trasferimento del contenuto multimediale dal server al client.

Il problema che si sta considerando è l'individuazione dei criteri da considerare per scegliere l'oggetto multimediale su cui operare l'adattamento (se necessario). Questi criteri ci permetteranno ad esempio di decidere se scegliere di operare su una presentazione del contenuto lontana dal cliente, ma che non ha bisogno di essere adattata perché esattamente corrispondente alle sue caratteristiche, o viceversa su una presentazione molto vicina alla posizione del cliente, ma da sottoporre al processo di adattamento. Le due linee guida indicate sono entrambe interessanti, tuttavia sono fortemente dipendenti dall'ambiente su cui il sistema opera: in una rete locale ad esempio, probabilmente la località delle informazioni è meno rilevante che in una rete geografica aperta come Internet. Dato che si vuole studiare un servizio che si collochi a livello middleware, bisognerà offrire agli sviluppatori il modo di poter esprimere i criteri che guideranno il sistema di adattamento nelle sue scelte.

### 1.3.2 Aspetti organizzativi dell'attività di trasformazione

Un altro aspetto molto importante è la definizione degli strumenti di adattamento da utilizzare per sintetizzare un flusso di informazione con certe qualità partendo da un altro. L'organizzazione dell'elaborazione di un flusso multimediale può seguire anche in questo caso varie direzioni:

1. Costruire tutto intorno a un unico componente adattatore molto potente in grado di compiere un qualsiasi tipo di trasformazione. Tuttavia la complessità di un componente di questo tipo in termini di progettazione e realizzazione è molto elevata. Inoltre la fragilità e l'inefficienza di un modello altamente centralizzato come questo, in cui esiste un'unica entità estremamente importante per il funzionamento dell'intero sistema, spingono a considerare soluzioni più distribuite capaci di sfruttare le potenzialità di molteplici entità cooperanti.
2. Scomporre l'attività di trasformazione del contenuto in un certo numero di sottoattività elementari svolte da componenti in grado di risolvere specifici problemi: come il downscaling della risoluzione video, la scolorazione delle immagini, eccetera. In un modello di questo tipo è molto importante individuare un'entità che si occupi del coordinamento delle operazioni da svolgere sul flusso, che potrebbe essere la stessa che si occupa della configurazione del processo di trasformazione. La Figura **1.6** schematizza il concetto di distribuzione appena espresso a parole.

   Anche questo secondo modello, tuttavia, comporta l'introduzione di complessità dovuta alla realizzazione della logica in grado di distribuire il processo di codifica del flusso sui vari componenti, in compenso però porta anche tutti i vantaggi che un modello distribuito può offrire rispetto ad uno centralizzato.

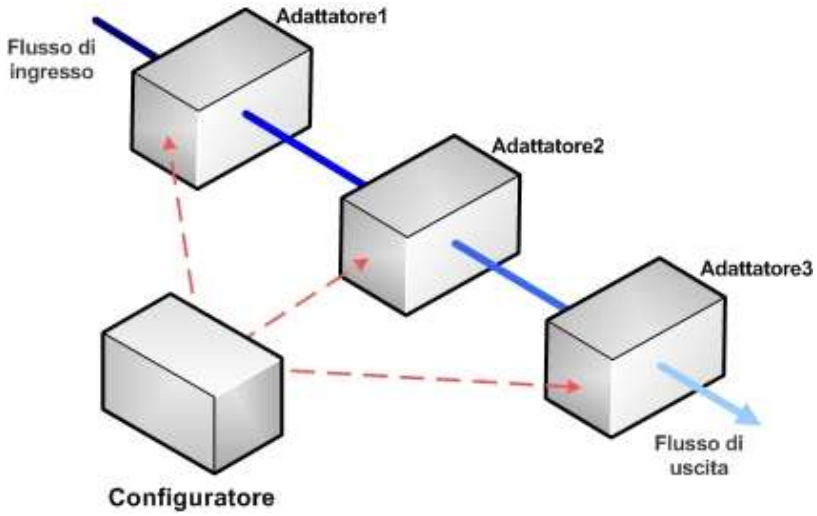



**Figura 1.6:** Struttura di un adattatore distribuito

## 1.4 Conclusione

In questo capitolo è stato presentato lo scenario preso in considerazione in questo lavoro di tesi, cercando di focalizzare l'attenzione sulla grande eterogeneità di contenuti, utenti e mezzi che popolano l'attuale panorama della distribuzione di contenuti multimediali. Sono stati individuati i servizi e gli strumenti di base utili alla realizzazione di un supporto all'integrazione e alla comunicazione di queste entità, basato sull'adattamento dei contenuti alle abilità e alle risorse degli utenti che ne fanno richiesta.

Nei prossimi capitoli questi temi verranno discussi più in dettaglio, al fine di raggiungere l'acquisizione del background di conoscenze necessario per la progettazione e lo sviluppo di un servizio di livello middleware in grado di fornire utili strumenti di supporto agli sviluppatori che trovano nell'adattamento un servizio necessario per la realizzazione di servizi multimediali.

*Capitolo 2*

# ADATTAMENTO DI CONTENUTI MULTIMEDIALI E STATO DELL'ARTE

Negli ultimi anni la comunità scientifica ha dimostrato molto interesse verso le problematiche legate all'adattamento di contenuti multimediali. La causa di questo crescente interesse è da ricercarsi nell'esigenza di costruire dei servizi in grado di gestire la forte eterogeneità di utenti e dispositivi che popolano l'attuale scenario dei sistemi distribuiti e nella forte diffusione di dati multimediali a cui abbiamo assistito negli ultimi anni, a causa soprattutto della nascita di dispositivi elettronici dalle capacità computazionali sempre più performanti.

Dopo aver presentato nel precedente capitolo i principi e le parte essenziali di un sistema di adattamento dei contenuti multimediali: il profilo dell'utente e il modello di descrizione dei dati multimediali, verrà discusso in questo capitolo come recenti lavori di ricerca utilizzino queste informazioni al fine di realizzare servizi adattabili alle preferenze di un utente e/o ai limiti fisici imposti dai dispositivi utilizzati.

Il capitolo è strutturato come segue: nella prima parte si focalizzerà l'attenzione sulle strategie di adattamento realizzabili presentandone gli aspetti caratterizzanti e una loro possibile classificazione basata su di essi; la seconda parte prenderà in considerazione aspetti di tipo architetturale che possono differenziare in modo determinante l'efficienza e l'efficacia delle tecniche di adattamento presentate; infine l'ultima parte sarà dedicata all'illustrazione di alcuni dei più rappresentativi lavori di ricerca, attualmente ancora in fase di sviluppo.

## 2.1 Strategie di Adattamento

In questo paragrafo verrà presentata una panoramica delle tecniche di adattamento correntemente utilizzate e discusse nei diversi lavori di ricerca che si occupano della

tematica dell'Adattamento basato sul profilo utente. Tra tutte le soluzioni presentate nei paragrafi successivi non ne esiste una migliore in assoluto, ognuna infatti presenta vantaggi più o meno importanti a seconda dello scenario applicativo nel quale la si usa.

In accordo a [Adapt], molteplici sono gli aspetti che possono essere presi in considerazione per classificare le tecniche di adattamento:

- Orientamento dell'adattamento;
- Il momento in cui l'adattamento viene effettuato;
- I tipi di media coinvolti nel processo di adattamento;
- Il livello di astrazione dell'adattamento.

Nei prossimi paragrafi, ognuno di questi aspetti verrà analizzato per comprendere le caratteristiche di comportamento che caratterizzano ogni strategia di adattamento.

### 2.1.1 Orientamento dell'Adattamento

L'adattamento in generale può essere orientato verso qualunque tipo di informazione caratterizzante il profilo dell'utente. Come discusso in precedenza le dimensioni descritte in un profilo sono quella delle preferenze dell'utente e quella delle infrastrutture tecniche utilizzate per accedere al servizio. Le strategie di adattamento quindi possono essere classificate in accordo alla dimensione scelta come "target" del processo di adattamento in:

- ***Strategie orientate alle preferenze dell'utente:***

  Gli utilizzatori dei servizi potrebbero avere caratteristiche molto diverse, appartenere ad aree geografiche diverse, a culture diverse o ricoprire ruoli diversi nella società. Ognuno dunque potrebbe essere interessato a un aspetto di una risorsa piuttosto che ad un altro. Ha senso quindi, in alcuni casi, adattare le risorse, sia nella presentazione che nel contenuto, rispetto alle preferenze dell'utente creando delle viste che focalizzano aspetti diversi della stessa risorsa.

- ***Strategie orientate al dispositivo utilizzato dal cliente:***

  I dispositivi utilizzati per accedere al servizio possono avere differenti caratteristiche tecniche in termini ad esempio di dimensione dello schermo,

profondità del colore, potenza di calcolo, sistema operativo, software utilizzato e così via. Anche le tipologie di connessione utilizzabili per la ricezione dei contenuti possono essere molto diverse: i terminali possono comunicare attraverso reti fisse o mobili. L'adattamento quindi può essere orientato alla creazione di contenuti adeguati a questo tipo di caratteristiche.

### 2.1.2 Quando effettuare l'adattamento

Una strategia di adattamento può essere classificata in accordo al momento in cui esso viene effettuato. Possiamo distinguere dunque le seguenti due tecniche di adattamento:

- ***Adattamento statico:***

  In fase di inizializzazione del sistema le informazioni multimediali vengono pre-elaborate e per ogni oggetto vengono create un certo numero di versioni caratterizzate da diversi livelli di qualità. A run-time, quando viene analizzata una richiesta, considerando il profilo dell'utente viene *"selezionata"* la versione ritenuta più opportuna e consegnata all'utente.

  Con questo tipo di approccio il provider dei contenuti ha un controllo forte su come questi vengono presentati sul cliente, tuttavia la gestione e il mantenimento sul server di un elevato numero di versioni, necessario per garantire la copertura di tutti i possibili contesti utente, risulta considerevolmente complesso e costoso.

- ***Adattamento dinamico:***

  Nell'approccio dinamico il contenuto viene elaborato durante la sua presentazione, si dice che la sua elaborazione viene svolta *"on the fly"*. Non esistono versioni pre-adattate, quindi l'azione fondamentale del processo di adattamento non è la *"selezione"* del contenuto, come nell'approccio statico, ma la sua *"trasformazione"*.

  L'approccio dinamico è in grado di trattare anche parametri del profilo con caratteristiche temporali real-time come l'ampiezza di banda della connessione di rete, molto sensibile a sollecitazioni esterne non prevedibili come il traffico. L'adattamento dinamico, in tempo reale, tiene conto delle

eventuali variazioni del profilo utente permettendo il supporto alla mobilità del terminale usato per accedere al servizio. Inoltre, non presenta problemi derivanti dalla gestione di versioni sul server, ma il processo di trasformazione che lo caratterizza potrebbe introdurre tempi di attesa troppo lunghi e compromettere la qualità di servizio percepita dall'utente.

### 2.1.3 Tipi di media coinvolti nel processo di Adattamento

Le trasformazioni e/o sostituzioni possono coinvolgere uno o più tipi di media. In accordo a questo criterio possiamo classificare le strategie di adattamento in due categorie:

- ***Adattamento orientato verso un singolo tipo di media:***

  In questa categoria possono essere posizionati la gran parte dei sistemi di adattamento attuali che si occupa di selezionare o trasformare contenuti dello stesso tipo. Il loro scopo principale è quello di comprimere i contenuti senza causare un imprecisato e intollerabile decadimento del loro livello di qualità. Tuttavia, non sempre questo obiettivo è perseguibile: se si cerca, ad esempio, di effettuare un forte ridimensionamento sul frame-size di un video è possibile che alcuni suoi dettagli, anche considerevolmente importanti dal punto di vista semantico, non siano più riconoscibili.

- ***Adattamento cross-media o multi-modale:***

  Un altro approccio possibile è considerare l'adattamento un processo capace di trasformare in contenuti da un tipo di media ad un altro, permettendo la loro fruizione anche da parte di dispositivi estremamente limitati dal punto di vista delle risorse e non in grado di riprodurre contenuti molto ricchi come i flussi video. In questi casi, potrebbe essere utile trasformare il flusso video in una sequenza di fotogrammi o in qualunque altro tipo di presentazione il terminale sia capace di gestire, in modo che anche l'utente più limitato dal punto di vista delle risorse di accesso non sia completamente tagliato fuori dalla sua ricezione.

### 2.1.4 Il livello di astrazione dell'adattamento

In generale con il termine "livello di astrazione" si indica i livello di dettaglio che si vuol prendere in considerazione nello svolgimento di un qualche processo di elaborazione. Si può pensare di applicare questo concetto anche al processo di adattamento e quindi distinguere:

- ***Adattamento semantico:***

  L'adattamento semantico è un processo selettivo, basato sulle informazioni richieste e quelle disponibili. Una presentazione viene vista come la combinazione di più contenuti informativi, più o meno importanti, che caratterizzano la sua struttura semantica. Ad esempio, la maggior parte dei siti web per il commercio elettronico contiene, oltre ai contenuti principali, contenuti di contorno come banners, loghi e pubblicità che consumano una buona parte della banda disponibile e sono ridondanti o di nessun interesse per l'utente. Se si riuscisse a definire una struttura semantica della pagina web, quando essa viene richiesta da terminali con scarse capacità fisiche si potrebbe creare una sua nuova versione rimuovendo i contenuti ridondanti e migliorando così l'efficienza della consegna delle informazioni.

- ***Adattamento fisico:***

  Il livello di astrazione dell'adattamento è più basso rispetto all'approccio presentato sopra. L'adattamento fisico può essere visto come combinazione di processi di conversione, compressione e riduzione del flusso multimediale, guidati dalle caratteristiche del suo formato e dal livello fisico di qualità di servizio che si vuol presentare.

## 2.2 Aspetti architetturali

Un'importante scelta di progetto è decidere dove l'adattamento deve essere svolto. A questo proposito le possibilità sono:

- Sul server;
- Su un intermediario, proxy;
- Sul client;

In questo paragrafo vengono discussi i pregi e di difetti di ognuna di queste soluzioni.

### 2.2.1 Adattamento lato server

Come mostra la Figura **2.1**, il server, fornitore dei contenuti, ha anche la responsabilità di analizzare il profilo utente per sintetizzare la strategia di adattamento più appropriata.

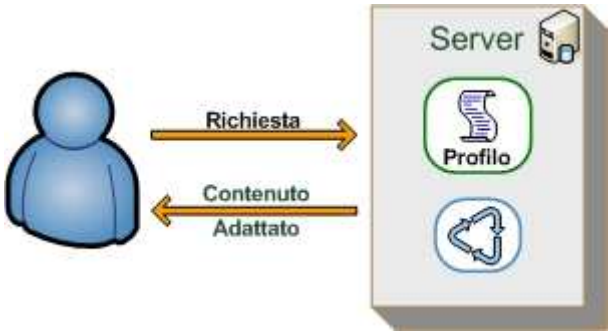



**Figura 2.1:** Architettura adattamento lato Server

Questo approccio ha il vantaggio di potersi avvalere dell'ampia visione delle informazioni sui contenuti presente lato server, quindi essere più efficace.

Tuttavia questo approccio architetturale presenta dei difetti:

- Complica l'implementazione del Server e degli algoritmi per la generazione di una presentazione del contenuto appropriata alle richieste.
- Aumenta il carico computazionale e il consumo di risorse di elaborazione e di memorizzazione sul server.

### 2.2.2 Adattamento lato proxy

Il Client è connesso al Server tramite un intermediario detto Proxy che intercetta le richieste del terminale utente, recupera dal Server il contenuto richiesto e lo adatta secondo le strategie e politiche che esso stesso decide di usare in base al conteso dell'utente. In Figura **2.2** si mostra l'architettura di questo approccio.

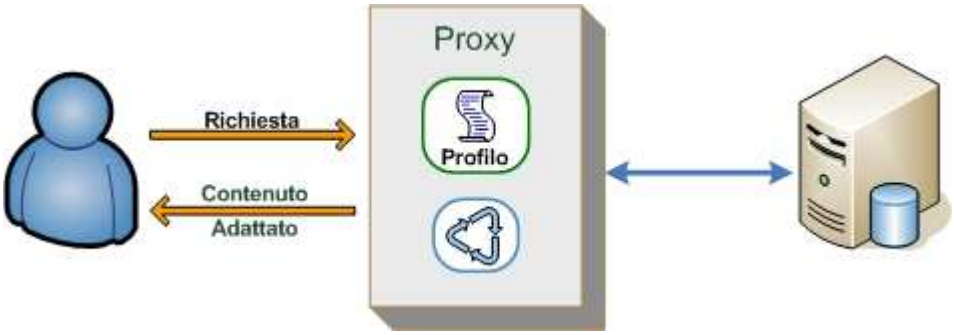



**Figura 2.2:** Architettura adattamento lato Proxy

Il notevole vantaggio di tale approccio è quello di essere totalmente trasparente sia nei confronti del client che del server che non devono subire nessun tipo di modifica. Tuttavia, il carico computazionale è completato concentrato sul proxy che, essendo un entità autonoma dal server, potrebbe introdurre ritardi imprecisabili ed incontrollabili sulla consegna dei contenuti. Per lo stesso motivo inoltre, per il server è molto difficile controllare il modo in cui i propri contenuti vengono presentati ai clienti.

### 2.2.3 Adattamento lato client

Come mostra la Figura **2.3**, il processo di adattamento viene effettuato dal terminale utente che riceve dal Server una risposta che include una lista di rappresentazioni del contenuto richiesto, dalla quale il terminale seleziona quella più appropriata alle proprie caratteristiche.

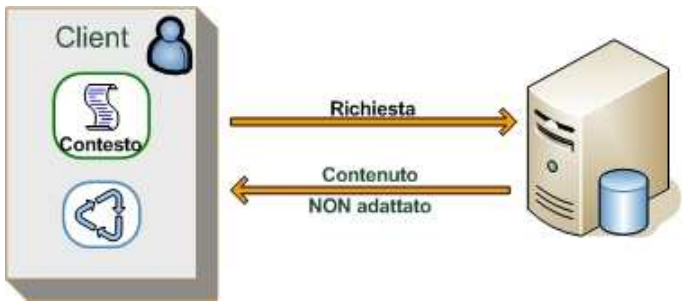



**Figura 2.3:** Architettura adattamento lato Client

Questo tipo di approccio è notevolmente vantaggioso quando il server non possiede le capacità di riconoscere ed estrarre il contesto utente dall'esame della richiesta e soprattutto quando si opera in sistemi che sfruttano un servizio di caching

distribuito dei contenuti per ridurre il carico del server e il costo di trasmissione delle informazioni. Tuttavia è chiaramente inutilizzabile se sono coinvolti terminali dalle capacità computazionali limitate, quindi non in grado di effettuare l'elaborazioni dei contenuti.

## 2.3 Sistemi esistenti per l'adattamento di dati multimediali

In questo paragrafo verranno illustrati alcuni dei framework per l'adattamento più significativi sviluppati negli ultimi anni dalla comunità scientifica, allo scopo di mostrare come i concetti spiegati precedentemente siano stati applicati e di verificare le problematiche sollevate dal loro utilizzo.

### 2.3.1 Un framework per l'adattamento statico e multi-modale

In questo paragrafo viene proposto un framework per l'adattamento costruito su *"InfoPyramid"*, sistema, realizzato dal Watson Research Center della IBM, che fornisce servizi per la gestione di contenuti, per la loro manipolazione, traduzione e codifica [InfoPyramid].

Il framework si basa su due tecnologie chiave:

1. ***InfoPyramid***, usato come strumento per la rappresentazione dei contenuti, in cui vengono create diverse versioni dei contenuti multimediali.
2. Un ***modulo di personalizzazione dei contenuti***, che ha la capacità di selezionare fra tutte le versioni presenti in *"InfoPyramid"* quella che riesce a soddisfare al meglio i vincoli imposti dalle risorse a disposizione del terminale cliente.

Una versione è caratterizzata non solo da un livello di qualità, espresso da una serie di parametri, ma anche da una modalità di presentazione. Ad esempio da un oggetto video possono essere codificate un certo numero di versioni di tipo video a diversi livelli di qualità e altre versioni di altro tipo, ad esempio una sequenza di fotogrammi o un testo. Questo tipo di adattamento chiamato *"multi-modale"*, come mostra la Figura **2.4**, permette di distribuire contenuti anche molto carichi dal punto di vista multimediale su piattaforme con disponibilità di risorse molto limitate.

| | Workstation/LAN | PC/Dialup | TV Browser | Gray PDA | BW PDA | Text Browser |
|---|---|---|---|---|---|---|
| Image |  |  |  |  |  | "bridge" |
| Color | 24 bit color | 24 bit color | 256 colors | 4 bit gray | B/W | |
| Size | 256x256 | 192x192 | 128x128 | 96x96 | 64x64 | |
| Bits | 34KB | 23KB | 8KB | 4KB | 0.6KB | 0.01KB |

**Figura 2.4:** Esempio di adattamento multi-modale.

Il framework di adattamento viene proposto come una estensione del server Web, quindi viene adottato un approccio completamente server-based. Le motivazioni di questa scelta derivano dall'analisi dei problemi che derivano dall'approccio proxy-based ampiamente sposato dalla maggior parte delle comunità di ricerca. Secondo gli autori del progetto infatti un approccio di tipo server-based garantisce un maggior controllo dei provider sul come vengono presentati i contenuti ai loro clienti, fattore di estrema importanza in ambienti in cui ci possono essere problemi legali sollevati dal copyright che possono precludere o severamente limitare la codifica delle informazioni lato proxy. Inoltre dal punto di vista tecnico, il beneficio maggiore che si può ottenere da sistemi di questo tipo è dovuto al fatto che il server ha una visione globale delle informazioni riguardanti i contenuti sicuramente migliore di un proxy e quindi sfruttando tale conoscenza è in grado di effettuare una personalizzazione del contenuto richiesto più efficace.

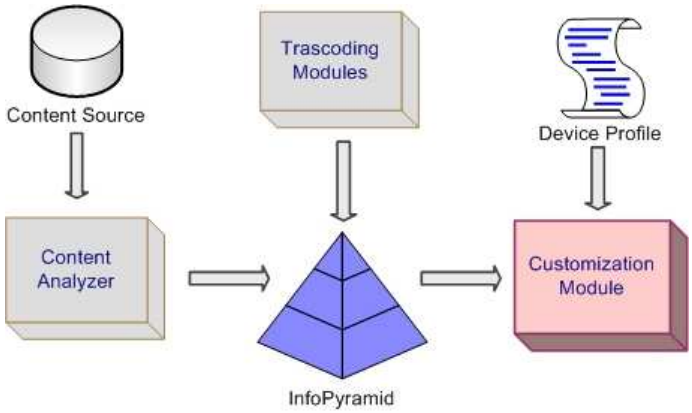



**Figura 2.5:** Architettura basata su InfoPyramid e il Customization Module

Nella Figura **2.5** viene mostrata l'architettura del framework proposto. Il *"Content Source"* contiene i dati multimediali che possono essere consegnati dal server. Ogni contenuto multimediale viene analizzato per poter estrarre da esso una sua descrizione che guiderà le fasi successive di selezione e personalizzazione.

Per ogni possibile profilo del terminale cliente, differenti moduli di codifica generano le versioni del contenuto con vari livelli di qualità e vari modelli di presentazione. Queste versioni vengono memorizzate in *"InfoPyramid"* off-line.

Quando il sistema riceve una richiesta, vengono innanzitutto recuperate le informazioni relative al terminale utente e il *"modulo di Personalizzazione"* dinamicamente seleziona fra le versioni contenute in *"InfoPyramid"* quella più idonea che viene dunque consegnata al cliente.

Il framework recupera le informazioni relative ai contenuti multimediali attraverso i servizi offerti da *"InfoPyramid"* che gestisce metadati relativi agli oggetti multimediali memorizzati, descritti attraverso lo standard MPEG-7.

Le informazioni relative al contesto utente vengono recuperate utilizzando vari meccanismi: il campo *"user-agent"* dell'header di una richiesta HTTP o attraverso profili creati in fase di registrazione e recuperati dopo aver identificato l'utente.

L'unità di personalizzazione del contenuto modella il problema di selezione come un problema di massimizzazione del punteggio, associato a ogni versione dell'oggetto multimediale da InfoPyramid, in modo che la dimensione fisica del contenuto selezionato sia inferiore a quella accettabile dal utente. Un'altra strategia testata modella il problema di selezione come uno di minimizzazione del punteggio con il vincolo che esso sia superiore a una certa soglia fissata. La prima strategia è volta a selezionare una versione qualitativamente più vicina al contenuto originale, la seconda invece a minimizzare il carico dovuto dell'adattamento.

Il framework presentato in questo paragrafo propone un approccio statico all'adattamento dei contenuti multimediali, questo da una parte permette un forte controllo da parte del Server di come i contenuti vengono presentati sui clienti, dall'altra introduce grossi problemi dovuti alla gestione e alla memorizzazione di un

elevato numero di varianti dei contenuti multimediali. Un altro limite forte di questo approccio è quello di non essere in grado di adattare i contenuti rispetto a parametri real-time come l'ampiezza di banda messa a disposizione dalla rete. Inoltre considerando statiche le caratteristiche fisiche del terminale preclude il suo supporto alla mobilità degli utenti.

### 2.3.2 Un framework per l'adattamento ibrido di dati multimediali

Il framework presentato in questo paragrafo, proposto e sviluppato nell'università di Hong Kong, si propone di affrontare la tematica dell'adattamento cercando di trovare un buon compromesso tra overhead del processo di codifica del flusso di dati e consumo di spazio per la memorizzazione di diverse "versioni" di una certa presentazione multimediale [Lau02].

Con il termine "versione" si intende una particolare istanza di una presentazione caratterizzata da un insieme attributi come la profondità del colore, il formato o il fattore di scala che ne determinano il livello di qualità.

L'idea di fondo è dunque di dotare il Server di un limitato numero di versioni pre-adattate di una certa presentazione da usare come punto di partenza per costruire dinamicamente quella più consona al contesto dell'utente. Il framework si propone di scegliere le caratteristiche delle versioni da pre-adattare e mantenere sul Server in modo da garantire la totale copertura della possibili richieste future.

Dal punto di vista architetturale il framework è composto da due moduli, il primo detto **Content Negotiaton Module** ha il compito di analizzare le informazioni di contesto dell'utente e configurare i parametri di adattamento che guideranno il secondo modulo nella costruzione della versione adattata desiderata, il secondo, il **Content Realization Module,** ha l'onere di applicare le decisioni prese dal primo modulo per costruire la versione più adeguata alla caratteristiche dell'utente.

La negoziazione viene svolta in due fasi:

1. Il ***Preprocessing*** viene svolto in fase di inizializzazione del sistema, prima dell'arrivo delle richieste. In questa fase il sistema crea una struttura di nodi detti "score nodes". Ognuno di essi rappresenta una delle possibili versioni della presentazione che il sistema di adattamento è in grado di creare e specifica i parametri di adattamento necessari per la sua generazione.

Quando un utente si registra, fra le altre informazioni, fornisce le sue preferenze indicando le caratteristiche di qualità verso le quali è più sensibile, permettendo di associare staticamente ad ogni "score node" un punteggio che indica la bontà della versione a cui si riferisce rispetto alle preferenze espresse dall'utente.

2. Il **Real-time Processing** viene svolto ad ogni richiesta dell'utente. Tiene in considerazione dei parametri real-time come quelli relativi alla connessione di rete utilizzata, alle caratteristiche del terminale utilizzato per accedere al servizio e all'oggetto richiesto. Questi parametri non potevano essere presi in considerazione dal punteggio assegnato nella fase precedente proprio per la loro natura dinamica e dipendente dalla richiesta. In questa fase l'algoritmo di negoziazione, presa conoscenza dei parametri real-time ha l'obiettivo di trovare lo "score node" con punteggio massimo in grado di soddisfare i vincoli da essi espressi.

Come illustra la Figura **2.6**, tratta da [Lum02], lo scopo della fase di negoziazione è quello di individuare le caratteristiche della versione più adatta alla preferenze dell'utente e alle caratteristiche del dispositivo attraverso il quale il servizio è stato richiesto e contemporaneamente individuare le linee guida da proporre al *"Content Realization Module"* che si occuperà della sua generazione.

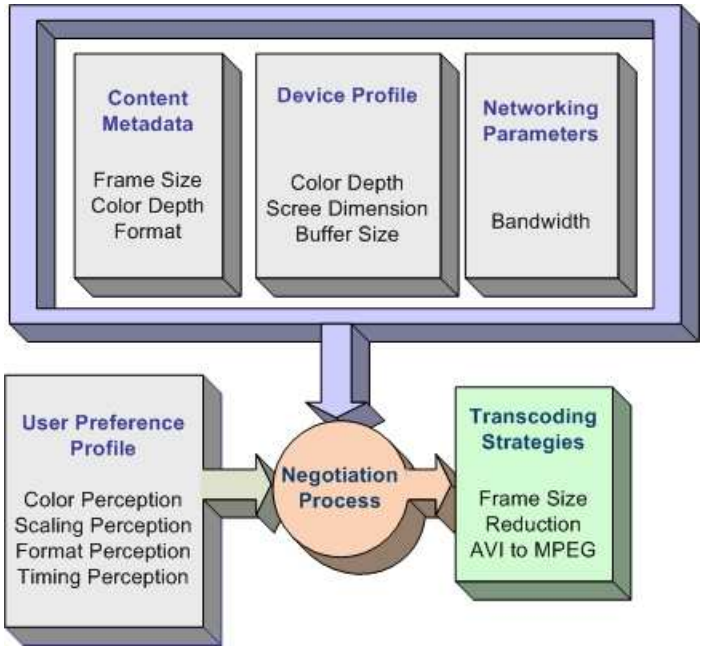



**Figura 2.6:** Schema dell'elaborazione compiuta in fase di negoziazione.

Il modulo di realizzazione del contenuto, attraverso una logica basata sulla definizione di algoritmi di similarità, seleziona la versione pre-adattata presente sul Server più vicina a quella richiesta dal negoziatore in modo da minimizzare il costo del processo di codifica.

Il processo di codifica viene visto come un processo multidimensionale, in cui ogni dimensione rappresenta una possibile direzione sulla quale indirizzare l'adattamento, ad esempio una dimensione potrebbe essere quella dei terminali usati per l'accesso al servizio, per la quale il numero di versioni di un contenuto è proporzionale al numero di terminali utilizzabili.

Una caratteristica interessante del framework è quella di considerare la natura combinatoria del processo di codifica fornendo un certo numero di "transcoder" capaci di effettuare delle operazioni di trasformazione elementari: riduzione della profondità del colore, trasformazione del formato, downscaling della dimensione e così via.

Gli adattatori (o transcoders) dichiarano le loro capacità usando una grammatica BNF (Backus Naur Form) chiamata *"Capability Advertisement"* (CA) che il modulo utilizza per interpretare chi è in grado di svolgere una certa operazione di trasformazione.

Infine per rappresentare le relazioni tra diverse versioni di uno stesso oggetto e le operazioni necessarie per passare da l'uno all'altro viene usato un grafo detto *"Transcoding Relation Graph"* (TRG) in cui ad ogni versione del contenuto associa un vertice e a ogni relazione di trasformazione un arco. La figura **2.7** mostra un possibile TRG relativo a un certo oggetto multimediale.

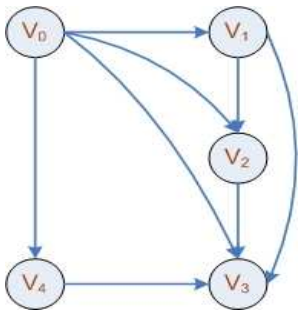



**Figura 2.7:** Il TRG rappresenta le relazioni tre le diverse versioni di un oggetto.

L'approccio presentato in questo paragrafo è dinamico nel senso che la versione del contenuto idonea alle caratteristiche del contesto utente viene creata on-line, ma allo stesso tempo presenta caratteristiche tipiche dell'approccio statico mantenendo sul server un insieme di versioni pre-adattate al fine di ridurre il costo di codifica del processo di adattamento a scapito del costo di memorizzazione sul Server. Tuttavia la gestione di un sistema in cui il numero delle versioni possibili è proporzionale ad esempio al numero di dispositivi utilizzabili per l'accesso al servizio è molto complessa, si pensi ad esempio all'introduzione di un nuovo terminale che richiederebbe l'aggiornamento di tutte le versioni.

### 2.3.3 NAC: Framework flessibile per l'adattamento dei dati

NAC (Negotation and Adaptation Core) è un framework sviluppato all'interno del progetto OPERA che si propone di affrontare la tematica dell'adattamento considerando la flessibilità come uno degli obiettivi primari del progetto, in modo che il framework sia in grado di fornire i suoi servizi in ambienti eterogenei anche pesantemente dinamici e sia facilmente estendibile e aperto verso modelli e strategie di adattamento non ancora studiate [NAC].

L'architettura di NAC è composta principalmente da tre componenti:

1. Il server dei contenuti;
2. L' *"Adaptation Negotiation Module"* (ANM) che può essere in esecuzione su un server o su un proxy;
3. Un modulo lato client detto *"User Context Module"*(UCM).

Il modulo UCM ha il ruolo di raccogliere le informazioni che caratterizzano il contesto utente, in particolare caratteristiche tecniche del terminale di accesso e preferenze dell'utente e inviare queste informazioni al modulo che si occupa dell'adattamento. Le informazioni raccolte vengono utilizzate per creare profili CC/PP, standard scelto per la descrizione del contesto dell'utente.

L'UCM può lavorare in due modalità:

1. ***Statica:*** Nelle situazioni in cui il profilo del cliente non cambia frequentemente esso può essere inviato una sola volta, in fase di registrazione;

2. ***Dinamica:*** Nei casi invece in cui il profilo è composte da proprietà frequentemente variabile, come quelle relative al mezzo di comunicazione, è necessario misurarle ed inviarle verso il modulo di personalizzazione ad ogni richiesta.

Il componente principale ai fini dell'adattamento è l'ANM che riceve le richieste dai clienti e risponde consegnando loro il contenuto richiesto in una forma negoziata con il Server sulla base del contesto utente, delle possibilità correnti del Server e alle condizioni di traffico della rete. Un'importante ruolo quindi viene giocato anche in questo caso dalla fase di negoziazione il cui risultato è la decisione delle tecniche di adattamento da adottare per creare la versione del contenuto da presentare al cliente.

L'insieme dei requisiti, che deve soddisfare la versione individuata nella fase di negoziazione, viene visto come insieme di vincoli che progressivamente vengono imposti al contenuto originale. Estratti i vincoli dal profilo del cliente, dal server e dal canale di comunicazione, il modulo ACM verifica se la risorsa richiesta è supportata dal cliente, in questo caso gli viene direttamente inviata, in caso contrario verifica l'esistenza di una versione alternativa della risorsa che soddisfi i vincoli imposti in grado dunque di sostituire quella richiesta. Se non esiste una versione compatibile, cerca di adattare quella originale. Per compiere questa operazione, viene confrontata la descrizione della versione richiesta con l'insieme degl'input richiesti dai metodi di adattamento disponibili per verificare l'esistenza di adattatori capaci di trattare la particolare versione richiesta. Se esistono, si verifica che l'output generato corrisponda a quello negoziato e in questo caso l'adattatore viene applicato sulla risorsa originale e il contenuto consegnato al cliente.

In modo molto flessibile, è possibile configurare NAC in modo da realizzare un adattamento dinamico o statico a seconda delle esigenze.

Dunque il framework presentato in questo paragrafo ha la qualità di essere molto configurabile alle varie esigenze applicative, senza dubbio però questa sua forte flessibilità viene pagata in termini di complessità in termini di implementazione e di interazione fra i vari componenti dell'architettura.

## 2.4 Conclusione

La numerosa documentazione sui lavori di ricerca e sviluppo che cercano di indirizzare il tema dell'adattamento in generale dimostra l'attuale interesse della comunità scientifica verso questo tema. Sono stati proposti molti approcci e modelli per la realizzazione di servizi di questo genere nell'ambito delle applicazioni multimediali, poiché sembra che la comunicazione digitale stia sempre più allargando le proprie prospettive verso questo tipo di contenuti.

In questo capitolo è stata fatta prima una classificazione degli approcci attualmente seguiti nell'affrontare la tematica della personalizzazione, poi sono stati presentati tre framework per l'adattamento di dati multimediali mostrando, per ognuno di essi, peculiarità, vantaggi e svantaggi con l'intento ti trarre da una loro analisi comparativa utili spunti per lo sviluppo di questo lavoro di tesi.

# *Capitolo 3*

# TECNOLOGIE E STANDARD PER IL SUPPORTO DELL'ETEROGENEITÀ

La creazione di servizi per il supporto della personalizzazione dei contenuti impone di trattare un argomento molto importante: la descrizione delle entità coinvolte nello scenario di comunicazione. Come detto nel capitolo introduttivo, da una parte è necessario conoscere in dettaglio le risorse che il Provider dei contenuti può offrire, dall'altra è importante sapere in che tipo di ambiente esse verranno "consumate" e le caratteristiche del mezzo trasmissivo attraverso il quale avverrà la loro fruizione. Queste informazioni vengono rappresentate all'interno di strutture descrittive che in letteratura vengono solitamente denominate, rispettivamente, "metadati" e "profili".

Nel primo capitolo di questa tesi si è discusso a proposito delle informazioni che queste entità dovrebbero contenere, giungendo alla conclusione che l'insieme di caratteristiche che esse dovrebbero descrivere può essere di vario genere, ma soprattutto può variare notevolmente da un campo applicativo ad un altro. In questo capitolo verranno esaminate gli standard utilizzabili per la loro rappresentazione, infatti, profili e metadati, per poter essere correttamente interpretati dal servizio di personalizzazione, devono essere rappresentati attraverso linguaggi standard capaci di assegnare ad ogni loro attributo un significato compreso e riconosciuto da chiunque. Un approccio possibile potrebbe essere l'utilizzo degli XML Schema, ma purtroppo attualmente non esiste ancora uno "schema" standard per la rappresentazione di profili e/o Metadati. Tuttavia, per soddisfare queste esigenze, sono stati definiti standard ad hoc che saranno l'oggetto di studio dei prossimi paragrafi.

## 3.1 Profilazione dell'utente

Un profilo, idealmente, dovrebbe includere tutte le informazioni che riguardano l'utente utili a migliorare il processo di personalizzazione dei contenuti. Deve dunque descrivere informazioni di varia natura e soprattutto gestite da diverse entità come l'operatore di rete e l'utente stesso. Inoltre, se si vuole costruire una framework per il supporto della profilazione in scenari abbastanza complessi che ammettono la possibilità dell'utente di spostarsi da un terminale ad un altro durante la ricezione del servizio, bisogna poter modellare gli attributi di un profilo in modo dinamico. Ad esempio, cambiamenti sulla banda supportata dal terminale causati dal passaggio dell'utente a un dispositivo diverso dovrebbero riflettersi sul valore dell'attributo che specifica la bit rate desiderata per lo streaming di un video.

### 3.1.1 Standard per la rappresentazione dei profili

In questa prima parte del paragrafo verranno presentati alcuni degli strumenti esistenti per la rappresentazione delle informazioni presenti in un profilo. Nella successiva si discuterà dell'opportunità di creare i profili dinamicamente come composizione di altri provenienti direttamente dalle entità coinvolte nello scenario di personalizzazione.

#### 3.1.1.1 RDF

RDF, Resource Description Framework, è un linguaggio progettato per rappresentare informazioni sulle risorse nel World Wide Web, quali ad esempio, titolo, autore, copyright, data di creazione e così via [RDF]. Tuttavia, generalizzando il concetto di risorsa web, RDF può essere usato anche per modellare altri generi di informazione comunicabili attraverso il web, nel caso specifico la descrizione delle preferenze di un utente o delle caratteristiche del dispositivo utilizzato per l'accesso al web.

RDF offre degli strumenti per rappresentare le informazioni in termini di "statement", ciascuno costituito da un soggetto, un predicato e un oggetto, fornendo una struttura comune per esprimere queste informazioni in modo che possano essere scambiate fra le applicazioni senza perdita di significato.

Per identificare le risorse, RDF utilizza gli URI (Universal Resource Identifier) che hanno la forma di un normale indirizzo web. Ogni risorsa, identificata da un URI e descritta in termini di coppie "proprietà/valore", può essere graficamente rappresentata da un nodo al quale possono esser associati uno o più "nodi valore" attraverso degli archi che rappresentano le proprietà della risorsa stessa. La Figura **3.1** mostra la rappresentazione RDF di una risorsa di esempio chiamata "device0" corrispondente alla seguente descrizione: "Il dispositivo device0 ha uno schermo di dimensione 320x240 pixel ed è capace di gestire una profondità di colore di 256 bit".

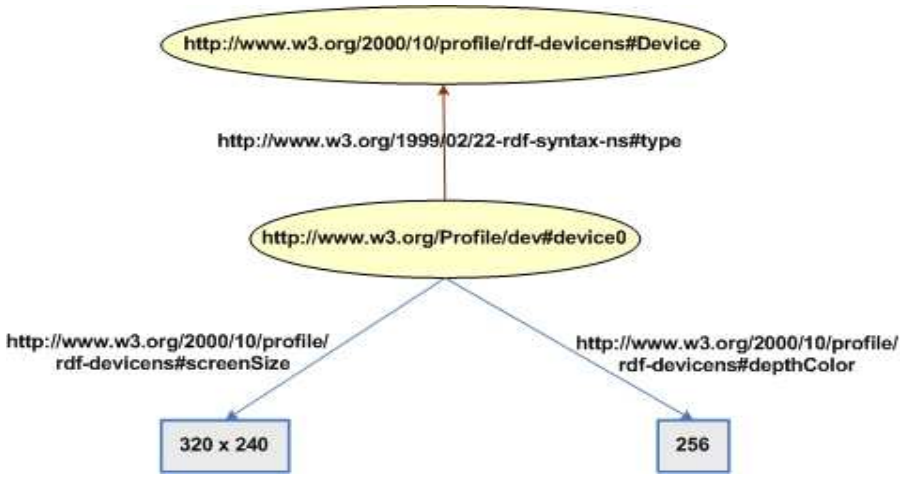



**Figura 3.1:** Esempio di grafo RDF di un oggetto.

Inoltre per facilitare lo scambio di questi grafi, RDF fornisce una loro rappresentazione XML denominata RDF/XML. Nella Figura **3.2**, viene mostrata la descrizione RDF/XML della risorsa rappresentata dal grafo mostrato in precedenza.

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntaxns#"
     xmlns:dev="http://www.w3.org/2000/10/profile/rdf-devicens#">

   <dev:Device rdf:about="http://www.w3.org/Profile/dev#device0>
      <dev:screeSize>320x240</dev:screeSize>
      <dev:depthColor>256</dev:depthColor>
   </dev:Device>
</rdf:RDF>
```

**Figura 3.2:** Esempio di rappresentazione XML/RDF di una risorsa

#### 3.1.1.2 CC/PP

CC/PP, Composite Capabilities/Preference Profile [CCPP], è una struttura basata su RDF per descrivere e manipolare profili contenenti informazioni riguardanti l'utente. In particolare CC/PP definisce la struttura generale di un profilo in modo completamente indipendente dalle informazioni che lo compongono, permettendo la loro definizione sintattica e semantica a entità esterne che adempiono questi compiti pubblicando un vocabolario RDF degli "statement" utilizzabili per la composizione dei profili.

La struttura definita da CC/PP prevede che un profilo sia articolato in una gerarchia di due livelli:

1. Ogni profilo è composto da un certo numero di componenti;
2. Ogni componente è a sua volta composto da uno o più attributi.

Questa strutturazione permette di rappresentare un profilo CC/PP come un albero, i cui rami iniziali solitamente rappresentano i suoi componenti chiave, ad esempio la piattaforma hardware, quella software, il canale di comunicazione o specifici componenti applicativi come il browser.

Ogni componente può essere descritto da altri sotto-componenti e/o da attributi CC/PP, quindi la sua rappresentazione è in generale un sotto-albero i cui rami sono le capacità o le preferenze connesse con quel componente.

CC/PP fornisce la possibilità di definire componenti di default, ovvero un gruppo di attributi identificabili in modo univoco che possono essere condivisi da più profili composti (formati appunto da componenti di default). La Figura **3.3** riassume la struttura di un profilo CC/PP.

Se il profilo definisce un attributo già considerato da un componente di default, il suo valore viene sovrascritto, dando la possibilità di personalizzare i componenti di default e di esprimere un profilo semplicemente puntualizzando le sue differenze da uno di riferimento.

Un attributo CC/PP può essere:

- ***Semplice*** (ad esempio interi, stringhe di caratteri)
- ***Complesso,*** ovvero di tipo:
    - ***Bag:*** Il valore dell'attributo è compreso all'interno di un insieme di valori specificati nel *Bag*.

- ***Seq:*** Il valore dell'attributo è definito da una sequenza ordinata di valori.

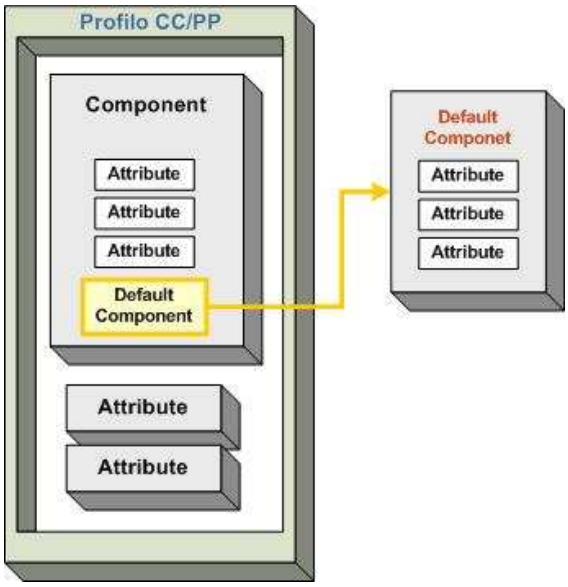



**Figura 3.3:** Struttura di un profilo composto CC/PP

L'aver definito solo la struttura generale del profilo, senza scendere nel particolare dei contenuti che lo compongono, rende CC/PP uno standard molto flessibile alle esigenze degli sviluppatori. Tuttavia, la facilità con cui possono essere creati nuovi vocabolari e modificati quelli esistenti ha causato la profilazione di un gran numero di versioni dei vocabolari, generando problemi di confusione fra le applicazioni che basano la loro interazione con l'ambiente esterno sullo standard CC/PP.

Comunque proprio le sue caratteristiche di flessibilità e di componibilità hanno reso CC/PP lo standard attualmente più diffuso per la rappresentazione dei profili utente.

### 3.1.1.3 UAProf

L'Open Mobile Alliance (in precedenza Forum WAP) ha definito un profilo, denominato UAProf (User Agent Profile) , che implementa le specifiche CC/PP per i terminali mobili con tecnologia WAP (Wireless Aplication Protocol), fornendo un punto di contatto tra le tecnologie mobili e quelle a rete fissa [UAProf]. Il forum WAP ha definito un vocabolario per la rappresentazione di profili che include

informazioni riguardanti cinque aree fondamentali per il riconoscimento del contesto di un utente WAP:

1. ***HardwarePlatform:*** Insieme di proprietà che descrivono in modo sufficientemente adeguato le caratteristiche della piattaforma hardware utilizzata dal utente per l'accesso ai servizi.
2. ***SoftwarePlatform:*** Collezione di attributi che fornisce informazioni a proposito del sistema operativo installato sul terminale, dei codificatori audio e video a sua disposizione e delle preferenze degli utenti sulla lingua e il set di caratteri utilizzabili.
3. ***BrowserUA:*** Insieme di attributi per descrivere il tipo di Browser installato sul terminale cliente.
4. ***NetworkCharacteristics:*** Informazioni relative al canale di comunicazione usato.
5. ***WapCharacteristcs:*** Insieme di attributi che descrivono i servizi WAP che il dispositivo è in grado di supportare.

Gli attributi di un profilo sono definiti utilizzando una sintassi che permette di esprimere a parole la loro semantica e di definire, oltre al loro tipo, l'insieme dei valori che possono assumere. In Figura **3.4** viene riportato uno stralcio dello schema RDF del vocabolario UAProf usato per la definizione di un suo attributo. Si può notare che per ogni attributo viene specificata una regola di risoluzione da adottare nel caso in cui esso sia definito più di una volta all'interno dello stesso profilo: il valore dell'attributo può essere sovrascritto solo se la regola lo permette.

```
<rdf:Description ID="CPU">
   <rdf:type rdf:resource="http://www.w3.org/TR/PR-rdf-schema#Property"/>
   <rdfs:domain rdf:resource="#HardwarePlatform"/>
   <rdfs:comment>
      Description:  Name and model number of the device CPU.
      Type:         Literal
      Resolution:   Locked
      Examples:     "Pentium III", "PowerPC 750"
   </rdfs:comment>
</rdf:Description>
```

**Figura 3.4**: Definizione RDF di un attributo di UAProf

Per il trasporto delle informazioni dell'UAProf, WAP 1.2.1 suggerisce l'utilizzo della rete Internet sfruttando le estensioni del protocollo HTTP (HTTPex),

originariamente proposto per il trasferimento dei profili CC/PP, ma a causa della mancanza di un'implementazione di questo protocollo, in WAP 2.0 viene proposta come alternativa l'estensione dell'HTTP 1.1 come Internet Protocol (W-HTTP) che usa le intestazioni abituali.

### 3.1.2 Approcci per una completa rappresentazione di un profilo

UAProf fornisce una descrizione abbastanza esaustiva delle proprietà di un dispositivo, tuttavia non riesce a fornire una descrizione completa delle caratteristiche utente necessaria per ottenere una buona personalizzazione dei servizi. Per raggiungere questo obiettivo dovrebbe descrivere altre informazioni quali interessi dell'utente, preferenze sui contenuti e sul modo in cui presentarli, informazioni sullo stato della rete di comunicazione, etc.
Per risolvere questo problema le strade percorribili sono fondamentalmente due:

1. Definire estensioni del vocabolario.
2. Integrare profili forniti dalle diverse entità che gestiscono le diverse informazioni di contesto.

#### 3.1.2.1 Definizione di estensioni del vocabolario

L'estensione di un vocabolario mira ad ampliare la capacità espressiva di un profilo rappresentabile con esso, definendo nuovi insiemi di attributi magari anche semplicemente attraverso la traduzione di ontologie già esistenti in attributi CC/PP. In questa direzione sono stati sviluppati molti lavori di ricerca che hanno permesso la creazione di vocabolari sempre più evoluti in grado di soddisfare le esigenze di specifiche applicazioni. Un esempio è l'estensione INTEL® del vocabolario UAProf [Intel2002], che include nuovi attributi per la descrizione delle caratteristiche hardware del dispositivo utilizzato dall'utente, ma soprattutto informazioni molto interessanti riguardo alle sue possibilità di comunicazione, come quelle relative alla bit rate supportata dal dispositivo e quella effettivamente utilizzata in un certo istante della comunicazione. Altri vocabolari sono stati definiti per la rappresentazione di alcune delle informazioni che compongono un profilo utente, interessanti esempi sono Expro [IntegratedProfile] e Liberty Alliance's ID Personal Profiles [Liberty2003] nati per la descrizione delle preferenze e degli interessi dell'utente.

UAProf se da un lato per la maggior parte dei contesti applicativi è abbastanza espressivo  per quanto riguarda le informazioni relative all'hardware dei dispositivi, non lo è quasi per nulla riguardo quelle relative alle caratteristiche di rete. In particolare, su questo campo, esso raccoglie solo alcune informazioni riguardo i protocolli di comunicazione e di sicurezza supportati, senza considerare altre informazioni importanti come quelle relative all'ampiezza di banda disponibile. La spiegazione di ciò potrebbe risiedere nel fatto che comunque il vocabolario UAProf, rivolgendosi a terminali per il WAP (Wireless Application), non è stato creato per la rappresentazione di descrizioni di generici dispositivi.

Questo problema è stato riscontrato anche da INTEL® quando, nella definizione della propria architettura PCA (Personal Internet Client Architecture), scelse di utilizzare lo standard CC/PP, basandosi sul vocabolario di base di UAProf ed estendendolo introducendo nuovi attributi. Molti di essi riguardano caratteristiche hardware del dispositivo tipo, frequenza della CPU, stato della batteria, eccetera, ma alcuni sono andati ad ampliare la descrizione delle risorse di rete disponibili da parte del dispositivo, rappresentate nel componente UAProf *NetworkCharacteristics*.

```
<rdf:Description rdf:ID="CurrentBearerMaximumBitRate">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
  <rdfs:domain rdf:resource="http://www.wapforum.org/profiles/UAPROF/
  ccppschema-20010330#NetworkCharacteristics"/>
  <rdfs:comment>Description:  The maximum bit rate in Kbps for the current
  bearer service
    Type:         Literal
    Resolution:   Locked
    Examples:     "56.6", "10000"
  </rdfs:comment>
</rdf:Description>

<rdf:Description rdf:ID="CurrentBearerActualBitRate">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Property"/>
  <rdfs:domain rdf:resource="http://www.wapforum.org/profiles/UAPROF/
  ccppschema-20010330#NetworkCharacteristics"/>
  <rdfs:comment>Description:  The current actual bit rate in Kbps for the
  current bearer service
    Type:         Literal
    Resolution:   Override
    Examples:     "26.4", "100"
  </rdfs:comment>
</rdf:Description>
```

**Figura 3.5:** Descrizione RDF di alcuni attributi introdotti dall'estensione Intel di UAProf

In Figura **3.5** viene riportato un estratto dell'estensione INTEL®, che definisce alcuni di questi attributi: rispettivamente essi descrivono la bit-rate massima supportata dal mezzo su cui è stata aperta la sessione e la bit-rate attualmente disponibile.

#### 3.1.2.2 Integrazione di profili

Nonostante l'elevato numero di vocabolari attualmente esistenti, ci sono informazioni difficili da descrivere usando un vocabolario predefinito ed unico. Consideriamo ad esempio la modellazione completa di un contesto, comprendente la descrizione delle attività che l'utente è capace di svolgere e di quelle in esecuzione in un certo istante, la sua posizione geografica, le condizioni generali dell'ambiente circostante, le risorse che interagiscono. Il valore di questi attributi è fortemente dinamico e soprattutto il set di informazioni considerate rilevanti è altamente variabile. Dunque è molto difficile rappresentare questo genere di informazioni specialmente se si vuole trattare il problema in modo generale ed esaustivo senza fare ipotesi sul tipo di applicazioni che le sfrutteranno.

Le informazioni che descrivono un profilo caratterizzano diverse entità: il dispositivo, l'utente, l'infrastruttura di comunicazione, i providers. Un modo per poter trattare tutte queste informazioni è quello di lasciare che siano le stesse entità a raccoglierle e rappresentarle attraverso opportuni vocabolari e quindi inserire componenti specifici per la loro interrogazione e per la loro integrazione in un profilo unico.

Uno dei problemi più importanti da risolvere in questo tipo di approccio è la risoluzione dei conflitti: diversi vocabolari possono identificare attributi diversi con lo stesso nome. Per evitare che sorgano ambiguità, ogni attributo deve essere associato a un significato ed è necessaria l'esistenza di un componente per la risoluzione dei conflitti e per la corrette gestione delle informazioni raccolte.

La Figura **3.5** mostra una possibile architettura, presentata in [IntegratedProfile], per un approccio di questo genere. Il Service Provider Profile Manager raccoglie i profili "intermedi" dai vari attori dello scenario di comunicazione per il Profile Interface (PI) che si prende carico di creare il profilo unico integrato. Il PI appoggiandosi a un componente interno, chiamato Merge, e in base a delle politiche

di risoluzione effettua la sintesi del profilo integrato selezionando gli attributi che lo descriveranno e determinando il loro valore.

Meccanismi di questo tipo riescono a risolvere il problema della rappresentazione dei profili in modo molto flessibile, fornendo agli sviluppatori di middleware la possibilità di progettare servizi di supporto per la gestione dei profili indipendenti dal contesto applicativo.

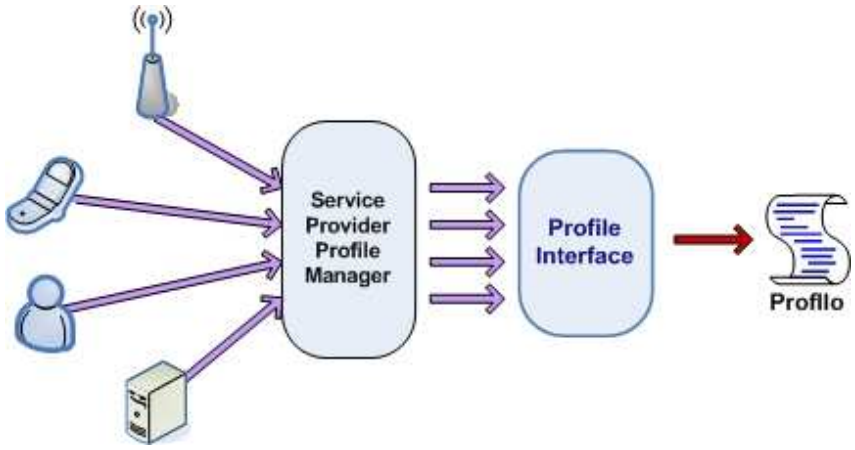



**Figura 3.5:** Architettura per l'integrazione di profili diversi

### 3.1.2.3 Confronto tra i due approcci

L'utilizzo di vocabolari ed estensioni esistenti è molto comodo e rappresenta un'ottima soluzione se sufficientemente espressiva per il contesto applicativo in cui viene utilizzata. Il difetto di questa soluzione sta nel fatto che le diverse esigenze applicative porterebbero alla definizione di un numero molto elevato di vocabolari che potrebbero causare confusione e disuniformità fra le applicazioni che le utilizzano. Al contrario ricordiamo che l'obiettivo fondamentale di uno standard per la descrizione del contesto è proprio quello di uniformare i linguaggi usati per descrivere le informazioni in esso contenute.

Il secondo approccio proposto risulta invece essere molto flessibile perchè lascia ai singoli gestori delle informazioni il compito di descrivere le risorse nel modo che ritengono più opportuno, ma il suo utilizzo richiede l'introduzione di complessità dovuta alla progettazione e allo sviluppo di un entità capace di tradurre i vari profili e integrarli in un unico profilo a cui l'applicazione fa riferimento.

## 3.2 Modelli per la rappresentazione di dati multimediali

L'esponenziale crescita della presenza di contenuti multimediali di vario tipo e genere in Internet e l'utilizzo sempre più massiccio di applicazioni multimediali distribuite hanno spinto negli ultimi anni la comunità scientifica a definire degli standard per la descrizione di dati multimediali, al fine di facilitare la loro ricerca e distribuzione. In realtà il tema della rappresentazione dei contenuti è stato studiato anche in passato per permettere una efficace ed efficiente catalogazione e ricerca di informazioni di tipo testuale, tipo libri, riviste e documenti. Tuttavia, la complessità delle informazioni multimediali non poteva essere supportata dai vecchi modelli di rappresentazione, da qui la necessità di cercare nuovi modelli che potessero rispondere alle esigenze di dati ad alto contenuto informativo.

In questo paragrafo verrà presentato dapprima Dublin Core, modello di rappresentazione dei contenuti di tipo bibliotecario utilizzato a partire dalla metà degli anni novanta. Dopo aver preso atto delle limitazioni imposte da Dublin Core verrà analizzato MPEG-7, standard di nuova generazione in grado di descrivere anche contenuti complessi come quelli multimediali.

### 3.2.1 Dublin Core

Dublin Core nasce nel 1995 dalla cooperazione dell'Online Computer Library Center (OCLC) di Dublin (in Ohaio, USA) e del National Center for Supercomputer Applications (NCSA), con l'obiettivo di sviluppare un insieme di metadati per la descrizione delle informazione elettroniche in rete [DC].

Lo standard Dublin Core è composto da quindici elementi la cui semantica è stata stabilita attraverso il lavoro congiunto di un gruppo di ricerca internazionale e interdisciplinare che comprendeva professionisti del mondo bibliotecario, dell'ambiente museale, esperti di informatica, codifica testi e di altri campi culturali.

Questi elementi descrivono caratteristiche della risorsa per lo più tipo bibliografico del tipo: titolo, autore, argomento, produttore, data di creazione, linguaggio, ect. .

La caratteristiche principali di Dublin Core sono le seguenti:

- *Semplicità di creazione e utilizzo:* Il set Dublin Core è composto da un limitato numero di elementi in modo da render facile il suo utilizzo anche per non specialisti della catalogazione.
- *Interoperabilità semantica:* La ricerca di informazioni di interesse per l'utente è molte volte resa difficile a causa dell'ambiguità dei termini e dei metodi di descrizione utilizzati nei diversi campi della conoscenza. Per far fronte a questo problema, Dublin Core definisce un insieme di dati e caratteristiche il cui significato e valore è universalmente compreso e accettato.
- *Infrastruttura internazionale:* Dublin Core è stato originariamente sviluppato in lingua inglese, ma per non cadere in problemi legati alla comprensione sono state create sue versioni in molte altre lingue.
- *Estendibilità:* Dublin Core supporta un meccanismo per estendere il set di elementi standard per soddisfare eventuali nuove esigenze di possibili nuovi contesti applicativi.

Risulta evidente che metadati contenti solo informazioni di tipo bibliografico non possono essere molto utili al fine di distribuire contenuti multimediali adattati al contesto utente. Come detto nel capitolo introdudittivo infatti, il metadato di una presentazione multimediale deve fornire informazioni riguardo le proprie caratteristiche di presentazione e qualità che indubbiamente Dublin Core non è in grado di gestire.

### 3.2.2 MPEG-7

MPEG-7 (Multimedia Content Descriptor Interface) si pone nell'ottica di fornire strumenti e standard per la descrizione di dati di tipo multimediale, non considerando la loro particolare tecnologia di memorizzazione e trasmissione, ma ponendosi come valido strumento per la gestione e interpretazione del contenuto informativo presente in essi [MPEG7].

MPEG-7 è uno standard aperto che non si rivolge a particolari applicazioni, ma piuttosto cerca di essere il riferimento del più vasto numero possibile di programmi e piattaforme, ponendosi anche l'obiettivo di esser compatibile con le soluzioni già esistenti.

MPEG-7 è formato dai seguenti sette moduli:

1. ***MPEG-7 System:*** Fornisce strumenti di supporto;
2. ***MPEG-7 Descriptor Definition Language:*** Linguaggio per la definizione di nuove caratteristiche e nuovi schemi di descrizione;
3. ***MPEG-7 Audio:*** Strumenti per la descrizione di oggetti di tipo audio;
4. ***MPEG-7 Video:*** Strumenti per la descrizione di oggetti di tipo video;
5. ***MPEG-7 Multimedia Description Schemas:*** Descrizione applicabili a contenuti multimediali di qualunque tipo.
6. ***MPEG-7 Reference Software:*** Implementazione software delle parti rilevanti dello Standard;
7. ***MPEG-7 Conformance:*** Linee guida e procedure per testare la conformità allo standard della varie implementazioni.

Il modulo *"MPEG-7 System"* definisce due elementi fondamentali dello Standard:

- ***Il Descriptor (D):*** Definisce la sintassi e la semantica di un dato multimediale, descrivendo la sue caratteristiche e le relazioni che intercorrono tra tutti gli elementi che lo compongono.

  La rappresentazione dei dati multimediali può essere svolta con differenti livelli di dettaglio e può coinvolgere vari tipi di informazione classificando i descrittori in due categorie:

  1. Quelli per la descrizione di informazioni legate al contenuto, come informazioni di produzione, di utilizzo, fisiche, strutturali, etc.;
  2. Quelli per la descrizione di informazioni non legate al contenuto, come il contesto.

- ***Il Descriptor Schema (DS):*** Specifica la struttura e la semantica delle relazioni tra i componenti che compongono la descrizione di un dato multimediale.

  Un Descriptor Schema estende il set di Descriptor MPEG-7, combinando tra loro elementi già esistenti, creando strutture più complesse e definendo le relazioni che intercorrono tra gli elementi che compongono queste nuove strutture.

Il modulo *"MPEG-7 Description Definition Language"* (DLL) fornisce un linguaggio mediante il quale gli sviluppatori creano gli schemi dei propri documenti e, se necessario, definiscono nuove descrizioni a partire da quelle esistenti nello standard.

Il DLL deve essere in grado di rappresentare le relazioni spaziali, temporali, strutturali e concettuali presenti all'interno di un oggetto multimediale, inoltre deve essere un linguaggio comprensibile sia dagli utenti che dai dispositivi quindi deve essere indipendente da piattaforme e applicazioni.

La descrizione di un contenuto multimediale viene memorizzata in un file di tipo XML, i cui elementi e attributi sono definiti attraverso uno XMLSchema [xmlSchema], che permette la rappresentazione del DLL. Questo approccio è molto comodo perché l'uso di XMLSchema permette una validazione automatica di una qualunque descrizione multimediale.

MPEG-7, in conclusione, fornisce quell'espressività necessaria agli sviluppatori di applicazioni multimediali per la descrizione dei contenuti, ma può risultare in alcune situazioni troppo complesso rispetto alle esigenze delle singoli applicazioni.

## 3.3 Conclusione

La necessità di esprimere in un linguaggio universale le proprietà che caratterizzano il contesto di un utente da un lato e i contenuti da consegnare dall'altro ha spinto alcuni organismi internazionali come W3C e ISO/IEC alla definizione e promozione di standard quali CC/PP e MPEG7, oggetto di studio in questo capitolo.

Dopo aver analizzato le caratteristiche di questi Standard, è stato evidenziato come in alcune situazioni sia necessario integrare una o più di queste tecnologie al fine di ottenere delle rappresentazioni delle risorse adeguate al particolare contesto applicativo in cui vengono utilizzate.

Nel prossimo capitolo verrà presentato MUM, un middleware per lo sviluppo di applicazioni multimediali, in cui si evidenzieranno gli approcci e le scelte effettuate durante la sua realizzazione per risolvere i problemi relativi alla rappresentazione di profili e metadati.

*Capitolo 4*

# MUM: MIDDLEWARE PER LE APPLICAZIONI MULTIMEDIALI

MUM (**M**obile agent-based **U**biquitous multimedia **M**iddleware), sviluppato presso il Dipartimento di Elettronica Informatica e Sistemistica (DEIS) dell'Università di Bologna, è una infrastruttura nata per fornire supporto allo sviluppo di applicazioni multimediali fornendo ai suoi utenti accessibilità ai servizi non solo in modo indipendente dai loro movimenti e dal luogo in cui accedono alla rete, ma anche considerando i vincoli tecnici imposti dal terminale che stanno utilizzando [MUM].

In realtà quest'ultimo punto non è ancora stato sviluppato, infatti è proprio l'obiettivo di questa tesi indagare la tematica dell'adattamento per cercare di sviluppare un servizio di personalizzazione dei contenuti multimediali da integrare nel sistema.

Questo capitolo non ha l'obiettivo di presentare MUM in modo esaustivo, ma solo di esporre le sue caratteristiche principali al fine di comprendere ciò che la piattaforma mette attualmente a disposizione e che può dunque essere utilizzato per lo sviluppo del progetto di tesi. In particolare, nella prima parte verranno introdotti i servizi principali offerti da MUM, nella seconda si studieranno più dettagliatamente quelli in stretto rapporto con il progetto portato avanti in questa tesi, cercando di individuare eventuali carenze e proponendo eventuali estensioni.

## 4.1 Caratteristiche di MUM

L'architettura distribuita di MUM è basata su un modello computazionale misto che cerca di integrare i vantaggi del tradizionale modello cliente/servitore con quello ad agenti mobili. MUM infatti è stato costruito al di sopra di SOMA, un ambiente ad

agenti mobili realizzato, come lo stesso MUM, presso il DEIS dell'Università di Bologna. Tuttavia, pur disponendo di una tecnologia per il supporto degli agenti mobili, non tutte le entità di MUM sono state realizzate come tali. Non conviene, infatti utilizzare questo potente paradigma di comunicazione in modo generalizzato per vari motivi. Primo fra tutti la richiesta di un ambiente di esecuzione uniforme sul quale far eseguire il codice mobile. Inoltre, il fatto che lo stesso codice mobile dovrà accedere alle risorse del sistema verso cui è migrato pone non pochi problemi di sicurezza, monitoraggio e autenticazione.

Molti dei servizi offerti da MUM sona basati sul principio di location-awarness cioè richiedono che la presenza di un supporto in grado di fornire la nozione di località fisica. A questa proposito SOMA definisce una gerarchia di astrazioni di località adatta alla descrizione di qualunque scenario di connessione, da quello di un rete estesa e aperta come Internet a quello di una rete locale LAN (Local Area Network). Essa definisce due livelli di astrazione:

1. Il **Place** rappresenta il contesto di esecuzione di un agente e sussume il concetto di nodo, infatti il place può corrispondere a una macchina fisica, ma su un nodo possono convivere anche più place.
2. Il **Dominio** rappresenta un'aggregazione di place che può corrispondere sia a un contesto reale (come una LAN) sia a una caratteristica logica (ad esempio l'insieme dei dispositivi dello stesso tipo).

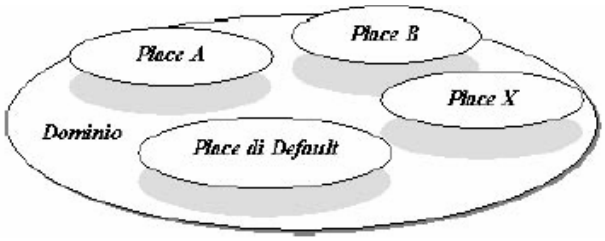



**Figura 4.1:** Astrazioni di località di SOMA

A livello implementativo, come mostra la Figura **4.1** il Dominio è realizzato come un place detto "place di default" che conosce tutti i place che lo costituiscono e rappresenta un punto di accesso da e verso l'esterno.
Per ulteriori approfondimenti sui servizi offerti da SOMA si rimanda a [Soma03].

### 4.1.1 Architettura di MUM

In Figura **4.2** viene rappresentata l'architettura di MUM, evidenziando i servizi di supporto offerti. Essa è composta da due livelli:

1. Il ***Mechanisms Layer***, che realizza i servizi di base;
2. *Il* ***Facilities Layer***, che incapsula le strategie e i servizi middleware direttamente utilizzabili dalle applicazioni costruite al di sopra di MUM.

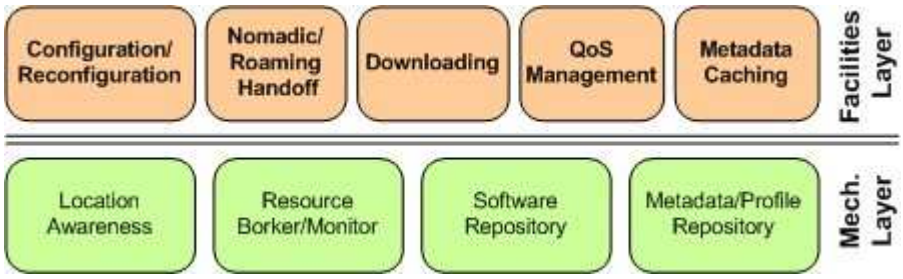



**Figura 4.2:** Architettura a livelli di MUM

Nel livello più basso, quello dei meccanismi, troviamo servizi essenziali come:

1. ***Location Awareness:*** Fornisce ai servizi del livello superiore visibilità sulla posizione dei nodi della rete, attraverso l'uso delle astrazioni di località di SOMA;
2. ***Resource Broker/Monitor:*** Ha il compito di effettuare il monitoraggio delle risorse;
3. ***Software Repository:*** Risorsa che mantiene il codice dei componenti che possono essere scaricati in fase di configurazione dei servizi;
4. ***Metadata/Profile Repository:*** Mantiene informazioni relative ai contenuti disponibili all'interno del sistema (ad esempio metadati sulle presentazioni) e ai profili dei clienti.

Questi meccanismi vengono utilizzati nello sviluppo dei servizi di supporto che MUM offre alle applicazioni. I servizi attualmente implementati sono:

1. ***Configuration/reconfiguration:*** Servizio per la configurazione del cammino che permette al servitore di raggiungere il cliente.

   Nello scenario di fruizione il client viene collegato a un server che dispone del contenuto richiesto attraverso un percorso che può coinvolgere nessuno, uno o più nodi intermedi detti proxy. Questo percorso, chiamato Service Path

(SP), attivo e riconfigurbile, è molto utile nella distribuzione dei vari componenti necessari alla elaborazione e al monitoraggio dei flussi inviati verso il cliente. Il servizio di configurazione si occupa quindi dell'inizializzazione dei nodi del SP affinché un certo servizio possa essere espletato. Inoltre, esso è anche in grado di gestire gli aspetti di negoziazione relativi alla gestione della qualità di servizio sia prima che durante l'erogazione del servizio richiesto;

2. ***Nomadic/Roaming Session Handoff:*** Realizza il supporto necessario a garantire continuità alla sessione utente. Quando, durante l'erogazione del servizio, l'utente si muove col proprio terminale mobile o comanda esplicitamente lo spostamento della propria sessione su un altro terminale, il sistema deve compiere delle operazioni che il letteratura vengono chiamate operazioni di gestione del *"Session handoff"*. Esse mirano a disconnettere i componenti applicativi dalle risorse del vecchio ambiente e a ricollegarle a quelle del nuovo in modo del tutto trasparente per l'utente. Il compimento di questa operazione è di fondamentale importanza per la preparazione dell'ambiente in cui la sessione dell'utente verrà spostata ed eseguita.
3. ***QoS Management:*** Incapsula le strategie per la gestione della qualità di servizio. Questo servizio collabora con il Monitor delle risorse, per verificare le condizioni generali della rete di comunicazione.
4. ***Downloading:*** Servizio per la ricerca e il dowloading del codice utilizzabile per la configurazione dei nodi che appartengono al Service Path.
5. ***Metadata Caching:*** Servizio per il caching distribuito dei metadati relativi ai contenuti multimediali presenti nel sistema.

### 4.1.2 Fruizione del materiale multimediale

Il modello di fruizione dei contenuti multimediali proposto da MUM introduce cinque entità fondamentali: Client, Server e Proxy realizzate come entità fisse, ClientAgent e ProxyAgent realizzate come entità mobili [Fos03].

Di seguito vengono analizzati brevemente il ruolo e le caratteristiche di ognuna di queste entità.

- ***Client:*** rappresenta l'end-point del Service Path che riceve i flussi multimediali richiesti. Per sua natura questa entità è fissa e risiede sulla macchina dalla quale l'utente effettua l'accesso al sistema. Il fatto che il Client non sia un agente mobile non implica che il software necessario per la sua esecuzione debba essere già presente sulla macchina dal quale verrà lanciato, infatti potrà essere utilizzato il servizio di downloading per scaricare a run-time tutti i componenti di cui si ha bisogno;

- ***Server:*** rappresenta l'altra estremità del Service Path, cioè l'entità che trasmette il contenuto multimediale comandato dall'utente. Così come il Client, anche il Server potrà usufruire del servizio di downloading per essere inizializzato con tutto il software di cui ha bisogno. Inoltre, una volta che un certo tipo di server viene lanciato su una macchina, viene utilizzato per servire tutte le richieste che da quel momento in poi arriveranno per quel tipo di server.

- ***Proxy:*** le due entità precedentemente presentate sono parte del ben noto modello Client/Server, il Proxy invece è un'entità non prevista da questo modello e ricopre un ruolo fondamentale nel modello computazionale di MUM. Il Proxy, posizionato sui nodi intermedi del Service Path, è un'entità che partecipa attivamente alla consegna del servizio richiesto. Nell'implementazione attuale di MUM, esso si occupa di:
  - Gestire l'intermittenza delle connessioni al Server. Questo problema, molto importante soprattutto per i dispositivi che accedono al sistema attraverso rete mobile (non sempre in grado di garantire una copertura permanente di tutto il territorio), viene risolto utilizzando il Proxy come una cache in cui memorizzare l'oggetto multimediale richiesto dall'utente.
  - Partecipare attivamente alla gestione della qualità di servizio, monitorando le condizioni delle risorse e chiedendo, se necessario, l'eventuale riconfigurazione del servizio in caso di

malfunzionamenti o in presenza di situazioni di sovraccarico della rete.

Nell'ottica di dotare MUM di un servizio di adattamento dei contenuti multimediali che tenga conto delle esigenze dei terminali dalle risorse limitate, uno o in generale più Proxy potrebbero essere responsabili della trasformazione del flusso multimediale in un formato adeguato alle caratteristiche del terminale cliente che lo ha richiesto.

- ***ClientAgent:*** Rappresenta l'entry-point del sistema dal lato cliente. Il suo compito è inizializzare la sessione per l'utilizzatore e poi gestire le interazioni con il resto del middleware. Svolge dunque un ruolo di mediatore che accetta le richieste fatte dall'utente attraverso un'opportuna interfaccia grafica permettendogli di interagire con MUM. Questa entità è stata realizzata come agente mobile per modellare il movimento dell'utente verso un altro terminale (nomadic user), e quello dell'utente insieme al proprio terminale verso un'altra area (roaming user).

- ***ProxyAgent:*** Agente mobile introdotto per supportare il movimento dei terminali che per la gestione vera e propria dei flussi si avvale dell'entità Proxy precedentemente introdotta.

### 4.1.3 Configurazione dinamica del sistema

In ambienti molto eterogenei non si può assumere che tutti i nodi che partecipano nel Service Path dispongano dei componenti necessari all'esecuzione di un certo servizio. La possibilità di scaricare codice solo al bisogno potrebbe portare notevoli benefici soprattutto nei casi in cui i nodi sono dispositivi con risorse limitate. Tuttavia, anche quando il Service Path è composto da macchine altamente performanti è utile disporre di un servizio di inizializzazione per evitare problemi legati all'indisponibilità dei componenti, sia di tipo applicativo che di tipo middleware, necessari per lo svolgimento del servizio richiesto.

Per questi motivi, MUM offre un servizio di supporto per la configurazione dinamica del sistema. Quando viene richiesto un servizio, prima viene scaricato il

software necessario alla sua trasmissione e ricezione, poi viene configurato il Service Path (scaricando, se necessario, il software indicato all'interno di un piano di inizializzazione) ed infine si inizia la sua erogazione.

In Figura **4.3** viene mostrata l'architettura del servizio di configurazione proposto evidenziando i meccanismi che vengono utilizzati per la sua realizzazione. In particolare, si può osservare che esso ha bisogno di definire e riconoscere le località dei partecipanti in modo da poter scaricare su ognuno il codice opportuno, inoltre si serve di un Resource Manager per supportare la riconfigurazione dinamica del sistema che viene garantita mediante la generazione di più piani di inizializzazione alternativi che possono essere utilizzati a run-time per riorganizzare il sistema.

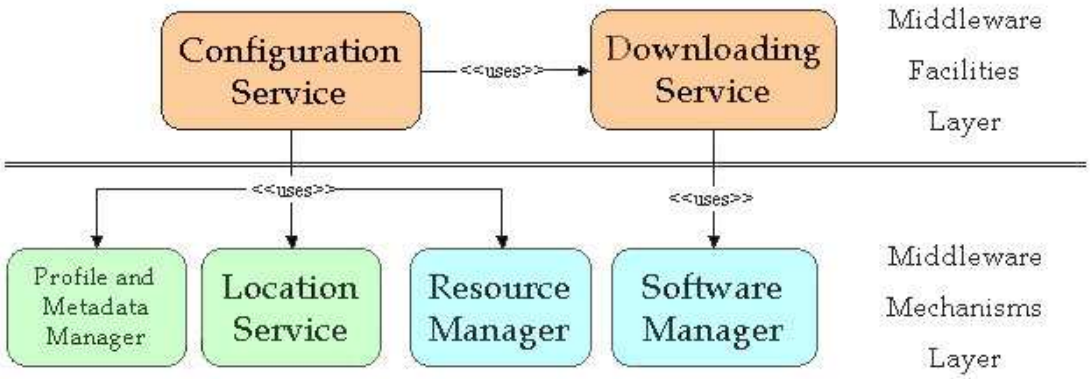



**Figura 4.3:** Architettura del servizio di configurazione proposto da MUM

Il servizio di configurazione viene realizzato attraverso due componenti middleware:

1. Il ***Decision Maker (DM)***, che incapsula le strategie che permettono di decidere come configurare il Service Path, producendo un piano che contiene una o più soluzioni corrispondenti a diversi livelli di qualità di servizio.
2. Il ***Plan Visitor Agent (PVA)***, agente mobile che viene spedito lungo il Service Path e che ha il compito di effettuare su ogni nodo la configurazione prevista dal piano generato e consegnatogli dal DM.

In Figura **4.4** viene mostrato lo scenario di configurazione: alle ricezione di una richiesta, il *ClientAgent* richiede l'inizializzazione del sistema indicando il DM da utilizzare per la generazione del piano di configurazione e il titolo che identifica il contenuto multimediale richiesto. Il servizio di configurazione, una volta ottenuto il

piano dal DM, lo passa a un PVA che attraversa il Service Path dal Client al Server per operare su ogni nodo:

- La negoziazione e la prenotazione delle risorse;
- Il downloading del codice necessario;
- L'inizializzazione di tutti i componenti necessari;

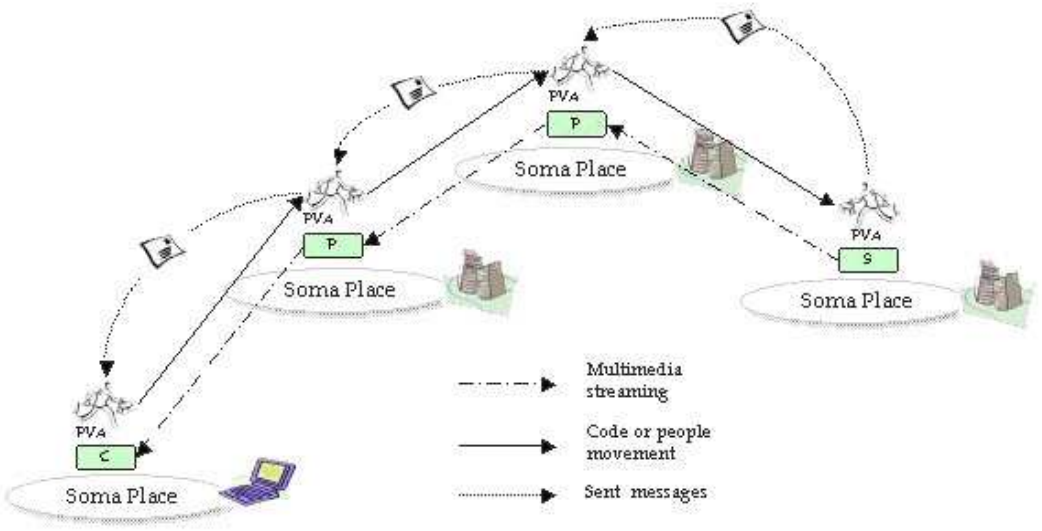



**Figura 4.4:** Protocollo per la configurazione del sistema

Il servizio di inizializzazione, quindi, provvede anche alla gestione della qualità di servizio. Appena giunto in un nuova località, infatti, il PVA per prima cosa vede se ci sono risorse sufficienti per la presentazione richiesta. In caso affermativo continua il processo di inizializzazione inviando un altro PVA sul nodo successivo, e se ci sono sufficienti risorse su tutto il path, il processo di inizializzazione termina con successo. Se al contrario in un certo place non sono disponibili sufficienti risorse, il Plan Visitor consulta il proprio piano di inizializzazione per vedere se esistono presentazioni alternative, a qualità più bassa e tenta di istanziare il percorso per tali presentazioni.

### 4.1.4 Mobilità di utenti e terminali

MUM considera due tipi di mobilità:

1. ***Nomadic-mobility:*** mobilità degli utenti da un terminale all'altro;

2. ***Roaming-mobility:*** mobilità degli utenti insieme ai dispositivi che stanno utilizzando.

Nel primo caso, rappresentato in Figura **4.5**, in risposta alla necessità dell'utente di cambiare il terminale dal quale sta consumando il servizio per spostarsi su un altro verso il quale si sta spostando fisicamente, MUM in modo trasparente all'utente trasferisce la sua sessione in modo che quando l'utente arrivi sul nuovo terminale possa continuare a fruire il materiale multimediale senza alcun ulteriore intervento da parte sua.

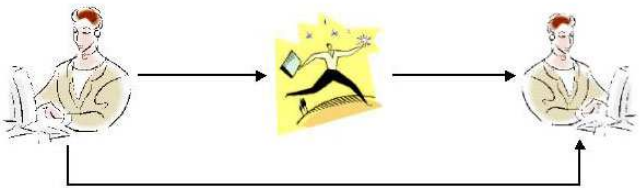


**Figura 4.5:** Modello di Nomadic-mobility

Nel secondo invece, rappresentato in Figura **4.6**, MUM assiste il movimento del cliente facendo in modo che l'infrastruttura si modifichi a run-time in modo da seguire i movimenti dell'utente.

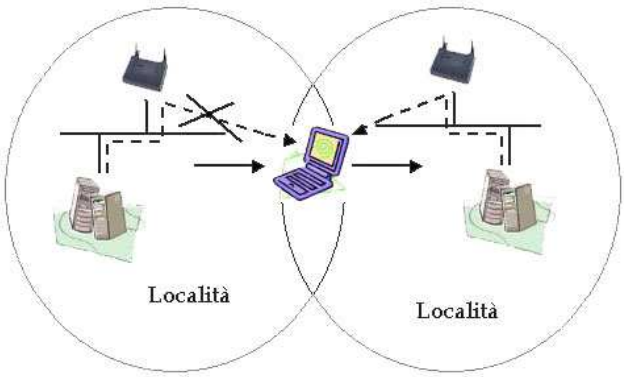



**Figura 4.6:** Modello di Roaming-mobility

In entrambi i casi è necessario effettuare un re-binding dinamico dei componenti di servizio, ed è compito del middleware compiere questa operazione in modo trasparente all'utente. Affinché la totale gestione del "Session handoff" sia trasparente, MUM offre un servizio di Session Continuity la cui architettura è rappresentata in Figura **4.7**.

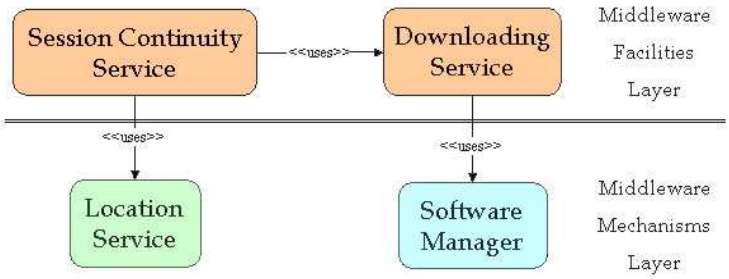



**Figura 4.7:** Architettura del Servizio di Session Continuity

L'approccio utilizzato da MUM nella realizzazione di questo servizio è rappresentato in Figura **4.8**: la richiesta dell'utente, catturata dal *ClientAgent* , viene passata all'*InitManager*, un componente middleware presente su ogni nodo, che ottiene dal DM un piano che descrive le azioni da compiere per spostare la sessione. Questo piano viene consegnato a un PVA che invia un altro PVA verso il nodo di destinazione della sessione. Una volta che quest'ultimo è arrivato scarica, se necessario, il codice ed inizializza il nuovo Client collegandolo al Proxy. Quando il "Session handoff" termina viene rispedito al place di partenza un messaggio che avvia la migrazione fisica del *ClientAgent* verso il nodo di destinazione.

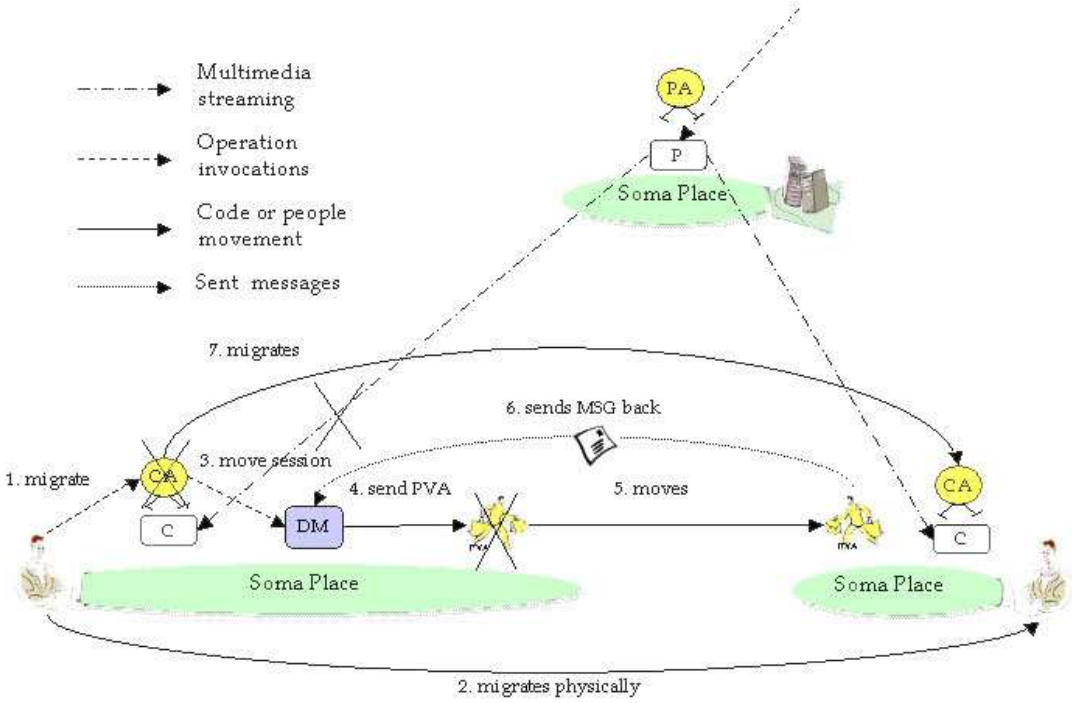



**Figura 4.8:** Protocollo per garantire continuità di sessione in MUM

Questo protocollo garantisce continuità di sessione, infatti quando avviene la migrazione il nuovo Client è già pronto permettendo di effettuarla senza interrompere l'erogazione del servizio.

Per la gestione dei roaming-users viene usato un approccio molto simile a quello esposto precedentemente per i nomadic-users, solo che questo tipo di mobilità non viene modellata come migrazione del *ClientAgent*, ma di uno o più *ProxyAgent* situati nelle vicinanze del Client.

## 4.2 MUM e Supporto all'Eterogeneità

Come già accennato ad inizio capitolo, attualmente MUM non fornisce un supporto per adattare le presentazioni multimediali alle esigenze e limitazioni imposte dai terminali utenti. D'altra parte per un middleware che mira, come MUM, a semplificare lo sviluppo di applicazioni multimediali soggette ad operare in scenari ampiamente eterogenei (come illustrato nel Capitolo 1), è di fondamentale importanza integrare un servizio di adattamento dei contenuti per far fronte a questo problema. L'obiettivo principale di questa tesi è arricchire MUM con un servizio di questo tipo.

Nel capitolo 1 sono stati individuati due servizi essenziali per lo sviluppo di un supporto per l'eterogeneità: il primo per la gestione dei metadati, il secondo per la gestione dei profili utente. Questi servizi sono già presenti in MUM, anche se, essendo stati progettatati e sviluppati per altri scopi, non hanno tutte le funzionalità necessarie per fornire supporto a un sistema di personalizzazione dei contenuti.

In questa sezione verranno presentati questi due servizi, per capire quelle che sono le carenze da superare nelle fasi successive del lavoro di tesi. Inoltre, si illustrerà il modo in cui intervenire per integrare in MUM un servizio di adattamento delle presentazioni orientato ai profili degli utenti che ne fanno richiesta.

### 4.2.1 Gestione dei Contenuti Multimediali e dei rispettivi Metadati

Il sistema di gestione dei contenuti di MUM è stato costruito nell'ottica di migliorare i tempi di risposta del sistema alle richieste del cliente, cercando di

limitare allo stesso tempo i consumi di risorse per la memorizzazione delle presentazioni multimediali.

L'architettura di questo sistema fornisce ad ogni place di default una cache. Ogni volta che uno di questi place viene coinvolto nella fruizione di una presentazione, esso memorizza nella propria cache solo un suo piccolo frammento iniziale, detto "prefisso", in modo da poter rispondere prontamente ad eventuali richieste successive della stessa presentazione iniziando a inviare il prefisso memorizzato in cache. Durante la trasmissione del prefisso, il place di default si procura la presentazione originale dal nodo server e quando l'ha ottenuta si occupa della sua trasmissione verso il cliente. I prefissi memorizzati possono avere caratteristiche qualitative molto diverse, al contrario la presentazione originale sul server è unica ed a qualità massima. Nelle cache dunque oltre ai prefissi vengono memorizzati anche i metadati che li descrivono.

MUM dispone di un componente per gestire le presentazioni multimediali in grado di:

- Fornire una loro opportuna descrizione espressa all'interno di metadati, rappresentati secondo le indicazioni di alcuni standard internazionali in modo che le applicazioni, anche se realizzate su piattaforme di supporto diverse, possano condividere informazioni sui contenuti multimediali gestiti.
- Creare dei meccanismi per la realizzazione del caching distribuito dei metadati con lo scopo di ridurre i tempi di partenza del servizio percepiti dall'utente finale.
- Definire e gestire politiche per l'accesso alle cache che contengono i metadati.

#### 4.2.1.1 Rappresentazione dei Meatadati

Nel capitolo 3, sono stati presentati due modelli standard per la rappresentazione digitale di informazioni caratterizzanti le risorse: DublinCore e MPEG-7. Le prima tecnologia, tuttavia, non è in grado da sola di descrivere in modo esaustivo il contenuto informativo di un dato multimediale; la seconda invece definisce un modello fin troppo complesso per la rappresentazione dei metadati. La scelta fatta in MUM è stata quella di fondere i due standard per creare uno schema per la

rappresentazione delle presentazioni multimediali che risulti più semplice di MPEG-7, ma allo stesso tempo più espressivo di DublinCore.

La fusione deriva dalla considerazione che le caratteristiche di un dato multimediale possono essere suddivise in informazioni bibliografiche ed informazioni fisico-tecniche. É stato scelto di utilizzare una parte del set di elementi del DublinCore per la rappresentazione delle prime e quattro elementi di MPEG-7 per la rappresentazione delle seconde.

In conclusione dunque lo schema di rappresentazione di un metadato di MUM è composto di cinque elementi fondamentali:

1. ***DCMES:*** Parte del set di elementi di DublinCore per specificare le informazioni di natura bibliografica degli oggetti multimediali. In particolare gli elementi utilizzati sono: *title, creator, subject, description, identifier, language* e *rights*.
2. ***MPEG-7 MediaLocator Descriptor*** per la definizione della locazione della risorsa.
3. ***MPEG-7 MediaTime Descriptor*** per la definizione degli attributi temporali associati alla risorsa (istante iniziale e durata della presentazione).
4. ***MPEG-7 MediaFormat Descriptor*** per la rappresentazione delle informazioni sul formato e sulla codifica della risorsa.
5. ***MPEG-7 TemporalDecomposition Descriptin Scheme*** per la descrizione delle componenti spazio-temporali del filmato.

In Figura **4.9** viene rappresentato lo schema del modello utilizzato da MUM per descrivere la singola entità multimediale, in cui in verde sono evidenziati gli elementi di DublinCore, in Blu quelli di MPEG-7.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:mpeg7="http://www.mpeg.org/MPEG/2000/" >

  <import namespace="http://purl.org/dc/elements/1.1/" />
  <import namespace="http://www.mpeg.org/MPEG/2000/" />

  <element name="MultimediaDescription">
    <complexType>
      <sequence>
        <element ref="dc:title" minOccurs="1" maxOccurs="1" />
        <element ref="dc:creator" minOccurs="0" maxOccurs="unbounded" />
        <element ref="dc:subject" minOccurs="0" maxOccurs="unbounded" />
        <element ref="dc:description" minOccurs="1" maxOccurs="1" />
        <element ref="dc:identifier" minOccurs="1" maxOccurs="1" />
        <element ref="dc:language" minOccurs="0" maxOccurs="1" />
        <element ref="dc:rights" minOccurs="0" maxOccurs="1" />
```

```
        <element name="MediaLocator" type="mpeg7:MediaLocatorType"
                 minOccurs="1" maxOccurs="1" />
        <element name="MediaTime" type="mpeg7:MediaTimeType"
                 minOccurs="1" maxOccurs="1" />
        <element name="MediaFormat" type="mpeg7:MediaFormatType"
                 minOccurs="1" maxOccurs="1" />
        <element name="TemporalDecomposition"
                 type="mpeg7:VideoSegmentTemporalDecompositionType"
                 minOccurs="0" maxOccurs="1" />
      </sequence>
    </complexType>
  </element>
</schema>
```

**Figura 4.9:** XML Schema per la descrizione di un oggetto multimediale in MUM

#### 4.2.1.2 Caching dei Metadati

Il servizio di caching distribuito dei metadati, predispone per ogni dominio una cache con un relativo gestore che si occupa della comunicazione con gli altri gestori del sistema seguendo un protocollo di tipo gerarchico, considerato molto efficiente per il trattamento di dati dalle dimensioni molto ristrette come quelle dei metadati.

Il modello gerarchico prevede che ogni cache possa comunicare solo con la cache di livello superiore o con gli eventuali figli. Si può dunque pensare di rappresentare questa architettura di caching, nel modo rappresentato in figura **4.10**, tratta da [Bor03], in cui l'intero sistema presenta la struttura di un albero. Ogni sotto-albero, definito a partire da ogni singolo dominio, rappresenta una località che comprende il dominio stesso e tutti i dominii che da esso discendono fino a quelli che costituiscono le foglie dell'albero. L'albero dei dominii è inoltre suddiviso orizzontalmente in modo da inserire ogni località in un livello.

Il gestore di una cache permette di effettuare l'inserimento, l'eliminazione e il recupero di un metadato relativo a un certa presentazione garantendo la consistenza di tutte le cache presenti nel sistema: quindi se, ad esempio, viene inserito un nuovo metadato in una cache esso dovrà essere inserito anche in tutte le altre cache che si trovano al di sopra di lui nella gerarchia (fino alla radice). Quando da un certo dominio viene richiesto il suo recupero, per prima cosa si controlla la cache locale al dominio: se essa contiene il metadato viene fornita direttamente una sua copia all'utente, se al contrario essa non è in grado di soddisfare la richiesta viene innescato un procedimento iterativo che la inoltra alla cache di livello superiore fino a quando o il metadato viene trovato o si raggiunge la cache relativa al nodo radice. In quest'ultimo caso, se anche esso non è in grado di rispondere alla richiesta

dell'utente significa che la presentazione richiesta non è disponibile all'interno del sistema.

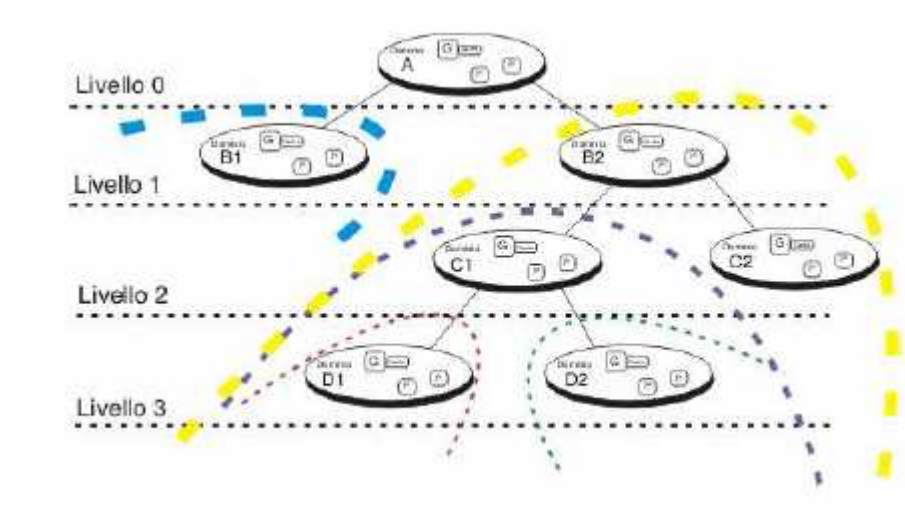



**Figura 4.10:** Architettura del caching gerarchico e definizione delle località

L'accesso alla cache è regolato da politiche di sistema interpretate da appositi gestori, che si prendono carico di analizzare le richieste e decidere sulla base della politica utilizzata se permette oppure no l'operazione richiesta.

In sintesi dunque su ogni dominio del sistema si ha un componente, dall'architettura rappresentata in Figura **4.11,** che si occupa del caching dei metadati.

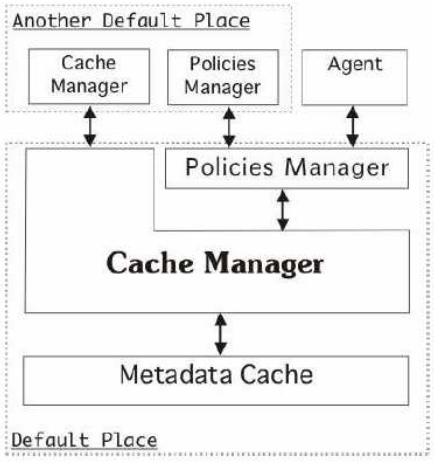



**Figura 4.11:** Struttura del gestore dei metadati all'interno di un Place di Default.

Le politiche di accesso attualmente definite all'interno del sistema riguardano esclusivamente l'inserimento di un nuovo metadato in cache. Le direzioni previste che possono essere seguite da una politica sono due:

1. Lo spazio occupato dai dati memorizzati nelle cache.
2. La prontezza del sistema nel rispondere alla richiesta di utente.

Infatti se da una parte inserire un presentazione in una cache comporta un maggiore diffusione e distribuzione dei contenuti migliorando la reattività del sistema, dall'altra comporta consumo di risorse per la memorizzazione di risorse già presenti all'interno del sistema.

#### 4.2.1.3 Carenze del modello per la gestione dei dati multimediali

L'espressività e la varietà delle informazioni contenute nei metadati definiti da MUM è molto ampia e le scelte effettuate per la loro rappresentazione dovrebbero soddisfare le esigenze di un modello per l'adattamento dei dati multimediali. Al contrario potrebbe rappresentare un vincolo la modalità di recupero dei metadati imposta dal modello gerarchico utilizzato dal sistema di caching. Infatti, secondo questa architettura la procedura di recupero finisce appena viene trovata una presentazione del contenuto richiesto, senza tener conto di alcun altro fattore. Per un sistema in grado di adattare i contenuti, invece, potrebbe essere molto utile, se possibile, selezionare più di una presentazione corrispondente al titolo richiesto al fine di aver più opzioni tra le quali scegliere per eseguire le operazioni di adattamento nel modo più efficiente possibile. Tuttavia potrebbe sembrare insensato seguire una politica di questo tipo poiché le cache sono state introdotte proprio evitare consumo di banda.

Inevitabilmente dunque, bisognerà sviluppare questa tematica per cercare possibili soluzioni di compromesso che garantiscano la possibilità al sistema di adattamento di poter avere più soluzioni su cui operare, ma allo stesso tempo tengano conto del consumo delle risorse di rete.

### 4.2.2 Gestione dei profili utente

La gestione dei profili utente è stata introdotta in MUM per svolgere al meglio la prenotazione delle risorse, quindi, per realizzare il sottosistema che si occupa della

gestione della qualità di servizio. Il profilo utente di MUM contiene informazioni riguardanti le piattaforme utilizzate dall'utente per accedere al sistema e le loro locazioni.

Dato che i descrittori delle piattaforme sono un dato che rimane piuttosto stabile nel tempo sono stati trattati in modo separato rispetto ai profili utente, che invece possono cambiare più frequentemente, dividendo logicamente i due database che contengono queste informazioni. Nel profilo utente quindi è contenuta un tabella nella quale sono indicati tutti i terminali da cui l'utente accede al sistema con i relativi identificatori univoci dei descrittori delle piattaforme utilizzate, che invece contengono informazioni specifiche che descrivono aspetti fisici del dispositivo: grandezza dello schermo e classe della piattaforma.

Un sistema di adattamento, come visto nel capitolo 1, potrebbe aver bisogno di informazioni più dettagliate riguardo il profilo dell'utente, la cui modellazione rappresenta uno dei temi principali nel suo sviluppo.

La rappresentazione del profilo, inoltre, nell'implementazione attuale di MUM non viene trattata. Al contrario nel capitolo 3 è stata sottolineata l'importanza di avere profili e metadati il più possibile conformi agli standard internazionali affinché possano essere interpretati in modo corretto ed uniforme. Quindi in visione degli studi effettuati e discussi nei precedenti capitoli di questo lavoro di tesi, è necessaria una ristrutturazione dell'intero servizio che vada incontro sia agli aspetti di modellazione che di rappresentazione dei profili, cercando di mantenere il più possibile la compatibilità di ciò che di nuovo verrà creato con quello che esiste già.

## 4.3 Integrazione dell'adattamento in MUM: Problematiche

Questa sezione non ha l'obiettivo di capire come effettuare nel dettaglio l'integrazione del servizio di adattamento che verrà sviluppato, ma piuttosto di identificare quali saranno i servizi coinvolti da questo processo proponendo per loro alcune importanti evoluzioni logiche e strutturali in modo da facilitare l'integrazione con il nuovo servizio di adattamento.

Nel capitolo 1, è stato delineato l'approccio fondamentale che verrà seguito nella costruzione del servizio per l'adattamento da integrare in MUM: un approccio basato

su un importante  principio dell'ingegneria del software, la modularità. Il processo di adattamento infatti, può essere pensato come il risultato di due fasi: la prima in cui l'adattamento viene configurato, la seconda in cui viene attuato (Figura **1.5**).

La prima fase interviene sul Configuration Service, l'altra invece direttamente sulla fruizione del materiale multimediale.

### 4.3.1 Configuration Service e Adattamento

Il servizio di configurazione di MUM permette all'atto di un richiesta utente, di istanziare lungo il Service Path che unisce il Cliente al Servitore tutti i componenti necessari all'erogazione del servizio, eventualmente scaricandoli se non sono presenti.

Il Configuration Service decide quale software utilizzare su un certo nodo sulla base delle decisioni prese da un componente middleware chiamato DecisionMaker (DM). Esso riveste senza un dubbio un ruolo fondamentale, avendo il compito di creare il piano di inizializzazione che verrà poi successivamente consegnato a un PVA che si occuperà della sua distribuzione seguendo il protocollo illustrato in 4.1.3. Come mostra la Figura **4.12**, la generazione del piano viene fatta dal DM in seguito all'elaborazione della profilo utente, del metadato relativo alla presentazione richiesta e alla locazione da dove esse proviene.

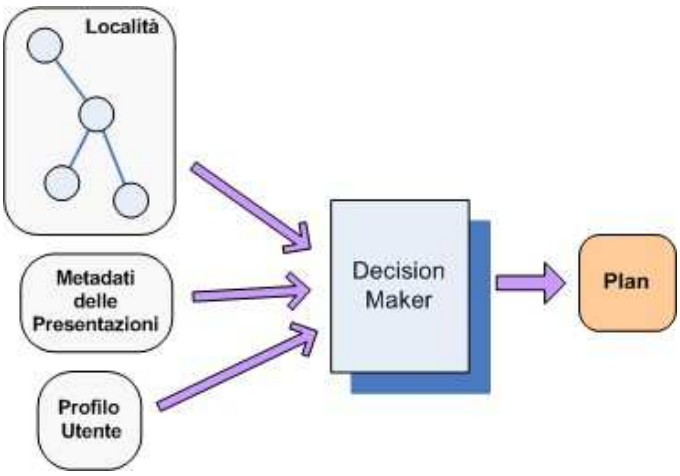



**Figura 4.12:** DecisionMaker e attuale generazione del Piano di Inzializzazione

Nella implementazione attuale tra tutte le presentazioni trovate il DM ne seleziona al massimo tre (se nella stessa cache sono state trovate almeno tre presentazioni

corrispondenti al titolo richiesto) corrispondenti a tre livelli di qualità di servizio diversi (alta, media, bassa) considerando l'indicazione di qualità della presentazione espressa nel metadato e la distanza del Server contente la presentazione dal Client. Per ognuna di queste soluzioni genera un piano che potrà essere utilizzato per configurazione o la riconfigurazione del Service Path. Il piano effettivamente utilizzato verrà selezionato dal PVA in collaborazione con il Resource Broker che valuta se il livello delle risorse disponibili è sufficiente per la consegna della presentazione a cui il piano fa riferimento. La decisione di quale presentazione utilizzare per l'erogazione del servizio è dunque presa direttamente dal DM.

Se si vuole introdurre nel sistema l'adattamento, bisogna fare in modo che le decisioni che verranno prese nella sua fase di configurazione vengano imposte al DM durante la generazione del piano, in modo che esso permetta ad esempio di scaricare il codice necessario per attuare l'adattamento sui nodi e sulla presentazione scelti dal *"configuratore"* . Quindi, come mostra la Figura **4.13**, il DM verrebbe sollevato dalla responsabilità di prendere le decisioni e rimarrebbe solo un entità capace di generare dei piani di inizializzazione dipendenti da parametri ottenuti come risultato della prima fase del processo di adattamento.

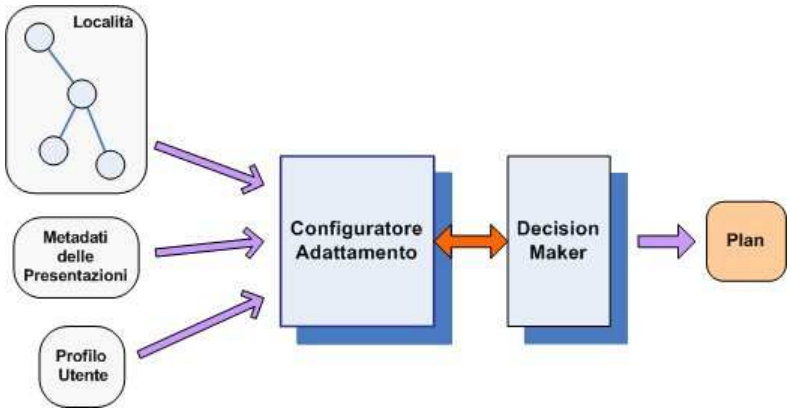



**Figura 4.13:** Collaborazione tra DecisionMaker e Configuratore dell'Adattamento

### 4.3.2 Fruizione del materiale multimediale e Adattamento

La fruizione del contenuto multimediale richiesto dall'utente coinvolge varie entità di MUM: il Client, i Proxy e il Server, opportunamente configurati in una fase di inizializzazione che precede l'erogazione del servizio.

Nell'attuale implementazione di MUM possiamo trovare due tipi fondamentali di Proxy, rappresentati rispettivamente in figura **4.14a** e **4.14b**: il primo non effettua nessun tipo di elaborazione del flusso in ingresso limitandosi semplicemente e riportarlo in uscita inoltrandolo verso il Client; il secondo oltre a eseguire l'operazione del primo memorizza un copia del flusso nella cache.

Nell'ottica di inserire nel sistema un servizio di adattamento, dopo aver capito dove operare per inserire il configuratore del processo, si ha la necessità di individuare il rispettivo componente sul quale attuarlo. Essendo l'attuazione dell'adattamento un processo che opera direttamente sul flusso, non può che essere svolto su una delle entità che compongono la catena di fruizione. Sicuramente la più adeguata a questo tipo di operazione è il Proxy, che ponendosi fra cliente e servitore svolgerebbe il processo di codifica in modo del tutto trasparente sia all'una che all'altra entità. Questa scelta non dovrebbe stupire visto che, fin dai capitoli iniziali, questa entità è stata considerata di fondamentale importanza per la realizzazione dei servizi middleware in grado di operare sul flusso destinato all'utente.

Supportare l'adattamento dunque significa anche creare un nuovo tipo di Proxy, rappresentato in figura **4.14c** per poter effettuare l'elaborazione del flusso.

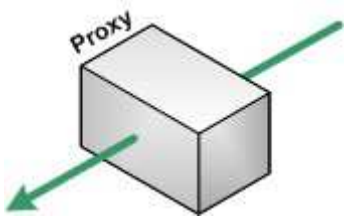



**Figura 4.14a:** Proxy semplice

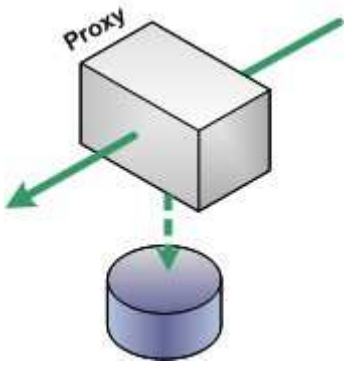



**Figura 4.14b:** Proxy per il caching del flusso

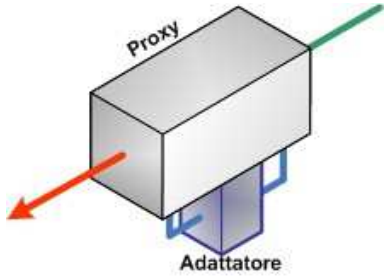



**Figura 4.14c:** Proxy per l'adattamento del flusso

## 4.4 Conclusione

In questo capitolo è stato presentato un middleware per le applicazioni multimediali: MUM. Tuttavia, essendo ancora in fase di sviluppo MUM è carente di alcuni servizi molto importanti per un middleware del suo genere, uno di questi è quello per l'adattamento dei contenuti. Nella prima parte sono state descritte alcune delle caratteristiche più interessanti realizzate dal sistema: il servizio di configurazione e il supporto alla mobilità. Nella seconda parte, il middleware è stato visto nella prospettiva di creare un servizio di adattamento dei contenuti multimediali da integrare in esso, discutendo in modo più dettaglio i servizi esistenti utili alla creazione di un sistema di questo genere cercando di sottolineare eventuali carenze e possibili estensioni. Infine si è cercato di individuare le problematiche principali imputabili alla integrazione del nuovo servizio nel middleware, evidenziando i servizi e componenti sui quali l'architettura di MUM impone di intervenire.

*Capitolo 5*

# ANALISI DEL SISTEMA PER L'ADATTAMENTO DI SERVIZI MULTIMEDIALI

I precedenti capitoli hanno cercato di creare il background di conoscenze necessario per comprendere le problematiche legate allo sviluppo di un sistema di personalizzazione di dati multimediali rispetto ai profili utente che descrivono l'ambiente in cui essi verranno consumati. Questo è infatti è l'obiettivo principale di questa tesi, che vuole integrare nell'architettura di MUM un sistema di questo genere.

Nella costruzione del componente sarà di estrema importanza tener conto che esso sarà parte di un Multimedia Middleware, quindi si cercheranno di rispettare fin

dall'inizio i principi architetturali fondamentali che guidano il loro sviluppo (modularità, flessibilità, scalabilità, ecc.), ricordando soprattutto che il principale obiettivo di ogni componente è quello di fornire servizi di supporto agli sviluppatori di applicazioni multimediali.

Nella prima parte di questo capitolo vengono esposti i requisiti che hanno guidato lo sviluppo del sistema sviluppato, la seconda invece è dedicata alla sua analisi e alla presentazione della sua architettura logica. In questo capitolo non si vuole entrare nei dettagli di progetto, ma si vogliono sottolineare le scelte architetturali che guideranno le fasi successive di progettazione e implementazione, evidenziando le parti in cui è possibile scomporre il sistema in esame e le interazione fra di esse.

La suddivisione della fase progettuale in produzione dell'architettura logica e del progetto concreto è una prassi ben consolidata dell'ingegneria del software, e viene qui adottata perché si ritiene di fondamentale importanza non asservirsi da subito alle tecnologie, per concentrarsi unicamente sulle reciproche interazione tra le parti e per facilitare apertura e reingegnerizzazione futura del prodotto.

## 5.1 Requisiti

È solitamente considerata buona norma nello sviluppo di un progetto software, e di qualunque altro genere di progetto in generale, individuare i suoi requisiti.

Possiamo distinguere i requisiti in funzionali e non funzionali. I primi esprimono le funzionalità che effettivamente il sistema costruito dovrà saper svolgere. I secondi, invece, indicano le linee guida strutturali che devono essere tenute in considerazione durante lo sviluppo dell'applicazione, per ottenere un prodotto di buona qualità, ben costruito e facilmente reingegneralizzabile.

### 5.1.1 Requisiti funzionali

Il sistema che si vuole realizzare, lavorando in cooperazione con gli altri servizi di MUM, deve supportare l'adattamento di presentazioni multimediali sulla base delle informazioni contenute nel profilo dell'utente che ha fatto la richiesta.

Con il termine "presentazione multimediale" si intende un aggregato di uno più oggetti multimediali che a loro volta rappresentano l'astrazione di un certo tipo di dato multimediale non complesso (audio, video, audio-video, ecc.), presente nel sistema. La presentazione multimediale è rappresentata da un metadato, utilizzato per

aggregare i diversi oggetti multimediali, contenente informazioni circa le sue caratteristiche bibliografiche, di formato, temporali e la sua posizione.

Il profilo utente è un contenitore di informazioni che descrivono le caratteristiche del dispositivo utilizzato dall'utente e che il sistema di adattamento considererà nello svolgimento del proprio lavoro. Alcune caratteristiche interessanti sul dispositivo potrebbero essere la dimensione dello schermo, la bit-rate disponibile, il numero di colori che esso è in grado di gestire e così via. Tuttavia, l'architettura del sistema dovrà essere costruita in modo da non essere legata dal punto di vista strutturale alle informazioni contenute all'interno del profilo.

In considerazione di quanto detto nel capitolo 2, quindi, il sistema in questione realizzerà un adattamento orientato al dispositivo e non considererà altre informazioni riguardo l'utente quali le sue preferenze, interessi, ecc. .

Il servizio di adattamento dovrà avere le seguenti responsabilità:

- Scegliere, fra tutte le presentazioni corrispondenti al titolo richiesto che è riuscito a recuperare dal sistema, quelle che considera convenienti da adattare in base a una politica di adattamento predefinita dallo sviluppatore che ha costruito la propria applicazione multimediale al di sopra di MUM.
- Scegliere, sulla base delle condizioni di carico del sistema, il place, o in generale i place, sul quale effettuare l'adattamento.
- Scegliere gli eventuali adattatori da utilizzare per trasformare il flusso multimediale richiesto nel formato considerato più idoneo alle caratteristiche del dispositivo utilizzato dal client.
- Effettuare la trasformazione vera e propria del flusso.

Una politica di adattamento, in base alle considerazioni fatte nel capitolo 1, deve indicare al sistema come comportarsi nel valutare le singole presentazioni. In particolare, definendo dei pesi, evidenzierà se esso dovrà privilegiare la località delle presentazioni, cercando di minimizzare il consumo delle risorse di rete, oppure la loro similarità alla miglior presentazione che può essere resa sul terminale utente, con il fine di minimizzare l'overhead dovuto alla trasformazione del flusso.

Il servizio di reperimento delle presentazioni multimediali presenti nel sistema dovrà considerare due aspetti:

- La possibilità di fornire al sistema di adattamento un set di presentazioni sui cui lavorare sufficientemente ampio, in modo che le sue scelte vengano svolte sulla base del maggior numero di alternative possibili.
- La considerazione dell'importanza delle risorse di rete. L'accesso alle cache dei metadati dell'intero sistema deve essere regolato da politiche anche quando si fanno operazioni di recupero.

Il servizio per la gestione dei profili utenti devrà offrire:

- Una rappresentazione standard dei profili, in modo che questi siano interpretabili in modo univoco da qualunque altro sistema.
- Un database all'interno del quale memorizzare i profili degli utenti e sul quale si possa effettuare operazioni per il loro inserimento, eliminazione e recupero.
- La capacità di tradurre il profilo espresso in formato standard e di estrapolare da esso le informazioni richieste dal sistema di adattamento.

### 5.1.2 Requisiti non funzionali

I principi a cui si cercherà di attenersi il più possibile nello sviluppo del sistema in esame sono i seguenti:

1. ***Modularità:*** Si cercherà di suddividere il sistema in parti, ognuna in grado di svolgere determinate attività, al fine di separare le varie logiche che guidano ognuna di esse. Il notevole vantaggio di un sistema sviluppato secondo questo principio è quello di assicurare incapsulamento, in modo che, se le cose sono state sviluppate nel modo corretto, sarà possibile modificare il sistema semplicemente sostituendo la vecchia versioni di un modulo con la nuova.

2. ***Flessibilità:*** Essendo il sistema in esame parte di un middleware, è molto importante attenersi a questo principio e non effettuare scelte architetturali e di progetto legate a un particolare contesto di utilizzo del sistema, ma invece cercare di risolvere i problemi sempre nel modo più generale possibile.

Ovviamente questo potrebbe richiedere un aumento di complessità considerevole durante le fasi di sviluppo e implementazione del sistema. Ci si riserva, dunque, di discutere di volta in volta la possibilità di effettuare opportune semplificazioni.

3. ***Scalabilità:*** Nell'ottica di inserire il componente che verrà realizzato in un ambiente distribuito, è di fondamentale importanza che il degrado delle sue prestazioni non lo rendano inutilizzabile all'aumentare delle dimensioni del sistema all'interno del quale verrà utilizzato. Tuttavia, solitamente i sistemi multimediali per l'alto fabbisogno di risorse richiesto, non sono altamente scalabili.

## 5.2 Analisi dei requisiti

Leggendo attentamente i requisiti funzionali del sistema che si vuole sviluppare, possiamo distinguere tre attività fondamentali: la gestione dei metadati relativi alle presentazioni multimediali, la gestione dei profili utente ed infine l'adattamento vero e proprio. D'altra parte questa suddivisione concettuale non dovrebbe stupire dato che fin dal primo capitolo sono state individuate queste tre problematiche che sono finora sempre state sviluppate separatamente. Possiamo pensare dunque di creare per ognuna di un esse un opportuno modulo che si occupa della loro realizzazione in modo che, come mostra la Figura **5.1**, il sistema venga suddiviso in tre parti indipendenti che ovviamente interagiscono fra di loro.


**Figura 5.1:** Suddivisione concettuale del sistema in moduli

Il Modulo principale, quello che si occupa dell'adattamento, dovrà svolgere essenzialmente due tipi di attività: di decisione e di codifica. Possiamo scindere dunque questo modulo in due componenti:

– L' ***Adaptation Engine***, rappresentato nello schema di Figura **5.2**, è il componente che si occupa di elaborare le informazioni sui metadati e sui profili, per decidere in base alla politica di adattamento corrente su quali presentazioni operare (eventualmente) l'adattamento. Questo componente corrisponde a quello che fino al capitolo precedente è stato chiamato "configuratore".

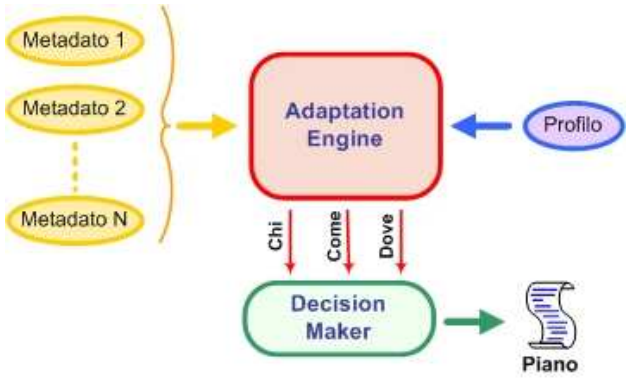



**Figura 5.2:** Ruolo dell'Adaptation Engine nel sistema di adattamento

Esso gioca un ruolo fondamentale all'interno del sistema di personalizzazione poiché, come spiegato in 4.3.1, imporrà al *DecisionMaker* le proprie scelte per la generazione dei piani di configurazione del servizio di erogazione. Le sue scelte dovranno inoltre riguardare anche la scelta degli adattatori da utilizzare per la eventuale trasformazione del flusso e le definizione di dove tale operazione dovrà essere svolta.

– L' ***Adaptor*** è invece il componente che si occupa di effettuare la traduzione di un flusso multimediale da un certo formato ad un altro scelto dall' *Adaption Engine*. Questo componente corrisponde a quello che fino al capitolo precedente è stato chiamato "adattatore".

I due componenti lavorano su livelli diversi, infatti mentre il primo opera su meta-informazioni (profili e metadati), il secondo elabora le informazioni vere e proprie (flussi multimediali).

L'adozione di un approccio modulare come questo permetterà di analizzare, progettare e realizzare i tre moduli presentati separatamente concentrandoci di volta in volta solo sugli aspetti caratteristici del componente considerato. In un approccio di tale genere, però, è di fondamentale importanza definire e tenere d'occhio le

interazioni fra i vari moduli che permetteranno loro di collaborare correttamente al fine di offrire i servizi richiesti.

## 5.3 Analisi

In questa sezione verrà effettuata l'analisi dei vari componenti del sistema e delle loro reciproche relazioni al fine di produrre un'architettura logica robusta che guiderà le successive fasi di progettazione e implementazione del sistema.

Nella modellazione dei moduli verrà utilizzato il linguaggio UML (**U**nified **M**odelling **L**anguage), ampiamente utilizzato per visualizzare costruire e documentare sistemi orientati agli oggetti.

### 5.3.1 Analisi del servizio di recupero dei metadati

Il servizio di recupero dei metadati non deve sostituirsi a quello già esistente, presentato nel capitolo 4, che funziona molto bene quando il recupero è finalizzato alla risoluzione delle politiche di inserimento in cache di nuovi metadati [Borr2004], bensì deve presentarsi come alternativo ad esso.

Si ricorda che l'obiettivo di questa nuova modalità di recupero dei metadati è cercare di offrire al modulo che si occupa dell'adattamento il maggior numero possibile di presentazioni da valutare, senza però consumare troppe risorse di rete con il rischio di provocare il suo congestionamento. Per raggiungere questo obiettivo, la ricerca delle presentazioni non dovrebbe fermarsi sul primo dominio in grado di rispondere alla richiesta dell'utente, ma proseguire su quelli di livello superiore poiché, secondo il modello di caching gerarchico usato da MUM, man mano che si risale la gerarchia la visibilità delle presentazioni da parte della cache aumenta. Il problema fondamentale è dunque quello di capire quando interrompere la ricerca accontentandosi dei risultati ottenuti.

Per risolvere questo problema sono state indagate due soluzioni:

1. La prima prevede che sia l'*Adaptation Engine* a comandare il numero di raffinamenti da fare sul risultato ottenuto dalla prima cache che risponde con un HIT, cioè da quella che per prima trova delle presentazioni corrispondenti a quella richiesta. Con "numero di raffinamenti" si intende il numero di livelli da salire nella gerarchia a partire appunto da questa cache. Questo parametro

può essere espresso all'interno della politica di adattamento poiché effettivamente caratterizza il processo di personalizzazione.

2. La seconda prevede invece che siano le stesse cache a decidere se una richiesta di recupero deve essere inoltrata oppure no a quella di livello superiore. In questa soluzione dunque, la politica di recupero viene distribuita tra vari i nodi del sistema.

La prima soluzione, inizialmente percorsa, è stata poi abbandonata perché permetterebbe a un *Adaptation Engine*, opportunamente configurato mediante le sua politica, di fare richieste che teoricamente potrebbero richiedere sempre l'intervento del nodo radice, comportando notevoli problemi di congestione. Il problema di questa approccio è che le cache non hanno nessun controllo sui dati che gestiscono.

La soluzione scelta è quindi la seconda, che permette di risolvere questo problema lasciando ad ogni nodo la responsabilità di decidere l'interruzione della propagazione della richiesta. Questo approccio, inoltre, permette di differenziare il comportamento delle varie cache nelle diverse località. Infatti, come mostra la Figura **5.2**, dato che all'aumentare del proprio livello all'interno della gerarchia la visibilità delle presentazioni da parte di una cache aumenta, ha senso dire che anche la ricchezza informativa delle risposte che essa fornisce aumenta e quindi, in generale, ha senso adottare per le cache politiche sempre più stringenti man mano che il loro livello gerarchico sale.

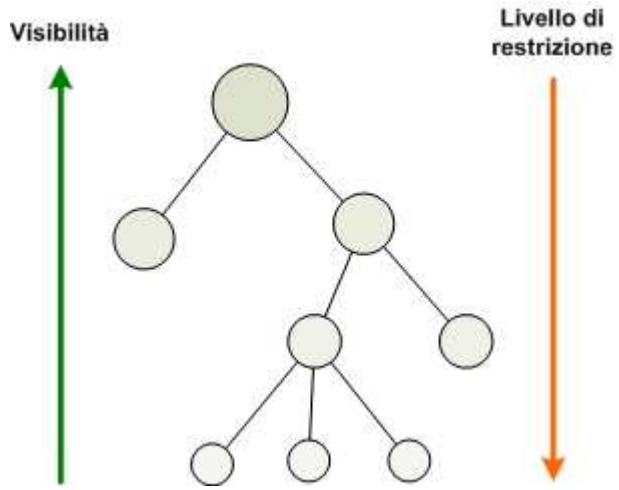



**Figura 5.2:** Visibilità dei metadati e livello di restrizione delle politiche per il recupero

La politica di ogni nodo dovrà contenere delle indicazioni che permetteranno alla cache di decidere a tempo di esecuzione se propagare una certa richiesta di recupero dei metadati. A questo proposito ogni politica definirà al proprio interno un attributo intero positivo che rappresenta il numero di raffinamenti precedentemente eseguiti su una richiesta sufficiente perché essa non venga inoltrata al nodo di livello superiore. Consideriamo ad esempio la politica rappresentata in Figura **5.3**: La richiesta viene fatta sul nodo **c**, il quale non trovando nelle propria cache una presentazione corrispondente al titolo richiesto, inoltra la richiesta alla cache **b** a prescindere dalla propria politica. Supponiamo che sulla cache di **b** esiste una presentazione compatibile con la richiesta: la politica di **b** non permetterebbe di inoltrare una richiesta raffinata già una volta, ma dato che il numero di raffinamenti fatti sul risultato corrente è zero (b è la prima cache ad aver trovato risultati pertinenti), la richiesta viene inoltrata al nodo **a** che, evidentemente scegliendo di non caricare il nodo di radice (valore 0 all'interno della politica), non inoltrerà mai nessuna richiesta al nodo radice avviando la consegna verso il cliente del risultato definitivo, composto dai metadati presenti nella propria cache corrispondenti al titolo richiesto.

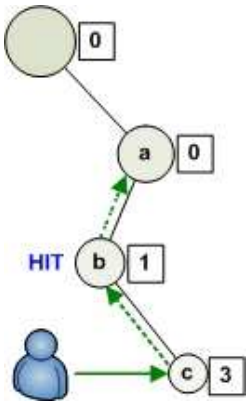



**Figura 5.3:** Esempio di una plausibile politica per il recupero dei metadati.

In accordo alle osservazioni fatte in precedenza riguardo la dipendenza del livello di restrizione delle politiche dal grado gerarchico della cache, i valori interi che definiscono il comportamento della cache andranno via via diminuendo man mano che ci si avvicina al nodo radice.

Il servizio di gestione dei metadati di cui dispone MUM, già offre dei meccanismi per la gestione delle politiche, verranno utilizzati dunque tali meccanismi per l'inserimento, l'eliminazione e il recupero delle politiche utilizzate.

### 5.3.2 Analisi del servizio di gestione dei profili utente

Gli aspetti che il servizio di gestione dei profili deve affrontare sono la scelta di uno standard per la rappresentazione dei profili utente e la realizzazione di meccanismi per la loro gestione, intesa come inserimento, eliminazione e recupero da un database utilizzato per la loro memorizzazione.

#### 5.3.2.1 Rappresentazione dei profili utente

La scelta di uno standard di rappresentazione è molto importante al fine di rendere le informazioni contenute nei profili significative ed in grado di essere interpretate in modo corretto ed uniforme da un servizio di adattamento. Nel capitolo 3 sono stati presentati alcune delle tecnologie attualmente disponibili per la rappresentazione dei profili, in particolare si è parlato di **CC/PP** che offre una visione del profilo molto flessibile basata su una sua composizione in componenti e attributi. Proprio per questa sua caratteristica si è scelto di utilizzare CC/PP per le rappresentazione dei profili e poiché essi dovranno contenere informazioni sui dispositivi utilizzati da clienti per accedere ai servizi, è stato utilizzato **UAProf** come vocabolario di riferimento. Tuttavia, la scelta di CCPP, proprio grazie alla sua flessibilità, non preclude un eventuale estensione del profilo utente con informazioni che riguardano le sue preferenze e/o i suoi interessi.

```
<rdf:Description ID="ScreenSize">
    <rdf:type rdf:resource="http://www.w3.org/TR/PR-rdf-schema#
       Property"/>
    <rdfs:domain rdf:resource="#HardwarePlatform"/>
    <rdfs:comment>
      Description:  The size of the device's screen in units of
                    pixels, composed of the screen width and the
                    screen height.
      Type:         Dimension
      Resolution:   Locked
      Examples:     "160x160", "640x480"
</rdfs:comment>
```

**Figura 5.4:** Attributo UAProf per la rappresentazione della dimensione dello schermo

Il profilo utente, stando alle specifiche, deve in generale fornire informazioni riguardo a N parametri che caratterizzano il dispositivo utilizzato dall'utente. Il problema fondamentale dunque è quello di trovare un vocabolario capace di esprimere e rappresentare le informazioni del profilo utente.

Valutando l'espressività di UAProf, come rilevato anche in precedenza in 3.1.2, si può notare che esso offre una descrizione abbastanza dettagliata della piattaforma hardware e software del dispositivo (ad esempio in Figura **5.4** viene mostrata la definizone UAProf di un attributo che descrive le dimensioni dello schermo), ma purtroppo non fa altro per la rappresentazione delle sue proprietà di comunicazione.

In 3.1.2 sono stati indicati due approcci per poter risolvere problemi di espressività di questo tipo:

1. L'utilizzo o la definizione di estensioni del vocabolario;
2. L'integrazione di profili di varia natura;

Si ritiene che in generale per un servizio middleware la miglior soluzione sia la seconda, perché garantisce più flessibilità, tuttavia la complessità che introduce hanno spinto per il momento a optare per la prima soluzione. Infatti, utilizzando l'estensione all'UAProf proposta da INTEL si riesce a risolvere il problema in modo efficace ed efficiente, però è probabile che possibili estensioni future del profilo utente (ad esempio che considerino anche informazioni sulle sue preferenze) non siano più rappresentabili per mezzo di questi vocabolari.

#### 5.3.2.2 Gestione dei profili

Il servizio di gestione dei profili dovrà essere basato su quello esistente, introducendo in esso le funzionalità necessarie per gestire la rappresentazione dei profili secondo lo standard CC/PP. In particolare sarà necessario prevedere l'integrazione di una funzionalità capace di tradurre la rappresentazione del profilo, eventualmente estraendo da esso le informazioni ritenuti essenziali. Considerando i problemi legati alla rappresentazione del profilo, di cui si parlava nel paragrafo precedente, questo processo di interpretazione deve essere legato il meno possibile al particolare vocabolario utilizzato.

In Figura **5.5**, viene utilizzato un diagramma a blocchi per dare un prima rappresentazione dell'architettura che questo servizio dovrà assumere.

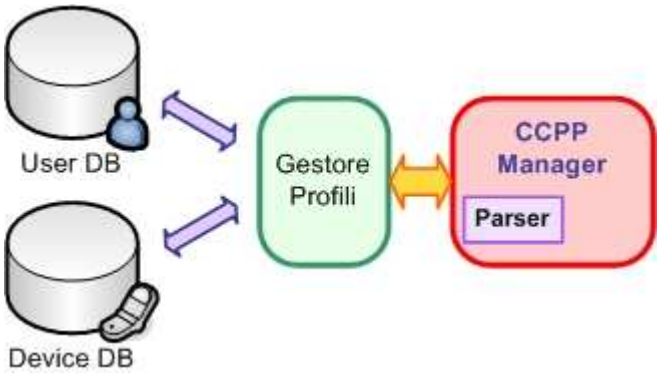



**Figura 5.5:** Architettura del servizio di gestione dei profili utente

Il servizio di gestione dei profili esistente, come spiegato in 4.2.2, divide concettualmente le informazioni relative alle piattaforme da quelle relative agli utenti, memorizzandole in database differenti entrambi gestiti da un entità dedicata che si occupa dell'inserimento, del recupero e dell'eliminazione dei profili. Il CCPPManager si appoggerà dunque a questa entità per realizzare queste funzionalità di base, inoltre dovrà supportare un servizio per l'interpretazione dei profili espressi nella sintassi CC/PP.

### 5.3.3 Il servizio di configurazione del processo di adattamento

Nell'analisi dei requisiti sono stati individuati i tre moduli principali in cui può essere scomposto il sistema. Tuttavia, quello principale è sicuramente quello che verrà trattato in questo paragrafo. Esso comanda gli altri due, sfruttando i servizi da loro offerti per avere delle rappresentazioni del contesto utente e delle presentazioni disponibili sul quale poter lavorare.

Analizzando questo componente si è scelto di dividere la logica che guida l'adattamento vero e proprio da quella che guida le interazioni con i moduli precedentemente esposti, in modo che eventuali sviluppi futuri su uno dei due servizi di supporto o sul modo in cui essi interagiscono con il sottosistema di adattamento siano facilmente integrabili senza causare la sconvolgimento dell'architettura interna

del componente in esame. La Figura **5.6** mostra schematicamente quanto appena detto.

Vengono dunque introdotti nell'architettura dell'*Adaptation Engine* due componenti:

1. Il ***ProfileAnalyer*** ha il compito di interagire con il servizio di gestione dei profili, quindi con il CCPPManager che funge da entry-point verso tutti i servizi di questo tipo. In particolare esso rappresenta lo strumento utilizzato dall'Adaptation Engine per recuperare i profili ed estrapolare da essi specifiche informazioni tipo la dimensione dello schermo e l'ampiezza di banda a disposizione del dispositivo considerato.

2. Il ***MetadataAnalyzer*** rappresenta il corrispettivo del ProfileAnalyzer dal lato del modulo di gestione dei metadati. La sua attività principale è quella di recuperare le presentazioni corrispondenti a un certo titolo disponibili nel sistema. Potrebbe inoltre effettuare sul risultato di ricerca delle operazioni di filtraggio, ad esempio per limitare il numero di presentazioni da fornire all'*Adaptation Engine* al fine di rendere il processo più efficiente.

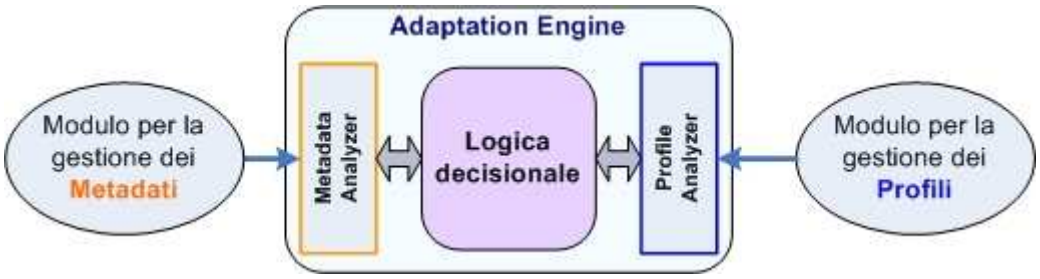



**Figura 5.6:** Separazione della logica decisionale dalla logica di interazione con gli altri moduli

La logica decisionale permette all'*Adaptation Engine* di scegliere:

- Quale presentazioni adattare;
- Quale/i Adaptor utilizzare per compiere la trasformazione;
- Dove effettuare il processo di codifica

Esso è stato quindi strutturato, come mostra la Figura **5.7**, in tre componenti, ognuno capace di prendere un certo tipo di decisione, essi sono rispettivamente l'

***AdaptationSubjectsSelector***, l'***AdaptorsSelector*** e l'***AdaptationPlaceSearcher***. Si noti che i primi due in generale, selezionano rispettivamente un insieme di presentazioni e un insieme di adattatori. Infatti queste informazioni serviranno per la costruzione di un piano di inizializzazione che è composto da più soluzioni di diversa qualità ordinate dalla migliore alla peggiore. Per ognuna di queste soluzioni l'AdaptationPlaceSearcher individua i place in cui gli adattatori dovranno essere scaricati. La decisione da esso presa potrebbe coinvolgere un solo place, ma in generale potrebbe essere privilegiata la distribuzione del carico di adattamento individuando più place su cui attuare le trasformazioni.

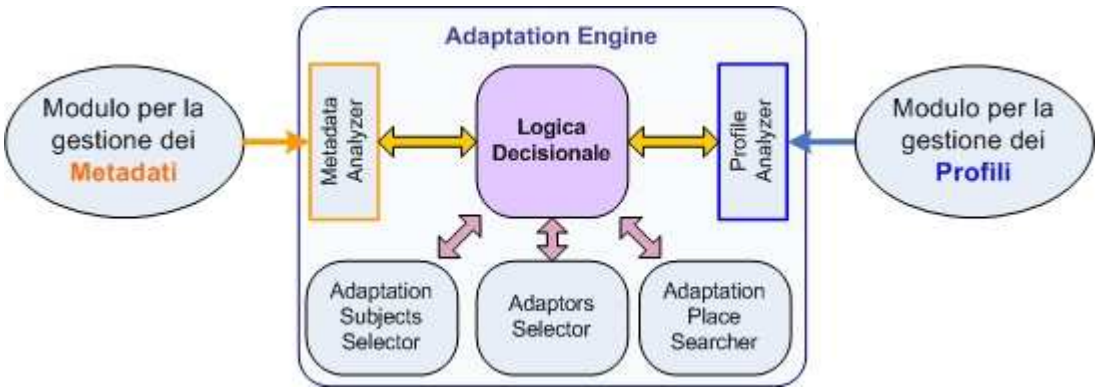



**Figura 5.7:** Struttura dell'Adaptation Engine

#### 5.3.3.1 Strategia di Adattamento

In considerazione delle ricerche effettuate e presentate nel capitolo 2, si è scelto di adottare una strategia di adattamento completamente dinamica perché il servizio che si sta sviluppando verrà integrato in una piattaforma software che, come spiegato nel capitolo 4, offre anche supporto alla mobilità degli utenti da un dispositivo ad un altro. Quindi nell'ottica di fornire un servizio che funzioni anche durante lo spostamento della sessione utente su un nuovo terminale, è necessario per l'adattamento utilizzare una strategia dinamica.

Scegliere l'approccio dinamico comporta per il servizio di configurazione dell'adattamento compiere una grossa mole di valutazioni a run-time per poter effettuare le proprie scelte. Ad esempio, per capire quale tra le presentazioni disponibili scegliere, ad ogni richiesta, esso dovrà valutare per ognuna di esse:

- il costo di adattamento, per trasformarla nel formato compatibile con il profilo dell'utente.

– la distanza della presentazione dal cliente.

Dato che queste operazioni dovrebbero essere eseguite a run-time, potrebbero introdurre tempi di latenza tali da compromettere la qualità di servizio percepita dall'utente. Inoltre, il sistema di adattamento dovrebbe essere in grado di gestire una moltitudine di profili e metadati che renderebbero la sua realizzazione molto complessa dal punto di vista ingegneristico. Per questi motivi, in questa tesi è stato introdotto un altro approccio che mira a rendere l'intero processo di configurazione dell'adattamento più efficiente e ingegneristicamente realizzabile, discretizzando le possibili descrizioni degli utenti e delle rappresentazioni su cui il sistema è capace di operare.

D'altra parte questo è la via intrapresa anche dal mondo industriale come dimostrano le specifiche del consorzio ISMA (Internet Streaming Media Alliance) che definiscono per i dati multimediali di tipo video ed audio due livelli di qualità descritti da due "profili": il *"Profile 0"* per la rappresentazione di un livello base ed il *"Profile 1"* per la rappresentazione di uno avanzato. Per ulteriori approfondimenti sulle specifiche ISMA si rimanda a [ISMASpec].

L'approccio utilizzato segue la direzione intrapresa da ISMA, infatti, prevede di fissare N rappresentazioni predefinite di "qualità di presentazione" dette *InternalProfile* che contengono esclusivamente informazioni utili ai fini dell'adattamento. Un *InternalProfile* può descrivere sia la qualità di un metadato, in termini ad esempio di frame-size della presentazione che rappresenta e bit-rate necessaria per la sua trasmissione, sia quella di un profilo utente in termini di qualità di presentazione che riesce a sostenere.

Da un punto di vista geometrico ognuna delle M caratteristiche utilizzate per la descrizione della qualità di una presentazione può essere rappresentata da una coordinata di uno spazio a M dimensioni. Quindi un generico punto di questo spazio definisce la generica qualità di un presentazione. Da qui possiamo ben capire che la costruzione di un sistema di adattamento capace di trasformare un qualunque formato della presentazione in un altro sia notevolmente complessa da realizzare.

L'introduzione degli InternalProfile, permette di individuare nel piano M-dimensionale, N punti che rappresentano i livelli di qualità su cui il sistema è in grado di operare. Il sistema di adattamento quindi prima mettere in azione i propri

componenti decisionali dovrà riconoscere il livello di qualità descritto dal profilo utente e dai metadati corrispondenti a una presentazione del contenuto richiesto. La Figura **5.8**, mostra quanto appena spiegato a parole nel caso in cui le caratteristiche di qualità considerate sono due (per semplicità di rappresentazione).

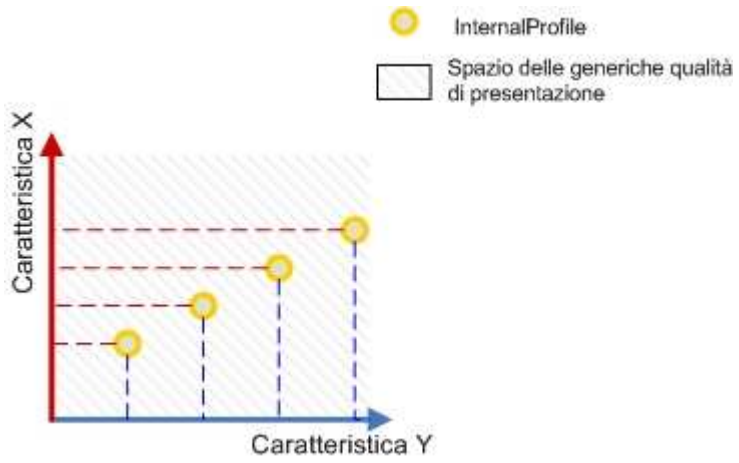



**Figura 5.8:** Discretizzazione della qualità di una presentazione mediante gli InternalProfile

Dopo aver fissato in fase di preparazione del sistema questo set di profili, possono essere valutate *off-line* le relazioni di similarità esistenti fra loro, al fine di individuare il costo di adattamento per le reciproche trasformazioni tra gli *InternalProfile*. Una volta calcolate queste informazioni possono essere quindi memorizzate in apposite strutture studiate per rendere il più efficiente possibile il loro recupero.

Questo approccio permette dunque di sollevare l'*Adaptation Engine* dal dover effettuare ad ogni richiesta tutte le valutazioni per stabilire i costi di adattamento delle presentazioni disponibili. Esso potrà infatti sfruttare a questo scopo la struttura in cui queste informazioni preventivamente calcolate sono state memorizzate ed inoltre dovrà continuare a occuparsi della valutazione della località della presentazione relativamente alla posizione del cliente (informazione che non è conveniente da pre-calcolare).

Tuttavia, la presenza di *InternalProfile* predefiniti comporta la necessità di introdurre dei meccanismi per la loro gestione e lo sviluppo di tecniche di riconoscimento che permettano di mappare un profilo CC/PP e un metadato in uno di essi.

Sono stati individuati quindi nuove funzionalità che il sistema deve realizzare per offrire i servizi richiesti, esse sono:

- La gestione degli *InternalProfile* e delle relazioni che tra essi intercorrono;
- Il riconoscimento di profili e metadati.

Affinchè l'approccio di adattamento presentato possa funzionare è necessario definire delle logiche per valutare le relazioni tra:

- *InternalProfile* e Metadati;
- *InternalProfile* e Profili utente;
- ogni *InternalProfile* e gli altri;

Le prime due sono necessarie per la realizzazione del servizio di riconoscimento, l'altra sarà fondamentale per l'individuazione delle presentazioni dal minor costo di adattamento. La relazione considerata per fare queste valutazioni è quella di similarità. L'esposizione del modo in cui questa viene valutata viene rimandata al capitolo successivo.

In Figura **5.9** si presenta una prima architettura del sistema di adattamento, che tuttavia non può essere ancora considerata definitiva poiché non ancora considera i meccanismi utilizzati dal sistema per la selezione degli adattatori e dei place in cui effettuare il processo di trasformazione che saranno analizzati nei prossimi paragrafi.

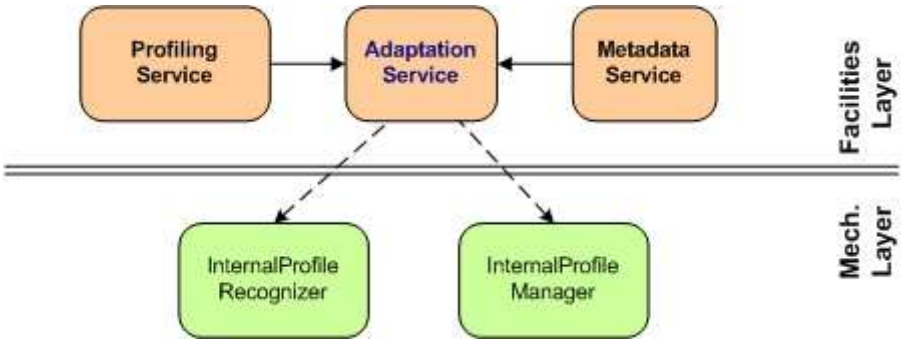



**Figura 5.9:** Prima architettura logica del servizio di adattamento.

### 5.3.3.2 Scelta delle presentazioni da adattare

Il componente denominato *AdaptationSubjectsSelector* è colui che ha la responsabilità di scegliere tra le presentazioni disponibili nel sistema, i cui metadati sono stati recuperati dal *MetadataAnalyzer*, quelle da inserire nel piano di configurazione prodotto dal *DecisionMaker*. Il componente oltre a svolgere una attività di selezione delle presentazioni ne effettua anche una di ordinamento. Infatti,

essendo il piano di configurazione composto da più soluzioni, si ritiene sensato cercare di integrare in esso tutte le possibilità disponibili ordinandole dalla migliore alla peggiore.

Una responsabilità fondamentale dell' *AdaptationSubjectSelector* è la valutazione della politica di adattamento proprio al fine di stabilire un ordinamento delle presentazioni considerando, sia la similarità tra l'InternalProfile rappresentante il profilo utente e quello rappresentate il metadato della presentazione, sia la distanza della presentazione stessa dal cliente.

Valutare una politica di adattamento significa considerare i pesi, contenuti nella politica, da dare alle informazioni di similarità e a quelle di località. A questo proposito ad ogni presentazione verranno assegnati dei punteggi per misurare quantitativamente queste caratteristiche e vengono utilizzati i pesi specificati dalla politica per calcolare la loro media pesata e ottenere un punteggio complessivo, utilizzato per ordinare le presentazioni.

Lo schema in Figura **5.10** mette in evidenza la stretta collaborazione dell'AdaptationSubjectsSelector con il servizio di riconoscimento, infatti, operando sugli InternalProfile, esso ha la necessità di riconoscere il profilo dell'utente e i metadati corrispondenti alle presentazioni da lui richieste.

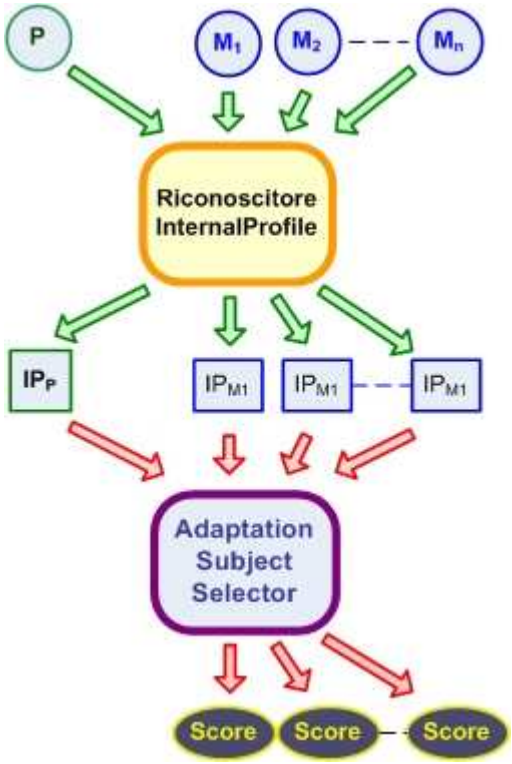


**Figura 5.10:** Modello UML dell'AdaptationSubjectSelector

#### 5.3.3.3 Scelta degli adattatori da utilizzare

Il compito di un adattatore è quello di trasformare un flusso di informazioni da un certo formato ad un altro. Da un punto di vista architetturale, le soluzioni percorribili, rappresentate in Figura **5.11** sono le seguenti:

1. Costruire un adattatore capace di effettuare una qualsiasi trasformazione.
2. Considerare l'adattatore come entità in grado di effettuare tutte le trasformazioni necessarie per passare da un certo InternalProfile ad un altro.
3. Costruire un'adattatore per ogni caratteristica considerata nella rappresentazione del livello di qualità di una presentazione (per ogni attributo dell'InternalProfile).

La prima soluzione è stata scartata subito poiché la complessità di un adattatore in grado di svolgere qualunque trasformazione la rende molto difficile da realizzare. Per risolvere questo problema si potrebbe pensare di adottare la seconda soluzione che sfrutta l'idea utilizzata nella definzione degli InternalProfile, predisponendo degli adattatori capaci di effettuare specifiche trasformazioni che permettono di passare da un InternalProfile all'altro. Tuttavia, se gli InternalProfile tra di loro compatibili all'interno di un sistema sono N, secondo questa soluzione sarebbe necessario definire $N \times (N-1)$ adattatori. Infine l'ultima soluzione prevede la realizzazione di adattatori in grado di svolgere solo "trasformazioni elementari", cioè trasformazioni che coinvolgono solo un attributo dell'InternalProfile. Si ritiene questa soluzione molto valida per i seguenti motivi:

1. La complessità di realizzazione degli adattatori è molto ridotta;
2. Il numero di adattatori da definire non dipende quadraticamente dal numero di InternalProfile definiti all'interno del sistema come nel caso della soluzione precedente, ma solo linearmente dal numero di attributi che li caratterizzano.
3. La modularità della soluzione permette di costruire adattatori complessi componendo gli adattatori in catene che possono essere allocate su diversi proxy del Service Path al fine di distribuire il carico computazionale del processo di trasformazione.

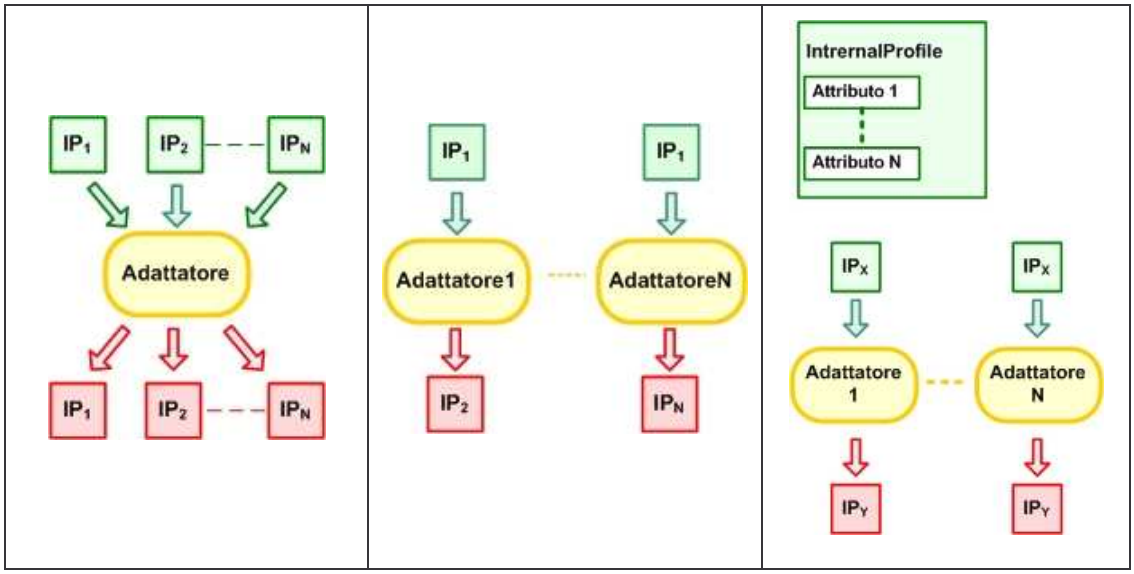



**Figura 5.11:** Possibili soluzioni architetturali dell'adattatore

Tuttavia, la scomposizione di un adattatore in parti potrebbe risultare costosa perché moltiplica i tempi necessari all'inizializzazione e alla configurazione degli adattatori. Nonostante questo difetto si è deciso di adottare quest'utlima soluzione perché introduce nell'architettura la possibilità di comporre gli adattatori che in evoluzioni future del sistema potrebbero esser capaci di compiere anche trasformazioni "meno" elementari riducendo il problema dei costi.

Il sistema di adattamento, quindi, dispone di **X** adattatori, ognuno in grado di operare su una certa caratteristica di qualità (ad esempio il FrameSize) in modo da ottenere un formato della presentazione con determinate caratteristiche. In generale, un adattatore può essere in grado di generare un certo formato, rappresentato da un InternalProfile, a partire da diversi livelli di qualità di un flusso. É possibile dunque che sia capace di effettuare più di una trasformazione.

Il componente denominato *AdaptorsSelector* individua per ogni presentazione multimediale, scelta dall' *AdaptationSubjectsSelector,* l'insieme degli adattatori da utilizzare per effettuare la trasformazione del flusso nel formato rappresentato dall'InternalProfile associato al terminale utente.

L'*AdaptorSelector* per raggiungere il suo obiettivo deve:

- Analizzare gli InternalProfile che caratterizzano il flusso di ingresso e di uscita, allo scopo di individuare le caratteristiche di qualità che necessitano di adattamento;
- Scegliere per ognuna di esse l'adattatore in grado di compiere la trasformazione.

Un ruolo molto importante nella scelta degli adattatori riveste la descrizione delle loro capacità in termini di trasformazioni elementari che essi sono in grado di attuare. Quindi per avere la possibilità di compiere anche trasformazioni complesse, cioè che coinvolgono più caratteristiche di qualità del flusso, il componente deve essere in grado di selezionare più adattatori, uno per ognuna di esse.

Le descrizione delle capacità di un certo adattatore viene affidata ad un apposito descrittore detto *AdaptorDescriptor* a cui esso è associato univocamente. Questo descrittore contiene tutte le informazioni riguardo alle trasformazioni che esso è in grado di effettuare e rappresenta uno strumento molto utile nella ricerca svolta dall'*AdaptorSelector*. Le gestione degli *AdaptorDescriptor* sarà affidata a un apposito *AdaptorDescriptorManager* che verrà presentato nei paragrafi successivi.

Lo schema in Figura **5.12** rappresenta l'architettura del sistema di selezione degli adattatori.

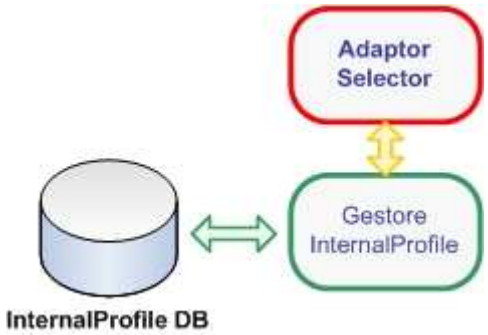



**Figura 5.12:** Diagramma UML del sistema di recupero degli adattatori

Una volta individuato il nome dell'adattatore da utilizzare per trasformare il flusso rappresentato da un certo InternalProfile in un formato espresso da un altro InternalProfile, si potrebbe memorizzare questa informazione nella struttura che mantiene le relazioni di similarità tra i vari InternalProfile in modo che la prossima

volta che viene richiesta quel tipo di trasformazione non ci sia bisogno di interpellare l'*AdaptorSelector*.

#### 5.3.3.4 Scelta del place in cui svolgere l'adattamento

Il componente dedicato alla scelta del place in cui saranno effettuate le operazioni di codifica è chiamato *AdaptationPlaceSearcher*. L'obbiettivo della sua ricerca è quello di trovare all'interno del Service Path il nodo più scarico a cui poter attribuire l'onere di attuare le trasformazioni sul flusso. Questa ricerca è basata sulle risposte date da un servizio che si dovrà occupare del monitoraggio delle risorse di rete in una certa località. Solo un dispositivo di tale genere infatti potrebbe riuscire ad individuare il place più scarico in un certo istante.

Il sistema potrebbe essere costruito in modo da considerare una possibile distribuzione del carico di adattamento sui nodi del ServicePath, individuando anche più di un place proxy sul quale scaricare gli adattatori coinvolti nel processo di codifica. In questo caso si dovrebbero specificare il gruppo di adattatori che oguno di essi dovrebbe ospitare, tenendo ad esempio in considerazione informazioni sull'eventuale loro presenza sui proxy. Inoltre, il servizio che si occupa di trovare il place più scarico, rappresentato dal *NetworkAnalyzer*, per il momento selezionerà sempre il place di default immediatamente superiore (il padre) a quello su cui è stata fatta la richiesta. Lo schema in Figura **5.13** rappresenta il modello del sistema per la ricerca del nodo in cui effettuare l'adattamento.


**Figura 5.13:** Sistema di selezione del nodo per l'adattamento.

#### 5.3.3.5 Servizio per il riconoscimento di Profili e Metadati

L'obbiettivo del servizio di riconoscimento è quello di mappare il profilo di un utente o il metadato di una presentazione in un InternalProfile, ovvero determinare il

livello di qualità sostenibile dal terminale utente o quello di una presentazione multimediale rispetto alle proprietà rappresentate in un InternalProfile.

Come detto nei paragrafi precedenti la logica per effettuare queste operazioni è contenuta all'interno degli stessi InternalProfile, il servizio di riconoscimento dunque, come mostra anche lo schema in Figura **5.14**. lavora in stretta collaborazione con quello che si occupa della loro gestione. Utilizzando questa logica il servizio in oggetto riesce a stilare una sorta di classifica degli InternalProfile contenuti nel database del sistema assegnando ad ognuno di essi un punteggio che indica quanto siano simili al profilo o metadato preso in considerazione. Quest'ultimo sarà ovviamente rappresentato dall'InternalProfile con il punteggio più alto.

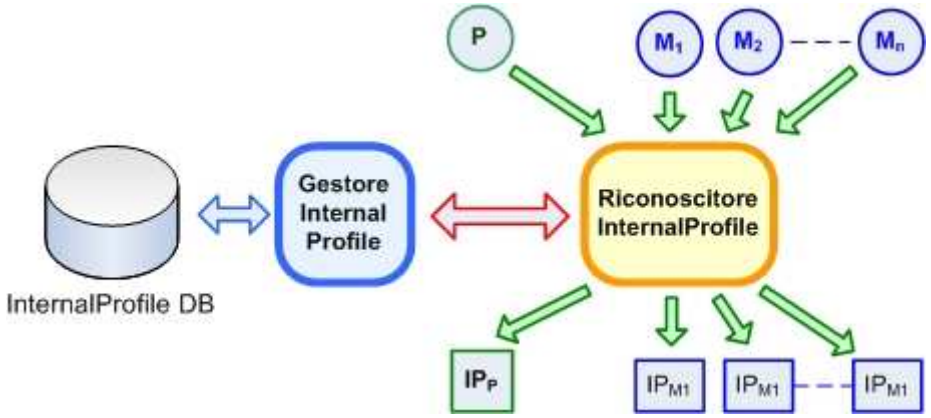



**Figura 5.14:** Schema del servizio di riconoscimento

Il problema del riconoscimento può essere visto come un problema di mapping: ad esempio volendo considerare per la definizione della qualità di una presentazione i due parametri precedentemente scelti: frame-size e bit-rate, questi potrebbero essere visti come dimensioni di uno spazio bidimensionale all'interno del quale la qualità di ogni presentazione potrebbe essere rappresentata con un punto. In quest'ottica, definire degli InternalProfile significa fissare dei punti in questo spazio e riconoscere un profilo/metadato consiste nel mappare un qualsiasi punto dello spazio in uno di questi punti fissati. Questa operazione di mapping deve però rispettare due vincoli fondamentali:

**1)** Ad un profilo utente non può essere riconosciuta la capacità di sostenere una presentazione che qualitativamente è al di sopra delle proprie possibilità (Figura **5.15a**);

**2)** La qualità di una presentazione multimediale non può essere sottostimata (Figura **5.15b**).

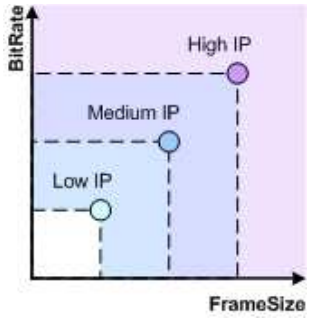



**Figura 5.15a:** Riconoscimento di un profilo

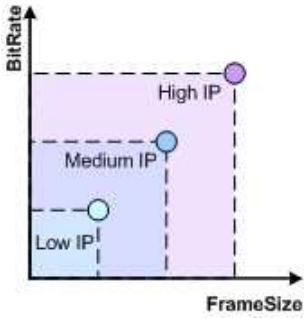



**Figura 5.15b:** Riconoscimento di un metadato

Il primo vincolo garantisce l'impossibilità di sopravvalutare un dispositivo orientando l'adattamento su capacità che invece in realtà non ha. Il secondo permette di evitare la consegna di un presentazione non adattata perchè apparentemente dello stesso livello di qualità del dispositivo.

Queste osservazioni vengono prese in considerazione nell'assegnamento dei punteggi di similarità agli InternalProfile, quindi all'interno dello loro funzioni di valutazione.

Il riconoscimento introduce inevitabilmente dell'overhead dovuto alla valutazione della similarità di tutti gli InternalProfile con il profilo utente o il metadato considerato. Questo suggerirebbe di limitare il numero di InternalProfile, ma se il passo di discretizzazione della qualità non è sufficientemente fine si rischia di perdere notevole informazione sui profili e sui metadati. Tuttavia si pensa che scegliendo degli opportuni livelli di qualità, ad esempio tre livelli (basso, medio, alto) per ogni tipo di terminale utilizzabile nell'accesso al sistema (smartphone, PDA, Personal Computer), si possano trovare delle buone soluzioni di compromesso.

#### 5.3.3.6 Servizio per la gestione degli InternalProfile

Il servizio oggetto di questo paragrafo deve occuparsi della modellazione del InternalProfile cercando di non legarlo agli specifici attributi che lo dovranno caratterizzare. Si vuole quindi realizzare una struttura dinamica in modo che eventuali estensioni sul tipo di caratteristiche che essa dovrà descrivere non

comportino sconvolgimenti sulla sua architettura. Inoltre, dovrà utilizzare un database in cui memorizzare gli InternalProfile definiti in fase di preparazione del sistema ed unitamente ad esso sarà utile la realizzazione di un suo Manager per gestire il loro inserimento, recupero ed eliminazione. Infine, è necessario prevedere la realizzazione di un struttura molto sintetica in grado di mantenere le relazione fra tutti gli InternalProfile contenuti nel database.

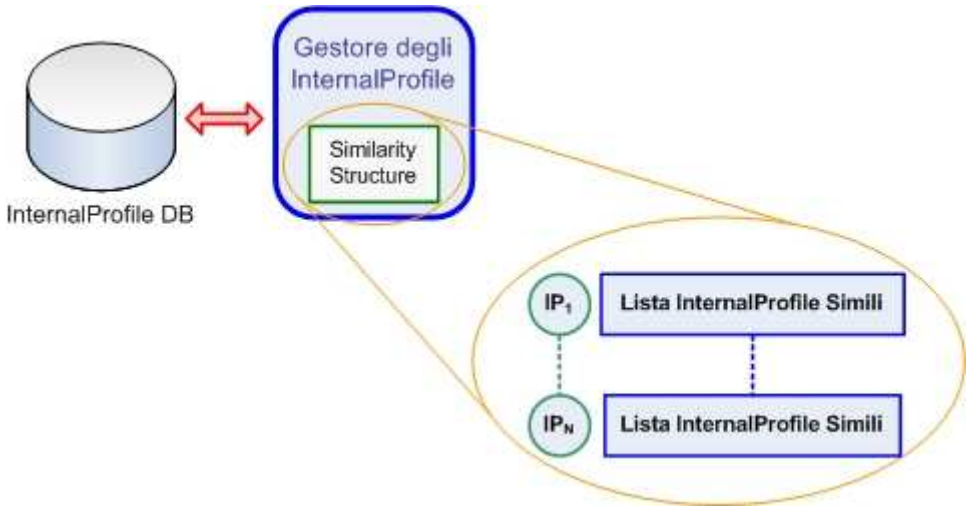



**Figura 5.16:** Modello di gestione degli InternalProfile

L'introduzione di InternalProfile predefiniti fornisce la possibilità di precalcolare off-line le relazioni di similarità che li legano in modo da rendere più efficiente la ricerca delle presentazioni compatibili con un certo InternalProfile, corrispondente al profilo dell'utente.

La Figura **5.16** mostra una schema del sistema di gestione degli InternalProfile, evidenziando l'esistenza di un struttura dati che ha il compito di mantenere le informazioni riguardo le relazioni di similarità che legano i profili contenuti nel database. Questa struttura viene aggiornata automaticamente all'inserimento o all'eliminazione di un InternalProfile nel database. Come mostra la stessa figura, essa sarà organizzata in modo che dato un InternalProfile sia immediato recuperare il tutti gli InternalProfile simili ad esso.

Infine, affinché sia sintetica ed accessibile in modo efficiente, conterrà solo le informazioni necessarie a individuare e definire una relazione tra InternalProfile, caratterizzata da:

- i nomi delle due entità partecipanti;
- il verso della relazione che indica quale entità può essere trasformata nell'altra;
- il punteggio di similarità che quantifica quanto le due entità sono simili;

Infine, si noti che nella relazione si potrebbe anche indicare il nome dell'adattatore da utilizzare per effettuare la trasformazione, ovviamente dopo che esso sia stato individuato.

#### 5.3.3.7 Servizio di gestione dei descrittori degli adattatori

Al fine di descrivere le capacità di ogni adattatore viene definito per ognuno di essi un descrittore, detto *AdaptorDescriptor,* che viene utilizzato per rappresentare le trasformazioni elementari che esso è in grado di attuare.

Una trasformazione elementare è caratterizzata da tre elementi: un riferimento agli InternalProfile, che descrivono le caratteristiche di qualità del flusso in ingresso ed in uscita, ed il nome dell'attributo sul quale viene effettuata la trasformazione.

Come mostra la Figura **5.17**, i descrittori vengono memorizzati in un apposito database sul quale viene costruito un gestore che permette il loro inserimento, recupero ed eliminazione. L' *AdaptorsSelector* interagendo con questo gestore riesce ad individuare il nome dell'*Adaptor* in grado di svolgere una certa trasformazione.

Si noti che nel database viene memorizzato solo il descrittore e non l'intero adattatore. Dal descrittore infatti si può risalire al nome univoco dell'Adaptor, necessario per poter interrogare il SoftwareRepository di MUM in cui risiede il suo codice.

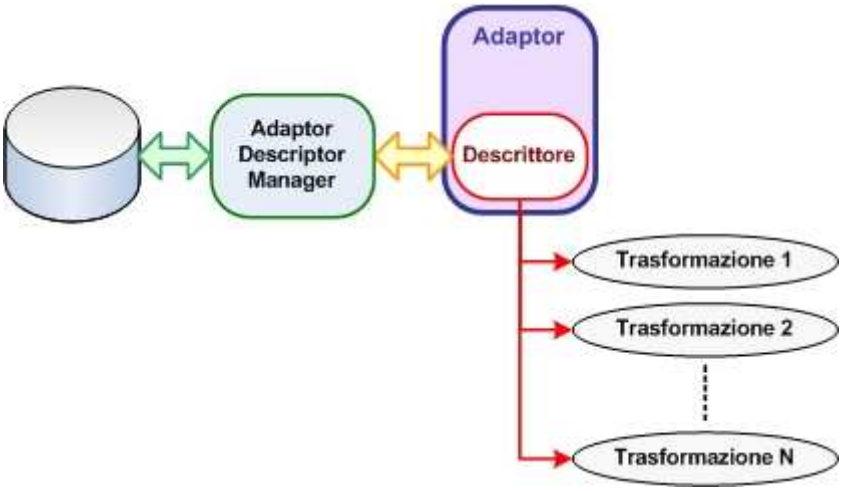



**Figura 5.17:** Modello di gestione dei descrittori degli adattatori

### 5.3.4 Analisi del sistema per la codifica di flussi multimediali

Nel capitolo 1 sono stati individuati due approcci organizzativi per lo svolgimento delle operazioni di trasformazione su un flusso di informazione, il primo di tipo centralizzato, il secondo di tipo distribuito. Si ritiene che quest'ultimo modello sia molto valido perché permette una miglior gestione del carico rispetto al primo, per questo motivo si cercherà di studiare un'architettura che preveda il suo utilizzo. Tuttavia, inizialmente, si è scelto di focalizzare l'attenzione più sulle problematiche di trasformazione del flusso che su questi aspetti, che quindi, seppur molto importanti, non verranno considerati nello sviluppo del primo prototipo.

L'introduzione degli **N** InternalProfile comporta notevoli vantaggi ai fini della realizzazione di un sistema di codifica dei flussi. Infatti, permette di conoscere a priori le possibili trasformazioni che ad esso possono essere richieste, limitando fortemente la complessità dell'entità *Adaptor* che si dovrà occupare della codifica.

Nel sistema, infatti, vengono definiti **M** adattatori ognuno in grado di operare delle trasformazioni sul flusso multimediale per renderlo in un formato caratterizzato da uno degli N livelli di qualità. Gli *Adaptor*, scelti dal servizio di configurazione dell'adattamento, durante l'inizializzazione del Service Path, verranno scaricati sul place proxy indicato come responsabile del processo di codifica e collegati fra di loro nell'ordine indicato dall'*AdaptorsSelecto*r a formare una catena di adattamento le cui estremità saranno a loro volta collegate alle altre entità utilizzate dal proxy per ricevere e spedire i flussi multimediali.

L'architettura del Proxy di MUM è organizzata in modo da separare la gestione della sessione, ottenuta mediante l'invio dei comandi provenienti dal client ed inoltrati al server, da quella dei flussi che al contrario provengono dal server per essere ridiretti sul client. MUM definisce, quindi all'interno della struttura del proxy quattro componenti chiave:

- Il ***ProxyInProtocolUnit*** riceve i comandi e li invia al ProxyOutProtocolUnit. Esso svolge inoltre il ruolo di coordinatore occupandosi dell'inizializzazione degli componenti del Proxy.
- Il ***ProxyOutProtocolUnit*** provvede all'invio dei comandi verso l'entità successiva del Service Path (un altro proxy o il server).
- Lo ***StramReceiver*** si occupa della ricezione del flusso proveniente dal Server.
- Ed il ***ServerVideoAgent*** spedisce il flusso, ricevuto dall'ultimo adattatore della catena, verso il cliente.

L'integrazione dell'adattatore nel Proxy di MUM segue lo schema di Figura **5.18**: gli adattatori, dopo essere opportunamente inizializzati, dovranno operare tra lo StreamReceiver ed il ServerVideoAgent.

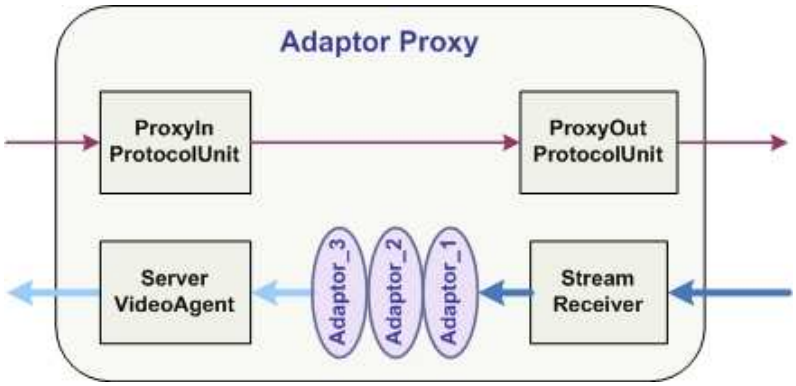



**Figura 5.18:** Integrazione degli Adaptor nel Proxy di MUM

## 5.4 Conclusione

In questo capitolo sono stati fissati i requisiti che il sistema di adattamento dovrà soddisfare. Dopo una loro prima analisi, il sistema in esame è stato suddiviso in moduli fondamentali e per ognuno di essi l'analisi ha permesso di individuare servizi e comportamenti che dovranno realizzare per assolvere alle loro responsabilità.

Il prodotto di questa fase può essere rappresentato dalla Figura **5.19**, in cui queste entità sono state mappate all'interno dell'architettura di MUM, distinguendo tra quelle che offrono supporto alle applicazioni costruite al disopra di MUM e quelle che forniscono funzionalità ed astrazioni utilizzate per la loro realizzazione. Questa architettura logica guiderà le fasi successive dello sviluppo: la progettazione e l'implementazione del sistema, che saranno oggetto di studio dei capitoli successivi.

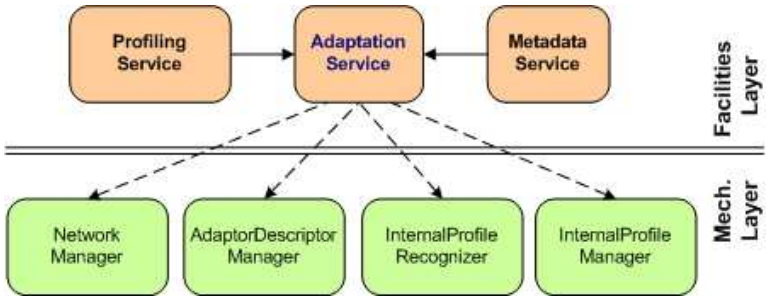



**Figura 5.19:** Architettura logica del servizio di adattamento prodotta dalla fase di analisi.

# *Capitolo 6*

# PROGETTO DEL SISTEMA PER L'ADATTAMENTO DI SERVIZI MULTIMEDIALI

In questo capitolo si presenterà la progettazione dei componenti e dei servizi individuati precedentemente nella fase di analisi. L'obiettivo della fase di progettazione è quello di stabilire come possono essere realizzate ed assolte funzionalità e responsabilità delle varie entità precedentemente analizzate, studiando in dettaglio i comportamenti e le interazioni fra le varie parti del sistema.

In generale, in questa fase dello sviluppo, sarebbe auspicabile rimanere ancora slegati dall'ambiente in cui il sistema verrà implementato, tuttavia essendo esso parte di un Middleware costruito sulla piattaforma SOMA, realizzata in linguaggio Java, si orienterà lo sviluppo dell'architettura verso il modello ad oggetti.

La struttura del capitolo segue quella dell'architettura logica prodotta in fase di analisi che divide le varie entità che compongono il sistema in due livelli: quello dei meccanismi e quello dei servizi.

## 6.1 Progetto dei meccanismi

Nel livello dei meccanismi trovano posto i servizi di base che offrono vari tipi di supporto ai componenti middleware. Di questo livello fanno parte i sistemi di gestione e di riconoscimento degli InternalProfile, il servizio di monitoraggio della rete e quello di descrizione degli adattatori.

### 6.1.1 Gestione degli InternalProfile

Al fine di realizzare una modellazione flessibile, l'entità InternalProfile viene rappresentata come una composizione di attributi rappresentanti una caratteristica di qualità del flusso multimediale (Figura **6.1**). Quindi di volta in volta lo sviluppatore che utilizza questo sevizio potrà decidere se per la sua applicazione utilizzare l'InternalProfile predefinito dal middleware, chiamato ***MUMInternalProfile*** oppure definirne uno nuovo composto da attributi che preferisce, che ovviamente però si prende carico di definire.

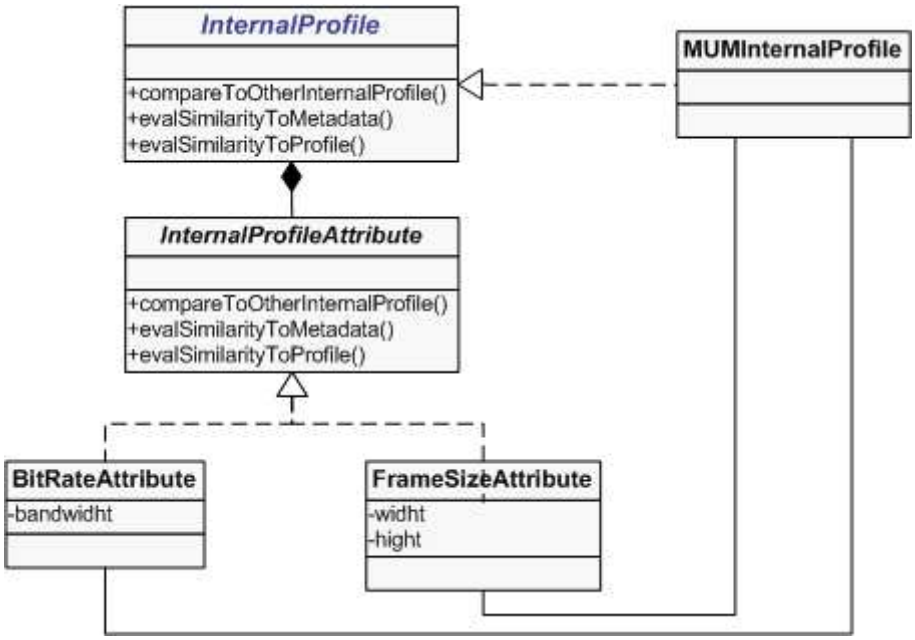



**Figura 6.1:** Modello flessibile e dinamico dell'InternalProfile

Affinché un sistema basato su questo modello possa funzionare correttamente è necessario che ogni InternalProfile incapsuli le logiche per la valutazione delle sue relazioni con i metadati, i profili utente e gli altri InternalProfile presenti all'interno del database. Esse rivestono un ruolo centrale sia nel riconoscimento di profili e metadati, sia nella selezione degli InternalProfile compatibili con quello che rappresenta le caratteristiche del profilo utente. Tuttavia la funzione utilizzata nei due casi appena presentati non è la stessa, infatti nel secondo si ha la necessità di esprimere più informazione, per considerare quelle situazioni in cui alle richieste provenienti da un terminale molto avanzato corrispondano nel sistema solo presentazioni di qualità inferiore.

#### 6.1.1.1 Relazione di similarità tra InternalProfile e profili o metadati

La valutazione della similarità rispetto a un Metadato o un Profilo, fondamentale all'atto del loro riconoscimento, viene realizzata attraverso la definizione di funzioni di scoring che indicano quanto esso sia simile all'InternalProfile considerato. Questa valutazione coinvolge tutti gli attributi dell'InternalProfile, infatti ognuno di essi incapsula la logica per effettuare il calcolo della similarità rispetto alla caratteristica che rappresenta. Un InternalProfile, dunque non deve far altro che trarre dai punteggi calcolati da ogni *AttributeInternalProfile* uno score complessivo che rappresenta la

sua effettiva similarità con il metado/profilo considerato. Il diagramma di sequenza in Figura **.2** spiega graficamente il funzionamento del meccanismo di valutazione per un metadato rispetto ad un *MUMInternalProfile*.

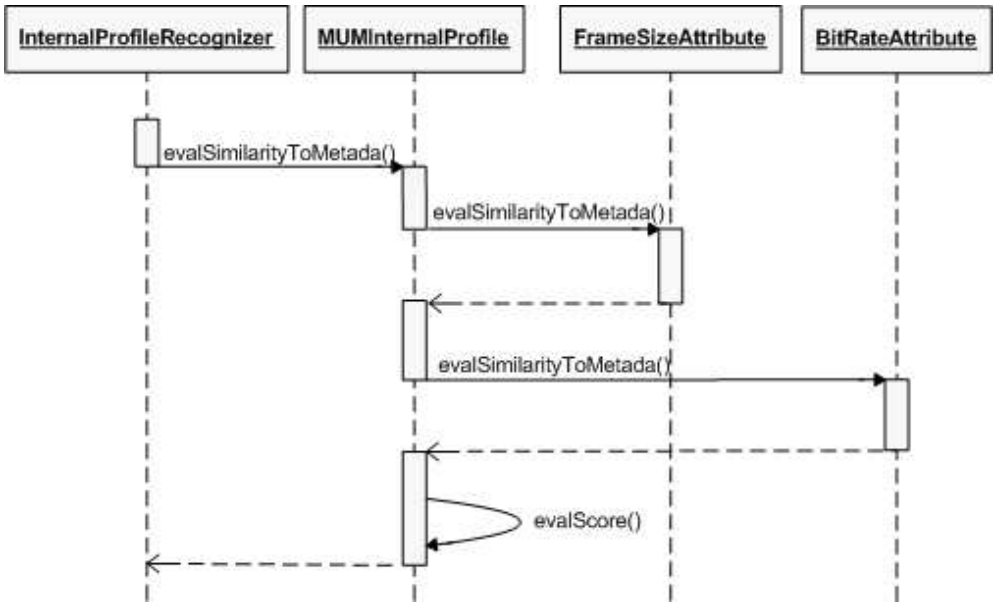



**Figura 6.2:** Meccanismo di valutazione della similarità

Le operazioni principali compiute all'interno di un *InternalProfileAttribute* per effettuare la valutazione di un Profilo/Metadato sono:

- Il recupero da esso del valore assunto dalla caratteristica che rappresenta;
- La valutazione della funzione di similarità.

Quest'ultima rappresenta l'unico aspetto che differenzia la logica di valutazione di un metadato da quella di un profilo, dovendo tener conto dei diversi principi, illustrati nel precedente capitolo, che guidano il riconoscimento dei metadati e quello dei profili e che dovranno essere presi in considerazione nella definizione delle funzioni di similarità.

Per il riconoscimento di un profilo utente il sistema utilizza la seguente funzione, dove $x_P$ rappresenta il valore dell'attributo $x$ all'interno del profilo utente considerato, mentre $x_{IP}$ il valore dello stesso attributo relativamente però all'InternalProfile rispetto al quale si sta calcolando la similarità.

$$sim = \begin{cases} \frac{x_{IP}}{x_P} & se \quad x_{IP} \leq x_P \\ -1 & altrimenti \end{cases}$$

Per il riconoscimento del metadato di una presentazione invece viene utilizzata quest'altra funzione, in cui $x_M$ rappresenta il valore dell'attributo x all'interno del metadato.

$$sim = \begin{cases} \frac{x_M}{x_{IP}} & se \quad x_M \leq x_{IP} \\ -1 & Altrimenti \end{cases}$$

Entrambi le funzioni permettono di assegnare ad ogni attributo dell'InternalProfile un punteggio di similarità rappresentato da un numero decimale compreso nell'intervallo [0,1] solo se i vincoli di riconoscimento sono soddisfatti.

#### 6.1.1.2 Relazione di similarità tra InternalProfile

La valutazione della similarità rispetto ad un altro InternalProfile presenta alcune differenze rispetto alle due precedentemente discusse non solo relativamente alla funzione di scoring da utilizzare, ma anche riguardo ai principi da seguire durante la sua elaborazione. In questo caso, infatti, la similarità deve tener in considerazione il principio secondo cui non è possibile ottenere un formato di un certa qualità se non a partire da uno di qualità superiore. Tuttavia, possono verificarsi situazioni in cui alle richieste provenienti da un terminale molto avanzato corrispondano nel sistema solo presentazioni di qualità inferiore. In questi casi si vorrebbe offrire al cliente la presentazione qualitativamente più vicina alle caratteristiche del proprio dispositivo.

La funzione di similarità, quindi, deve essere in grado di evidenziare la legittimità di una trasformazione quantificando la distanza tra gli InternalProfile coinvolti anche nei casi in cui essa non è possibile.

La funzione di similarità utilizzata dal sistema e incapsulata all'interno degli attributi che compongono il *MUMInternalProfile* è rappresentata dalla seguente formula, dove $x_a$ rappresenta il valore dell'attributo *x* relativo all'InternalProfile che rappresenta il livello di qualità da ottenere, mentre $x_{da}$ il valore dello stesso attributo

relativo però all'InternalProfile dal quale si vuol partire.

$$sim = \begin{cases} \dfrac{x_a}{x_{da}} & se \quad x_a \le x_{da} \\ -\dfrac{x_{da}}{x_a} & se \quad x_a > x_{da} \end{cases}$$

Il punteggio di similarità è, quindi, un numero decimale compreso nell'intervallo ]-1,1] che indica la possibilità della trasformazione solo nel caso sia positivo. Inoltre, in questo caso i due InternalProfile sono tanto più simili quanto più esso è prossimo a 1. Al contrario, se la trasformazione non è possibile, essi sono tanti più vicini quanto più il punteggio è prossimo a -1.

La similarità tra gli InternalProfile viene calcolata soltanto una volta, in fase di preparazione del sistema, nella quale tutte le relazioni vengono salvate in una struttura dati appositamente costruita all'interno dell'*InternalProfileManager*.

Questa struttura permette di associare all'InternalProfile, riconosciuto a partire da un metadato o da un profilo utente, un insieme di oggetti, chiamati *SimilarInternalProfile*, ognuno dei quali rappresenta una relazione con esso, rappresentata dal nome dell'InternalProfile rispetto al quale la similarità è stata calcolata, il suo punteggio e, se noto, il nome dell'adattatore per effettuare l'eventuale trasformazione. Il diagramma di Figura **6.3** mostra la struttura predisposta in un InternalProfile per ospitare le proprie informazioni di similarità:

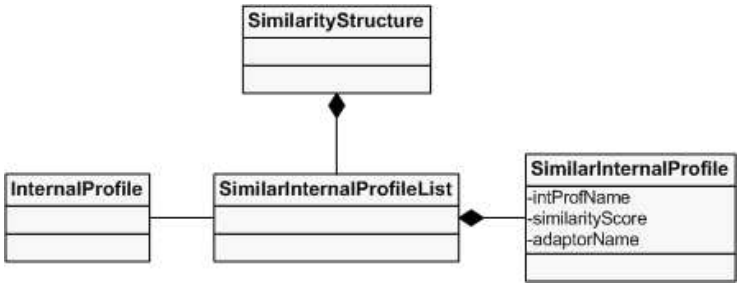



**Figura 6.3:** InternalProfile e informazioni di similarità

La struttura dati all'interno dell' *InternalProfileManager* è stata realizzata come un insieme di *SimilarInternalProfileList,* ognuna con un riferimento a un InternalProfile presente nel database. La struttura viene creata dinamicamente, infatti

inizialmente essendo il database degli InternalProfile vuoto, essa non contiene nessun elemento, in seguito ad ogni nuovo inserimento viene aggiornata creando una nuova *SimilarInternalProfileList* per l'elemento inserito ed eventualmente modificando quelli esistenti.

Similmente anche l'eliminazione di un InternalProfile dovrà essere gestita in modo da garantire sempre la coerenza della struttura di similarità con il contenuto dell'*InternalProfileRepository*.

### 6.1.2 Riconoscimento di profili e metadati

Il riconoscimento, svolto dall'*InternalProfileRecognizer*, è un processo basato sulla valutazione della similarità tra l'oggetto da riconoscere e tutti gli InternalProfile definiti in fase di preparazione del sistema. Il diagramma in Figura **6.4** mostra il modello UML di tale servizio evidenziando il suo stretto rapporto con quello che si occupa della gestione degli InternalProfile.

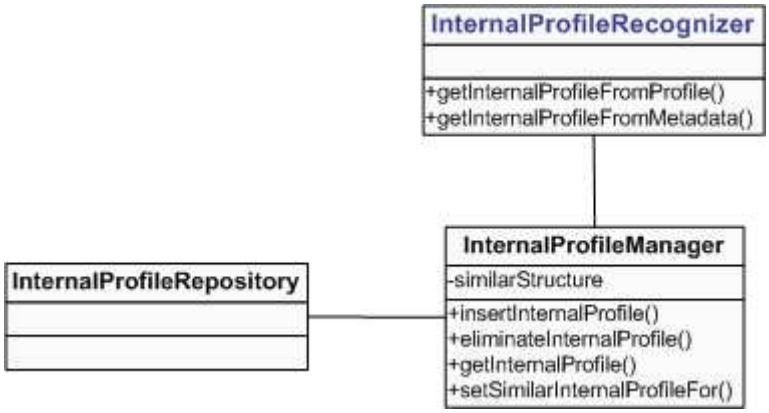



**Figura 6.4:** Modello del processo di riconoscimento

Alla richiesta di riconoscimento di un metadato, o similmente di un profilo, *l'InternalProfileRecognizer* si procura tutti gli InternalProfile presenti nel repository e per ognuno di essi comanda la valutazione della similarità individuando quello che meglio descrive le caratteristiche del profilo o del metadato sottoposto al riconoscimento. Infine, dopo che questo InternalProfile è stato trovato, viene passato all'*InternalProfileManager* che si occupa di assegnarli la corretta *SimilarInteralProfileList*.

I vincoli che bisogna tener in considerazione nel processo di riconoscimento, illustrati in 5.3.3.6, vengono automaticamente risolti utilizzando le funzioni di similarità presentate nel precedente paragrafoche assegnano all'InternalProfile un punteggio positivo solo se compatibile con l'oggetto da riconoscere.

### 6.1.3 Descrittori degli adattatori e loro gestione

La visibilità degli adattatori all'interno del sistema è strettamente legata alla presenza dei loro descrittori all'interno del database appositamente creato per la loro gestione. Questo database, chiamato *AdaptorDescriptorDB*, è di fondamentale importanza per il sistema perché permette di referenziare uno o più *Adaptor*, il cui codice è mantenuto all'interno del *SoftwareRepository* messo a disposizione da MUM, a partire dalle caratteristiche di una trasformazione di interesse. Il suo gestore, l'*AdaptorDescriptorManager*, permette l'inserimento di nuovi descrittori nel database, ma ovviamente questi non potranno essere utilizzati se prima il codice degli adattatori a cui essi si riferiscono non viene caricato nel *SoftwareRepository*.

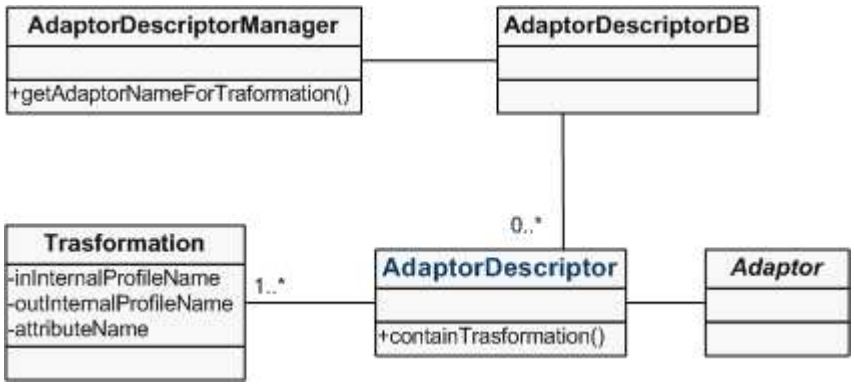



**Figura 6.5:** Modello degli AdaptorDescriptor e loro gestione

L'*AdaptorDescriptorManager,* quindi, è in grado di recuperare il riferimento ad un adattatore capace di compiere una certa trasformazione. Questo servizio offre un comodo supporto alle attività dell'*AdaptorsSelector* a cui rimane il compito di rilevare le trasformazioni elementari necessarie per il passaggio da un InternalProfile ad un altro, ma che viene sollevato dall'incarico di cercare per queste un apposito adattatore. Quest'ultima responsabilità viene assolta molto semplicemente dall'*AdaptorDescriptorManager* fornendo al database di cui è gestore un

identificatore della trasformazione di interesse. L'AdaptorDescriptorDB infatti è organizzato come una tabella che ad ogni trasformazione associa il nome univoco dell'adattatore in grado di effettuarla.

## 6.2 Progettazione dei servizi

Questa seconda parte del capitolo è dedicata alla progettazione dei servizi middleware direttamente accessibili dagli sviluppatori di applicazioni multimediali, facendo riferimento in particolare al servizio di adattamento e ad alcuni servizi a suo supporto come quello di profilazione degli utenti e di recupero dei metadati.

### 6.2.1 Servizio per il recupero dei metadati

Il servizio di recupero dei metadati si appoggia sull'interpretazione di una politica, definibile per ogni place del sistema, che permette di differenziare la gestione delle richieste considerando il numero di raffinamenti precedentemente eseguiti su di esse da parte degli altri place.

Questo servizio è stato progettato come estensione di quello che MUM mette a disposizione per la gestione dei metadati, la cui architettura è stata introdotta nel capitolo 4. Il punto di accesso al servizio quindi, è rappresentato da un oggetto di tipo MetadataServiceManager, disponibile su ogni place del sistema, che permette anche di accedere ad altre funzionalità di gestione dei metadati come inserimento ed eliminazione. La modalità di recupero tradizionale, utile nella realizzazione di altri servizi, non verrà toccata ma solo affiancata dalla nuova.

Seguendo l'architettura proposta dal servizio di gestione dei metadati, quando l'*AdaptationEngine* richiede il recupero di un metadato per un certa presentazione, la sua richiesta viene dapprima elaborata dal gestore delle politiche (il *PoliciesManager*) che dovrebbe valutare la propria politica per decidere se concedere alla richiesta l'accesso alla cache gestito dal *CacheManager*. In realtà nelle operazioni di recupero non ha senso valutare la politica almeno fino a quando un "Place di default" non trova un metadato corrispondente alla richiesta effettuata. Il comportamento del nuovo servizio di recupero fino a questo istante è dunque il medesimo di quello già implementato in MUM, solo che mentre quest'ultimo arrivato a questo punto restituisce i risultati ottenuti, l'altro richiede la valutazione

della politica e quindi decide se continuare la ricerca inoltrando la richiesta al CacheManager padre oppure restituire il risultato.

Il meccanismo di recupero proposto mira ad ottenere risultati più ricchi, in modo che l'AdaptationEngine abbia più opzioni da valutare per effettuare le proprie scelte, e allo stesso tempo a distribuire il controllo della propagazione della richiesta trai vari domini del sistema.

### 6.2.2 Servizio di profilazione dell'utente

In fase di creazione su ogni dominio viene creato un CCPPManager che incapsula il riferimento ai database degli utenti e delle piattaforme. Il CCPPManager viene utilizzato dunque come punto di accesso al servizio di profilazione rendendolo disponibile su ogni place di default del sistema presentando l'interfaccia mostrata in Figura **6.6**.

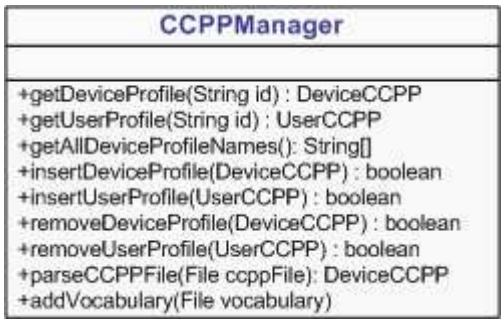



**Figura 6.6:** Interfaccia del servizio di profilazione

Come mostra l'interfaccia appena presentata, per poter richiedere un profilo è necessario conoscere il nome che identifica la piattaforma che supponiamo ci venga fornito dall'utente ad ogni sua richiesta.

La rappresentazione CC/PP di un profilo utente può essere vista come una qualsiasi rappresentazione XML, quindi trasformarla in un oggetto che la rappresenta significa semplicemente effettuare una operazione di parsing sulla stessa.

Come mostra la Figura **6.7**, il modello di gestione dei profili è basato su quello esistente che divide concettualmente le informazioni relative alle piattaforme da quelle relative agli utenti, dando la responsabilità della loro a gestione a entità differenti unicamente accessibili da un entità *ProfileContainerManager*.

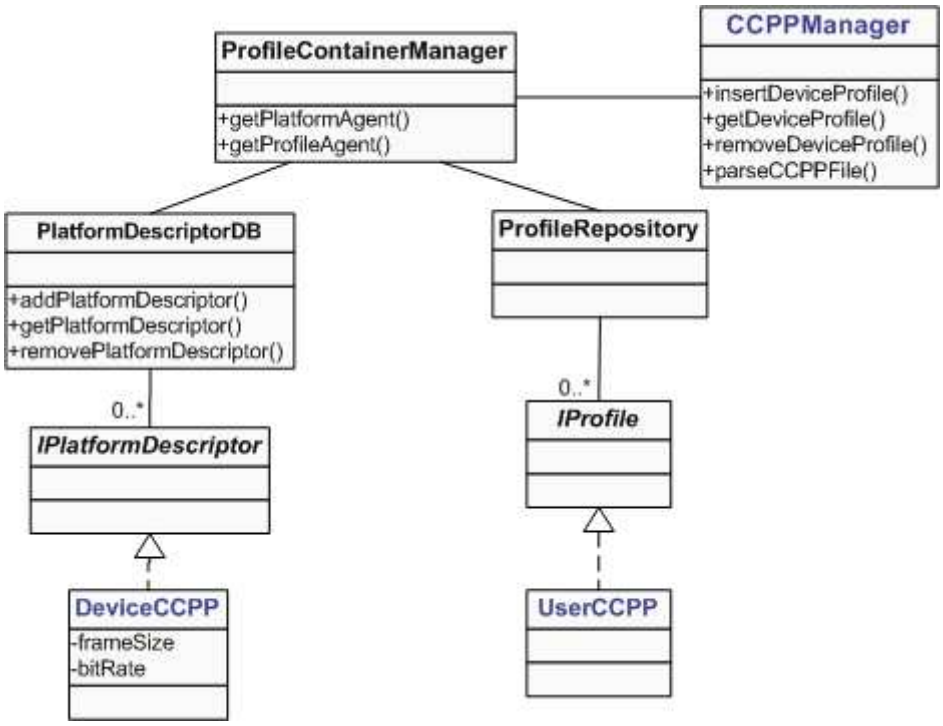



**Figura 6.7:** Architettura del servizio di gestione dei profili utente

L'architettura adottata prevede l'introduzione di due nuove classi per la rappresentazione dei due tipi di profili:

1. ***DeviceCCPP:*** Descrive la struttura CC/PP di un profilo contente informazioni sulla piattaforma utilizzata.
2. ***UserCCPP:*** Analogo al precedente, ma utilizzato per la rappresentazione di informazioni riguardanti l'utente.

In realtà, quest'ultimo profilo è stato introdotto solo per completezza nella definizione dell'architettura, infatti il tipo di adattamento che si vuol realizzare è orientato alla sola considerazione delle caratteristiche del dispositivo.

Si noti che essendo i due profili classificati rispettivamente come IPlatoformDescriptor e IProfile, non occorre costruire appositi database per la loro memorizzazione, ma possono essere utilizzati quelli già a disposizione del sistema.

Il gestore dei profili CC/PP, chiamato ***CCPPManager***, oltre a operare le classiche funzionalità di inserimento, eliminazione e recupero dei profili per le quali si appoggerà alle funzionalità offerte dai due contenitori, supporta un servizio per la traduzione dei profili espressi nella sintassi CC/PP in oggetti *DeviceCCPP*.

La traduzione dei profili, terrà ovviamente conto delle scelte fatte a proposito dei vocabolari usati per la loro rappresentazione , ma per non essere vincolato da esse il *CCPPManager* offre la possibilità di inserire a tempo di esecuzione nuovi vocabolari in modo da poter ampliare le proprie capacità di traduzione.

### 6.2.3 Servizio di Configurazione dell'adattamento

In fase di analisi sono state individuate nel servizio di configurazione dell'adattamento, gestito dall'*AdaptationEngine*, due logiche: quella che gli permette di interagire con i servizi precedentemente esposti da una parte e quella che si occupa delle attività decisionali dall'altra. La prima è gestita dal *MetadaAnalyzer* per quanto riguarda l'interazione con il servizio di gestione dei metadati e dal *ProfileAnalyzer* per quello che riguarda il servizio di profilazione. Il diagramma riportato in Figura **6.8** mostra l'interfaccia delle funzionalità che questi due componenti mettono a disposizione dell'*AdaptationEngine* per poter interagire con questi servizi.

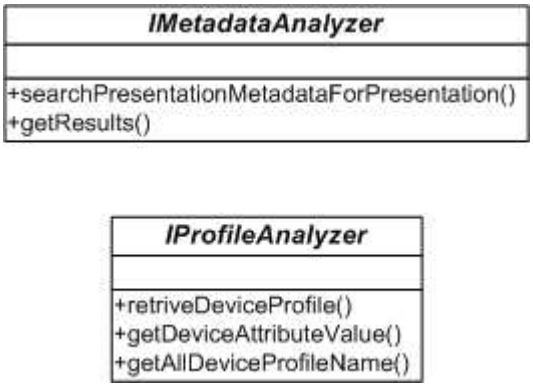



**Figura 6.8:** Interfacce dei due componenti per l'interazione con i servizi di supporto

Utilizzando questo tipo di architettura si riesce a garantire totale indipendenza dei componenti che si occupano delle attività decisionali dal modo in cui i servizi di supporto discussi vengono progettati e realizzati.

La logica decisionale è invece gestita da tre componenti, ognuno specializzato nel prendere un certo tipo di decisione. Come mostra il diagramma di collaborazione della Figura **6.9**, l'unico metodo esposto dall'*AdaptationEngine* viene utilizzato dal *DecisionMaker* per avviare il processo di configurazione dell'adattamento. Ad ogni richiesta l'*AdaptationEngine* dapprima recupera il profilo dell'utente e l'insieme dei

metadati corrispondenti alla presentazione da lui richiesta, poi considerando le informazioni ottenute comanda i componenti decisionali che le elaborano e restituiscono all'*AdaptationEngine* lo loro decisioni, che vengono raccolte in un oggetto di tipo *AdaptationResponse* e consegnate al *DecisionMaker*.

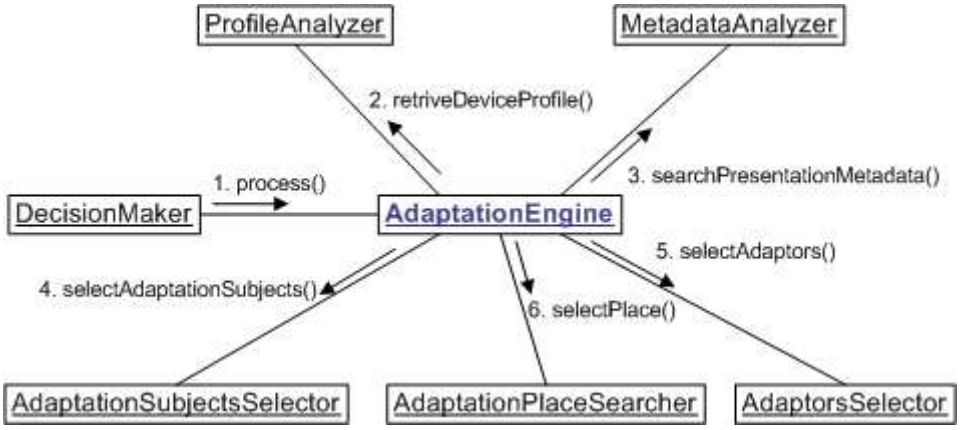



**Figura 6.9:** Interazione tra i vari componenti dell'AdaptationEngine per la sintesi delle decisioni

Nei prossimi paragrafi si studieranno in dettaglio il modo in cui le informazioni vengono elaborate dai vari componenti dell'*AdaptationEngine* per sintetizzare le decisioni che caratterizzeranno il processo di adattamento.

#### 6.2.3.1 Scelta delle presentazioni da adattare

La selezione delle presentazioni effettuata dall'*AdaptationSubjectsSelector* viene svolta attraverso le seguenti attività:

- Dalla lista dei metadati trovata dal *MetadataAnalyzer* vengono selezionati tutti quelli relativi a presentazioni che possono essere trasformate in un formato idoneo al profilo del terminale utente.
- Per ogni metadato che ha superato la selezione viene valutata la politica di adattamento al fine di ottenere una lista di presentazioni candidate al processo di trasformazione ordinata dalla migliore alla peggiore secondo i criteri espressi nella politica.

La prima attività richiede il riconoscimento del profilo del terminale utente e dei metadati. Si ricorda che questa operazione permette di associare ad un InternalProfile una lista che descrive le sue relazioni di similarità con gli altri InternalProfile. L'*AdaptationSubjectsSelecotor* quindi, utilizza questa lista per conoscere il

punteggio di similarità che lega ogni metadato al profilo utente e se esso indica la possibilità di effettuare la trasformazione il metadato viene etichettato come candidato al processo di adattamento. L'etichettamento avviene attraverso la creazione di un oggetto di tipo *AdaptationSubject* che contiene informazioni considerate rilevanti ai fini delle fasi successive del processo di configurazione dell'adattamento. In particolare per necessità e comodità si è scelto di incapsulare in questo oggetto la posizione della presentazione, il nome dell'InternalProfile che descrive il livello di qualità della presentazione ed infine il punteggio di similarità che lo lega all'InternalProfile del terminale utente.

La seconda attività svolta dall'*AdaptationSubjectsSelector* elabora la lista degli *AdaptationSubject* prodotta nella fase di selezione ordinandola in considerazione della politica di adattamento. La valutazione di questa politica comporta la stima del costo del processo di adattamento, effettuata a partire dalle informazioni di similarità fra i due InternalProfile, ed il calcolo della distanza della presentazione dal Client.

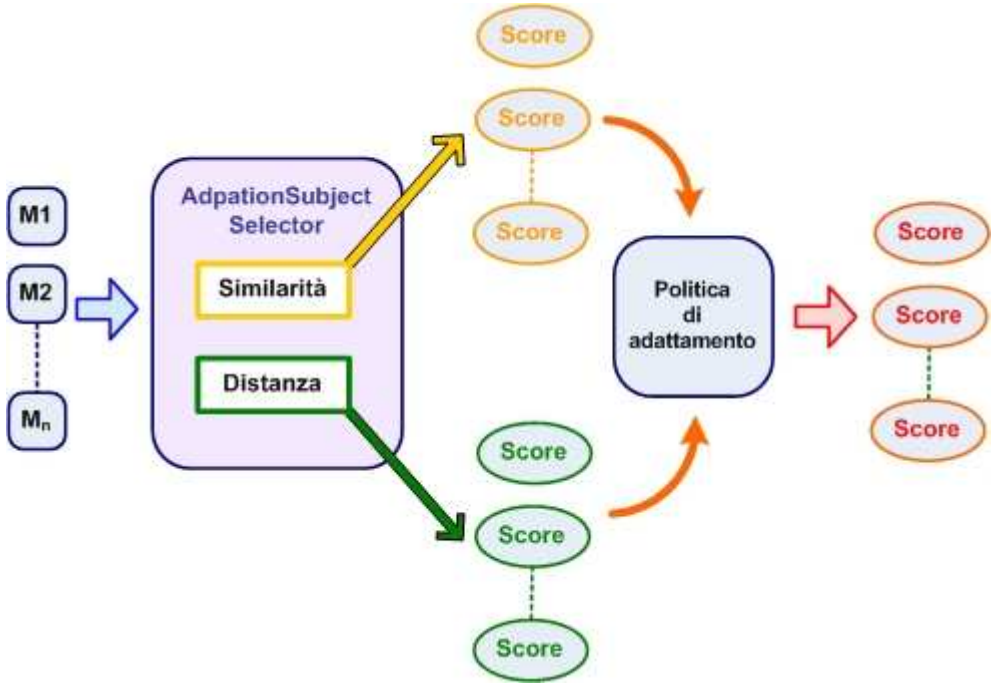



**Figura 6.10:** Meccanismo di valutazione delle presentazioni candidate all'adattamento

Nella Figura **6.10** viene rappresentato il meccanismo di assegnamento dei punteggi agli InternalProfile corrispondenti alle presentazioni compatibili con il profilo dell'utente: il punteggio complessivo viene ottenuto come media pesata di quello che misura la loro similarità con quello corrispondente al profilo utente e di

quello che rappresenta la distanza della presentazione. I pesi utilizzati per il calcolo di questa media sono quelli espressi nella politica di adattamento.

#### 6.2.3.2 Scelta degli adattatori da utilizzare

Come evidenziato in fase di analisi, il processo di scelta degli adattatori può essere scomposto in due fasi distinte: nella prima si ricercano le caratteristiche del flusso da adattare, esaminando le differenze tra l'InternalProfile di un candidato all'adattamento e quello relativo al profilo dell'utente, nella seconda quindi si procede alla selezione degli adattatori in grado di operare sulle proprietà del flusso individuate al fine di trasformarlo nel formato considerato idoneo alle qualità del terminale utente.

Lo svolgimento della prima fase è strettamente legato all'individuazione degli AdaptationSubject che deve essere effettuata prima di iniziare il processo di selezione degli adattatori. Infatti, le attività appena presentate dovranno essere svolte per ognuno dei candidati all'adattamento allo scopo di trovare la catena di adattatori da scaricare sul proxy nel caso in cui il DecisionMaker decida di adottare la soluzione che li coinvolge.

La seconda fase, come mostra la Figura **6.11**, viene gestita dall' *AdaptorsSelector* in collaborazione con il gestore dei descrittori degli adattatori. Ques'utlimo infatti ha la capacità di individuare l'adattatore in grado di attuare una certa trasformazione basandosi sulle informazioni presenti nell'apposito database ed introdotte all'inserimento degli adattatori nel sistema. Quindi dopo l'individuazione delle trasformazioni da effettuare all'*AdaptorsSelector* non rimane che effettuare l'opportuna richiesta al *AdaptorDescriptorManager*.

Infine, al termine del processo di selezione vengono notificate all'InternalProfileManager le scelte effettuate dall'AdaptorSelector, che le memorizza all'interno della struttura di similarità in modo da non doverlo interpellare di nuovo nel caso sia necessario rieseguire la stessa trasformazione in seguito ad un'altra richiesta.

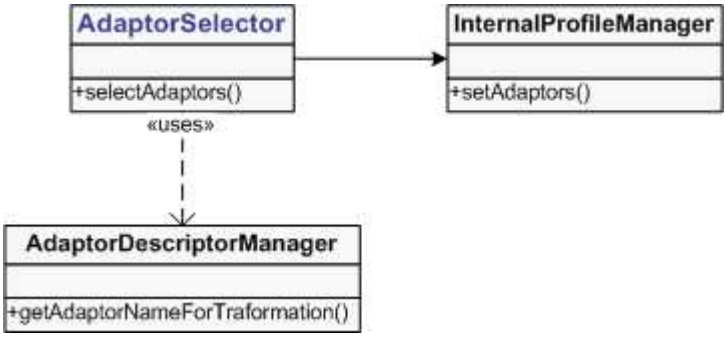



**Figura 6.11:** Modello del servizio di selezione degli adattatori

#### 6.2.3.3 Scelta dei place in cui svolgere l'adattamento

La scelta dei place in cui attuare l'adattamento è di fondamentale importanza per la distribuzione del carico computazionale sui vari nodi del Service Path in modo da rendere il processo di trasformazione del flusso il meno invasivo possibile.

Il servizio di individuazione dei place di adattamento potrebbe scegliere, a seconda della corrente situazione del traffico o delle risorse disponibili sui vari nodi del Service Path, di selezionare uno o più proxy in cui posizionare gli adattatori precedentemente selezionati dall'*AdaptorsSelector*.

Per la realizzazione di un sistema di questo tipo si è ipotizzato di disporre di un servizio di monitoraggio delle risorse che dato un Service Path ed una lista di adattatori restituisca una lista di place che specifica per ognuno di essi gli adattatori che in esso dovranno essere scaricati. Il punto di accesso a questo servizio è il *NetworkAnalyzer* al quale dunque l'*AdaptationPlaceSearcher* farà riferimento nel momento in cui verrà chiamato ad individuare i place sul quale posizionare gli adattatori per ogni candidato all'adattamento precedentemente selezionato dall'*AdaptationSubjectsSelector*.

### 6.2.4 Progetto del sistema di codifica dei flussi multimediali

Il risultato della configurazione dell'adattamento è la generazione di un piano di inizializzazione del Service Path che segue le decisioni prese dall'*AdaptationEngine*. Il piano generato, attraverso il meccanismo spiegato in 4.1.3, permette di scaricare sugli opportuni place gli adattatori che permetteranno la trasformazione del flusso.

La parte dell'adattatore che incapsula la logica in grado di trasformare i flussi è stata chiamata *Adaptor*. Un oggetto di questo tipo fornisce ai componenti utilizzatori

le funzionalità per la gestione di comandi base per iniziare e fermare le operazioni di trasformazione su una certa sorgente dati.

L'introduzione dell'adattatore nei place intermedi del Service Path ha portato allo sviluppo di un nuovo tipo di proxy che nell'Architettura MUM si inserisce nel modo rappresentato in Figura **6.14**. Questo nuovo proxy come gli altri già presenti in MUM utilizza una *IProxyInProtocolUnit* per scaricare il software necessario, istanziare ed inizializzare i suoi componenti, tra i quali anche la catena di adattatori da applicare sui flussi provenienti dal server per trasformarli nel flusso destinato al client. Le parti del proxy utilizzate per l'inoltramento dei comandi verso il server, la ricezione dei dati dallo stesso server e, dopo la loro elaborazione, l'invio verso il client vengono gestiti dai componenti già realizzati per la realizzazione del proxy tradizionale.

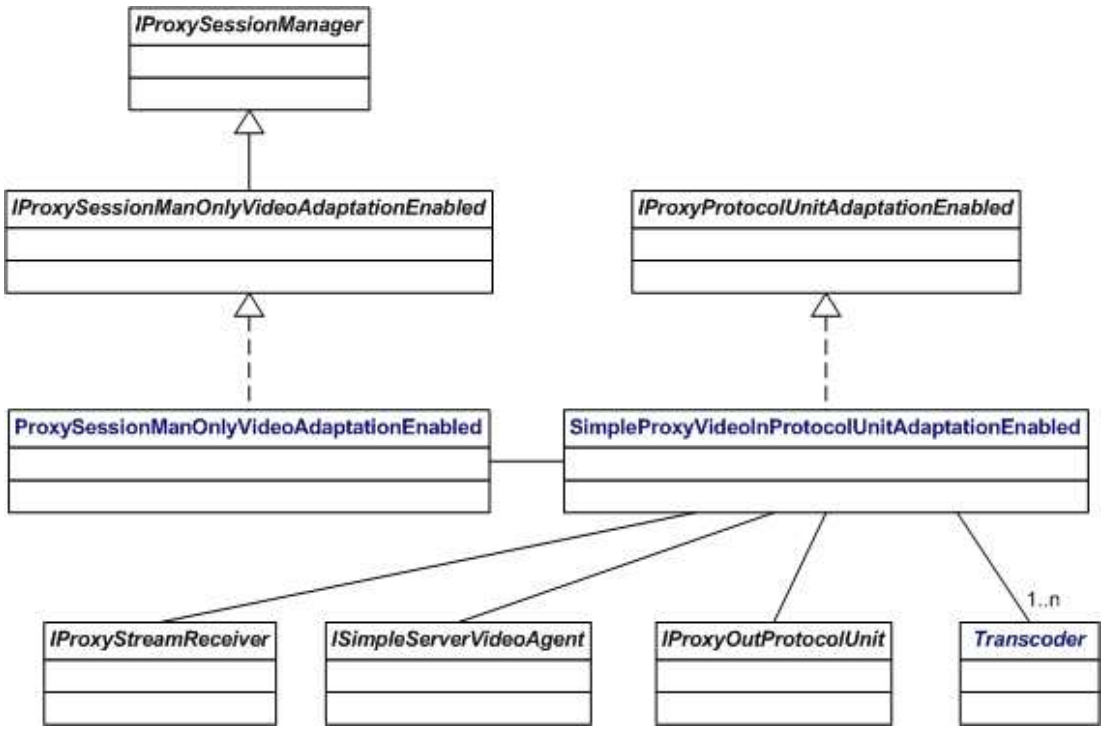



**Figura 6.14:** Inserimento del nuovo proxy nell'architettura di MUM

## 6.3 Conclusione

La progettazione del sistema partendo dall'architettura logica individuata nel capitolo di analisi ha permesso di capire il modo in cui realizzare un insieme di

servizi cooperanti al fine di integrare in MUM un servizio di adattamento che risponde ai requisiti posti in precedenza.

In questo capitolo si è discusso di problemi legati alla progettazione dei meccanismi e dei servizi middleware che compongono il sistema sviluppato presentando le scelte e gli approcci utilizzati per risolverli.

*Capitolo 7*

# IMPLEMENTAZIONE E TEST DEL SISTEMA DI ADATTAMENTO

L'ultima fase del processo di sviluppo del sistema è l'implementazione, in cui l'architettura e le soluzioni individuate nella fase di progettazione vengono realizzate attraverso l'utilizzo degli specifici strumenti che la piattaforma di sviluppo scelta mette a disposizione. In realtà, dopo l'implementazione, è di fondamentale importanza sottoporre il sistema creato a una fase di test per verificare che tutte le funzionalità progettate si comportino nel modo previsto e per valutare l'efficienza delle scelte effettuate in fase di progetto.

La prima parte del capitolo sarà dedicata alla presentazione di alcune caratteristiche implementative del sistema di adattamento, dando prima spazio agli strumenti utilizzati per la sua realizzazione; nella seconda, invece, si focalizzerà l'attenzione sui risultati ottenuti dalla fase di test del sistema cercando di trarre dalla loro analisi considerazioni e utili spunti per lo sviluppo di eventuali miglioramenti del sistema.

## 7.1 Strumenti per l'implementazione

Come anticipatamente segnalato in fase di progettazione, la piattaforma utilizzata per la realizzazione del sistema progettato è quella JAVA, linguaggio di programmazione in cui è stato sviluppato SOMA e quindi conseguentemente utilizzato per l'implementazione di MUM. Questa piattaforma offre utili strumenti sia per il trattamento dei profili CC/PP, sia per lo sviluppo di applicazioni multimediali. Entrambi sono utilizzati nella realizzazione del prototipo del sistema progettato e verranno qui di seguito presentati. In questo paragrafo, inoltre, verranno illustrate alcune delle scelte implementative svolte durante la realizzazione del prototipo, con riferimento in particolare alle funzioni di scoring utilizzate per la

valutazione della similarità al fine di stimare il costo di elaborazione di una trasformazione e di riconoscere metadati e profili dei terminali utente.

### 7.1.1 JSR-188 API e rappresentazione dei profili CC/PP

All'interno del Java Community Process (JCP) è stato sviluppato un insieme di API (Application Programming Interface), identificate con la sigla JSR-188, per la gestione e l'elaborazione di informazioni relative allo standard CC/PP in modo da facilitare lo sviluppo di applicazioni Java che vogliono offrire servizi tenendo conto delle capacità del dispositivo utilizzato dall'utente e delle sue preferenze [JSR188].

Le API proposte sono organizzate nei seguenti package:

1. *javax.ccpp* contiene le classi e le interfacce che permettono di creare ed accedere agli oggetti CC/PP rappresentati in termini di componenti e attributi.
2. *javax.ccpp.uaprof* fornisce le classi di base per la rappresentazione dei vari componenti secondo le specifiche di UAProf.
3. *com.sun.ccpp* definisce delle classi per configurare l'elaborazione dei profili e per la gestione dei vocabolari.

Queste funzionalità consentono di eseguire agevolmente le tradizionali operazioni di elaborazione e gestione che possono essere condotte su dei profili. Molto interessante, ad esempio, è la classe *Profile* che rappresenta un profilo come oggetto composto di componenti a loro volta composti di attributi. Una sua istanza può essere ottenuta dalla classe *ProfileFactory* che segue il pattern Abstract Factory e in modo completamente trasparente all'utilizzatore permette ad esempio di effettuare il parsing di un documento CC/PP.

La configurazione dei vocabolari usati nella interpretazione dei profili è gestita da un entità chiamata *DescriptionManager* che, fra le altre cose, si occupa della validazione dei vocabolari e delle estensioni propostogli rispetto a uno specifico XML Schema che definisce la struttura che essi dovrebbero seguire per essere considerati coerenti con le specifiche CC/PP.

### 7.1.2 Il Java Media Framework

Sun, in collaborazione con IBM, ha sviluppato un pacchetto software opzionale per estendere le funzionalità della piattaforma JAVA2SE™ in modo da poter

incorporare in applicazioni e applet materiale di tipo multimediale. Questo componente mette a disposizione degli sviluppatori delle API, dette JMF (Java Media Framework), che premettono la gestione dei più comuni standard di memorizzazione dei dati multimediali, quali: MPEG-1, MPEG-2, Quick Time, AVI, WAV, AU e MIDI.

Il trattamento dei dati multimediali in applicazioni JAVA era molto complesso prima dell'introduzione delle API JMF, infatti il meccanismo di funzionamento della piattaforma JAVA, basato sull'interpretazione del byte-code generato precedentemente da un compilatore, non permette di avere una velocità di elaborazione dei dati sufficiente da garantire una loro gestione ottimale dal punto di vista delle prestazioni. Lo sviluppatore che voleva operare con dati di tipo multimediale in JAVA ottenendo buone prestazioni era obbligato a interagire direttamente con il codice nativo della piattaforma di interesse rendendo inoltre l'applicazione sviluppata non più portabile sulle altre piattaforme. L'introduzione dell' API JMF risolve questo problema mettendo a disposizione una serie di chiamate a procedura di alto livello per la gestione del codice nativo, che semplificano lo sviluppo di applicazioni multimediali e soprattutto lo rendono trasparente al tipo di piattaforma software utilizzata.

L'attuale versione dell' API JMF, la 2.1.1, fornisce delle classi che permettono la riproduzione, la cattura, l'elaborazione, la memorizzazione e lo streaming di dati multimediali.

Ai fini del progetto sviluppato la possibilità di elaborare dati multimediali è sicuramente l'aspetto più interessante che JMF permette di gestire.

JMF 2.1.1 definisce a questo scopo la classe Processor, che operando su una certa sorgente dati ne restituisce una nuova che può essere riprodotta da un Player, memorizzata all'interno di un file, o di nuovo manipolata da un altro Processor.

Le operazioni che un Processor dovrà applicare su una certa sorgente sono configurabili solo quando esso si trova in un certo stato detto "Configured" . La Figura **7.1**, tratta da [JMF] a cui si rimanda per ulteriori approfondimenti, mostra un diagramma di stato che descrive il comportamento del *Processor* durante il suo ciclo di vita. Gli stadi evidenziati sono gli unici non presenti in un Player, vengono infatti

utilizzati dal Processor per connettersi alla sorgente dati e accedere alle informazioni che permettono di caratterizzarne il formato.

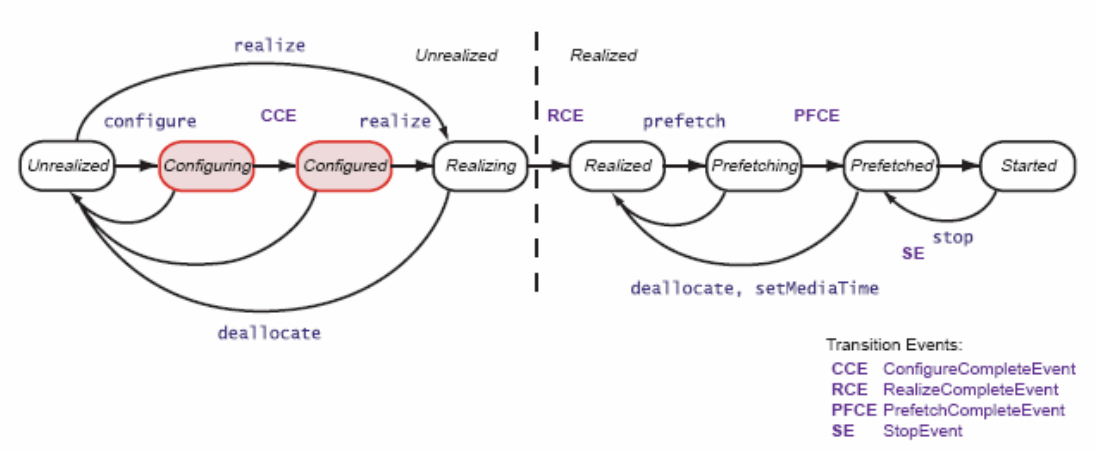



**Figura 7.1:** Stati di un Processor

Una volta che queste operazioni sono state effettuate il Processor passa allo stato "Configured", in cui può essere programmato in modo che vengano condotto sulla sorgente le trasformazioni volute.

Al termine della programmazione il Processor viene realizzato e si può ottenere da esso un riferimento alla sorgente dati che verrà creata quando partirà il processo di trasformazione.

## 7.2 Implementazione del sistema di adattamento

L'obiettivo di questa sezione non è quello di illustrare la completa implementazione del sistema, per la quale si rimanda al cd in allegato, ma solo di soffermarsi sui punti cruciali del lavoro svolto mettendo in risalto le peculiarità implementative del prototipo realizzato.

### 7.2.1 Implementazione del profilo utente

Nel prototipo del sistema progettato, il profilo del terminale utente, chiamato *DeviceCCPP* viene rappresentato utilizzando l'oggetto *Profile* delle JSR-188 API. La Figura **7.2** mostra come questo oggetto sia incapsulato nel *DeviceCCPP* che quindi lo usa come contenitore delle informazioni di tipo CC/PP alle quali integra quelle che ciascun descrittore di piattaforma deve fornire, cioè un nome per la sua

identificazione e un oggetto *Capability* che riassume alcune delle caratteristiche rappresentative del profilo direttamente utilizzabili dagli altri componenti del sistema. Continuare ad utilizzare l'oggetto *Capability* potrebbe sembrare ridondandante visto che non contiene nessuna informazione aggiuntiva a quelle del *Profile*, in realtà si scelto di tenerlo, da una parte per rendere il profilo utente compatibile con i servizi già sviluppati, dall'altra perché, essendo l'attività di recupero delle informazioni strettamente dipendente dal tipo di vocabolario utilizzato nella rappresentazione dei profili, potrebbe essere utile incapsulare la sua logica all'interno del *DeviceCCPP* stesso, in modo da poter creare un opportuno oggetto *Capability* (descrittore del profilo), contente le sole informazioni utilizzate dal sistema sviluppato.

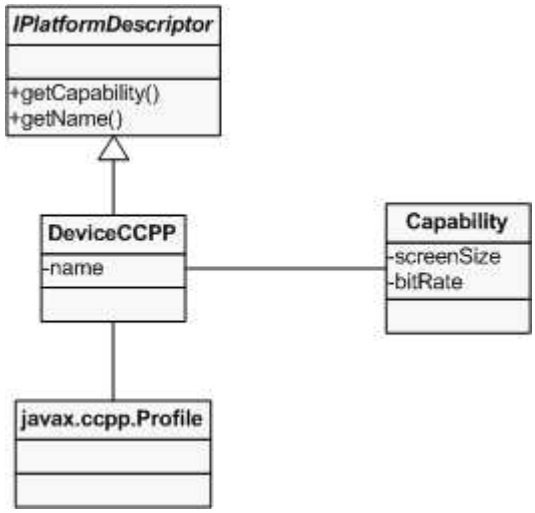



**Figura 7.2:** Implementazione del Profilo del dispositivo per mezzo di JSR-188

### 7.2.1 Recupero dei metadati e dei profili

Essendo le funzioni di recupero le uniche che richiedono l'esecuzione di operazioni di comunicazione con gli altri place del sistema, si teme che esse possano pesare molto in termini di costi sull'intero processo di configurazione dell'adattamento. Per questo motivo i due componenti preposti allo svolgimento di queste attività, il MetadataAnalyzer ed il ProfileAnalyzer sono stati realizzati cercando di ottimizzare il più possibile il loro funzionamento.

In Figura **7.3** si riporta il metodo del ProfileAnalyzer che si occupa dell'interrogazione del CCPPManager. Si può osservare che quando un profilo viene recuperato dal database attraverso l'intervento del suddetto "manager", esso viene

memorizzato all'interno di una variabile del ProfileAnalyzer che ha il compito di mantenere l'utlimo profilo recuperato in modo che un'eventuale richiesta successiva dello stesso profilo venga gestita senza richiedere di nuovo l'intervento del CCPPManager.

```
/**Per richiedere il recupero dell'DeviceProfile specificato.
 *
 * @param id Stringa univoca per l'identificazione del profilo
 * @return true se l'operazione ha esito positivo.*/
public boolean retriveDeviceProfile(String id){
       if(deviceProfile!=null && deviceProfile.getUniqueId().equals(id))
            return true;
       this.deviceProfile = this.profileManager.getDeviceProfileFromRepository(id);
       if(this.deviceProfile!=null) return true;
       return false;
}
```

**Figura 7.3:** Metodo del ProfileAnalyzer per il recupero di un profilo

Si potrebbe pensare di stressare questo approccio memorizzando tutti i profili che vengono recuperati, ma questa soluzione è stata scartata perché il sistema di recupero diventerebbe statico e non più sensibile all'inserimento di nuove profili nel sistema. Un approccio simile viene utilizzato anche per il recupero dei metadati.

### 7.2.2 InternalProfile e riconoscimeto di profili e metadati

La struttura dell'InternalProfile considerata nel prototipo è composta da due attributi: il primo rappresenta la dimensione dei frame della presentazione, il secondo la quantità di informazione necessaria per un secondo di riproduzione, ovvero la bit-rate della presentazione espressa in bps. Tuttavia come previsto anche dall'architettura presentata in precedenza è possibile definire InternalProfile personalizzati semplicemente definendo gli attributi di cui comporli.

Le funzioni di similarità utilizzate per il riconoscimento dei metadati e dei profili e quella per la valutazione della similarità con gli altri InternlProfile ricalcano quelle presentate in fase di progetto. Al fine di realizzare una struttura dell'InternalProfile flessibile queste funzioni vengono definite all'interno degli attributi che lo compongono. Essi, quindi, calcolano un punteggio di similarità parziale, relativo alla caratteristica rappresentata, che l'InternalProfile raccoglie per valutare il punteggio

complessivo facendo la loro media aritmetica. La figura **7.4** riporta il metodo di valutazione della similarità di un generico InternalProfile rispetto a un metadato. Si può notare che questa funzione è indipendente dal tipo di caratteristiche descritte dall'InternalProfile, quindi non necessita di essere ridefinita qualora uno sviluppatore ne voglia utilizzare uno personalizzato.

```
/** Funzione per valutare la similarità del profilo rispetto a un metadato
 *
 * @param metadata Il metadato rispetto al quale viene valutata la similarità
 * @return Valore di similarità compreso tra 0 e 1.
          -1 se il metadato non è compatibile con il profilo*/
 public float evalSimilarityToMetadata(Metadata metadata){
    float similarity = 0;
    for(Iterator j = this.getAttributes().iterator(); j.hasNext();){
        InternalProfileAttribute ia = (InternalProfileAttribute)j.next();
        float sim = ia.evalSimilarityToMetadata(metadata);
        if(sim>0) similarity += sim;
        else return -1;
    }
    /*La similirità del metadato al profilo viene calcolata come media aritmetica
      delle similarità del metadato agli attributi del profilo interno.*/
    return similarity/this.getAttributes().size();
}
```

**Figura 7.4:** Metodo per la valutazione della similarità di un metadato rispetto all'InternalProfile.

Il meccanismo di riconoscimento dei metadati è ovviamente basato sulla valutazione di questo metodo per tutti gli InternalProfile presenti all'interno del sistema. Attraverso l'utilizzo di un meccanismo analogo viene effettuato il riconoscimento dei profili.

### 7.2.3 Scelta degli adattatori

I descrittori su cui è basata la scelta degli adattatori sono stati realizzati come contenitori delle trasformazioni che essi sono capaci di effettuare. Per rendere più efficiente la loro gestione, quando un descrittore viene inserito nell'apposito database, realizzato attraverso una tabella hash, vi vengono inserite tante entry quante sono le trasformazioni che esso contiene. La chiave di ognuna di queste entry

è l'identificatore della trasformazione, mentre il valore memorizzato è il nome univoco dell'adattatore capace di effettuarla.

Questo approccio, data una trasformazione, permette di recuperare immediatamente dal database un riferimento all'adattatore necessario per attuarla. Il meccanismo qui presentato sarà molto utile all'*AdaptorsSelector* che, come mostra il codice riportato in Figura **7.5**, dopo aver individuato le trasformazioni necessarie per ottenere il formato richiesto non deve fare altro che interrogare il database.

```
/** Permette di selezionare l'Adattore da utilizzare per effettuare una
 * trasformazione elementare che coivolge un singolo attributo.
 * @param pInputName Nome del Profilo interno di ingresso all'adattatore.
 * @param pOutput Profilo interno di uscita dall'adattatore.
 * @param attributeName Nome dell'attributo coinvolto nella traformazione
 * @return Il nome univoco dell'adattatore capace di effettuare la trasformazione.*/
 private String selectAdaptor(InternalProfile pOutput,String pInputName,
                              String attributeName){
        String nameAdaptor = pOutput.getAdaptorNameFor(pInputName);
        if (nameAdaptor != null)
            return nameAdaptor;

        //Interrogazione del repository
        nameAdaptor =this.adaptorsManager.getAdaptorNameFor(pInputName,
                                                            pOutput.getName(),
                                                            attributeName);
        //Memorizzazione del nome nella stuttura di similarità
        this.profileManager.setAdaptorFor(pOutput.getName(),pInputName,nameAdaptor);

        return nameAdaptor;
 }
```

**Figura 7.5:** Metodo per la selezione dell'adattatore in grado di svolgere una trasformazione

Dopo l'interrogazione del repository, l'identificatore dell'adattatore viene memorizzato nella struttura dati presente nell'InternalProfileManager in modo che eventuali richieste successive per la stessa trasformazione vengano immediatamente servite senza accedere alla tabella hash.

### 7.2.4 Scelta dei place in cui effettuare l'adattamento

Il sistema di adattamento coinvolge nel proprio funzionamento molti servizi di contorno. La maggior parte di essi è stata realizzata seguendo l'architettura proposta

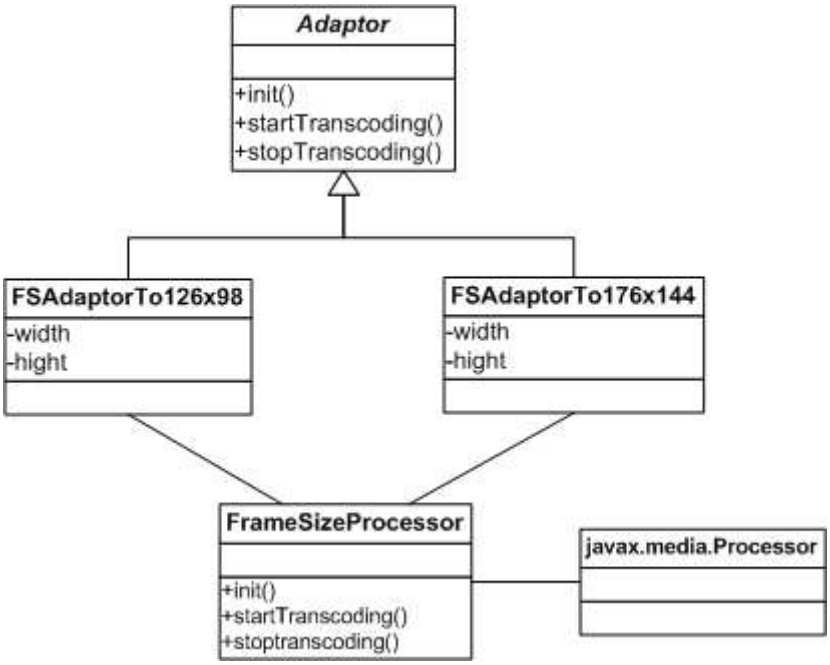
e presentata nei precedenti capitoli, tuttavia in alcuni casi sono state fatte delle ipotesi semplificative al fine di non perdere di vista il tema di questo lavoro di tesi: l'adattamento. È questo il caso del servizio per il monitoraggio delle risorse a disposizione dei nodi che compongono il Service Path al fine di individuare i place è più conveniente attuare l'adattamento. Per la realizzazione di questo servizio, come accennato nel precedente capitolo si fa l'ipotesi di avere a disposizione un componente che dato l'insieme degli adattatori da utilizzare per effettuare la trasformazioni e il Service Path all'interno del quale essi possono essere distribuiti, si occupa di produrre in uscita una lista di oggetti detti AdaptationPlace che specifica per ogni place coinvolto nel processo di trasformazione la lista degli Adaptor che in esso dovranno operare.

Il prototipo, per facilitare una eventuale integrazione futura di questo componente nel sistema, prevede a livello architetturale la sua esistenza, ma dal punto di vista realizzativo non implementa alcun tipo di logica per effettuare le scelte, ma semplicemente staticamente alloca tutti gli adattatori in uno stesso place, che è il padre di quello in cui è stata effettuata la richiesta di adattamento. Questa scelta è stata effettuata per permettere di valutare separatamente il carico derivante dalla configurazione del processo di adattamento da quello dovuto al processo di trasformazione del flusso.

### 7.2.5 Processor e Adattatori

Nel prototipo sviluppato gli oggetti *Adaptor* vengono realizzati attraverso un *Processor* dell'API JMF. In particolare ognuno di essi incapsula lo specifico *Processor* da utilizzare per effettuare le trasformazioni dichiarate attraverso il proprio *AdaptorDescriptor* ed inoltre gestisce i parametri che permetteranno la sua opportuna configurazione. La Figura **7.6** mostra attraverso un diagramma delle classi il modo in cui vengono realizzati due *Adaptor* che operano sulla stessa caratteristica del dato multimediale: il frame-size, ma con parametri di altezza e ampiezza diversi.

La concatenazione degli *Adaptor* all'interno di un proxy può essere realizzata come concatenazione dei *Processor* a cui si riferiscono.




**Figuta 7.6:** Relazione tra Adaptor e Processor

Una possibile alternativa alla soluzione appena presentata potrebbe essere quella di sfruttare la possibilità di programmazione dei Processor JMF, per la realizzazione di adattatori configurabili in grado di svolgere più di una trasformazione.

## 7.3 Test del sistema

Quest'ultima sezione del capitolo è dedicata alla presentazione dei risultati raccolti e a una loro interpretazione e discussione finalizzata all'individuazione delle eventuali inefficienze e carenze del sistema implementato. Essa è dunque di estrema importanza perché rappresenterà il punto di partenza di eventuali sviluppi futuri del sistema di adattamento.

I componenti principali del sistema realizzato sono due:

- l'*AdaptationEngine* che elabora le scelte che caratterizzeranno l'intero processo di consegna delle presentazioni.
- il proxy adattatore che effettua durante lo streaming del flusso multimediale le trasformazioni indicate dal precedente componente.

In questo paragrafo, dunque, le capacità di questi componenti verranno presentate e valutate separatamente cercando ogni volta di focalizzare l'attenzione solo su quello considerato.

Le topologie di rete sulle quali sono stati svolti i test sono le due rappresentate in Figura **7.7**:

- la prima è composta di due place di default. Uno fa da nodo radice e svolge sia il ruolo di servitore, dato che in esso si trovano tutte le presentazioni del sistema, sia da proxy adattatore; l'atro svolge invece il ruolo di cliente.
- la seconda prevede l'utilizzo di una terza entità nel Service Path che svolga esclusivamente il ruolo di proxy.

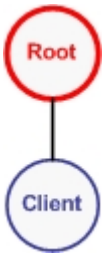



**Figura 7.7a:** Topologia senza proxy

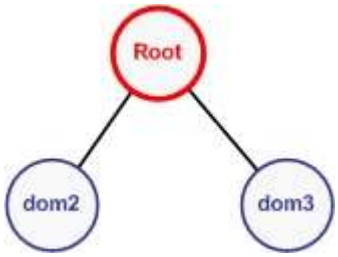



**Figura 7.7b:** Topologia con proxy

I computer coinvolti nell'esecuzione dei test, connessi attraverso una rete Ethernet LAN a 100 Mbps, sono equipaggiati nel seguente modo:

- ***Configurazione Hardware:***
  Sun Blade 2000 workstation con processore a 900 Mhz e 1024 MB di RAM (Random Access Memory).
- ***Configurazione Software:***
  - Sistema operativo SunOS 5.9;
  - Java Virtual Machine (JVM) versione 1.4.0_00_b05;
  - Java Media Framework (JVM) Performance Pack per Solaris versione 2.1.1e.

### 7.3.1 Valutazione dei ritardi introdotti dall'AdaptationEngine

L'AdaptationEngine, entrando in gioco ad ogni richiesta presentata dall'utente deve garantire efficienza. Senza dubbio durante la fase di configurazione del ServicePath il componente introdurrà dei ritardi che in questo paragrafo verranno valutati, in modo da individuare eventuali colli di bottiglia nella sua realizzazione.

#### 7.3.1.1 Costi delle singole attività dell'AdaptationEngine

L'AdaptationEngine svolge essenzialmente due attività: quella di recupero delle informazioni riguardo l'utente e le presentazioni corrispondenti al titolo richiesto; e quella di elaborazione. Sarebbe quindi interessante valutare il peso di queste due attività per capire su quale agire in futuro per migliorare le prestazioni complessive del componete. A questo scopo, sono stati effettuati dei test sulla topologia di Figura **7.7a**, in cui sul nodo "root" viene registrata una presentazione di livello medio, e sul nodo client viene lanciato un agente che simula la richiesta della presentazione da un profilo di basso livello, le cui caratteristiche sono state discusse nel paragrafo precedente. I risultati raccolti, ottenuti effettuando cinque prove in sequenza e riportati in **Tabella 1**, mostrano che le attività più costose, in termini di tempo (in millisecondi) necessario per il loro svolgimento, sono quelle volte al recupero delle informazioni. In particolare si sottolinea che il dato relativo al recupero del profilo utente ingloba il tempo necessario al prelievo del file CCPP dal database e all'azione di parsing effettuata dalla classe *Profile* delle API JSR-188.

| Attività | Prova1 | Prova2 | Prova3 | Prova4 | Prova5 | Media |
|---|---|---|---|---|---|---|
| **Recupero metadati** | 511 | 0 | 0 | 0 | 0 | **102,2** |
| **Rucpero Profilo** | 1400 | 0 | 0 | 0 | 0 | **280** |
| **Totale Recupero** | 1911 | 0 | 0 | 0 | 0 | **382,2** |
| **Selezione Presentazioni** | 15 | 10 | 8 | 5 | 4 | **8,4** |
| **Selezione Adattatori** | 3 | 1 | 2 | 1 | 1 | **1,6** |
| **Selezione Place** | 7 | 0 | 1 | 0 | 1 | **1,8** |
| **Totale Decisione** | 25 | 11 | 11 | 6 | 6 | **11.8** |
| **Totale** | 1951 | 16 | 13 | 10 | 8 | **399,6** |

**Tabella 1:** Costi delle singole sottoattività dell'AdaptationEngine

Come mostra questo esperimento, il peso medio dell'attività decisionale rappresenta circa il 3% del carico dell'AdaptationEngine. La maggior parte del ritardo introdotto dall'AdaptationEngine è quindi introdotto dal recupero delle informazioni. Questa situazione era stata prevista in fase di progettazione del

sistema, in cui per limitare il problema si era pensato di effettuare il caching dell'ultimo risultato recuperato. I risultati confortano questa soluzione, infatti dalla seconda prova in poi il tempo per il recupero delle informazioni è non rilevabile, tuttavia in generale non è affatto detto che i clienti abbiamo sempre lo stesso profilo.

In **Tabella 2** vengono mostrati i tempi necessari per il recupero delle informazioni nel caso in cui il caching dell'ultimo risultato è disabilitato.

| Attività | Prova1 | Prova2 | Prova3 | Prova4 | Prova5 | Media |
|---|---|---|---|---|---|---|
| **Recupero metadati** | 511 | 16 | 15 | 8 | 8 | **111,6** |
| **Rucpero profilo** | 1403 | 540 | 256 | 286 | 254 | **547,8** |
| **Totale recupero** | 1914 | 556 | 271 | 294 | 262 | **659,4** |

**Tabella 2:** Tempi di recupero con caching dell'utlimo risultato disabilitato

Ancora una volta si può notare il netto miglioramento dei tempi dalla seconda prova in poi. Questo è un risultato aspettato per il recupero dei meatadati perché MUM si serve di un sistema di caching dei metadati che permette di migliorare l'efficienza nel loro recupero. Il miglioramento dei risultati sul fronte del recupero dei profili potrebbe invece essere dipendente dai metodi utilizzati da JSR-188 per l'effettuazione del parsing dei file CCPP, che probabilmente utilizzano anche essi sistemi di caching, tuttavia non è stata trovata documentazione a conferma di questa ipotesi.

#### 7.3.1.2 AdaptationEngine e inizializzazione dello streaming

Con inizializzazione dello streaming si intende la fase che comincia quando l'utente richiede un certo titolo e termina quando tutte le entità sono state inizializzate dal place del cliente fino al place in cui si trova la presentazione prescelta. La terminazione della fase di inizializzazione coincide con l'inizio della visione del video da parte dell'utilizzatore.

L'adattamento introduce dei ritardi sia in fase di costruzione del piano di inizializzazione costruito sulla base scelte compiute dall'AdaptationEngine, sia durante la sua attuazione per far si che sui proxy vengano scaricati, inizializzati e configurati adeguatamente gli adattatori necessari alla trasformazione del flusso.

In questo sezione si vuole focalizzare l'attenzione sul ritardo introdotto nella prima fase cercando non solo di quantificarlo, come fatto nei paragrafi precedenti, ma anche di valutarlo relativamente al ritardo percepito dall'utente dovuto all'intero processo di inizializzazione dello streaming. I risultati, presenti in **Tabella 3**, riportano in prima colonna questo ritardo, in seconda il contributo dell'AdaptationEngine alla realizzazione di questo ritardo. I risultati raccolti sono relativi a un test svolto sulla topologia di Figura **7.7b**, in cui su "dom2" si ha una presentazione di livello medio richiesta da una agente su "dom3". Le prove sono state svolte in sequenza.

| | **Inizializzazione dello streming** | **Ritardo dell' AdaptationEngine** | **Incidenza in % dell'AdaptationEngine** |
|---|---|---|---|
| Prova1 | 21149 | 1944 | **9,19** |
| Prova2 | 3799 | 12 | **0,31** |
| Prova3 | 3331 | 15 | **0,45** |
| Prova4 | 3754 | 11 | **0,29** |
| Prova5 | 3838 | 10 | **0,26** |
| **Media** | **7174,2** | **398,4** | **2,10** |

**Tabella 3:** Incidenza dell'AdaptationEngine su presentazioni di media qualità

Il forte scostamento della prima misurazione dalle successive è dovuta al fatto che la prima volta che viene effettuata l'inizializzazione è necessario procedere dapprima alla configurazione dei vari nodi che compongono il Service Path, scaricando, dove necessario, il software per lo svolgimento dello streaming. Questo scostamento quindi esiste anche nel caso in cui si effettua l'erogazione del flusso multimediale senza operare adattamento, tuttavia, come vedremo meglio negli esperimenti successivi l'inserimento degli adattatori lungo il Service Path comporta un suo aumento.

I risultati mostrano che a regime ed in considerazioni favorevoli (utenti con lo stesso tipo di dispositivo), l'incidenza dell'AdaptationEngine sul tempo necessario per l'inizializzazione dello streaming è inferiore all'1%.

#### 7.3.1.3 Dipendenza dei costi dalla distanza tra profilo e metadato

Un'altra interessante verifica da svolgere sull'AdaptationEngine consiste nel

capire se il ritardo da esso introdotto è dipendente dalla distanza del profilo utente dal metadato. A questo scopo è stato svolto un secondo test sull'AdaptationEngine in cui si utilizza la stessa topologia del test presentato in precedenza, ma a differenza di quest'ultimo il livello di qualità corrisponde a un InternalProfile di livello alto.

I risultati, riportati in **Tabella 4**, permettono di osservare che il ritardo introdotto dall'AdaptationEngine aumenta rispetto al test condotto in precedenza di meno di due millesimi di secondo.

| Attività | Prova1 | Prova2 | Prova3 | Prova4 | Prova5 | Media |
|---|---|---|---|---|---|---|
| **Recupero metadati** | 338 | 0 | 0 | 0 | 0 | **67.6** |
| **Rucpero profilo** | 1611 | 0 | 0 | 0 | 0 | **322.2** |
| **Totale recupero** | 1999 | 0 | 0 | 0 | 0 | **389.8** |
| **Selezione Presentazioni** | 13 | 9 | 7 | 5 | 5 | **7.8** |
| **Selezione Adattatori** | 5 | 1 | 2 | 1 | 1 | **2.2** |
| **Selezione Place** | 8 | 1 | 1 | 1 | 1 | **2.4** |
| **Totale decisione** | 26 | 11 | 10 | 7 | 7 | **12.2** |
| **Totale** | 1951 | 16 | 13 | 10 | 8 | **402** |

**Tabella 4:** Risultati dell'AdaptationEngine su presentazioni di alta qualità

Questa osservazione quindi suggerirebbe di concludere che l'*AdaptationEngine* non sia sensibile alla distanza tra gli InternalProfile. In realtà probabilmente questo esperimento da solo non è sufficiente per arrivare ad una conclusione definitiva del problema. Infatti, nel particolare esperimento considerato, per scelte legate all'implementazione del prototipo, un InternalProfile può differenziarsi da un altro solo per un singolo attributo: la dimensione dei frame della presentazione. Considerando invece il caso generale, InternalProfile molto diversi richiederebbero l'esecuzione di un numero di trasformazioni elementari elevato, quindi ci si aspetta che aumenti il tempo necessario per la selezione degli adattatori. Ulteriori esperimenti saranno svolti in futuro per verificare queste considerazioni.

Considerando i risultati, ottenuti dallo stesso test e riportati in **Tabella 5**, relativi all'incidenza delle attività svolte dall'AdaptationEngine sull'inizializzazione dello

streaming si possono fare le seguenti osservazioni:

1. L'incidenza dell'AdaptationEngine sull'inzializzazione dello streaming è circa la stessa di quella riscontrata nel precedente test in cui la presentazione adattata era di livello medio, a conferma della presunta indipendenza del ritardo introdotto da questo componente dalla distanza tra profilo e metadato.
2. Sempre rispetto al test svolto in precedenza, si rileva un aumento del tempo di inizializzazione medio dello streaming (da 7,1 a 8 sec) a indicare il fatto che la maggiore distanza tra profilo e metadato implica un maggior sforzo da parte dell'adattatore. L'esperimento quindi conforta la scelta di utilizzare la similarità tra gli InternalProfile per valutare la convenienza di una trasformazione.

| | **Inizializzazione dello streming** | **Ritardo dell' AdaptationEngine** | **Incidenza in % dell'AdaptationEngine** |
|---|---|---|---|
| Prova1 | 21743 | 2193 | **10.03** |
| Prova2 | 5106 | 17 | **0.31** |
| Prova3 | 4271 | 12 | **0.38** |
| Prova4 | 4034 | 10 | **0.24** |
| Prova5 | 5237 | 9 | **0.17** |
| **Media** | **8078.2** | **398.4** | **2.22** |

**Tabella 5:** Incidenza dell'AdaptationEngine per presentazioni ad alto livello di qualità

#### 7.3.1.4 Considerazioni sulla politica di adattamento

Attraverso il test presentato nel precedente paragrafo è stata compreso l'effettiva rilevanza della similarità tra InternalProfile. In questa sezione invece, si intende valutare il peso della distanza della locazione fisica della presentazione dal place cliente. A questo scopo, si prenderà in considerazione il tempo di inizializzazione del Service Path a partire dalla richiesta di una presentazione di livello medio, nel caso di una topologia con tre nodi, già a disposizione nella **Tabella 3**, e lo si confronterà con quello di un topologia con due nodi del tipo di Figura **7.7a**.

Il confronto dei risultati ottenuti, rappresentato in Tabella 6, mostra come all'aumentare del numero di nodi coinvolti nel ServicePath il tempo necessario alla prima inizializzazione dello streaming cresca.

| | |
|---|---|
| 2 nodi | 15110 |
| 3 nodi | 21149 |

**Tabella 6:** Confronto sul tempo di inizializzazione dello Streaming

Il costo di consegna di una presentazione collocata in un place lontano dal quello cliente è molto più elevato del costo di adattamento di lontana dal punto di vista qualitativo a quella voluta. Quindi potrebbe essere conveniente utilizzare politiche di adattamento che diano più peso alla località delle presentazioni piuttosto che alla loro similarità con l'InteranlProfile relativa al profilo utente.

### 7.3.2 Ritardo dell'adattamento percepito dall'utente

Un modo per valutare il ritardo introdotto dal processo di adattamento è quello di far partire contemporaneamente due richieste relative alla stessa presentazione: su una di queste viene effettuato l'adattamento, sull'altra invece viene fatto semplicemente lo streaming del flusso multimediale richiesto. La differenza trai i tempi di inizializzazione dello streaming ottenuti, viene chiamata delay ed indica proprio il ritardo introdotto dal processo di adattamento nell'inizializzazione dello streaming.

Sono stati svolti due test di questo tipo sulla stessa topologia (quella di Figura **7.7b**) in cui il primo considera una presentazione di livello medio (risultati in **Tabella 7**), mentre il secondo una di livello alto (risultati in **Tabella 8**).

| | Adattamento abilitato | Streaming tradizionale | **Delay** |
|---|---|---|---|
| Prova1 | 21149 | 17049 | 4100 |
| Prova2 | 3799 | 1893 | 1906 |
| Prova3 | 3331 | 1302 | 2029 |
| Prova4 | 3754 | 1512 | 2242 |
| Prova5 | 3622 | 1422 | 2200 |
| Prova6 | 3640 | 1368 | 2636 |
| ***Media senza prima inizializzazione*** | | | **2202.6** |

**Tabella 7:** Risultati sul delay con presentazione di livello medio

| | Adattamento abilitato | Fruzione tradizionale | **Delay** |
|---|---|---|---|
| Prova1 | 21743 | 17013 | 4640 |
| Prova2 | 5106 | 1967 | 3139 |
| Prova3 | 4271 | 1554 | 2717 |
| Prova4 | 4034 | 1434 | 3033 |
| Prova5 | 4530 | 1497 | 3017 |
| Prova6 | 4330 | 1313 | 2857 |
| ***Media senza prima inizializzazione*** | | | **2952.6** |

**Tabella 8:** Risultati sul delay con presentazione di livello alto

Comparando i risultati ottenuti nei due casi si può osservare come il flusso adattato viene ricevuto con ritardo medio di circa due secondi e mezzo. Inoltre a conferma di quanto detto in precedenza l'adattamento di una presentazione più distante dal profilo dal profilo utente comporta una aumento del ritardo, dovuto al maggior sforzo che deve essere compiuto dagli adattatori per compiere la trasformazione.

Inoltre, i risultati mostrano che il ritardo introdotto alla prima inizializzazione è in media quasi doppio a quello introdotto nelle inizializzazioni successive. Questo risultato può essere spiegato considerando che alla prima inizializzazione bisogna configurare ogni nodo del Service Path, scaricando, se necessario il codice su ogni place appartenente ad esso, ed eventualmente attivando il server.

Tuttavia il delay misurato si riferisce solo alla inizializzazione dello streaming e non a quello introdotto sul singolo frame, che ci si aspetta essere molto meno pesante e che sarà oggetto di valutazione in ulteriori esperimenti futuri.

### 7.3.3 Scalabilità dell'AdaptationEngine

Al fine di valutare il comportamento del sistema nelle varie situazioni di carico sono stati condotti dei test di scalabilità. Nel primo test effettuato, una sola presentazione corrispondente al titolo richiesto I test effettuati consistono nel lanciare un numero N di richieste crescente per valutare ad ogni prova il tempo medio di risposta del sistema verso ognuno di essi. Le richieste richiedono tutte l'adattamento di una presentazione di media qualità ad un profilo di basso livello. In **Tabella 9** sono riportati i risultati raccolti in questi test in cui il tempo di medio di risposta dell'AdaptationEngine è espresso in millisecondi

| Numero di clienti in parallelo | Tempo medio di risposta |
|---|---|
| **1** | 984 |
| **10** | 1390 |
| **20** | 1861 |
| **30** | 2319 |
| **40** | 2853 |
| **60** | 3949 |
| **80** | 4266 |
| **100** | 4578 |
| **150** | 5728 |
| **200** | 7147 |
| **250** | 7669 |
| **300** | 9014 |
| **350** | 10540 |
| **400** | 11440 |
| **450** | 12108 |
| **500** | 13308 |
| **750** | 19255 |
| **1000** | 25588 |

**Tabella 9:** Tempo medio di risposta dell'AdaptationEngine all'aumentare dei clienti

In Figura **7.8** i risultati raccolti vengono riportati in grafico che mostra una crescita lineare del tempo medio di risposta all'aumentare richieste effettuate in parallelo. Questo è il migliore risultato che si poteva ottenere considerando che l'AdaptationEngine lavora utilizzando solo le risorse di elaborazione della macchina locale e che non utilizza strategie di esecuzione che gli permetto di elaborare una sola risposta per gruppi di richieste dello stesso tipo. Il numero di clienti in grado di essere gestiti dipende dunque solo dalla disponibilità delle risorse locali.

Si può notare che la pendenza iniziale della curva è leggermente superiore rispetto a quella di regime. Questa diminuzione della pendenza può essere spiegata poiché dopo un certa quantità di tempo qualcuno degli agenti lanciati effettua il recupero delle informazioni che vengono memorizzate dal sistema e riutilizzate quando anche gli altri clienti le richiedono.

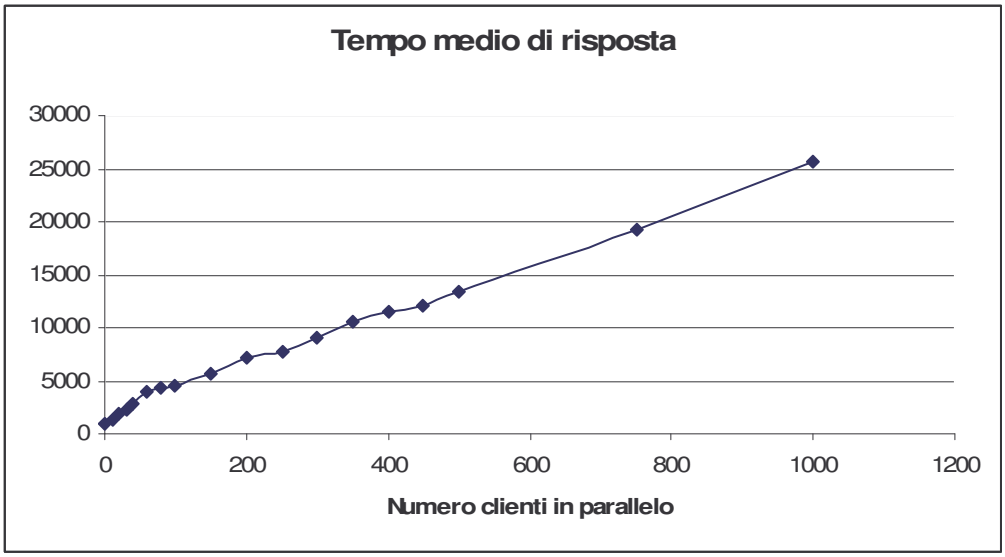



**Figura 7.8:** Tempo medio di risposta dell'AdaptationEngine all'aumentare del numero di richieste

## 7.4 Conclusione

In quest'ultimo capitolo sono stati affrontati gli aspetti legati all'implementazione del servizio di adattamento evidenziando le scelte e le ipotesi effettuate durante la realizzazione di un prototipo del sistema di adattamento. Inoltre sono stati presentati le prove fatte sul sistema durante la fase di testing. Esse hanno dimostrato che l'incidenza dell'AdaptationEngine sull'inizializzazione dello streaming è molto ridotta. Al contrario essa risulta essere abbastanza appesantita dall'inizializzazione dei proxy che effettuano l'adattamento. Una possibile soluzione a questo problema potrebbe essere la realizzazione dell'approccio, solamente discusso nei capitoli precedenti, di adattare il prefisso della presentazione richiesta più vicino al cliente, in modo da anticipare la ricezione dello streaming da parte dell'utente. Questo approccio permetterebbe di svolgere durante l'erogazione del prefisso, l'inizializzazione del parte restante del Service Path in modo del tutto trasparente all'utente. Altri esprimenti hanno mostrato che l'AdaptationEngine ha problemi di scalabilità dovuti al fatto che la sua elaborazione sfrutta solo le risorse della macchina locale in cui sono arrivate le richieste.

# CONCLUSIONI

Il lavoro di tesi svolto ha individuato nell'adattamento dei contenuti un valido strumento per la gestione dell'eterogeneità che caratterizza i servizi multimediali.

Dopo aver indagato sulle varie strategie di adattamento utilizzabili nella realizzazione di un sistema di personalizzazione dei contenuti, è stata sviluppata un'architettura che configura e svolge un adattamento dinamico di flussi multimediali orientato alle caratteristiche del dispositivo utilizzato dall'utente per accedere al servizio.

Il sistema sviluppato è stato progettato per essere integrato in MUM, una infrastruttura di supporto, volta alla semplificazione dello sviluppo applicativo delle applicazioni multimediali, che attraverso questo progetto di tesi è stata arricchita di un servizio di adattamento che sfrutta il servizio di configurazione dei nodi che partecipano alla fruizione del servizio.

L'adattamento svolto viene effettuato sulla base delle informazioni che caratterizzano il dispositivo dell'unente raccolte all'interno di un "profilo" e di quelle che definiscono il formato delle presentazioni rappresentate all'interno di un "metadato".

L'architettura proposta separa concettualmente il sistema in due parti, la prima, dedicata alla configurazione del servizio di adattamento si occupa di decidere quali tra le presentazioni presenti nel sistema è conveniente adattare, quali adattatori utilizzare per svolgere la trasformazione e come distribuirli lungo il Service Path; la seconda invece realizza un componente proxy configurabile che si occupa di effettuare le operazioni sul flusso multimediale utilizzando gli adattatori prescelti in fase di configurazione.

I primi esprimenti condotti sul sistema realizzato mostrano che l'incidenza del componente di configurazione dell'adattamento sull'inizializzazione dello streaming è molto bassa. Al contrario sembra essere più pesante l'inizializzazione degli adattatori lungo il service path. Ulteriori accertamenti, comunque sono in fase di svolgimento per verificare l'efficienza del sistema una volta inizializzato.

Sviluppi futuri del sistema potranno essere svolti nell'intento di offrire un adattamento orientato non solo alla caratteristiche del dispositivo, ma anche alle

preferenze dell'utente. Inoltre, in situazioni in cui i vincoli del dispositivo sono così limitanti da non renderlo abilitato alla ricezione di flussi audio-video, potrebbe essere interessante realizzare un adattamento multi-modale per trasformare le informazioni in un tipo di media diverso da quello originale, come ad esempio immagini o testo.

# BIBLIOGRAFIA


**[Adapt]** Zhijum Lei, Georganos, "Context-Based Adaptation in Pervasive Computing.

**[Anind00]** Anind K., Gregory A., "Towards a better understanding of context and context-awarness", 2000.

**[CCPP]** Composite Capability/Preference Profiles: Structure and Vocabularies 1.0, WRC Raccomandation 15/01/2004.
http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

**[DC]** DublinCore
http://www.dublincore.org

**[Expro]** Agostini A., Bettini C., Cesa-Bianchi N, Maggiorini D., Riboni D., "Integrated Profile and Policy Management for mobile oriented Internet service", Technical Report FIRB-Web Minds, 2004.

**[Fos03]** Foschini L., "Gestione di flussi multimediali in reti integrate fisse e moblili", tesi di laurea 2003.

**[Fugg04]** Fuggetta A., "Integrare la molteplicità: il futuro dei sistemi di telecomunicazioni", rubrica Espansione 2004.

**[InfoPyramid]** Raikesh Mohan, J.R. Smith, Chung-Sheg-Li, Adapting Multimedia intent context for Universal Access".

**[Intel2002]** Intel PCA Device Profile, CC/PP Client Profile for Intel PCA Device. Design guide, August 2002.

**[ISMASpec]** Internet Streaming Media Alliance Implementation Specifiations v 1.0, 03/06/2004.
http://www.isma.org

**[ITUH264]** ITU H264 Video Compression
http://www.h263l.com

**[JMF]** Java Media Framework home page
http://java.sun.com/products/javamedia/jmf/

**[JSR188]** Qusay H. Mahmoud, "Getting Started with Conposite/Preferences

Profile and JSR-188, ottobre 2004.
http://www.devolepers.sun.com/

**[Lau02]** Wai Lum, Francis Lau, "A Context-aware Decision Engine for Content Adaptation" , IEEE 2002.

**[Liberty03]** Liberty Alliance Liberty ID-SIS Personal Profile Service Specification Version 1.0, 2003.

**[Log4j]** Log4j Home Page.
http://logging.apache.org/log4j/docs/

**[Lum02]** Wai Lum, Francis Lau, "On Balancing Between Transcoding Overhead ad Spatial Consumption in Content Adaptation".

**[MPEG7]** MPEG-7 documentation
http://www.archive.dstc.edu.au/mpeg7-dll/

**[MUM]** MUM Home Page.
http://www.lia.deis.unibo.it/Research/MUM/

**[MUM04]** P. Bellavista, A. Corradi, L. Foschini, "MUM: a middleware for the provisioning of Continuous Services to Mobile Users", 2004.

**[MUMOC]** P. Bellavista, A. Corradi, L. Foschini, "MUMOC: an active infrastructure for open Video Caching", 2005.

**[NAC]** Tayeb Lemloula, Nabi Layeida, "NAC: A basic core for the Adaptation and Negotiation of multimedia service", 2002.

**[RDF]** RDF/XML Syntax Specification
http://www.w3.org/TR/2003/WD-rdf-syntax-grammar-20030123

**[Shilit]** Context-aware computing Applications, 1° International Workshop on Mobile Computing System and Applications, 1994.

**[SOMA03]** Bellavista P., Corradi A., Stefanelli C.,"Mobile Agent Middleware to support Mobile Computing", IEEE Computer, Vol. 34, No 3, pag 73-81, March 2001.

**[Torl2004]** Torlone R., Cuffio A., Del Nostro P., "Strumenti per la produzione di siti web adattativi: Coordinate di Adattamento", 2003.

**[UAProf]** UAProf User Agent Profiling Specification, 1999.
http://www.wapforum.org/what/tecnical.htm

**[Video]** Materiale sulla compressione video
http://www.cs.sfu.ca/CourseCentral/365/li/material/notes/Chap4/

**[Want92]** Wants. R., Happer A., Falcao V.,Gibbons J., "The active badge Location System", ACM Transaction and Information Systems, 1992.

**[XMLSchema]** XML Schema Part 0,
World Wide Web Consortium Raccomandation 02/05/01.
http://www.w3.org./TR/2001/REC-xmlschema-0-20010502/

# RINGRAZIAMENTI

Vorrei ringraziare la mia famiglia, per aver creduto in me e per avermi dato la possibilità di raggiungere questo importante traguardo ed in particolare mio fratello Simone per essermi stato vicino nei momenti di difficoltà e a cui auguro presto di riuscire ad ottenere la soddisfazione di terminare il percorso di studio che ha scelto.
Un caloroso e affettuoso ringraziamento ai tutti i miei amici, sia quelli che hanno condiviso con me questa esperienza universitaria sia quelli che seppur lontani mi hanno sempre fatto sentire il loro affetto e la loro amicizia.
In fine un particolare ringraziamento al Prof. Corradi e al Dott. Foschini per avermi dato la possibilità di svolgere questa tesi e per la disponibilità dimostrata nel fornire sempre preziosi consigli e suggerimenti utili al miglioramento del lavoro svolto.