Servizi mobili in modalità “push” nell’architettura “Simple Mobile Services”

1 Premessa

Il progetto Simple Mobile Services [1] definisce una architettura per l’esecuzione e la creazione di servizi per terminali mobili. L’architettura è basata sui principi della “SOA” Service Oriented Architecture e prevede un insieme di componenti che possono interagire. L’interazione tra i componenti è supportata da un livello di middleware, detto “SMILE” [2] che fornisce un insieme di primitive per la ricerca di componenti remoti e per lo scambio di messaggi sia sincrono che asincrono. Con riferimento alla Figura 1, il middleware SMILE è quella parte della “SMS Service Execution Plarform” che consente agli “SMS Component Services” di interagire tra di loro.

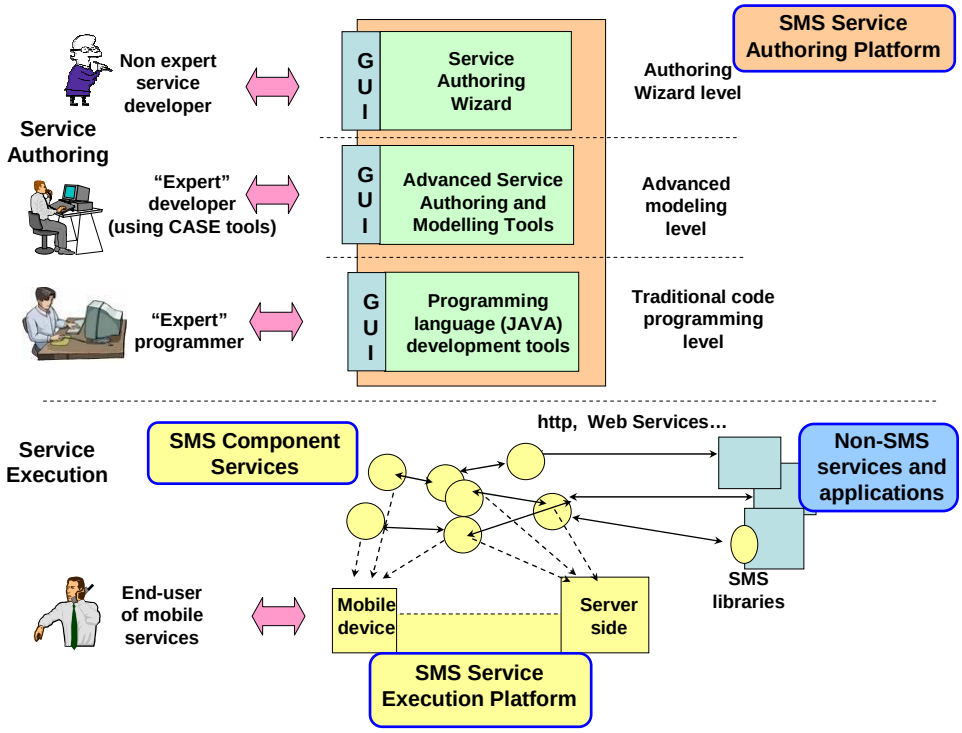

Figura 1: Visione d’insieme dell’architettura Simple Mobile Service

Dal lato dei terminali mobili si utilizza una applicazione sviluppata in J2ME chiamata MOVE, che include un insieme di componenti lato “mobile device”.
Dal lato server i componenti sono sviluppati principalmente in JAVA e possono essere delle applicazioni “stand alone” o delle “servlets” che girano su un “application server”.
L’applicazione MOVE utilizza un “GUI” framework chiamato Thinlet che offre un insieme di GUI “widgets” e consente di definire l’interfaccia utente attraverso file XML utilizzando un linguaggio simile a “XUL”. All’interno delle pagine XML sono definite le “azioni” che MOVE deve eseguire in risposta agli input da parte dell’utente (es. la pressione di un tasto o la selezione di un item di un menu).

Un servizio lato “mobile device” può essere realizzato con una sequenza di pagine Thinlet/XML, in modo simile a come un servizio può essere realizzato su un browser con una sequenza di pagine html. Una pagina Thinlet può contenere come “azione” la richiesta di caricare un'altra pagina Thinet, esattamente come gli hyperlink in http.

A titolo esemplificativo, ecco una possibile lista di “servizi” fruibili sulla applicazione MOVE

“Cerca un negozio”
“Cera un ristorante nell’aeroporto”
    “Ristorante X”
    “Ristorante Y”
    “Ristorante Z”
“Cerca un ristorante ad Atene – Guida Michelin”
“Cerca un ristorante ad Atene – Guida Slow Food”
“Elenco delle guide dei ristoranti di Atene”
“Cerca il gate”
“Trova i tuoi amici nell’aeroporto”
“Compra un biglietto aereo”
“Gioca al gioco xxx on-line”
“Compra un biglietto per l’opera”
“Eventi sportivi oggi ad Atene”
“Risultati delle partite di calcio in Italia”
“Notizie del telegiornale (italiano)”

Le pagine Thinlet/XML corrispondenti ai servizi possono essere pre-caricate sulla applicazione MOVE oppure scaricate dinamicamente. Nel caso vengano scaricate dinamicamente si può pensare di procedere in modo simile al web in cui l’utente cerca i servizi (es. con google) e poi dalla lista di servizi risultati scarica quelli che gli interessano. Chiamiamo questo approccio “pull” perché il client richiede di scaricare le pagine contenenti i servizi dal server.

Nel caso dei servizi mobili questo approccio di tipo “pull” ha degli svantaggi, dovuti alla lentezza dello scaricamento delle pagine, limitate capacità grafiche del terminale per mostrare le liste dei servizi, minore “pazienza” degli utenti nell’aspettare prima di ricevere le pagine e nel fare diversi tentativi prima di trovare il servizio giusto.

Idealmente, un’architettura per i servizi mobili dovrebbe consentire di avere disponibili i servizi giusti al momento giusto ed evitare per quanto possibile le attese dovute allo scaricamento delle pagine. Per questo si può introdurre un approccio diverso, detto “push” in cui i server inviano le pagine al “mobile client” cercando di anticipare i suoi bisogni.

2 Descrizione del progetto

Il componente da sviluppare è un “broker” che si preoccupa di fornire in modalità push al mobile client i servizi più “adatti”, cercando di anticipare i bisogni del cliente stesso. Il broker cercherà di “ordinare” i possibili servizi in ordine di priorità per uno specifico utente. Il broker potrà poi implementare dei “vincoli” sulla quantità di servizi/pagine che è possibile inviare all’utente in un certo intervallo di tempo (ad esempio max 1 MB / giorno o 200 KB/ ora). Inoltre va tenuto in conto la capacità del terminale (numero di pagine o spazio di memoria disponibile per le pagine dei servizi).

Il broker gestisce un catalogo di servizi (implementato con un database), per ciascun servizio possono essere associate informazioni come:

1) localizzazione del servizio (es. posizione e raggio di interesse)
2) eventuali relazioni con ora e data di possibile maggior uso

3) profilo del servizio (es. categoria: ristorante/negozio/turismo ecc.)

Per ordinare i servizi in ordine di rilevanze il “broker” si può avvalere delle seguenti informazioni:
1) localizzazione corrente dell’utente
2) ora e data
3) profilo dell’utente (es. preferenze)

In più il broker può basare le sue informazioni sulla base del comportamento degli utenti: es. popolarità dei servizi. Si può pensare che il mobile client comunichi al server i servizi che ha effettivamente utilizzato, in modo che il server potrà costruire un “profilo di utente” con le informazione relative ai servizi che l’utente ha usato. Il server potrebbe basare le sue scelte sulla combinazione della popolarità dei servizi (quanto tutti gli utenti hanno utilizzato i servizi) e sulla storia dello specifico utente (quanto l’utente ha utilizzato quel servizio in precedenza o servizi simili a questo).

Non si richiede che si utilizzino tutte le informazioni elencate sopra, il sistema potrà essere inizialmente sviluppato in forma semplificata, ad esempio sfruttando solo l’informazione di localizzazione, ma dovrà essere pensato in modo modulare per includere estensioni ulteriori.

2.1 Primitive e funzionalità offerte dalla piattaforma esistente.

Per sviluppare il componente si sfrutteranno ove possibile le funzionalità già sviluppate dell’architettura Simple Mobile Services [3].

Gestione del profilo degli utenti

Gli utenti con i loro profili sono memorizzati in un DB mySQL, la piattaforma Joomla viene utilizzata per la registrazione / modifica dei dati di profilo (questo è già disponibile dal progetto SMS).

Gestione del catalogo dei servizi

[Da chiarire se ci saranno funzionalità rese disponibili dalla piattaforma SMS. Si può assumere per il momento che il catalogo dei servizi debba essere realizzato.]

Primitive di comunicazione tra componenti:

PushPage

Allows a server to send a page to a SMS user. It is the page server which, according to its specific contents, can originate autonomously the notification.
This is typically implemented as an asynchronous message. No confirmation of reception is provided. Unlike RSS, which is based on HTTP and thus on a polling mechanism, this is a push based mechanism and doesn't keep the typical heavy overhead brought by the polling mechanism.

OUT PushPageMessage

pageID	String	Rapresents the page's unique identifier.
page	String	String Contains the page represented as an XML stream compliant with XUL.
displayPoint	String	Describes where to show the entry point inside the client.

Update position

Allows a user to notify a server about his current position.

OUT UpdatePositionMessage

position	Position	Contains the position of the user
userID	UserID	The ID of the user

Le primitive di comunicazione tra componenti vengono descritte utilizzando un semplice linguaggio di descrizione delle interfacce detto IDLight (Interface Definition Light) [4], che è un sottoinsieme di WSDL [5] con una sintassi semplificata. Ad esempio la primitiva di update position mostrata sopra si definisce in questo modo:

```
@Notification UpdatePosition
        @Summary
                Allows a user to notify a server about his current position
        @Out UpdatePositionMessage
                @Param position:Position      Contains the position of the user
                @Param userID:UserID          The ID of the user
```

2.2 Funzionalità da realizzare

Lista dei servizi rilevanti per un utente: data la posizione dell'utente (e le eventuali altre informazioni disponibili), fornire una lista ordinata dei servizi "adatti" all'utente.

> definizione della funzione di ranking

Decisione di quali servizi inviare. Il broker decide a partire dalla lista ordinata quali servizi devono essere inviati. Il broker deve tenere in conto di quali servizi sono già disponibili sul terminale utente e qual è la capacità del terminale stesso.

> vanno definiti i campi del DB dei servizi, in particolare i dati che serviranno alla funzione di ranking

> va definito come memorizzare i servizi che sono già disponibili sul terminale utente: servono dei messaggi per sincronizzare l'informazione tra client e server ?

Realizzazione di un catalogo di servizi (saranno servizi fittizi!) con associate informazioni di profilo di servizio, localizzazione ecc. Il catalogo dei servizi può essere realizzato con un DB mySQL, non è necessario realizzare interfacce di utente apposite per la gestione di questo DB, bastano le interfacce fornite dai tool grafici di gestione di mySQL (es. phpmyadmin).

Simulazione dei movimenti degli utenti e dell'invio dei dati di posizione al server di localizzazione.

Valutazione della "popolarità" dei servizi: quanto i servizi vengono usati. - "Profilazione" degli utenti: memorizzazione per ciascun utente di quali servizi ha usufruito. Il mobile client invierà dei messaggi al broker comunicando quali servizi vengono usati.

> va definito il DB per memorizzare queste informazioni di utilizzo dei servizi

vanno definiti i messaggi tra mobile client e broker per comunicare l'utilizzo dei servizi e le modalità di comunicazione (es. dopo ciascun uso di un servizio, riepilogo periodico, interrogazione da parte del broker)

Le interfacce tra componenti remoti vanno definite utilizzando il linguaggio IDLigth.

3 Riferimenti

[1] Simple Mobile Service project: http://www.ist-sms.org
[2] G. Bartolomeo, S. Salsano, R. Glaschick, "A Glimpse into SMILE Programming", Technical Report, available at http://netgroup.uniroma2.it/twiki/bin/view.cgi/SMS/TechnicalReports
[3] Simple Mobile Services project: on-line platform and documentation: http://netgroup.uniroma2.it/twiki/bin/view.cgi/SMS/SMSPlatformHome
[4] S. Salsano, G. Bartolomeo, R. Glaschick, "IDLight (Interface Definition Light)", Technical Report, available at http://netgroup.uniroma2.it/twiki/bin/view.cgi/SMS/TechnicalReports
[5] WSDL – Web Services Description Language, http://en.wikipedia.org/wiki/Web_Services_Description_Language