UNIVERSITÀ DEGLI STUDI DI PISA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Tecnologie Informatiche

TESI DI LAUREA

Progettazione e Generazione Automatica di Interfacce Utenti per TV Digitale su Piattaforma MHP

Candidato
Sandro Sansone

Relatore
Fabio Paternò

Controrelatore
Antonio Cisternino

Anno Accademico 2005–2006

Alla mia famiglia

Sommario

Lo scopo della tesi è quello di creare un ambiente di progettazione e generazione di interfacce utenti per la TV digitale. È stato realizzato un editor per la progettazione di interfacce utenti per la TV digitale su piattaforma MHP usando descrizioni logiche dalle quali, mediante una trasformazione, si ottiene l'implementazione in *Xlet* (applicazioni Java per TV digitale). Tale trasformazione è stata poi integrata in un ambiente di supporto per la migrazione dinamica di interfacce utenti da dispositivi, quali Desktop o PDA, verso TV digitale.

Indice

Capitolo 1

Introduzione

1.1 Introduzione alla TV interattiva

Come è accaduto diversi anni fa con internet e i telefoni cellulari, sembra essere arrivato il momento di una nuova mutazione tecnologica: la Televisione Digitale. Questa nuova tecnologia è nata da un progetto europeo, il DVB (Digital Video Broadcasting) [20], promosso dalla commissione europea al fine di definire uno standard comune per le trasmissioni televisive digitali via satellite (DTV-S), via cavo (DVB-C) e via terra (DVB-T). Questi standard sono successivamente stati adottati anche da paesi non europei come il Giappone.

Le specifiche tecniche approvate dal DVB vengono successivamente ratificate dall'ETSI [22] che ne attribuisce la veste di standard europei. I vantaggi di questo standard sono riassumibili in due categorie:

- potenziamento del servizio in termini di quantità e qualità;
- offerta di una serie di servizi di tipo interattivo.

Infatti a parità di frequenze utilizzate per le reti televisive analogiche, il numero di canali digitali irradiabili potrebbe quintuplicarsi. Inoltre la trasmissione digitale offre una migliore qualità audio/video e permette l'utilizzo di schermi televisivi di grande formato. Il broadcaster può inoltre utilizzare i mezzi di trasmissione in modo più flessibile, potendo, in certe zone, ridurre il numero

di programmi trasmessi e migliorarne la qualità. Con la televisione digitale è altresì possibile diffondere una serie di servizi interattivi utilizzabili dal ricevitore televisivo, che è stato trasformato in un vero e proprio terminale, mediante l'uso del ricevitore digitale terrestre, il Set-Top-Box (STB), con la capacità di elaborazione e immagazzinamento di dati. Quindi la televisione sembra destinata a far parte della nostra rete domestica e si presuppone possa fruire dei servizi che tale rete offre. Ciò significa che anche nelle case prive di personal computer sarà possibile usufruire dei servizi associati ad internet.

L'utilizzo di applicazioni interattive sul STB è reso possibile dalla piattaforma MHP [22] (Multimedia Home Platform) standardizzata da DVB. Il DVB-MHP è il nuovo standard aperto, accettato sia in Europa che in Asia, per la televisione digitale interattiva. Esso racchiude un insieme di API Java e definisce l'interfaccia tra le applicazioni interattive e i terminali sulle quali le applicazioni vengono eseguite.

MHP è inoltre alla base della Open Cable Specification [36] (OCAP, specifiche aperte per la TV via cavo) negli USA. Tali specifiche sono state definite e sviluppate dalla Open Cable che quindi definisce lo standard per lo sviluppo di applicazioni interattive negli Stati Uniti.

1.2 Struttura della tesi

Il lavoro svolto in questa tesi si inserisce in un contesto di ricerca più ampio che mira a fornire supporto, sia in fase di progettazione che in fase di esecuzione, per ottenere interfacce utenti in ambienti multi dispositivi partendo da descrizioni logiche. Inizialmente questa ricerca si era focalizzata solo in ambienti Web. Successivamente ci si è resi conto che la TV digitale sta assumendo una importanza crescente.

Quindi lo scopo della tesi è quello di colmare questa lacuna ed affrontare le problematiche specifiche per supportare la progettazione e la generazione di interfacce utenti per questa piattaforma.

Questa tesi è organizzata in sei capitoli. L'ordine dei capitoli rispecchia in maniera reale il susseguirsi del lavoro, che può essere suddiviso in fasi.

Fase di studio delle caratteristiche della TV digitale e delle interfacce utenti:

- Studio della piattaforma MHP e delle tecnologie esistenti per lo sviluppo di applicazioni per la TV digitale. [Capitolo 2]
- Analisi di linee guida per la progettazione (anche in termini di usabilità) di applicazioni interattive. Verranno date delle *guidelines* per lo sviluppo di tali applicazioni. In particolare si analizzeranno le peculiarità della televisione digitale, e in base a tali caratteristiche, si spiegherà come costruire una interfaccia utente usabile. [Capitolo 3]

Fase di progettazione e implementazione:

- Estensione del tool TERESA [1] per permettere la progettazione e la generazione di interfacce utenti per la TV digitale. [Capitolo 4]
- Integrazione del processo di generazione di interfacce utenti per la TV digitale a partire da descrizioni logiche in un ambiente di migrazione [7]. È stata aggiunta la possibilità di migrare su un dispositivo televisivo digitale a partire da un altro dispositivo come un PC desktop o un palmare. In tale processo viene generata un'interfaccia utente per il dispositivo destinatario (in questo caso una TV digitale) mantenente lo stato in cui si trovava nel dispositivo sorgente (ad esempio se nel dispositivo sorgente sono stati compilati dei campi di una form, tali campi risulteranno già compilati anche nel dispositivo verso il quale è stata effettuata la migrazione). [Capitolo 5]

Inoltre la tesi inizia con questo capitolo introduttivo [Capitolo 1] e termina con un capitolo conclusivo [Capitolo 6] contenente un riassunto dei punti visti in precedenza e la presentazione di possibili estensioni future.

Capitolo 2

Multimedia Home Platform

2.1 Introduzione

Il Multimedia Home Platform è uno standard progettato dal DVB Project per l'utilizzo di applicazioni interattive nella televisione digitale.

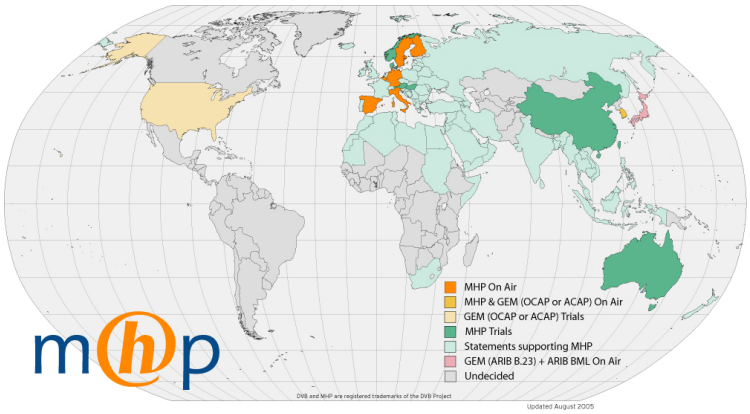

Figura 2.1: Diffusione di MHP nel mondo

Nella sua concezione più semplice, MHP è un insieme di API Java che permettono la scrittura di applicazioni interoperabili che vengono trasmesse sullo schermo sotto forma di stream MPEG-2. Un ricevitore MHP collegato all'apparecchio televisivo è in grado di eseguire tali applicazioni. MHP quin-

di definisce una interfaccia tra l'applicazione ed il terminale sul quale verrà eseguita.
Aver definito uno standard è stato quindi utile al fine di separare le applicazioni scritte da diversi fornitori di servizi dall'hardware specifico del terminale sul quale verrà eseguita l'applicazione.

Si inizia a parlare di MHP in una conferenza nel 1994, ma la prima versione delle specifiche MHP fu rilasciata, dopo anni di duro lavoro, nel 2000 dall'ETSI [14]. MHP estende gli standard definiti da DVB riguardo le trasmissioni televisive che utilizzano il segnale digitale, includendo i servizi interattivi, e per questo motivo le applicazioni DVB-MHP sono destinate alla TV digitale in tutte le sue versioni: satellitare, terrestre o via cavo. Il DVB Project ha scelto Java come linguaggio per lo sviluppo di MHP. In tal modo è possibile sviluppare applicazioni indipendentemente dallo strato sottostante, ovvero dal tipo di decoder, sia esso via satellite, via cavo o via terra, sul quale l'applicazione verrà eseguita.

2.2 I profili MHP

MHP definisce tre profili:

1. The Enhanced Broadcast Profile (Profilo 1): questo profilo permette il normale flusso audio e video e inoltre dà la possibilità alle applicazioni di poter essere eseguite in locale. È il profilo base ed è adatto a Set-Top-Box con nessuna capacità del canale di ritorno;

2. The Interactive Broadcast Profile (Profilo 2): questo profilo include il supporto per il canale di ritorno (tipicamente PSTN v.90). Le applicazioni possono scaricare le classi tramite il canale di ritorno, mentre nell' Enhanced Broadcast Profile ciò è possibile solamente utilizzando il canale broadcast;

3. The Internet Access Profile (Profilo 3): questo profilo include il supporto per applicazioni internet come e-mail o browser.

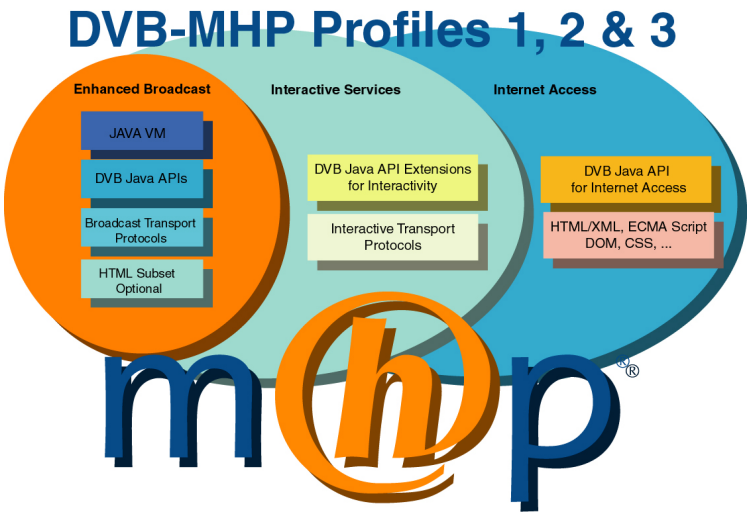

Figura 2.2: I profili MHP

MHP 1.0 (TS 201 812) [14] è lo standard di base. Esso specifica l'ambiente dove si possono eseguire le applicazioni per la TV digitale indipendentemente dall'hardware e dal software sottostante che sono specifici di chi produce Set-Top-Box.
MHP 1.0 implementa i profili Enhanced Broadcast Profile e Interactive Broadcast Profile e quindi contiene:

- Java Virtual Machine;
- API Java DVB;
- diversi formati come: MPEG-2, JPEG, GIF e PNG;
- protocolli di trasporto broadcast;
- protocolli di trasporto per il canale di ritorno (include il protocollo IP).

MHP 1.1 (TS 102 812) è l'estensione di MHP 1.0, nata per implementare il profilo di Internet Access. Essa include:

- API per l'utilizzo di client internet;

- API per la comunicazione con le smart card;
- possibilità di memorizzazione di dati in memoria persistente;
- download delle applicazioni mediante il canale di ritorno;
- possibilità di eseguire applicazioni DVB-HTML.

2.3 Applicazioni DVB-J

Il core di MHP è sviluppato in Java, adottando una Java Virtual Machine ed una serie di API Java. Ruolo fondamentale viene assunto dalle API JavaTV [25]. Tali API permettono di poter sviluppare applicazioni DVB-J, più comunemente conosciute sotto il nome di Xlet. Le Xlet sono particolari applicazioni Java, concepite per essere eseguite sui decoder interattivi nei quali un software specifico, l'Application Manager, è in grado di controllarne il ciclo di vita. Esse sono molto simili alle Applet, dove l'application manager risulta essere il browser. Tali applicazioni sono scritte in java e consistono in un insieme di classi scaricabili via broadcast o, nel caso di MHP 1.1, potendo utilizzare il canale di ritorno.
Le API JavaTV forniscono la possibilità di poter accedere ai servizi offerti dai ricevitori digitali. Inoltre sfruttano le Java Media Framework (JMF) per la gestione di dati multimediali (audio/video) in applicazioni (Xlet nel nostro caso). Esse sono dotate di un livello di astrazione che permette di poter eseguire l'applicazione su un qualunque dispositivo in grado di supportarle, a differenza dello standard MHP, che permette di eseguirle solamente su sistemi DVB per la TV digitale. Le API JavaTV non costituiscono uno standard per la TV digitale. Tuttavia sono incluse in molti standard emanati.

2.3.1 Ciclo di vita delle Xlet

Il package più importante delle API JavaTV è `javax.tv.xlet` che contiene la definizione dell'interfaccia Xlet:

```
public interface Xlet {

   public void initXlet(XletContext ctx)
      throws XletstateChangeException;

   public void startXlet()
      throws XletstateChangeException;

   public void pauseXlet();

   public void destroyXlet(boolean unconditional)
      throws XletstateChangeException;

}
```

Come per le applet, anche per le Xlet esistono metodi che permettono all'Application Manager di poter inizializzare, eseguire, mettere in pausa o distruggere una Xlet. La differenza sostanziale con le Applet è che una Xlet può essere messa in pausa e poi essere riavviata successivamente, una applet no. Per ragioni legate alla capacità dell'hardware, nel caso in cui più Xlet debbano essere eseguite contemporaneamente, è possibile che solo una sia visibile, e quindi è necessario che le altre siano poste in stato di pausa in modo da non occupare risorse necessarie alla Xlet attualmente in esecuzione. Il ciclo di vita delle Xlet è riassunto di seguito.

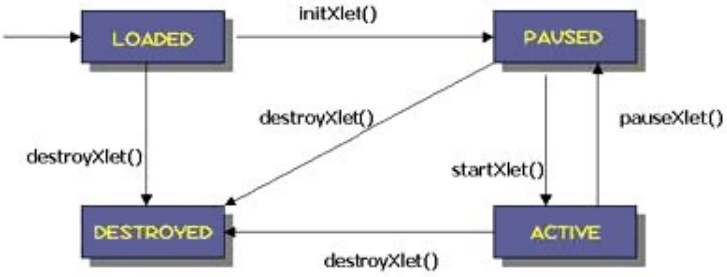

Figura 2.3: Il ciclo di vita delle Xlet

Una Xlet può trovarsi in cinque stati: *Not Loaded, Loaded, Paused, Destroyed, Active*. Un ulteriore stato è rappresentato da *Invalid*: quando non può essere eseguita e non è stata ancora cancellata dal garbage collector. Quando una Xlet si muove da uno stato all'altro, il middleware sottostante lancerà un evento `AppStateChangeEvent` a tutti i listener registrati a quell'evento, comunicandogli lo stato attuale in cui si trova l'applicazione e lo stato precedente. È importante ricordarsi che una Xlet non è una applicazione Java standard. Contemporaneamente potrebbero esserci più Xlet in esecuzione, ed è necessario che non eseguano azioni che potrebbero influenzare la Java Virtual Machine. È quindi necessario che le Xlet non richiamino mai il metodo `System.exit()` per evitare che quando una applicazione termina, debba terminare l'intera Java VM.

Esempio di ciclo di vita di una Xlet

1. Loading: quando il ricevitore MHP riceve le informazioni su una applicazione, che si trova nello stato *Not Loaded*. A questo punto, quando la Xlet viene caricata, l'application manager invoca il costruttore di default e crea quindi una istanza della Xlet che viene posta in stato di *Loaded*.

2. Initialization: quando l'utente sceglie di eseguire la xlet, l'application manager chiama il metodo `initXlet()` passando nel nuovo `XletContext` (discusso in seguito). La Xlet può usare il suo `XletContext` per inizializ-

zarsi ed effettuare il pre-caricamento di oggetti, come immagini. Quando l'inizializzazione è stata completata la Xlet si trova nello stato di *Paused* ed è pronta per il caricamento.

3. Execution: una volta che il metodo `initXlet()` è stato eseguito con successo, l'application manager può chiamare il metodo `startXlet()` che farà transitare la Xlet dallo stato di *Paused* a quello di *Loaded*, nel quale può avvenire l'interazione con l'utente.

4. Pausing and Resuming: durante l'esecuzione di una Xlet, l'application manager può chiamare il metodo `pauseXlet()` che causerà la transizione di stato, da *Started* a *Paused*. L'applicazione verrà posta successivamente nello stato di *Started* mediante l'esecuzione del metodo `startXlet()` .

5. Termination: quando un utente sceglie di terminare una Xlet, o quando viene comunicato al ricevitore MHP di terminare la Xlet, l'application manager invocherà il metodo `destroyXlet()` che farà transitare la Xlet nello stato di *Destroyed*. A questo punto tutte le risorse associate a tale Xlet verranno deallocate.

2.3.2 XletContext

Come le applet, anche le Xlet vengono eseguite in un contesto. Per le applet è rappresentato da `java.appplet.AppletContext`, invece per le Xlet da `javax.tv.xlet.XletContext`. In entrambi i casi il contesto serve ad isolare l'applicazione dal resto della VM quando interagisce con l'ambiente nel quale viene eseguita. Ad esempio, nel caso in cui più Xlet possano avere valori differenti per determinate proprietà, oppure nel caso in cui debbano comunicare all'application manager i cambiamenti di stato.
L'interfaccia della classe `javax.tv.xlet.XletContext` è la seguente:

```
public interface XletContext {
```

```
    public static final String ARGS = "javax.tv.xlet.args";

    public void notifyDestroyed();

    public void notifyPaused();

    public void resumeRequest();

    public Object getXletProperty(String key);

}
```

I metodi `notifyDestroyed()` and `notifyPaused()` permettono alla Xlet di comunicare al Set-Top-Box lo stato di *Destroyed* o *Paused*. In tal modo è possibile conoscere lo stato di ogni applicazione. Questi metodi devono essere chiamati immediatamente prima che la Xlet entri in stato di *Destroyed* o *Paused*, perché il ricevitore MHP potrebbe effettuare delle operazioni per le quali la Xlet potrebbe non essere pronta. Ad esempio, la Xlet potrebbe non aver rilasciato le risorse che sta utilizzando. Ciò potrebbe causare problemi al resto del middleware. Una applicazione può richiedere di essere posta dallo stato di *Paused* allo stato di *Started* usando il metodo `resumeRequest()` che permette ad una applicazione di porsi dallo stato di pausa a quello di esecuzione quando un certo evento è stato ricevuto. Il metodo `getXletProperty()` permette alla Xlet di accedere a informazioni riguardanti il suo ambiente. Queste informazioni sono definite dal broadcaster. L'unica proprietà attualmente definita è `XletContext.ARGS`.

2.3.3 Il Modello Grafico

Una delle maggiori differenze tra lo sviluppo di applicazioni per PC e lo sviluppo di applicazioni per la TV è il modo con cui la piattaforma gestisce la grafica.

Le Java Abstract Window Toolkit (AWT) sono state progettate specificatamente per i PC e non sono state concepite per supportare caratteristiche specifiche della TV digitale. Proprio per questo motivo sono nate le API HAVi (Home Audio/Video Interoperability) [24] per interfacce grafiche.
I principali problemi da affrontare quando si sviluppano interfacce grafiche si applicazioni per TV digitale sono:

- *Dimensione dei pixel*: Un motivo per cui TV e monitor hanno risoluzioni differenti è che le applicazioni per la DTV usano pixel non quadrati; invece le API per computer graphic assumono che i pixel siano quadrati.

- Cambiamenti dell'aspect ratio: La TV può utilizzare due formati di visualizzazione (Aaspect ratio) 4:3 (standard) o 16:9 (widescreen). Inoltre, ad esempio, il modo in cui viene visualizzata una immagine su TV è diverso a seconda del formato di visualizzazione che si utilizza.

- *Trasparenza*: come possiamo rendere gli oggetti grafici trasparenti in modo da poter visualizzare lo sfondo? Se utilizzassimo Java 2 Platform potremmo utilizzare il supporto alla trasparenza delle AWT. Purtroppo non c'è garanzia che sia usato proprio Java 2 Platform.

- *Color space*: come mappare il formato RGB usato da java, nel formato YUV usato dalla TV? Questo è più complesso di quello che sembra considerando che RGB e YUV non si sovrappongono completamente.

- *Cursore*: la mancanza di un cursore, come un mouse, fa sì che la navigazione della pagina deve essere affidata ad altri strumenti, come il telecomando.

Lo schermo

È possibile suddividere lo schermo di una TV digitale in 3 layer. Dal più profondo al meno profondo sono: *background layer*, *video layer* e *graphic layer*.

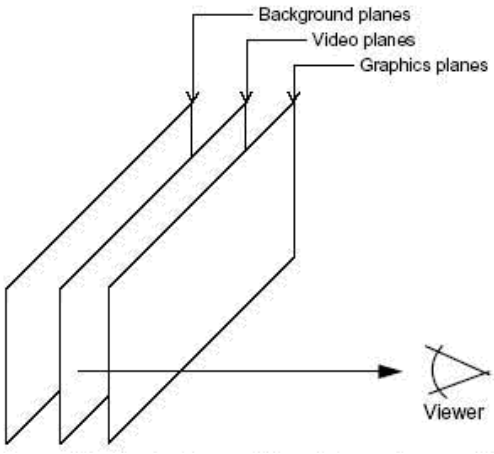

Figura 2.4: I layer grafici della piattaforma MHP

Il **background layer** può contenere un *solid color* oppure una immagine su tutte le aree (rappresentata da un frame MPEG-2), anche quelle non coperte dal video. Il **video layer** contiene il flusso Audio/Video del canale TV o di una qualunque fonte MPEG-2. Il livello più alto corrisponde al **graphic layer** che può contenere la grafica creata nella Xlet.
Questo modello grafico è supportato da vari package messi a disposizione tra i quali spicca `HAVi`. In tale package viene messa a disposizione la classe `org.havi.ui.HScreen` che rappresenta il dispositivo fisico. Il ricevitore MHP avrà al più un solo oggetto `HScreen` attivo. Ogni oggetto `HScreen` ha un numero di oggetti `HScreenDevice` associati ad esso che rappresentano i layer dello schermo.
In particolare:

- `HBackgroundDevice` rappresenta il background layer;
- `HVideoDevice` rappresenta il video layer;
- `HGraphicsDevice` rappresenta il graphic layer.

La risoluzione supportata è 720x576 pixel con un minimo di 256 colori. Questi sono solo valori minimi. Alcuni ricevitori MHP potrebbero avere risoluzioni più alte.

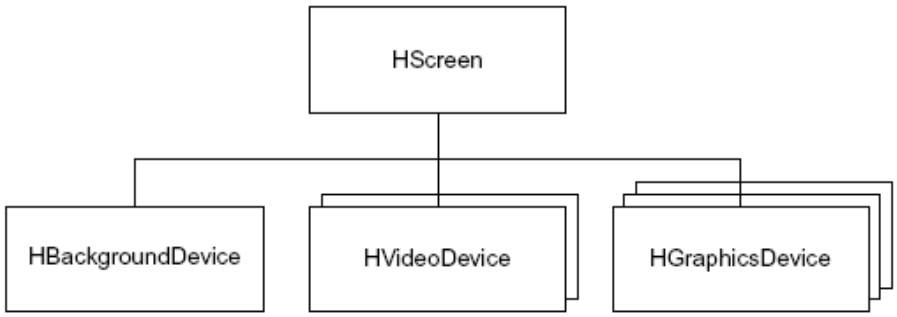

Figura 2.5: Il modello grafico di HAVi

In genere si tende a non utilizzare tutti i pixel a disposizione per vari motivi. Innanzitutto solo il 90% centrale dello schermo viene visualizzato e solamente l'80% centrale dello schermo non è affetto da distorsione. Se sommiamo questi numeri ai vari formati, 4:3 e 16:9, se ne deduce che ci sono varie parti di schermo che possono risultare distorte o essere nascoste. Per evitare ciò si consiglia che la grandezza della scena ricada all'interno della 'graphics-safe area', conosciuta anche come 'title safe area'.
Questa area è chiamata sicura perché corrisponde alla parte centrale dello schermo che è sicuramente visibile e che non contiene alcuna distorsione su qualunque TV. Le sue caratteristiche sono riportate nella figura sottostante.

Standard	*Safe Area Width*	*Safe Area Height*
PAL	562	460
NTSC	384	512

Figura 2.6: 'Title safe area' per schermi PAL e NTSC con formato 4:3

2.3.4 Immagini

Un ricevitore MHP può visualizzare immagini in vari formati come PNG, GIF o JPEG. Alternativamente possiamo utilizzare MPEG I-frames per visualizzare

immagini a schermo intero.

2.3.5 Visualizzazione del Testo

La visualizzazione del testo in uno schermo televisivo non è una cosa semplicissima. La dimensione infatti varia a seconda della risoluzione e dell'*aspect ratio* utilizzato.

Le AWT mettono a disposizione dei metodi per la visualizzazione del testo come `drawString()` o `drawChar()` di `java.awt.Graphics`. Questi metodi però necessitano che si conoscano a priori delle cose essenziali affinché il testo venga disegnato. Ad esempio, per quanto riguarda la risoluzione verticale non ci sono problemi: infatti un carattere di dimensione 12 pt, sarà alto 12 pixel. La cosa è differente per la risoluzione orizzontale. Infatti, essendo i pixel non quadrati, a seconda dell'*aspect ratio* e della risoluzione la dimensione del carattere varia. Ad esempio, con risoluzione 720x576 e formato 4:3 l'altezza di un carattere corrisponde a 45/48 dell'altezza.
È quindi necessario calcolare la dimensione del testo prima di poterlo visualizzare, proprio perché per costruire l'oggetto rappresentante il testo (ad esempio usando la classe `HText`) è necessario conoscerne altezza e larghezza.

2.3.6 Font da utilizzare

MHP include solamente `Tiresias` come font, ma altre potrebbero essere scaricate e utilizzate dall'applicazione. `Tiresias` è stata realizzata proprio per gli schermi televisivi, garantisce una migliore leggibilità e utilizza un set di caratteri Western European (ISO-8859-1).
Tuttavia, se volessimo utilizzare altre font, le potremmo scaricare ed utilizzare. Visto che il middleware non sa come scaricare le font, non è possibile utilizzare il normale meccanismo per istanziarle. Invece dobbiamo utilizzare la classe `org.dvb.ui.FontFactory` avente la seguente firma:

```
public class FontFactory {
```

```
    public FontFactory()
       throws FontFormatException, IOException;

    public FontFactory(java.net.URL u)
       throws FontFormatException, IOException;

    public java.awt.Font createFont(String name, int style, int size)
       throws FontNotAvailableException,
              FontFormatException,
              IOException;

}
```

Ogni font è associata ad una sorgente, che può essere quella di default o quella specificata nell'URL del costruttore. I costruttori sono sincroni, per cui si bloccheranno fin quando la font non è stata scaricata. Una volta creata una istanza di `FontFactory` è possibile utilizzare il metodo `createFont()` per creare la nuova font con dimensione e stile che vogliamo.

2.3.7 Input, Eventi e Focus

La TV digitale tipicamente non usa tastiera o mouse per la gestione dell'input. L'unico input possibile è rappresentato dai tasti del telecomando.
Gestendo opportunamente gli eventi associati al telecomando si può emulare una tastiera, un po' come accade per i telefonini. Inoltre bisogna dare all'utente la possibilità di navigare nell'applicazione creata e quindi muoversi tra gli oggetti.
Fortunatamente per noi, HAVi definisce una serie di classi che possono aiutarci a gestire l'input. Il package `org.havi.ui.event` estende gli eventi delle AWT aggiungendo delle caratteristiche proprie dei ricevitori per DTV. Inoltre

definisce una serie di costanti che rappresentano i tasti del telecomando, come mostrato nella seguente figura.

Key	*Constant Name*
Up arrow	VK_UP
Down arrow	VK_DOWN
Left arrow	VK_LEFT
Right arrow	VK_RIGHT
Enter (also known as Select or OK)	VK_ENTER
Number keys	VK_0 to VK_9
Teletext key	VK_TELETEXT
First colored key	VK_COLORED_KEY_0
Second colored key	VK_COLORED_KEY_1
Third colored key	VK_COLORED_KEY_2
Fourth colored key	VK_COLORED_KEY_3

Figura 2.7: Tasti standard sui decoder MHP

La semantica associata ai tasti, ovvero che tasti utilizzare per avviare l'applicazione oppure per muoversi fra gli oggetti, verrà trattata nel capitolo successivo.

Il modello descritto permette di poter catturare gli eventi del telecomando solamente quando l'applicazione è messa a fuoco. In un ambiente come MHP questa restrizione può causare alcuni problemi. Immaginiamo ad esempio di voler avviare un pop-up dopo che un particolare tasto è stato premuto. Non essendo questo pop-up messo a fuoco, non sarebbe in grado di poter ricevere l'evento che lo faccia visualizzare.
Fortunatamente, il package `org.dvb.event` risolve il problema sopra citato. In particolare la classe `UserEventListener` può ricevere un input anche se l'applicazione non è messa a fuoco. Gli eventi generati da queste API non sono i classici eventi AWT, ma istanze della classe `UserEvent`.
Prima che l'applicazione possa ricevere degli eventi, è necessario definire un `UserEventRepository` contenente i tipi di eventi che l'applicazione vuole catturare. Essa fornisce metodi per permettere alle applicazioni di aggiungere e rimuovere varie combinazioni di tasti.

```
public class UserEventRepository {

   public UserEventRepository (String name);

   public void addUserEvent(UserEvent event);
   public UserEvent[] getUserEvent();
   public void removeUserEvent(UserEvent event);

   public void addKey(int keycode);
   public void removeKey(int keycode);

   public void addAllNumericKeys();
   public void addAllColourKeys();
   public void addAllArrowKeys();

   public void removeAllNumericKeys();
   public void removeAllColourKeys();
   public void removeAllArrowKeys();

}
```

2.3.8 Java su Piattaforma MHP

Come detto in precedenza, il core di MHP è sviluppato in Java. La JVM presente sui decoder MHP non è la standard edition (J2SE), ma la J2ME Personal Basis Profile (PBP) [26]. Inoltre sono state incluse una serie di API tra cui spiccano JavaTV e HAVi.

2.3.9 Esempio di Xlet

Vediamo il più classico degli esempi: 'Hello World'.

```
import javax.tv.xlet.*;
import org.havi.ui.*;
import java.awt.*;

public class HelloWorld implements Xlet{

   private XletContext context;
   private HScene scene;

   public Browser() {
      super();
      HSceneFactory factory =HSceneFactory.getInstance();
      scene = factory.getFullScreenScene(
      HScreen.getDefaultHScreen().getDefaultHGraphicsDevice());
      scene.setSize(720, 576);
   }

   public void initXlet(XletContext ctx)
      throws XletStateChangeException {
         this.context = ctx;
   }

   public void pauseXlet() {
      scene.setVisible(false);
   }

   public void destroyXlet(boolean b) {
      if (scene != null) {
         scene.setVisible(false);
         HSceneFactory.getInstance().dispose(scene);
```

```
            scene = null;
        }
        context.notifyDestroyed();
    }

    public void startXlet() throws XletStateChangeException {
        HText htext = new Htext(
            "Hello World",
            new Font("Tiresias", Font.PLAIN, 24),
                     Color.BLACK, Color.WHITE,
                     new HDefaultTextLayoutManager());

        htext.setSize(200, 50);

        scene.add(htext);

        scene.repaint();
        scene.setVisible(true);

    }

}
```

2.3.10 Emulatori per TV digitale

Esistono due tools open source che permettono di emulare la visualizzazione delle Xlet su PC:

- XletView [28]
- OpenMHP [29]

Questi due software non sono sufficienti per emulare realmente una applicazione per la TV digitale, ma supportano solamente alcune delle funzionalità dei decoder MHP. Inoltre lo stack MHP non è certificato. I due tools sono stati messi a confronto ed è risultato che XletView è il più completo, ed è per questo motivo che è stato scelto per testare l'applicazione realizzata.

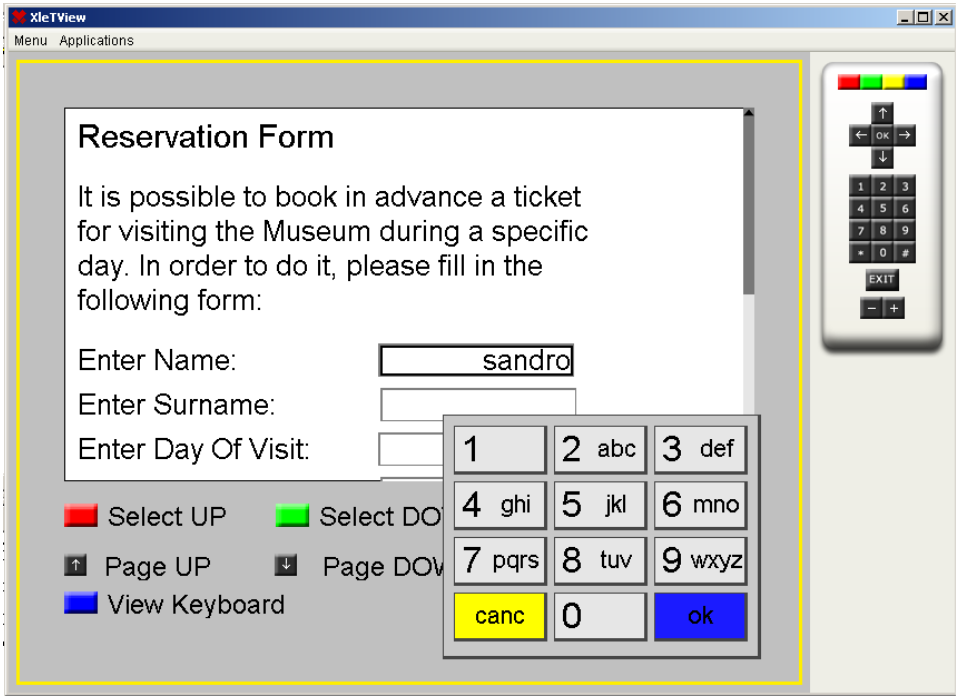

Figura 2.8: Il tool XletView durante l'esecuzione di una applicazione

Capitolo 3

Linee Guida per lo sviluppo di interfacce utenti per DTV

3.1 Introduzione

Il primo passo nello sviluppo di applicazioni per TV Digitale Interattiva (iTV) è quello di costruire una interfaccia utente usabile, ovvero che risponda a criteri di efficacia, efficienza e soddisfazione nella navigazione del servizio (norma ISO 9241).

Essendo il campo ancora tutto da scoprire non esistono dei veri e propri standard a riguardo ma solo dei consigli promossi da esperti del settore basati sull'esperienza. In particolare ci baseremo su degli studi effettuati dalla Fondazione Ugo Bordoni [14].

3.2 Usabilità della TV digitale

Il successo di una interfaccia utente, sia essa sviluppata in ambito web o per piattaforma televisiva, è dato dal grado di usabilità della stessa.

Compito degli studi di usabilità è fare in modo che il modello mentale di chi ha progettato il software (design model), da cui deriva il suo reale funzionamento,

corrisponda il più possibile al modello mentale del funzionamento del software così come se lo costruisce l'utente finale (user model).

L'usabilità nasce dunque soprattutto come ausilio alla progettazione, e si applica in particolare alle interfacce. È con l'interfaccia di un software, infatti, che l'utente si relaziona. Ad ogni sua azione l'interfaccia proporrà un risultato, un cambiamento di stato. Poco importa, ai fini dell'usabilità, come l'interfaccia sia giunta a quello stato, attraverso cioè quali meccanismi di programmazione, che rimangono racchiusi in una vera e propria scatola nera impermeabile all'utente. Se questa considerazione è valida per le applicazioni informatiche in generale, a maggior ragione è valida nell'ambito di servizi applicati a nuove tecnologie come la TV digitale dove la popolazione che ha accesso ai servizi corrisponde a tutta la popolazione, e quindi non solo al sottoinsieme di persone aventi competenze informatiche.
La televisione si pone, quindi, come mezzo che più facilmente riesce a raggiungere un vasto pubblico e adatta a non escludere nessun possibile utente: un mezzo che potenzialmente elimina il divario che esiste fra persone informatizzate e persone che non hanno accesso a sistemi informatici.

3.3 Caratteristiche della TV digitale

Analizziamo brevemente le principali caratteristiche della TV digitale che saranno argomento di successive discussioni.
Le principali problematiche che verranno analizzate sono:

- Contesto e accesso all'applicazione: uno scenario tipico è rappresentato dall'utente che assiste passivamente ad un programma televisivo. La presenza di una applicazione deve essere debitamente segnalata, e l'accesso ad essa deve essere semplice, veloce e chiaro.

- Struttura dell'applicazione: una importante caratteristica che l'interfaccia utente deve possedere è la semplicità d'uso. Questo è motivato dal fatto che la iTV eredita tutti gli utenti della TV analogica, quindi anche

utenti che non hanno alcuna competenza informatica. Per questo motivo la struttura deve essere chiara ed intuitiva. E' consigliato separare i contenuti dai comandi ed è fondamentale prevedere aiuti per gli utenti.

- Navigazione: bisogna tenere conto che a differenza di un pc non è possibile utilizzare un mouse per navigare all'interno dell'applicazione. Bisogna quindi definire un nuovo meccanismo di navigazione tenendo sempre conto dell'aspetto della semplicità e che l'unico mezzo di interazione è il telecomando.

- Orientamento: a differenza di una comune applicazione informatica dove l'utente non è in grado di vedere immediatamente l'effetto delle sue azioni, in una applicazione televisiva è importante, durante l'interazione, che l'utente riceva dei feedback prima che il processo cominciato con la sua applicazione si concluda.

- Look & feel: non essendoci un modello da seguire dobbiamo inventare noi un nuovo modello sapendo che ci sono dei forti vincoli sulla piattaforma televisiva. Un esempio è la scelta della dimensione del carattere: si potrebbe variare da 24, per delle note, a 36 per i titoli.

- Interazione: alcune applicazioni potrebbero richiedere immissione di input. La mancanza di una tastiera delega all'applicazione il supporto all'immissione di dati. Bisognerà, ad esempio, realizzare una tastiera virtuale per ovviare a tale mancanza.

- Scrivere per la TV: la natura del mezzo televisivo è diversa da quella di altri mezzi di comunicazione che erogano informazioni di tipo testuale. E' quindi necessario porre particolare attenzione ad una serie di accorgimenti volti a rendere la lettura la meno faticosa possibile.

3.3.1 Contesto e accesso all'applicazione

Come anticipato in precedenza, la presenza di una applicazione interattiva deve essere segnalata chiaramente, e l'accesso ad essa deve essere semplice

ed intuitivo. Di seguito verranno riportate una serie di raccomandazioni a riguardo.

Segnalazione della presenza on air dell'applicazione

Si raccomanda di segnalare sempre con un reminder la presenza dell' applicazione interattiva. Il reminder spesso indica graficamente l'azione da compiere per eseguire l'applicazione. Ad esempio, se per eseguire un'applicazione che visualizza eventi sportivi si deve premere il tasto rosso, il reminder potrebbe essere: 'Premi per visualizzare gli eventi sportivi".

Tasto per accedere all'applicazione interattiva

Come già detto non esiste uno standard vero e proprio, però è di uso comune, nei broadcaster nazionali e internazionali, l'utilizzo del tasto rosso per accedere alle applicazioni.

Tasto per uscire

È obbligatorio che nell'applicazione ci sia sempre un tasto per uscire e tornare al canale TV. Tale tasto sarà il tasto EXIT, a causa dell'interpretazione semantica associata.

Tasto per lo zoom del flusso audio/video

Si consiglia di utilizzare un tasto che riduca ad icona l'applicazione e permetta di vedere il flusso audio/video a tutto schermo senza che si debba uscire dall'applicazione. Si utilizza lo stesso tasto per tornare allo stato precedente, con il flusso audio video in resize.

Conferma dell'uscita dall'applicazione

È buona norma che l'applicazione preveda messaggi di conferma per l'uscita dall'applicazione, per evitare uscite accidentali dal servizio.

Messaggio di caricamento

Si raccomanda di segnalare sempre lo stato di caricamento delle applicazioni con effetti visivi dinamici, per evitare che all'utente possa venire il dubbio che l'applicazione possa essere in stallo.

3.3.2 Struttura dell'applicazione

La struttura dell'applicazione deve essere chiara ed intuitiva. Come detto in precedenza è buona norma separare i contenuti dai comandi e fornire l'applicazione di aiuti per gli utenti.

Contenuti e comandi

Si raccomanda di separare visivamente la sezione dei comandi da quella dei contenuti. Inoltre ogni elemento, all'interno della stessa applicazione, deve comparire sempre nella stessa posizione.

Posizione area brand, logotipi e titoli

Poiché lo sguardo corre da sinistra verso destra, è consigliato posizionare gli elementi di maggior rilievo per l'identificazione del servizio in alto a sinistra.

Posizione area flussi audio/video della trasmissione televisiva

Durante l'esecuzione dell'applicazione è utile posizionare in alto a destra, in modalità *resized*, il flusso televisivo audio/video, con dimensioni ridotte ad $\frac{1}{4}$ di schermo. Tale area corrisponde a 360x288 pixel.

Area comandi

Si raccomanda di dedicare una sezione dello schermo ai comandi, rendendo visibili solamente quelli attivi. Si suggerisce di posizionare quest'area in basso. Come detto in precedenza, i comandi attivabili dovrebbero trovarsi sempre nella stessa posizione.

3.3.3 Navigazione

Per la navigazione bisogna utilizzare un metodo di selezione degli oggetti semplice ed intuitivo.

Gerarchia dell'albero dei contenuti

Si raccomanda di strutturare i contenuti su livelli gerarchici, senza superare tre livelli di gerarchia al fine di evitare il disorientamento dell'utente. Bisogna prestare molta attenzione alla modalità con cui vengono raggruppate le opzioni nei menu, in modo che le unità logiche create siano intuitive e quindi facilmente memorizzabili.

Coerenza funzionale

Si raccomanda di assegnare ai tasti una funzione coerente e univoca all'interno della stessa applicazione. È inoltre importante che le label associate a tali comandi siano anch'esse coerenti. La coerenza (anche relativa al posizionamento) facilita l'apprendimento da parte dell'utente, e minimizza la ricerca del comando all'interno della pagina. Come già detto in precedenza si consiglia di usare la parte inferiore per la visualizzazione dei comandi.

Coerenza tra schermo e telecomando

Si suggerisce di garantire coerenza tra l'ordine dei tasti presenti sul telecomando e la loro rappresentazione sullo schermo. La coerenza layout/tasti automatizza la pressione dei tasti del telecomando sulla base della visualizzazione degli stessi sullo schermo. Quindi, ad esempio, l'ordine dei tasti colorati deve essere rosso,verde,giallo,blu. È buona norma non solo rispettare l'ordine, ma che anche le label dei tasti fisici del telecomando, associati ad esempio ai tasti OK e EXIT siano conformi con quelli visualizzati sullo schermo. Questa regola è di più difficile attuazione in quanto lo stile delle label può differire da un telecomando ad un altro. In questo caso si preferisce associare alla funzione un comando a video attivabile con le frecce di spostamento e successiva pressione del tasto OK.

Navigazione dei menu

Si raccomanda che la navigazione dei menu sia effettuata mediante una delle seguenti modalità:

1. Frecce direzionali $\leftarrow$, $\uparrow$, $\rightarrow$, $\downarrow$ per il posizionamento sull'oggetto con successiva pressione del tasto OK / ENTER / SELECT.

2. Tasti numerici associati alle opzioni disponibili. Questo caso necessita che il numero di opzioni sia ovviamente inferiore a 10.

3. Associare i tasti colorati solamente alle principali funzionalità della pagina.

Navigazione di menu di livello successivo al primo

Si suggerisce, in presenza di livelli di menu successivi al primo, di rendere visibile solo il menu in uso nascondendo gli altri in modo da non affollare eccessivamente la pagina di opzioni non selezionabili. Può però essere utile visualizzare in un'area dedicata della schermata la localizzazione della navigazione in modo da facilitarne l'orientamento.

Navigazione di servizio

Si raccomanda l'utilizzo di un sistema di navigazione di servizio. Tale navigazione contempla tutte le modalità di navigazione eccetto quelle della navigazione dei contenuti. Si consiglia di utilizzare sia i tasti colorati che i tasti fisici del telecomando ed eventualmente bottoni a video.

1. Tasti colorati: si consiglia di accorpare i 4 tasti colorati in una parte bassa della schermata, mantenendo la stessa coerenza del layout tra la rappresentazione a video e quella del telecomando. Si raccomanda quindi la posizione orizzontale, oltre che il mantenimento della stessa sequenza di colori.

2. Tasti fisici del telecomando: si consiglia di mantenere la semantica dei tasti fisici (OK, EXIT, ...).

3. Bottoni a video: nel caso in cui non siano disponibili tasti fisici e tasti colorati è possibile visualizzare a video dei bottoni selezionabili con le frecce direzionali e attivabili mediante la pressione del tasto OK/ENTER. Nel caso in cui si utilizzassero più bottoni a video è possibile raggrupparli in sequenza nella parte bassa della schermata raggiungibili mediante le frecce.

3.3.4 Orientamento

L'interfaccia di una applicazione informatica è una interfaccia 'nascosta' nel senso che l'utente non è in grado di vedere immediatamente l'effetto delle sue azioni. Per vedere se l'azione è andata a buon fine l'utente ha bisogno di *feedback* per conoscere lo stato di avanzamento del servizio richiesto onde evitare, nel caso in cui i tempi di attesa debbano essere lunghi, che l'utente possa pensare che l'applicazione sia in stato di stallo.

Messaggi

Bisogna fare molta attenzione all'aspetto della messaggistica riguardante l'interazione con l'utente in seguito ad una azione intrapresa dallo stesso. In particolare si raccomandano:

- messaggi che richiedano conferma dall'uscita dell'applicazione;
- messaggi che segnalino il caricamento delle applicazioni per la gestione dell'attesa dell'utente;
- messaggi che segnalino, nel caso in cui l'applicazione ne faccia uso, il collegamento del STB alla presa telefonica, il costo e la durata della chiamata;
- messaggi che indicano lo stato della connessione;

- messaggi di aiuto nel caso in cui si fosse verificato un errore.

Oltre ai messaggi, dare altre indicazioni all'utente, come:

- in caso di liste, indicare se esse sono circolari e qual è il tasto da utilizzare per scorrere gli elementi di tale lista;
- nel caso di più pagine che visualizzino lunghi elenchi, riportare il numero della pagina corrente e il numero di pagine totali.

Funzioni di aiuto

Per favorire l'orientamento da parte dell'utente si consiglia l'utilizzo di aiuti testuali o grafici. Inoltre è buona norma che l'applicazione sia dotata di una pagina di aiuto contenente tutte le informazioni dettagliate che riguardano la navigazione, raggiungibile da qualunque pagina dell'applicazione.

3.3.5 Look & Feel

Nella progettazione di un'interfaccia grafica per la TV digitale bisogna avere ben chiare quali sono le peculiarità di questo mezzo e che, a differenza di un PC, ci sono differenze di resa di immagini o di testo sullo schermo.

Tipologia di Font

Per garantire una leggibilità compatibile con la 'distanza televisiva', si utilizza una dimensione minima di 20 punti (per le note) fino ad un massimo di 36 punti (per i titoli). È comunque consigliato, onde evitare che l'utente si possa stancare durante la lettura, l'utilizzo di una font media di 24/26 punti. Si consiglia di utilizzare il font Tiresias.

Effetti di sfarfallio (flickering)

Lo sfarfallio si verifica quando i pixel dell'interfaccia scompaiono e compaiono a causa dell'interlacciamento dello schermo televisivo. Ciò si nota vistosamente nel caso di linee della dimensione di 1 pixel, che subiscono il cosiddetto flickering.

È quindi importante che si utilizzi, ad esempio per i bordi, una dimensione minima di 3 pixel al fine di evitare questo sgradevole effetto. Inoltre a differenza di uno schermo per PC, i pixel non sono quadrati ma hanno una forma rettangolare, per cui le immagini assumono una leggera distorsione orizzontale.

Contrasti di colore

Bisogna prestare particolare attenzione alle aree di colore contigue fortemente contrastate in quanto possono provocare l'effetto flickering.

È importante scegliere il giusto contrasto tra primo e secondo piano per rendere la lettura gradevole e meno stancante. Inoltre si consiglia di testare la grafica in modalità 'scala di grigio' per verificare la resa del contrasto e quindi garantire alle persone affette da daltonismo una buona leggibilità.

Utilizzo dei colori

La scelta dei colori deve essere fatta con cura in quanto il monitor televisivo non utilizza la stessa gamma di colori di un monitor per PC. Lo standard DVB-MPH supporta un numero limitato di colori (palette a 188 colori) e se ne consiglia la scelta in conformità con gli standard attuali.

Si suggerisce l'utilizzo di immagini con colori desaturati e definiti. Bianco, rosso e colori accesi sono da evitare a causa dell'effetto flickering. È necessario testare su schermi televisivi la resa dei colori per migliorarne la visualizzazione.

L'uso del colore è importante nel caso del focus degli oggetti. Infatti bisogna scegliere i giusti colori per poter identificare chiaramente che l'oggetto è in stato di focus.

3.3.6 Interazione

In alcune applicazioni interattive potrebbe essere richiesta l'immissione di dati da parte dell'utente. Nel caso di immissione di stringhe di testo bisogna porre particolare attenzione al fatto che sulla TV digitale non esiste alcuna tastiera fisica ed è quindi importante trovare un metodo per l'immissione di dati facilitata mediante la creazione di una tastiera virtuale. È inoltre utile realizzare un metodo di scrittura facilitata di cui sono dotati i cellulari.

Numero di campi da compilare

Si consiglia di limitare il numero di campi della form in quanto l'immissione di dati è una operazione lunga. E' quindi utile differenziare i campi da compilare obbligatoriamente da quelli facoltativi utilizzando dei simboli (come gli asterischi) e delle note che ne spieghino il significato.

Differenti modalità di inserimento dati

La Fondazione Ugo Bordoni ha realizzato dei test riguardo la scelta della modalità di inserimento dei dati: dove un giovane preferisce l'utilizzo di una tastiera in 'stile cellulare', una persona meno giovane con poca esperienza a riguardo potrebbe preferire l'utilizzo di una tastiera virtuale sulla quale muoversi mediante l'utilizzo delle frecce. Tale metodo è sicuramente più lento ma più efficace, riducendo la possibilità di errori.

Si raccomanda quindi di contemplare diverse modalità di inserimento dati a seconda dell'utente che utilizza il servizio.

Tastiera del telecomando

Si consiglia di utilizzare la tastiera del telecomando per l'immissione di numeri. Nel caso di stringhe alfanumeriche, bisogna tenere presente che diversi utenti (ad eccezione di quelli che scrivono SMS) hanno difficoltà nel trovare le lettere sui tasti numerici del telecomando. Si consiglia inoltre che la cancellazione dei caratteri avvenga con il tasto BACK (←). Per alcuni utenti ciò potrebbe non

essere intuitivo. Si potrebbe prevedere anche l'utilizzo di scrittura facilitata T9.

Tastiera virtuale

La tastiera virtuale prevede la rappresentazione a video di tutti i caratteri digitabili. L'utente seleziona, mediante l'uso delle frecce, il carattere prescelto e preme OK per confermarlo. Tale tastiera è utile per gli utenti che non hanno dimestichezza con le tastiere dei cellulari.

Bisogna anche gestire il focus dei tasti della tastiera virtuale in maniera appropriata, visualizzando chiaramente quale tasto è selezionato. È consigliato affiancare la tastiera ad un help in linea per aiutare gli utenti meno esperti a conoscere questo nuovo metodo di immissione di dati.

La grafica della tastiera deve essere semplice, le lettere disposte in ordine alfabetico e i numeri raggruppati nella parte superiore o laterale. Come detto in precedenza si devono utilizzare font leggibili e colori adeguati.

Slot machine

Un'altra modalità di inserimento di dati è la cosiddetta *slot machine*. Tale metodo alternativo consiste nello scorrere le lettere, mediante uso di frecce, fino al raggiungimento della lettera prescelta. Si usano quindi le frecce $\uparrow$ e $\downarrow$ per scorrere le lettere. Ad esempio se siamo posizionati sulla lettera 'a', per raggiungere la lettera 'c' basterà premere due volte il tasto $\uparrow$. Inoltre si utilizza la freccia $\leftarrow$ per cancellare un carattere e la freccia $\rightarrow$ per inserire il successivo.

Come nel caso della tastiera virtuale, tale metodo è nuovo e bisogna quindi documentarlo opportunamente.

Informativa sul canale di ritorno

È importante presentare sempre all'utente quando il STB deve utilizzare la linea telefonica, visualizzando costo e durata della connessione. È importante che l'utente capisca chiaramente quando sta utilizzando il telefono e quando invece non lo sta utilizzando. Ciò è fattibile mediante visualizzazione grafica

e l'uso di messaggi che ricordano all'utente che sta utilizzando la connessione.

Feedback della transazione

Si raccomanda l'utilizzo di messaggi semplici ed intuitivi che segnalino l'esito della transazione ed eventualmente il tipo di errore. Si consiglia di prevedere diversi metodi per l'invio della ricevuta come SMS, e-mail o posta ordinaria che confermino l'esito della transazione. L'attuale versione di MHP (MHP 1.0.2) non supporta il riconoscimento di eventi legati alla connessione. Ad esempio, con il termine *connessione fallita* si indicano una serie di eventi diversi fra di loro che vanno dal cavo scollegato alla linea occupata.
Infine, al termine della chiamata, è utile informare l'utente della durata della chiamata e del costo della connessione.

3.3.7 Scrivere per la TV

Per quanto riguarda la visualizzazione del testo bisogna utilizzare una serie di accorgimenti per rendere la lettura il meno faticosa possibile.
Chi sta davanti alla TV tende a finalizzare l'interazione verso la fruizione di contenuti televisivi, il che limita notevolmente l'impegno e la concentrazione che un utente dedicherà al testo sulla TV.

Presentazione del testo

È consigliabile porre le informazioni importanti all'inizio del testo. In tal modo se l'utente non dovesse arrivare in fondo, avrà comunque letto le cose più importanti. È inoltre consigliato non scrivere periodi troppo lunghi e non superare le 5 righe di testo.
Ulteriormente sullo schermo è più facile leggere informazioni schematizzate piuttosto che un flusso di testo continuo. A tal fine si possono quindi utilizzare elenchi puntati. Si consiglia di andare spesso 'a capo' per separare i periodi e facilitare la lettura.
Si raccomanda inoltre di lasciare diversi spazi bianchi per rallentare la lettura e affaticare meno il lettore.

Stile del testo

Si preferisce l'utilizzo di forme verbali semplici, come il presente indicativo, piuttosto che un gerundio, al fine di evitare di rendere ulteriormente faticosa la lettura.

Si consiglia inoltre di non usare abbreviazioni non chiare senza opportuna documentazione.

Titoli, sottotitoli, abstract

L'utilizzo della dimensione dei caratteri maggiore o minore e quindi la rappresentazione di titoli e sottotitoli facilita il lettore nella ricerca di contenuti. È bene che ci si renda conto dell'argomento riguardante il testo a colpo d'occhio senza che ci sia la necessità di dover leggere tutto il testo. L'utilizzo di riassunti, ad esempio sotto forma di slogan, facilita l'utente nel capire immediatamente l'argomento del testo.

Impaginazione

Nel caso in cui il testo non entri interamente nella schermata è utile inserire una barra di *scrolling* per lo scorrimento del testo mediante l'utilizzo delle frecce verticali $\uparrow$ e $\downarrow$ del telecomando.

Quando invece è necessario gestire il tutto nella stessa schermata è possibile utilizzare la modalità del 'volta pagina' per scorrere le pagine mediante l'utilizzo delle frecce orizzontali $\leftarrow$ e $\rightarrow$ del telecomando.

Utilizzo dei colori a fini semantici

E' bene tenere presente che i colori in genere richiamano alla mente sensazioni. Ad esempio il rosso esprime pericolo, il giallo attenzione e bianco o grigio invece neutralità. Bisogna anche ricordare che queste codificazioni variano di paese in paese.

Bisogna ricordarsi particolarmente di utilizzare la semantica dei colori dei tasti del telecomando e utilizzare stessi colori per informazioni dello stesso tipo.

Linguaggio

È importante che il linguaggio utilizzi espressioni di uso comune, senza termini ricercati ma con espressioni semplici. Ciò faciliterà la lettura. Inoltre, essendo la TV un mezzo confidenziale, si preferisce l'uso della forma confidenziale del 'tu'.

Per rendere il testo più sintetico è necessario evitare di scendere nei particolari. Si consiglia l'utilizzo di parole chiave e l'eliminazione di eventuali ridondanze.

Capitolo 4

Progettazione di Interfacce Utenti per TV digitale con TERESA

4.1 Introduzione

TERESA è un tool per la progettazione di interfacce utenti multi piattaforma. Parte integrante di questo tool è lo User Interface Editor (UI Editor) che permette di editare interfacce utenti per dispositivi quali, ad esempio, PC o palmari. Tale editor è stato esteso in modo che si potessero progettare e generare interfacce utenti per TV digitale.
TERESA UI Editor considera due livelli di astrazione [4]:

- AUI (Abstract User Interface)
- CUI (Concrete User Interface)

L'AUI definisce un'interfaccia astratta, indipendente dal dispositivo, dalla quale è possibile generare una CUI per un particolare dispositivo come la TV digitale. La AUI faceva già parte del sistema. Invece è stato opportuno definire una CUI per TV digitale come raffinamento dell'interfaccia astratta esistente.

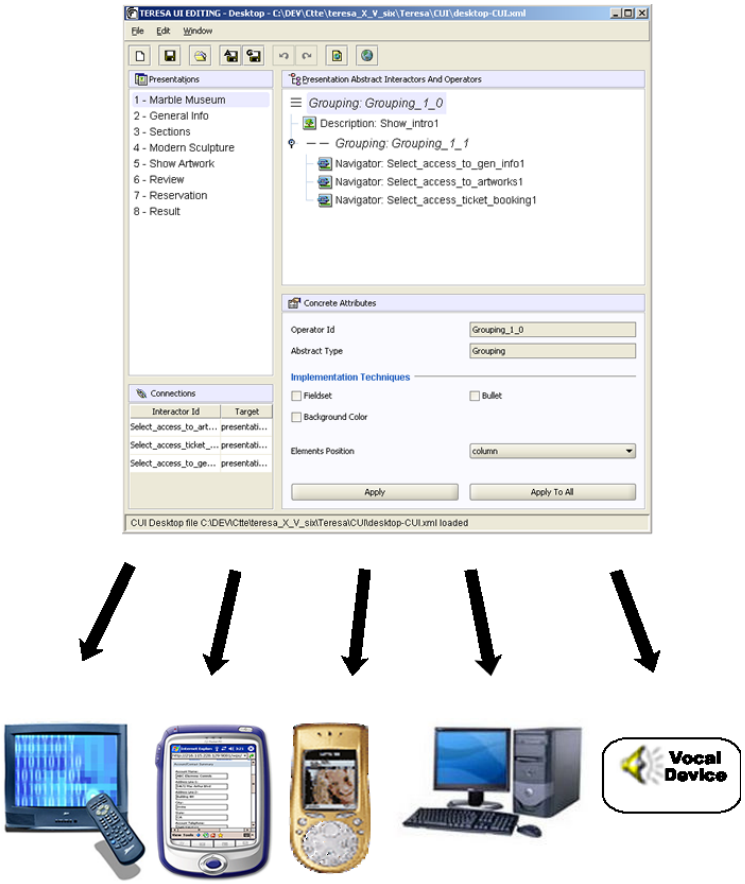

Figura 4.1: TERESA UI Editor

Definire un'interfaccia astratta indipendente dal dispositivo ci permetterà di poter progettare interfacce per più dispositivi e successivamente, mediante trasformazioni, generare l'interfaccia per un particolare dispositivo.

Figura 4.2: I livelli di astrazione di TERESA UI Editor

Di seguito verranno analizzati in modo più dettagliato sia l'interfaccia astratta che quella concreta per TV digitale. Sarà a partire da quest'ultima che verrà generato il codice Java che rappresenta l'interfaccia utente finale.

4.2 Abstract User Interface

Un'interfaccia astratta è composta da un numero di presentazioni astratte e connessioni che legano le varie presentazioni. Le connessioni definiscono la caratteristica dinamica dell'interfaccia utente. Più precisamente indicano quali interazioni causano un cambiamento della presentazione e quale sarà la prossima presentazione. Ogni presentazione astratta è composta da un numero di interattori (*interactors*), che rappresentano la descrizione astratta di oggetti raggruppati in base alla loro semantica.

4.2.1 Interactor Composition

La struttura della presentazione è definita in termini di interattori e di operatori di composizione.

Interactors
Viene fatta distinzione tra gli interattori che supportano l'interazione con l'utente (*interaction elements*) e quelli che presentano risultati derivanti dall'elaborazione di applicazioni (*only output elements*).

Gli *interaction elements* implicano un'interazione tra utente e applicazione. Ne esistono diversi tipi a seconda dell'operazione:

- selection;
- edit;
- control.

Gli elementi di selezione (*selection elements*) sono utilizzati per selezionare elementi da un insieme, quelli di edit per editare un oggetto e quelli di controllo per sollevare un evento utile ad attivare una funzionalità o transire in una nuova presentazione.

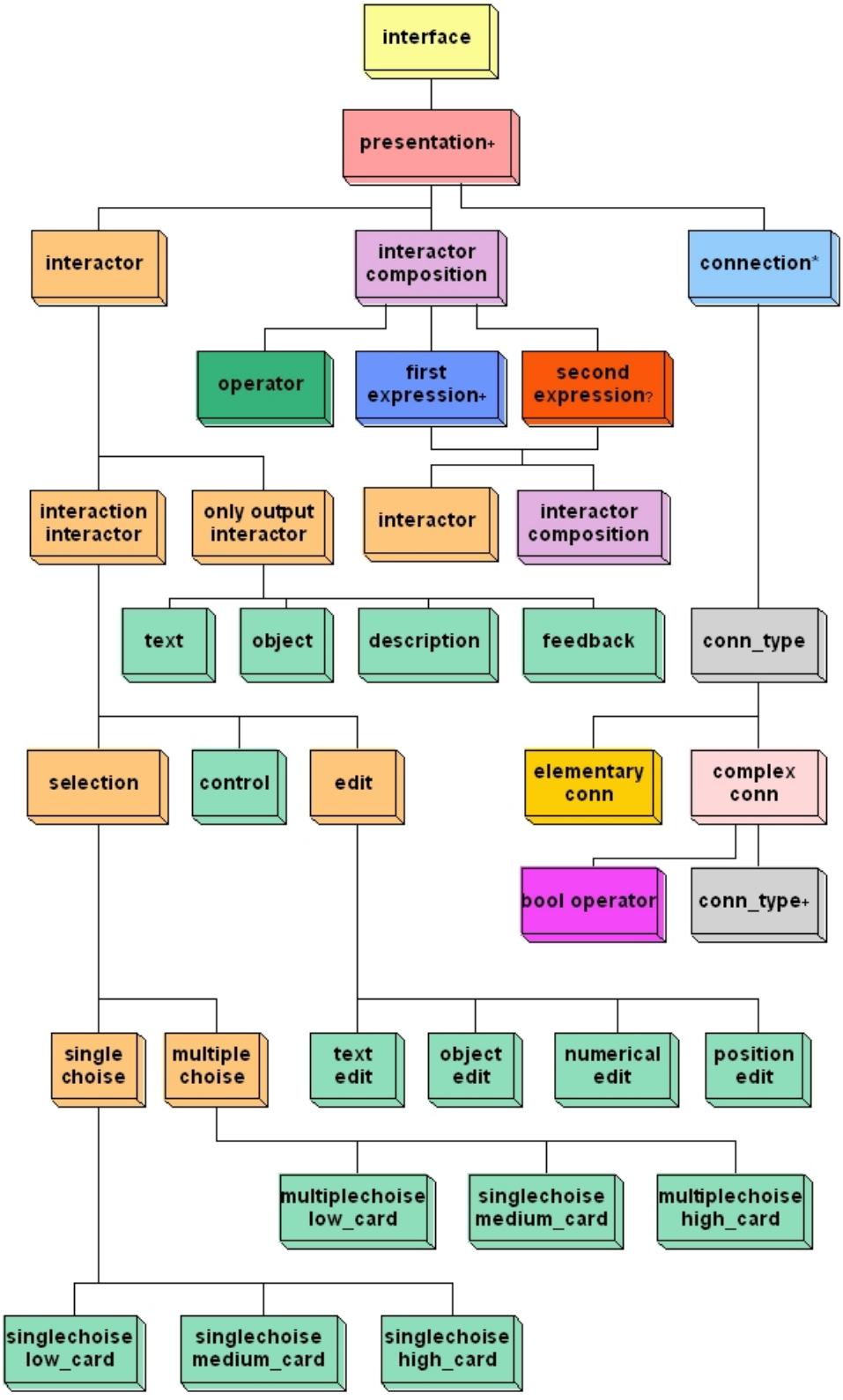

Figura 4.3: Struttura dell'abstract user interface in TERESA

Differentemente, un *only output element* si riferisce ad azioni provenienti solamente dalla stessa applicazione. Esistono diversi di *only output elements*:

- text;
- object;
- description;
- feedback.

Ognuno di questi elementi rappresenta un tipo output diverso, a seconda del tipo del tipo di output che l'applicazione fornisce all'utente (testuale, oggetto, descrizione, o feedback riguardante un particolare stato dell'interfaccia).

Composition Operators

Gli operatori di composizione possono contenere una o due espressioni, ognuna delle quali contenenti uno o più interattori o, alternativamente, composizione di interattori. Più semplicemente un operatore di composizione è un contenitore, con determinate caratteristiche, che può contenere altri operatori di composizione oppure degli interattori. Quindi una presentazione che non abbia operatori di composizione può contenere al massimo un solo interattore.
In particolare gli operatori di composizione sono:

- Grouping (G): indica un insieme di elementi logicamente connessi ad altri;
- Relation (R): rappresenta una relazione uno a molti tra elementi, un elemento ha determinati effetti sugli altri elementi della relazione;
- Ordering (O): evidenzia un certo ordinamento tra elementi di un insieme;
- Hierarchy (H): definisce diversi livelli di importanza fra un insieme di elementi.

4.3 Concrete User Interface

In questa sezione descriveremo la CUI per la piattaforma TV digitale. La struttura risulta essere molto simile a quella dell'abstract user interface. Inoltre la CUI per TV digitale risulta essere molto simile a quella per piattaforma Desktop a causa del fatto che sono due dispositivi molto simili (ad esempio la dimensione dello schermo è grande in entrambi i casi). Le differenze, che riguardano principalmente l'input e la navigazione, non influenzeranno di molto la definizione della CUI.

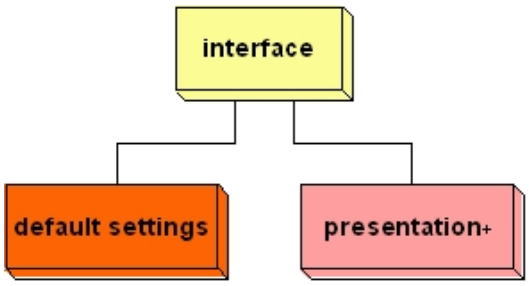

Figura 4.4: Struttura della concrete user interface in TERESA

Un'interfaccia concreta è definita da un certo numero di presentazioni e da delle impostazioni di default.

```
<!ELEMENT concrete_dtv_interface (default_settings,
                                  presentation+)>

<!ELEMENT default_settings (background, font_settings,
                            operators_settings,
                            interactors_settings)>
```

Le impostazioni di default si riferiscono a tutti gli elementi della presentazione, e vengono utilizzate se non si definiscono determinate proprietà relative alla presentazione (che vedremo in seguito).

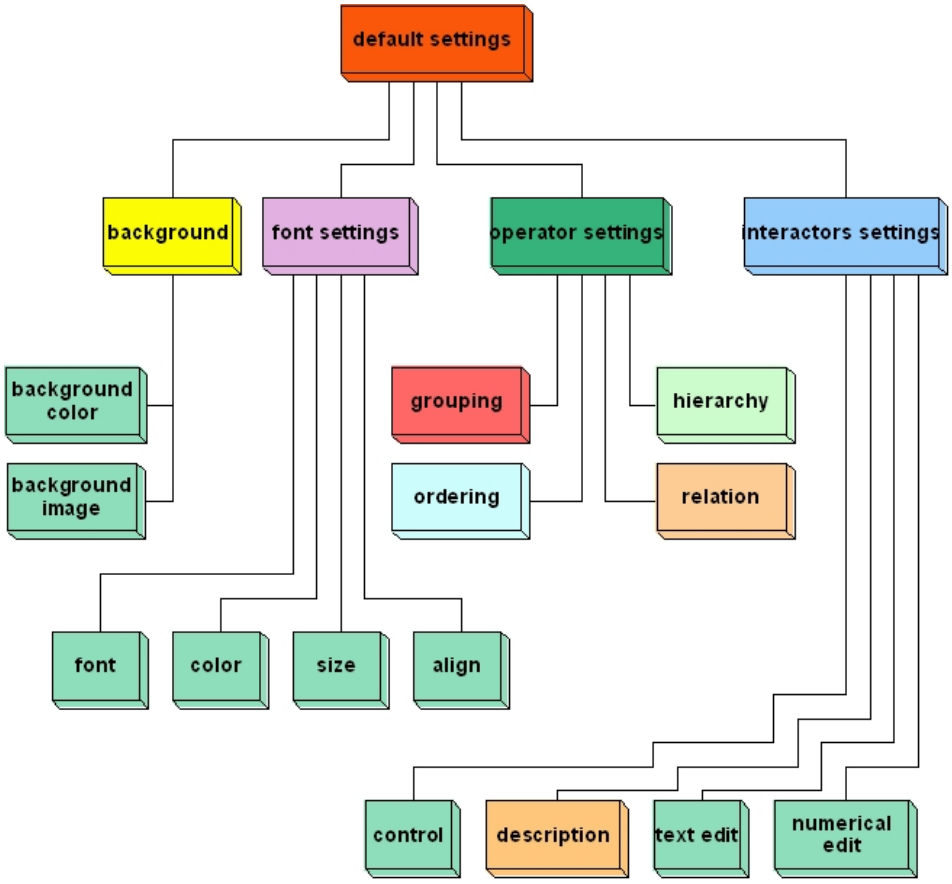

Figura 4.5: Struttura di *default settings*

Ogni presentazione contiene informazioni riguardanti organizzazione e connessione degli interattori con le altre presentazioni e le proprietà della presentazione stessa, come titolo, sfondo o dimensione della font.

```
<!ELEMENT presentation (presentation_properties,
                        connection*,
                        (interactor |
                           interactor_composition))>

<!ELEMENT presentation_properties (title, background,
                                   font_settings, top)>
```

Connection e *interactor composition* sono definiti come nell'AUI e vengono utilizzati per ogni tipo di piattaforma.

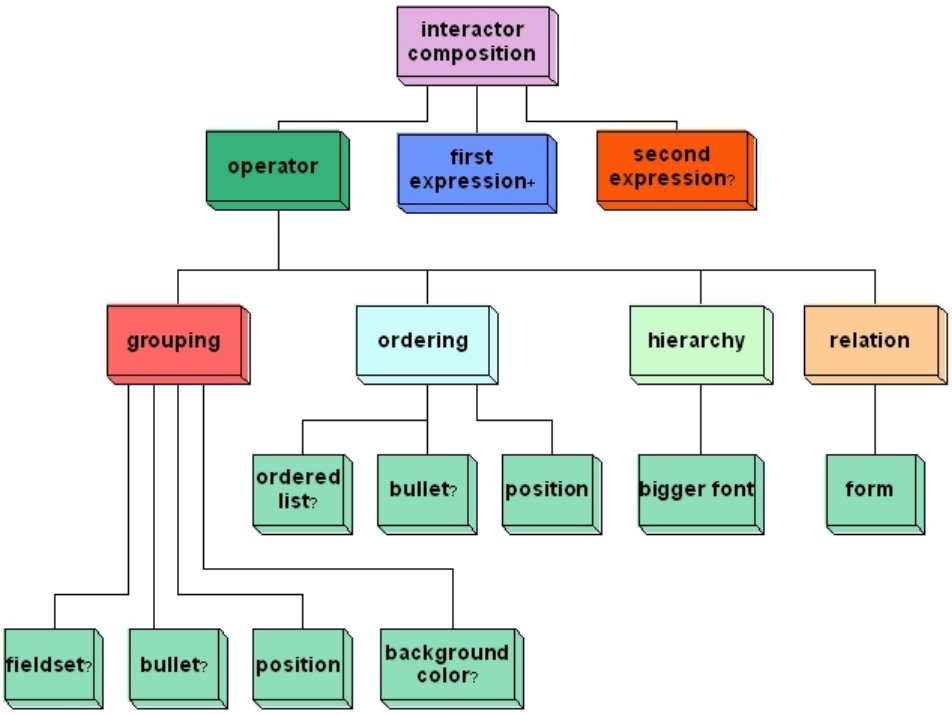

Figura 4.6: Struttura di *interactor composition*

```
<!ELEMENT connection (conn_type)>
<!ATTLIST connection presentation_name IDREF #REQUIRED>

<!ELEMENT conn_type (elementary_conn | complex_conn)>

<!ELEMENT interactor (interaction | only_output)>

<!ELEMENT interactor_composition (operator,
                                  first_expression+,
```

```
                                        second_expression?)>

<!ELEMENT operator (grouping | ordering | hierarchy | relation)>
```

Man mano che si analizzeranno gli interattori in profondità si scenderà più nel dettaglio della piattaforma in questione.

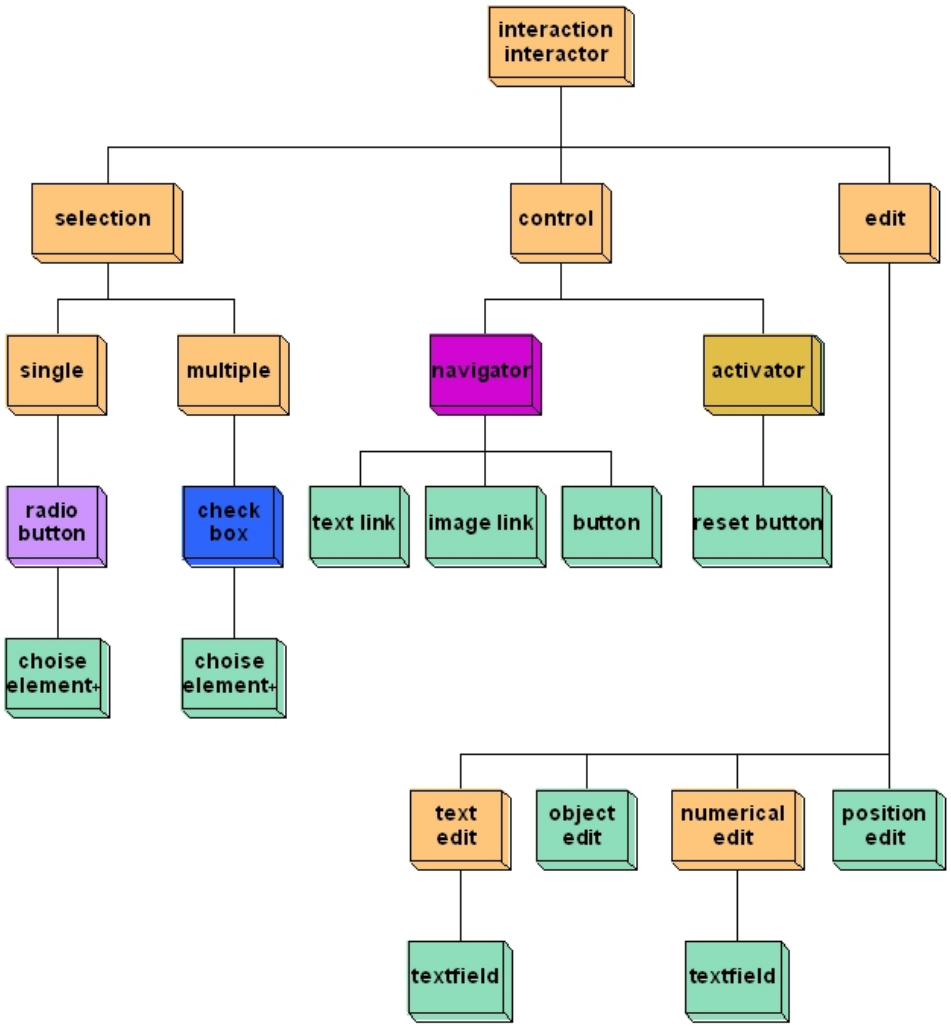

Figura 4.7: Struttura di *interaction interactor*

Ad esempio, consideriamo il caso della selezione singola sulla TV digitale:

```
<!ELEMENT interaction (selection  | editing | control)>

<!ELEMENT selection (single | multiple)>

<!ELEMENT single (radio_button)>

<!ELEMENT radio_button (choice_element+)>
<!ATTLIST radio_button
      label CDATA #REQUIRED
      alignment (%element_selection_alignment;) #REQUIRED>

<!ELEMENT choice_element EMPTY>
<!ATTLIST choice_element
      label CDATA #REQUIRED
      value CDATA #REQUIRED>
```

Come è possibile vedere, la selezione singola sulla TV digitale è implementata mediante *radio button*. Sarebbe stato possibile rimanere più aderenti al modello Desktop in cui la selezione singola è rappresentata nel seguente modo:

```
<!ELEMENT single (radio_button | list_box | drop_down_list)>
```

Ma c'è da considerare che su una TV, non avendo a disposizione alcuna tastiera o mouse, l'implementazione della selezione singola mediante *radio button* risulterebbe di più semplice utilizzo per l'utente.
Questo piccolo esempio permette di capire come l'interfaccia concreta rappresenti proprio le caratteristiche del dispositivo al quale è associato, in termini di operatori e interattori concreti disponibili su quella piattaforma.

Un altro esempio è quello in cui uno stesso interattore è rappresentato allo stesso modo su più piattaforme, ma essendo le caratteristiche dei dispostivi differenti, cambiano gli attributi associati a tali interattori. Ad esempio, l'immissione di testo su TV digitale e su palmare è rappresentato in entrambi i casi mediante *text field*.

```
<!ELEMENT text_edit (textfield)>

<!ELEMENT textfield EMPTY>
<!ATTLIST textfield
      label CDATA #REQUIRED
      length (%length_value;) #REQUIRED
      password (%option;) #REQUIRED>
```

Tuttavia, mentre sul palmare, a causa della ridotta dimensione dello schermo, sono permessi bassi valori per la lunghezza del campo testo

```
<!ENTITY % length_value "4 | 5 | 6 | 7 | 8 | 9 | 10 ">
```

sulla TV è permessa una lunghezza maggiore

```
<!ENTITY % length_value "8 | 9 | 10 | 11 | 12 |13 | 14 |
                         15 | 16 | 17 | 18 |19 | 20">
```

Per quanto riguarda la selezione multipla invece è stata implementata mediante *check box*:

```
<!ELEMENT multiple (check_box)>
<!ATTLIST multiple
      cardinality (%cardinality_value;) #REQUIRED>

<!ELEMENT check_box (choice_element+)>
<!ATTLIST check_box
      label CDATA #REQUIRED>
```

Anche in questo caso la scelta del *check box* permette una migliore facilità di utilizzo sulla TV.
Altro elemento importante è il *control*. Come visto in precedenza tale elemento può essere un *navigator* o un *activator*. Sulla TV digitale si possono esprimere nel seguente modo:

```
<!ELEMENT control (navigator | activator)>

<!ELEMENT navigator (text_link | image_link | button)>
<!ATTLIST navigator
      target CDATA #IMPLIED>

<!ELEMENT text_link EMPTY>

<!ELEMENT image_link EMPTY>

<!ELEMENT button EMPTY>

<!ELEMENT activator (reset_button)>

<!ELEMENT reset_button EMPTY>
```

Un *navigator*, che permette di passare da una presentazione all'altra, viene implementato mediante link (testuale o grafico) e bottoni. Un *activator* invece permette di compiere delle azioni sul altri elementi della presentazione. Il *reset button* permetterà, ad esempio, di azzerare i campi di input precedentemente compilati o annullare le selezioni effettuate. La CUI per piattaforma Desktop permette inoltre di definire un *activator* come *button and script* permettendo di associare alla pressione del bottone l'attivazione di uno script. Non è stato possibile definire tale *activator* sulla piattaforma TV digitale per la mancanza di un interprete JavaScript, ma ciò non significa che non sia fattibile.
Infine rimane da analizzare come sono stati definiti gli operatori di composizione (*grouping, ordering, hierarchy, relation*) sulla TV digitale.

```
<!ELEMENT operator (grouping | ordering | hierarchy | relation)>

<!ELEMENT grouping (fieldset?, bullet?, background_color?, position)>

<!ELEMENT fieldset EMPTY>
<!ELEMENT bullet EMPTY>
<!ELEMENT position EMPTY>

<!ELEMENT ordering ((ordered_list | bullet)?, position)>
<!ELEMENT ordered_list EMPTY>

<!ELEMENT hierarchy (bigger_font)>
<!ELEMENT bigger_font EMPTY>

<!ELEMENT relation (form)>
<!ELEMENT form EMPTY>
```

Il primo operatore è il Grouping. Rappresenta un contenitore di elementi sui

quali è possibile effettuare degli elenchi puntati (*bullet*), incorniciarli (*fieldset*), definire uno sfondo o raggrupparli orizzontalmente o verticalmente (*position*).
L'operatore di Ordering è simile al precedente, ma a differenza di quest'ultimo gli elementi al suo interno sono ordinati e possono essere visualizzati mediante elenchi puntati o numerati.
L'operatore di Hierarchy stabilisce una gerarchia tra gli elementi contenuti. Tali elementi vengono visualizzati in ordine di importanza in base alla dimensione del carattere: più importante è un elemento, maggiore sarà la dimensione del carattere.
Infine si ha il Relation, implementato mediante un contenitore di elementi contenente degli *activator* il cui effetto si ripercuote su tutti gli elementi dell'insieme.
Di seguito è riportata una tabella riassuntiva che rappresenta come i principali elementi logici sono implementati sulla TV digitale.

Only output	**Implementazione su TV Digitale**
Textual	È possibile visualizzare il testo utilizzando un particolare tipo di font: Tiresias. Tale font è stata creata per essere ben leggibile sulla TV. La dimensione del testo dovrebbe essere più grande del normale. Si consiglia una dimensione che varia dai 20/22 per le note a 36 per i titoli.
Object	Vengono visualizzate le immagini nella loro dimensione reale, oppure scalate nel caso in cui la dimensione dovesse essere maggiore dello schermo televisivo. Inoltre è possibile effettuare operazioni di zooming e scrolling.
Description	La descrizione rappresenta testo e/o immagini.

Tabella 4.1: Implementazione degli *only output interactors* sulla TV digitale

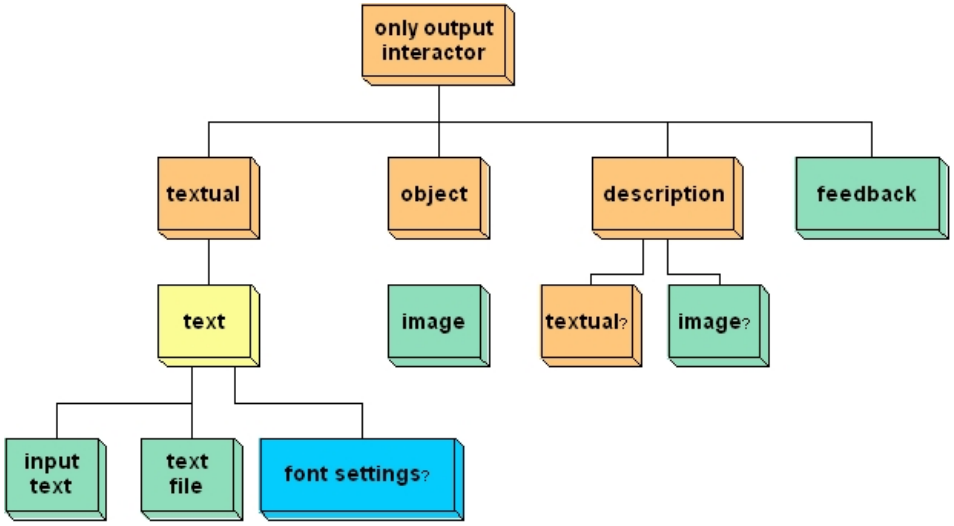

Figura 4.8: Struttura di *only output interactor*

Interaction	Implementazione su TV Digitale
Navigator	Sono stati implementati come *Textual Link, Graphical Link* e *Button*. A tali oggetti sono associate delle operazioni che vengono eseguite alla loro selezione.
Activator	Il Reset Button resetta tutti i campi di una form.
Text Edit	Implementati mediante *text field*. L'inserimento del testo può avvenire selezionando il campo da editare e l'inserimento dei dati avviene mediante una tastiera virtuale.
Numerical Edit	Come il *text edit*, ma l'input è formato da soli numeri che corrispondono direttamente ai tasti 0-9 del telecomando. Non necessita di alcuna tastiera virtuale.
Single Selection	Implementato come *radio button*.
Multiple Selection	Implementato come check box.

Tabella 4.2: Implementazione degli *interaction interactors* sulla TV digitale

Composition	Implementazione su TV Digitale
Grouping	Raggruppa degli elementi in modo orizzontale o verticale. Si possono visualizzare all'interno di un riquadro o meno. Inoltre è possibile visualizzarli come elenco puntato (*bullet*).
Ordering	Ordina gli elementi mediante elenco numerato.
Hierarchy	Visualizza gli elementi in ordine di importanza dall'alto verso il basso. Più importante è un elemento, maggiore è la dimensione del carattere.
Relation	Implementato attraverso un contenitore di interattori e operatori di composizione, nel quale le azioni degli *activator* si ripercuotono su tutti gli elementi dell'insieme.

Tabella 4.3: Implementazione degli *operator composition* sulla TV digitale

4.4 Final User Interface

La generazione dell'interfaccia finale avviene, come detto in precedenza, a partire dall'interfaccia concreta. Nel caso di piattaforma Desktop, ad esempio, viene generato codice XHTML, o nel caso di dispositivi vocali viene generato codice VoiceXML [27]. Per quanto riguarda la TV digitale viene generato e compilato del codice java (Xlet) in grado di poter essere eseguito sui Set-Top-Box. Tale processo può essere riassunto nella figura sottostante.

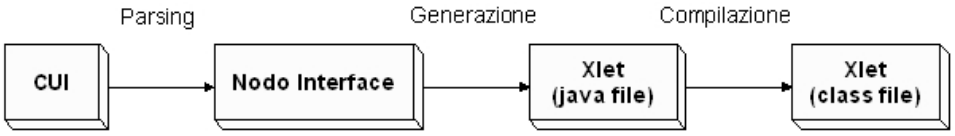

Figura 4.9: Processo di generazione dell'interfaccia finale con TERESA

La Xlet generata è composta da due parti:

- statica: un package xlet che viene utilizzato da tutte le xlet generate;
- dinamica: classe creata a runtime che richiama i metodi del package xlet.

4.4.1 Parte Statica: il package xlet

È stato realizzato un insieme di classi che permettessero la visualizzazione degli interattori e operatori di composizione, in modo da sopperire alla assenza di un browser open source per piattaforma MHP.
Il package xlet contiene le classi e i metodi per la visualizzazione degli interattori e degli operatori di composizione discussi in precedenza.

Only Output	Package xlet
Textual	DTVText
Object	DTVImage
Description	DTVDescription
Only Output	Package xlet
Navigator	DTVLink DTVGraphicalLink DTVButton
Activator	DTVResetButton : DTVButton
Text Edit	DTVTextEdit
Numerical Edit	DTVNumericalEdit
Single Selection	DTVSingleSelection : DTVSelection
Multiple Selection	DTVMultipleSelection : DTVSelection
Composition	Package xlet
Grouping	DTVGrouping : DTVComposition
Ordering	DTVOrdering : DTVComposition
Ordering	DTVHierarchy : DTVComposition
Hierarchy	DTVRelation : DTVComposition

Tabella 4.4: Package Xlet: Interattori ed operatori di composizione

Ognuna di queste classi utilizza il `package HAVi` per la visualizzazione grafica. A queste classi va aggiunta la classe `DTVPresentation`, utilizzata dalla parte dinamica, che contiene i metodi per aggiungere gli elementi alla presentazione, senza entrare nel dettaglio tecnico degli stessi.

```
class DTVPresentation
{

   public void addText();
```

```
    public void addImage();

    public void addDescription();

    public void addButton();

    public void addGraphicalLink();

    public void addResetButton();

    public void addTextEdit();

    public void addNumericalEdit();

    public void addSelection();

    public void addGrouping();

    public void addOrdering()

    public void addHierarchy();

    public void addRelation();

}
```

4.4.2 Parte Dinamica: generazione della FUI

Dopo aver effettuato il parsing dell'interfaccia concreta, si ottiene una struttura dati ad albero che rappresenta la DTD della CUI discussa in precedenza.
È a partire da questa struttura dati che viene generata la Xlet: si tratta di

un file java che chiameremo `Browser.java`. La generazione avviene scorrendo l'albero, contenente tutti gli elementi che compongono l'interfaccia utente, e per ogni elemento, si analizzano i valori, e si aggiorna il file `Browser.java`. Ad esempio, se durante la discesa dell'albero si incontra un *only output element* che rappresenta un campo testo, verrà aggiunta una linea nel file `Browser.java` che richiama il metodo `addText()` della classe `DTVPresentation`. Dopo aver analizzato interamente l'albero, il file `Browser.java` viene salvato e compilato. Il file risultante, `Browser.class`, è pronto per poter essere eseguito sull'emulatore per TV digitale.

Esempio di generazione

Supponiamo che la CUI di partenza si la seguente:

```
<concrete_dtv_interface>
  <default_settings>
   [...]
  </default_settings>
  <presentation name="presentation_1">
    <presentation_properties>
      <title value="presentation_1" />
      <background>
        <background_color value="#FFFFFF" />
      </background>
      <font_settings>
        <font font="Tiresias" />
        <color color="#000000" />
        <size size="28_pt" />
        <align align="Left" />
      </font_settings>
      <top>
        <input_text value="Titolo Pagina" />
      </top>
```

```
      </presentation_properties>
      <interactor id="Text_1_0">
        <only_output>
          <textual>
            <text>
              <input_text value="Hello World!" />
            </text>
          </textual>
        </only_output>
      </interactor>
    </presentation>
  </concrete_dtv_interface>
```

Tale CUI rappresenta una interfaccia avente un titolo 'Titolo Pagina' e un campo testo: 'Hello World!'.
Il risultato (semplificato) della generazione, a partire dalla precedente CUI è il seguente:

```
public class Browser implements Xlet, IDTVConstant {

   [...]

   public void startXlet() throws XletStateChangeException {

      //Aggiunge la confirm Exit
      confirmExitPanel.setVisible(false);
      scene.add(confirmExitPanel);

      //Aggiunge la presentazione presentation_1
      DTVPresentation presentation_1 = new DTVPresentation(
         "presentation_1",X, Y, DTV_WIDTH,PAGE_SPACE,
         "#000000", infoPanel, keyboard,
```

```
            confirmExitPanel, presHash);

        presHash.put("presentation_1", presentation_1);

        //Testo
        presentation_1.addText("Titolo Pagina",3,"Left");

        //Testo
        presentation_1.addText("Text_1_0",2,"Left");
        scene.add(presentation_1);

        //Aggiunge lo sfondo
        presentation_1.addBackgroundColor("#FFFFFF");

        //Aggiunge la barra verticale
        presentation_1.addVerticalBar();

        presentation_1.requestFocus();
        presentation_1.setVisible(true);

        scene.add(infoPanel);
        scene.repaint();
        scene.setVisible(true);
    }
}
```

Come è possibile notare, all'interfaccia utente finale vengono aggiunti ulteriori elementi che non compaiono nella CUI. Si tratta di elementi che non fanno parte della presentazione ma che sono necessari per il corretto funzionamento della pagina. Tali elementi sono:

- information panel: pannello che compare nella parte bassa dell'inter-

faccia e contiene informazioni sull'utilizzo dell'applicazione (pulsanti da premere, come muoversi nell'applicazione, ...);

- vertical bar: barra verticale per effettuare lo *scrolling* della pagina;
- keyboard: tastiera virtuale utilizzata per l'immissione di valori alfanumerici;
- exit panel: pannello che chiede conferma per l'uscita dall'applicazione.

Alcuni di questi oggetti (keyboard, exit panel e vertical bar) sono settati come *not visible* e vengono resi visibili solo quando l'applicazione necessita di un loro utilizzo.

4.4.3 Gestione degli eventi

Ruolo fondamentale per il corretto funzionamento di una applicazione per TV digitale è dato dalla gestione degli eventi, che passa spesso inosservata e per la quale è stato speso molto tempo.
L'assenza di un linguaggio, in particolare l'assenza del supporto al linguaggio DVB-HTML (ovvero di un browser), obbliga, a chi scrive applicazioni, la gestione del supporto all'applicazione stessa. Ad esempio, chi scrive una pagina HTML non si preoccupa di implementare il supporto per un *radio button*. Egli lo dichiara e basta. È insito nel *radio button* stesso che, durante la selezione di un elemento, il precedente venga deselezionato. Ciò non accade in applicazioni MHP, dove attualmente, è chi programma l'applicazione che definisce un supporto per essa.

Le applicazioni per TV digitale generate con TERESA gestiscono contemporaneamente gli eventi in due modi differenti:

- centralizzato: vengono gestiti in modo centralizzato gli eventi relativi, ad esempio, alla selezione degli oggetti della pagina o allo scrolling della pagina stessa;

- distribuito: ogni oggetto (selezione, editing o bottone) gestisce gli eventi in base alle sue peculiarità. Ad esempio, selezionando un campo di editing, sarà esso stesso che si occuperà di mettere a disposizione dell'utente una tastiera virtuale, oppure selezionando un *radio button* bisognerà gestire gli eventi in modo da mantenere la semantica di tale oggetto.

4.5 Authoring

In questa sezione verrà analizzato l'ambiente di authoring che è stato sviluppato per questa tesi, e si discuteranno anche altri ambienti presenti sul mercato.

4.5.1 TERESA UI Editor

Le descrizioni logiche e le trasformazioni discusse sono state incluse in un ambiente di authoring: TERESA. In particolare ci riferiremo allo User Interface editor di TERESA (TERESA UI Editor). La figura sottostante mostra tale ambiente in relazione all'esempio di una applicazione museale.

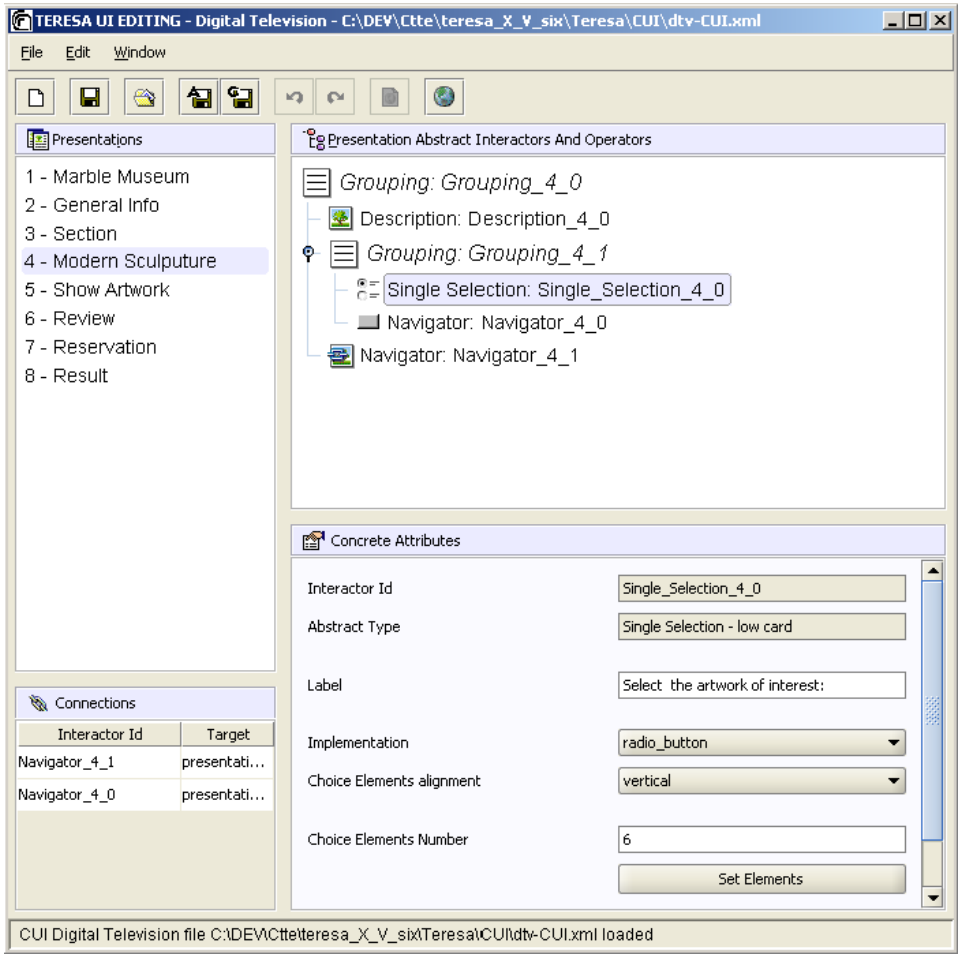

Figura 4.10: TERESA UI Editor: editing di interfacce per TV digitale

L'interfaccia è divisa in quattro pannelli: il primo (alto a sinistra) mostra la lista delle presentazioni facenti parte dell'applicazione; in alto a destra vengono mostrati gli interattori e gli operatori di composizione mediante struttura ad albero; gli attributi concreti relativi all'elemento astratto selezionato sono mostrati nel pannello in basso a destra; infine, in basso a sinistra compare la lista delle connessioni relative a determinati interattori (navigator).
È inoltre possibile modificare le impostazioni relative ad una presentazione (sfondo, titolo, font, ...) mediante apposito pannello.

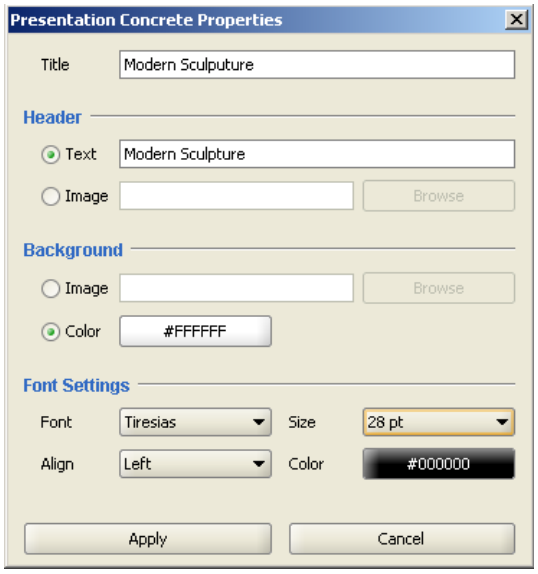

Figura 4.11: TERESA UI Editor: pannello delle proprietà

L'UI Editor di TERESA permette quindi di poter generare l'interfaccia utente finale. Cliccando sull'apposito bottone, il tool permetterà di generare e compilare l'interfaccia utente per la piattaforma televisiva.
Una volta generata, l'interfaccia finale va eseguita su un emulatore, come XletView (discusso in precedenza).

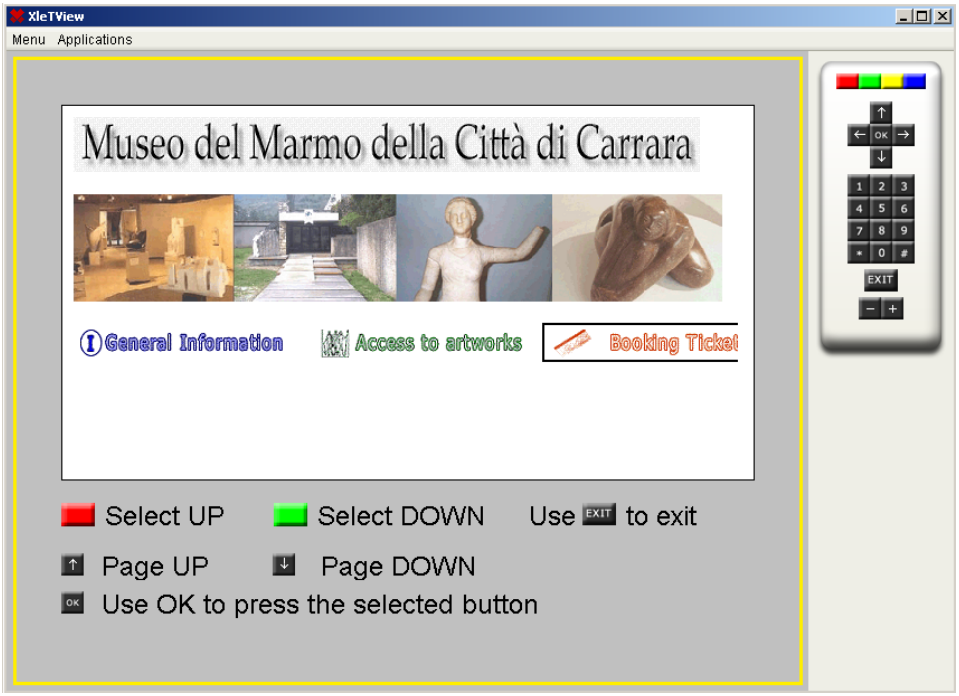

Figura 4.12: L'applicazione museale eseguita sulla TV digitale

Come si può vedere dalla figura, è possibile utilizzare diversi bottoni sul telecomando per poter navigare all'interno dell'interfaccia e selezionare gli elementi che interessano. In questo caso è possibile selezionare tre oggetti, ovvero i tre bottoni: general information, access to artworks, booking ticket.
Scegliendo l'ultimo link (booking ticket) e premendo il tasto 'OK' è possibile navigare all'interno dell'applicazione per accedere alla sezione successiva (figura successiva).

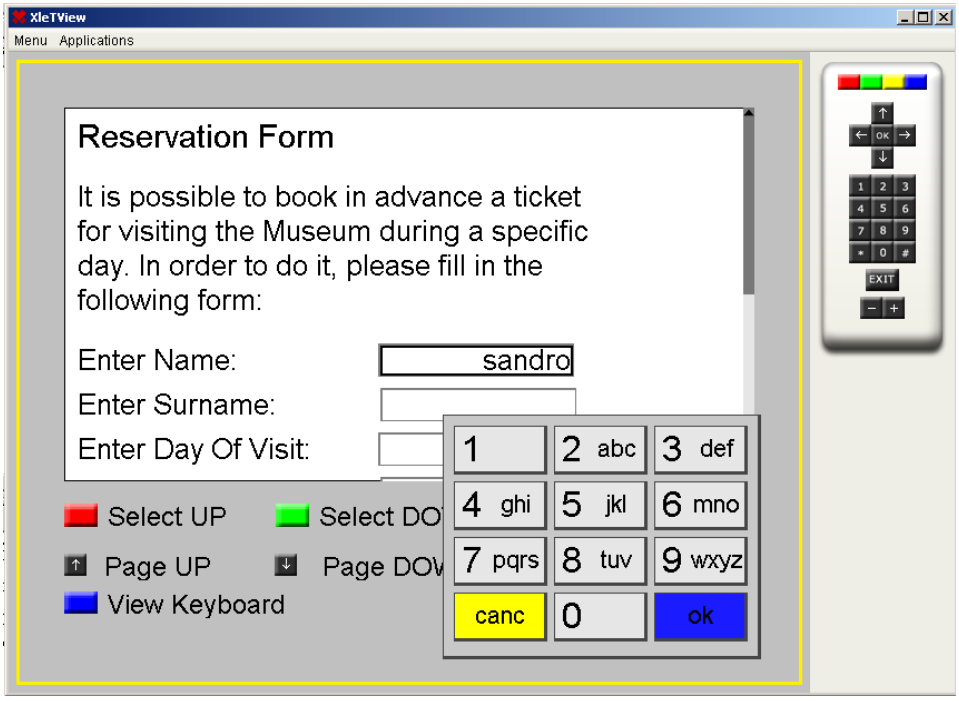

Figura 4.13: Conclusione della prenotazione

Gestendo opportunamente il meccanismo degli eventi è possibile, a seconda dell'oggetto selezionato, attivare determinate funzionalità. Ad esempio, selezionando un bottone o un link è possibile premere 'OK' o selezionando un campo di editing possibile premere il pulsante blu per visualizzare una tastiera.

4.5.2 Altri ambienti di authoring

Non si può dire che TERESA UI Editor sia l'unico ambiente di authoring per sviluppare progetti interattivi basati su standard MHP, però possiamo affermare che è l'unico che sfrutta descrizioni logiche per la generazione di interfacce utenti finali. Inoltre la maggior parte degli ambienti presenti sul mercato sono a pagamento. Alcuni di questi sono stati testati, di altri invece sono state lette le recensioni [30] e analizzate le caratteristiche. In questa sezione verranno analizzati in breve i più importanti tra gli ambienti presenti sul mercato.

Icareus: iTV suite

Con iTV Suite [31] lo sviluppatore può concentrarsi sul contenuto e il look & feel dell'interfaccia e sulla struttura del servizio interattivo. Il sistema è pensato per generare con semplicità sistemi ipertestuali in MHP, mentre per quanto riguarda servizi dinamici con canale di ritorno occorre una maggior dimestichezza con il codice. Ben fatto l'emulatore interno che permette di testare le pagine in tempo reale.

Figura 4.14: Icareus: iTV suite

Gli sviluppatori finlandesi di Icareus rilasciano versioni aggiornate del prodotto, correggendo bugs e seguendo il cliente nelle fasi di installazione e training on the job.

Pontrega

La società Nionex, attiva nei servizi IT, offre la possibilità di scaricare gratuitamente (trial) la piattaforma 'pontegra' [32] basata sull'HTML.

Con tale piattaforma si possono sviluppare in modo facile e veloce i servizi TV interattivi. È sufficiente disporre di conoscenze di HTML per trasmettere i contenuti nella televisione interattiva. Pontegra funziona come un browser che supporta in modo del tutto compatibile un subset dello standard DVB-HTML.

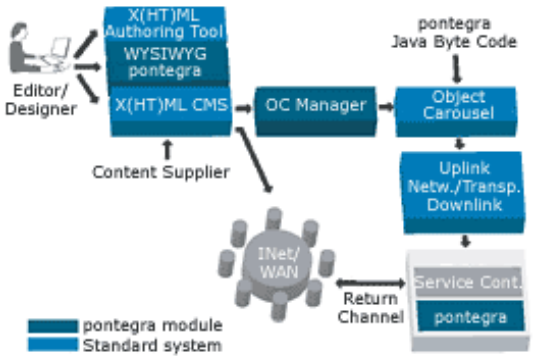

Figura 4.15: Pontrega HTML browser

I contenuti e i servizi possono essere caricati sui set-top box in commercio, come avviene per altre applicazioni MHP.

My DTT

Forte dell'esperienza maturata nella creazione della prima televisione interattiva di massa, MyTV [33] ha raccolto la sfida della TV digitale, giocando da subito un ruolo di primo piano.
MyTV è un punto di riferimento in Italia nella sperimentazione sulla TV digitale. Da oltre quattro anni sviluppa sistemi, prodotti e soluzioni esclusive che saranno presto distribuiti anche sul mercato internazionale.
Tra i più importanti my-dtt©, la piattaforma scelta da RAI come ambiente di riferimento che abilita le emittenti televisive e le istituzioni allo sviluppo e all'erogazione di servizi informativi per la TV digitale. Tra questi:

- Superteletext
- Guida TV
- T-Government
- Sport events
- Smart card applications

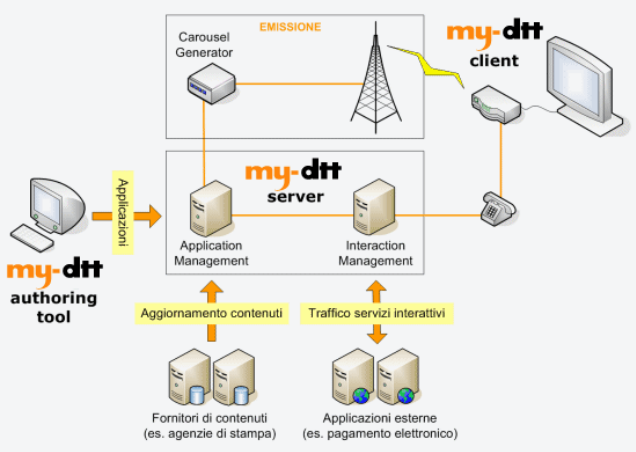

Figura 4.16: My DTT

- Enhanced TV channel
- Content related application
- T-learning
- Interactive advertising application

SetMedia

SetMedia [34] è una suite per la gestione dei contenuti digitali, da trasmettere via DTT ma anche via Internet e rete cellulare.

Figura 4.17: SetMedia

Dal punto di vista delle funzioni, SetMedia si divide in due prodotti: SetMedia Studio e SetMedia Suite. Il primo si occupa della creazione e gestione dei

contenuti in una filosofia per gruppi di lavoro, arrivando sino alla distribuzione dei contenuti medesimi secondo palinsesto e alla gestione del canale di ritorno, quando esso è presente. Il controllo del 'return channel' permette tra l'altro di attivare funzioni di profilazione degli utenti e di inviare contenuti personalizzati. L'elemento caratterizzante di SetMedia Studio sta nel suo essere indipendente dal media prescelto: il Digitale Terrestre è quello più innovativo, ma la suite di Interattiva controlla anche la distribuzione di contenuti via Internet e via dispositivi mobili.
SetMedia Suite è invece un insieme di applicazioni MHP pre-sviluppate per alcuni ambiti specifici: dalla comunicazione di marketing all'intrattenimento, dai servizi informativi sino a quelli di e-government.

4.6 Esempio di progettazione di una applicazione interattiva

In questa sezione vedremo come progettare una applicazione interattiva per TV digitale utilizzando TERESA.

Supponiamo di voler costruire una semplice applicazione di commercio elettronico, composta da due pagine. La prima pagina conterrà un link alla seconda. La prima pagina vogliamo che sia fatta nel seguente modo:

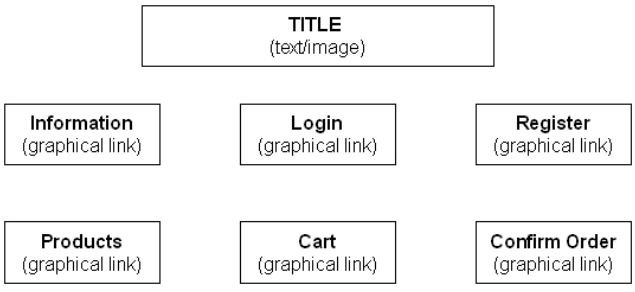

Figura 4.18: Layout della pagina principale dell'applicazione

Innanzitutto è necessario eseguire TERESA UI Editor per progettare l'applicazione interattiva per TV digitale.

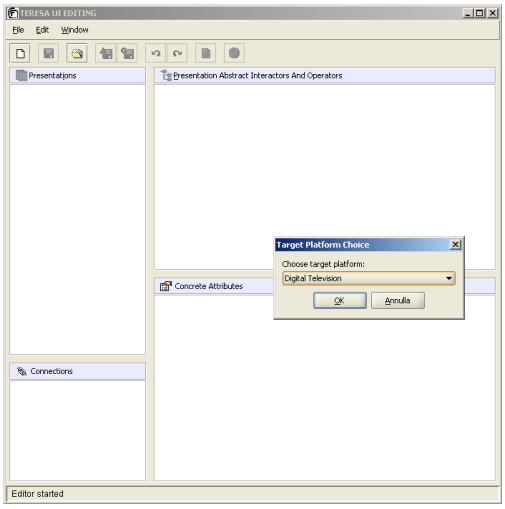

Figura 4.19: TERESA UI Editor: selezione della piattaforma

A questo punto bisogna aggiungere una nuova presentazione, corrispondente alla prima pagina della nostra applicazione.

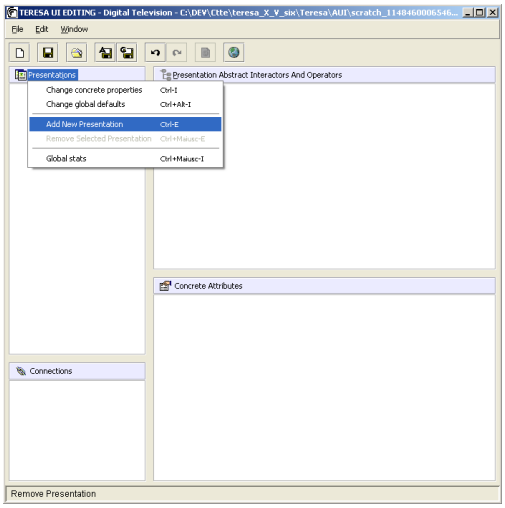

Figura 4.20: TERESA UI Editor: aggiunta prima presentazione

Adesso siamo pronti per poter aggiungere gli oggetti alla pagina. È però importante utilizzare gli operatori di composizione in modo appropriato affinché gli oggetti vengano disposti all'interno della pagina nel modo corretto. La pagina quindi conterrà tre *grouping*. Il primo, corrispondente a quello principale, disporrà gli elementi verticalmente, su 3 righe: la prima riga sarà occupata dal titolo della pagina; la seconda e la terza riga conterranno ognuna i tre link disposti in modo orizzontale. Per far ciò è necessario che tali link siano contenuti a loro volta in un altro *grouping* che li disporrà orizzontalmente.
Tale schema è raffigurato nella figura seguente.

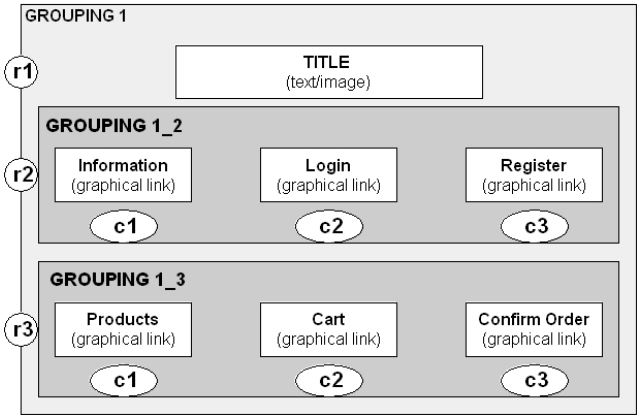

Figura 4.21: Utilizzo degli operatori di composizione per un corretto layout

Adesso possiamo procedere con l'inserimento del *grouping* principale su TERESA (fig 4.22) e con l'inserimento della prima riga, corrispondente all'header della pagina (title), e della seconda, contenente i tre link (Information, Login e Register) (fig. 4.23).

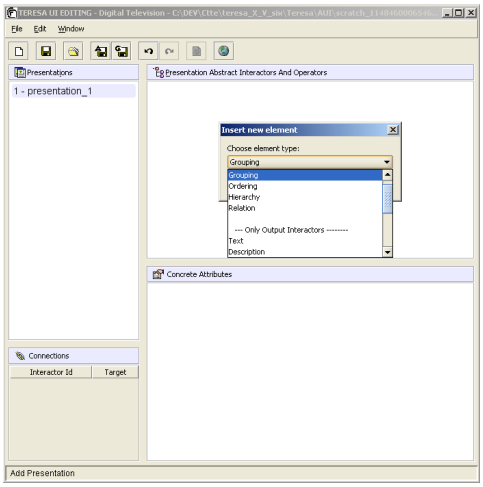

Figura 4.22: TERESA UI Editor: aggiunta del grouping principale

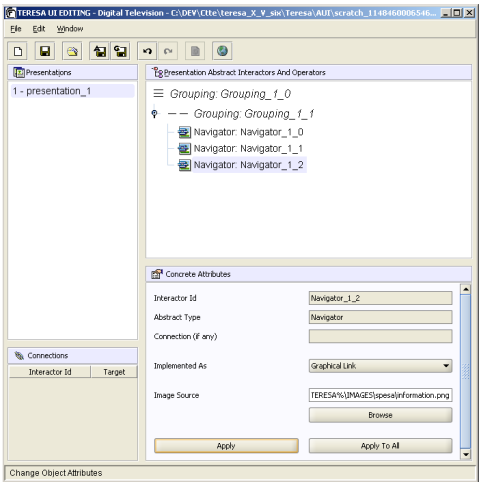

Figura 4.23: TERESA UI Editor: aggiunta dei link

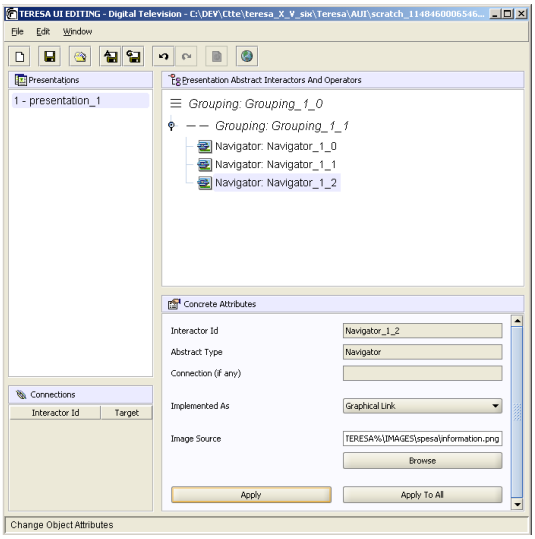

Figura 4.24: TERESA UI Editor: aggiunta dell'immagine corrispondente all'header

La pagina non è ancora completa, ma è utile cominciare a vedere come verrà visualizzata sulla TV digitale.

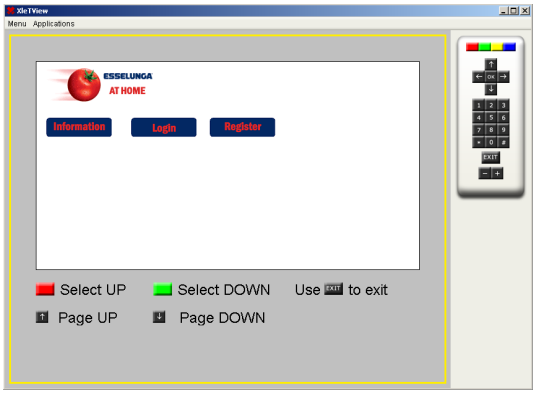

Figura 4.25: Applicazione interattiva su XletView

Il layout della pagina è risultato corretto. Quindi possiamo procedere con l'inserimento della terza riga, contenente agli altri tre link.

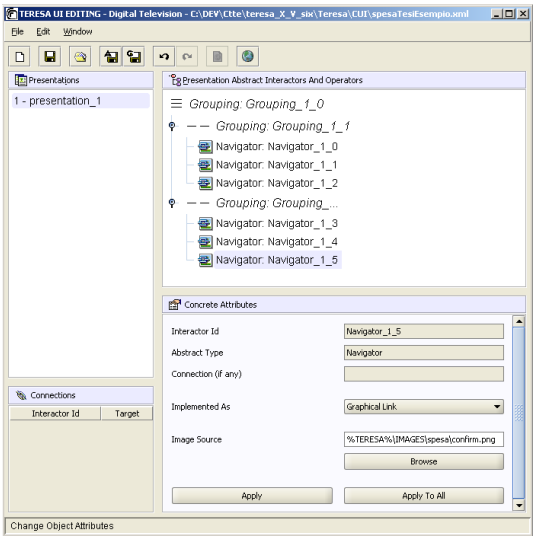

Figura 4.26: TERESA UI Editor: aggiunta degli altri link

La pagina principale è completa e possiamo procedere con la creazione della seconda presentazione.

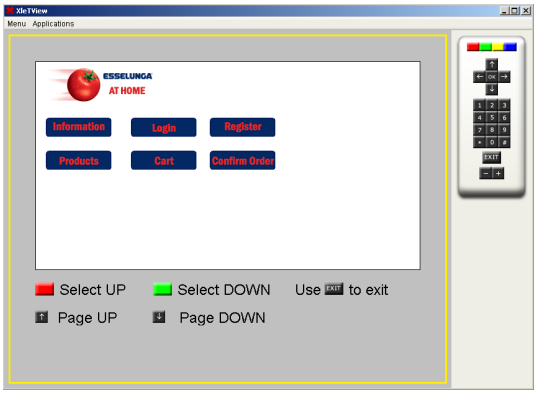

Figura 4.27: Prima presentazione eseguita su XletView

Supponiamo di voler creare la presentazione relativa al link 'Confirm Order'. Tale presentazione risulterà essere simile alla precedente, ma con l'aggiunta di:

- un campo testo che identifica la pagina 'Confirm Order';
- due campi per l'immissione di login e password;

- un bottone 'Reset';
- un bottone 'Submit'.

Questa volta la pagina non avrà come contenitore principale un *grouping* ma un *relation* affinche il 'Reset' possa avere effetto sui campi della presentazione. Il layout della nuova presentazione dovrà essere quello mostrato nella figura seguente.

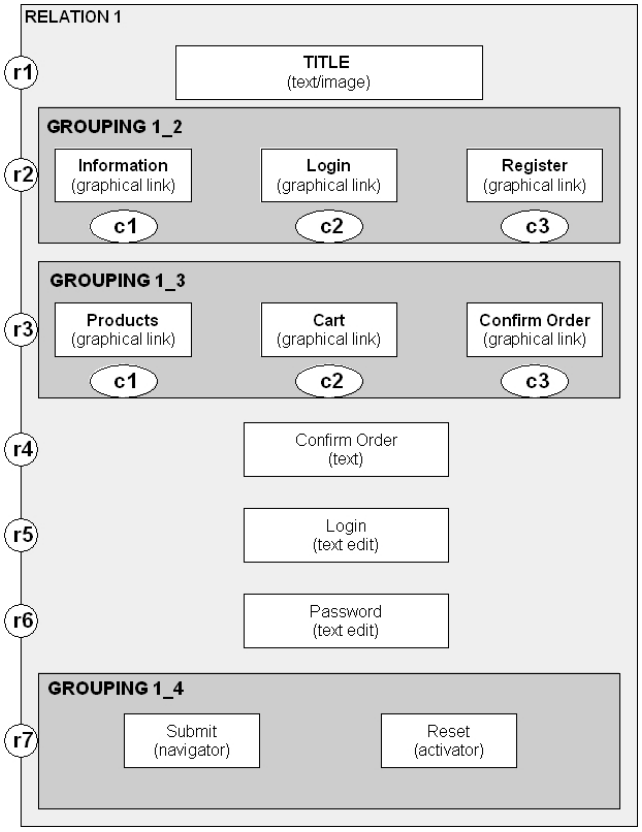

Figura 4.28: Layout della seconda presentazione

Il procedimento utilizzato per la progettazione della prima presentazione sarà lo stesso a quello che utilizzeremo per la seconda presentazione. Infatti, in TERESA, si aggiungerà una nuova presentazione contenente gli oggetti schematizzati in precedenza.

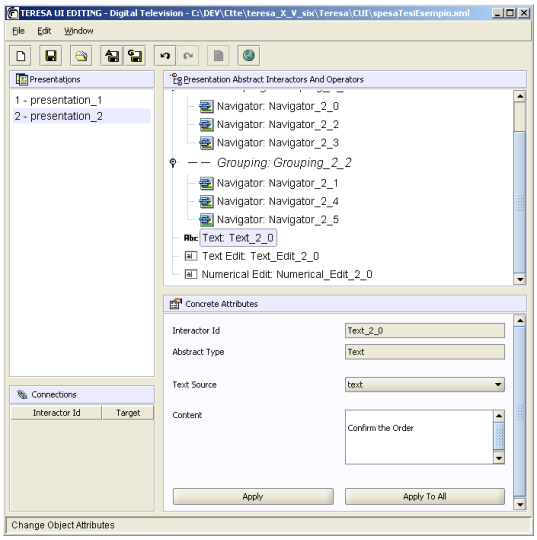

Figura 4.29: TERESA UI Editor: editing della seconda presentazione

Essendo completata anche la seconda presentazione è possibile visualizzarla con XletView.

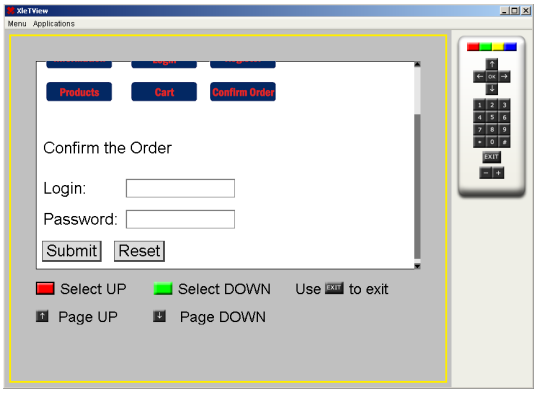

Figura 4.30: Seconda presentazione eseguita su XletView

Rimane infine da collegare le due presentazioni mediante una connessione.

Figura 4.31: Connessione fra le due presentazioni

Tale operazione bisogna eseguirla da TERESA

Capitolo 5

Ambiente di migrazione

5.1 Introduzione

La generazione dell'interfaccia finale, quella che poi potrà essere scaricata ed eseguita sul set-top-box, può avvenire secondo due schemi:

- Editing dell'interfaccia
- Migrazione

Il primo, visto nel capitolo precedente, permette di utilizzare un editor per la realizzazione dell'interfaccia, lo User Interface Editor, parte integrante del tool TERESA, che permette di editare descrizioni logiche di interfacce utenti per la TV digitale.
Il secondo, che analizzeremo in questo capitolo, farà si che l'interfaccia utente verrà generata automaticamente a partire da un altro dispositivo. In tal caso si parla di interfacce migratorie.

Le interfacce migratorie sono interfacce utenti che si spostano da un dispositivo ad un altro mantenendo invariato il contesto dell'applicazione. Tali interfacce sono un aspetto fondamentale per gli imminenti sistemi ubiqui, permessi dallo sviluppo della tecnologia wireless e dalla proliferazione di una vasta gamma di dispositivi interattivi. I sistemi ubiqui costituiscono una particolare classe degli odierni sistemi distribuiti. Essi sono caratterizzati da una forte

eterogeneità (nei paradigmi di comunicazione adottati e nelle architetture degli elementi costituenti) e dalla mobilità dei dispositivi.
Nel prossimo futuro, gli utenti potranno accedere a diversi tipi di dispositivi, e il sistema permetterà loro di potersi muovere da un dispositivo all'altro in modo attivo. Ad esempio durante la navigazione su un PC l'utente può scegliere di migrare su un altro dispositivo (in tal caso si tratta di TV digitale). Un proxy server apposito si occuperà di generare l'interfaccia utente per TV digitale a partire dalla pagina sorgente (una pagina XHTML). Sarà compito del processo di migrazione che il contesto del dispositivo sorgente sia mantenuto sul dispositivo target. Ad esempio, se un utente ha compilato dei campi sulla pagina HTML sorgente, questi devono risultare compilati anche sulla TV digitale dopo che il processo di migrazione è terminato.
Le interfacce migratorie richiedono quindi che la sessione sia persistente e che l'interfaccia utente sia capace di riadattarsi al nuovo dispositivo. L'effetto delle interfacce migratorie è che l'utente si può muovere da un dispositivo ad un altro in modo da continuare il lavoro che stava svolgendo, potendo scegliere dinamicamente il dispositivo di destinazione.

5.2 Architettura del Sistema

L'architettura del sistema di migrazione si basa su un migration/proxy server (GeReMi) [7] che riceve richieste di migrazione ed effettua a runtime l'adattamento dell'interfaccia per il dispositivo sul quale è stato deciso di migrare. Inizialmente l'ambiente supporta applicazioni web sviluppate per piattaforma desktop mettendo a disposizione un'interfaccia per la migrazione.
Gli utenti che vogliono accedere al servizio, devono caricare un client di migrazione sui loro dispositivi. Questa parte di software si occupa della comunicazione con il server per l'invio delle richieste di migrazione.
Le caratteristiche del migration/proxy server possono essere spiegate analizzando situazioni differenti:

- Simple Proxy: l'utente accede al web attraverso un dispositivo che ap-

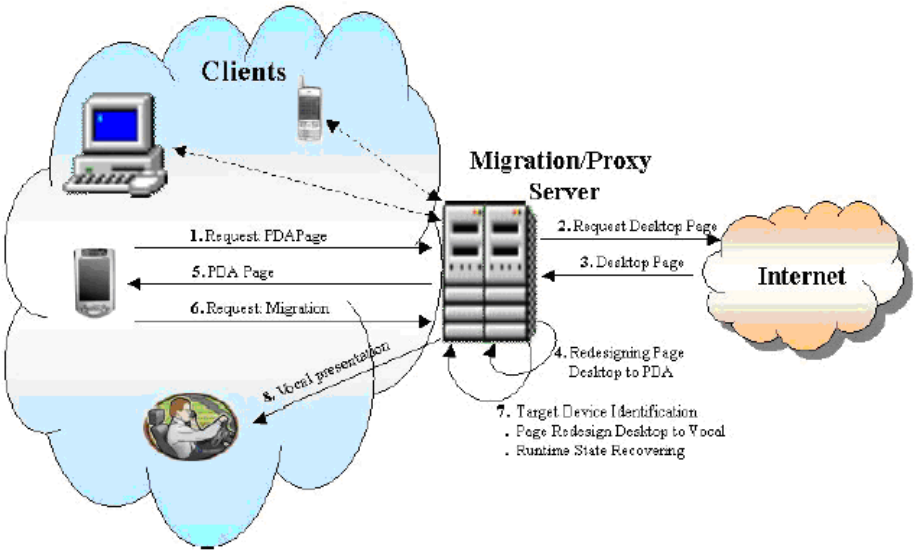

Figura 5.1: Architettura del sistema di migrazione

partiene alla stessa piattaforma per la quale la pagina era stata creata. Il migration/proxy server ottiene la pagina richiesta dal web server e la passa al client.

- Interface Redesign: l'utente accede al web attraverso un dispositivo che appartiene ad una piattaforma differente per la quale la pagina era stata creata. Ad esempio, un dispositivo televisivo accede a pagine create per Desktop. Inizialmente, il migration/proxy server recupera la pagina dal web server, la pagina viene riadattata (redesign) per la TV digitale ed è così che può essere eseguita sulla TV.

- Migration from Desktop: l'utente accede al web attraverso un dispositivo che appartiene alla stessa piattaforma per cui la pagina era stata creata ed a un certo punto decide di migrare verso un altro dispositivo (ad esempio TV digitale). In prima istanza viene selezionato il dispositivo verso il quale si vuole migrare. La pagina viene scaricata e viene effettuato automaticamente il redesign per TV digitale preservando la continuità. Viene acceduto il DOM della pagina per mantenere nella nuova interfaccia i campi già inseriti nel dispositivo sorgente. Alla fine di

tale processo, nel caso in cui il dispositivo target sia un PDA, la pagina viene inviata a tale dispositivo. Nel caso in cui il dispositivo target sia la TV digitale, viene creata la Xlet che rappresenta la pagina in questione, per poter essere successivamente eseguita sull'emulatore.

- Migration from non-Desktop platform: l'utente accede al web attraverso un dispositivo differente per cui la pagina è stata creata (ad esempio PDA) e successivamente viene effettuata una migrazione su piattaforma televisiva. Questo è il caso più complesso e interessante che coinvolge tutte le funzionalità del migration/proxy server. Questo caso differisce dai precedenti in quanto sia il dispositivo sorgente che il dispositivo target non appartengono all'insieme per cui la pagina web era stata creata e in entrambi i casi deve essere effettuato il redesign della pagina. Durante il redesign della pagina per il dispositivo target, il migration/proxy server sfrutta la descrizione logica relativa all'interfaccia della piattaforma sorgente.

5.3 Generazione della descrizione logica

La generazione dell'interfaccia utente per la nuova piattaforma inizia con il *reverse engineering*. Dalla pagina web sorgente si ottengono tre descrizioni logiche: concrete user interface, abstract user interface e task model (quest'ultima non verrà trattata).

A livello implementativo le informazioni riguardanti la descrizione concreta sono integrate in quella astratta. Infatti, la descrizione concreta è un rifinimento che aggiunge informazioni riguardanti attributi concreti alle strutture fornite dalla descrizione astratta. L'interfaccia astratta è indipendente dalla piattaforma, così gli elementi sono identificati in termini di ciò che supportano (ad esempio: selezione, editing, activator, ...). L'interfaccia concreta, dipendendo dalla piattaforma, fornisce la descrizione dell'interfaccia utente, definendo come gli interattori e gli operatori di composizione sono implementati in quella determinata piattaforma.

La descrizione astratta è usata nella fase di redesign per guidare i cambiamenti di alcuni oggetti e delle loro caratteristiche, riorganizzando la loro distribuzione nelle pagine riprogettate e mantenendo la semantica nella nuova piattaforma.

Dall'interfaccia utente alla sua descrizione logica

L'interfaccia concreta è il livello logico più vicino all'interfaccia utente finale. Come descritto nel capitolo precedente, è organizzata in presentazioni collegate tramite connessioni.

Viene effettuato il reverse dell'interfaccia utente (pagine web), una pagina per volta, in una presentazione concreta. L'algoritmo di reverse lavora sul DOM della pagina, ed è quindi necessario che le pagine X/HTML siano ben formate (valide e conformi agli standard). Spesso ciò non accade ed è quindi necessario, prima di iniziare la fase di reverse, che venga effettuato il parsing della pagina usando il parser Tidy W3C [37] che corregge i possibili *mismatch* e restituisce un DOM corretto. Il DOM della pagina viene quindi analizzato e per ogni nodo vengono estratte le informazioni necessarie per generare i rispettivi interattori o operatori di composizione. In relazione al DOM X/HTML è possibile avere tre casi:

- Il nodo è mappato in un interattore concreto, inserendolo nell'interfaccia concreta (CUI). Il nodo in questione è una foglia. Questo è l'esempio di tag `<img>`, `<a>` o `<select>` che causano la generazione dei seguenti interattori: *object, navigator e selection.*

- Il nodo corrisponde ad un operatore di composizione. Viene quindi generato il relativo operatore di composizione e viene effettuata una chiamata ricorsiva sul sottoalbero per generare gli elementi facenti parte di questo operatore di composizione (che a sua volta possono essere interattori o operatori di composizione). Questo è l'esempio del tag `<form>` che corrisponde all'operatore di composizione *Relation.*

- Il nodo non richiede la creazione di un'istanza di un interattore nell'in-

terfaccia concreta. In tal caso nessun nuovo elemento viene aggiunto. Ciò può avvenire nel caso di paragrafi vuoti `<p></p>` o `<br>`.

5.4 Redesign

Il passo di redesign produce vari cambiamenti nell'interfaccia utente. In particolare, il redesign è necessario per riadattare l'interfaccia alla nuova piattaforma, mantenendo la semantica della descrizione logica (creata nella fase di reverse). Ad esempio, se si decide di migrare su un PDA, bisogna tenere conto della dimensione dello schermo, e quindi dividere la presentazione in più pagine.

Per quanto riguarda la migrazione su TV digitale non vengono effettuati particolari cambiamenti in quanto la piattaforma televisiva è molto simile a quella Desktop. A esempio, nel caso in cui la lunghezza della presentazione sia maggiore di una pagina, verrà data la possibilità all'utente di effettuare lo scrolling della pagina.

Piccoli cambiamenti si hanno sulla selezione, singola e multipla. Per questione di usabilità si è preferito implementare la selezione singola mediante *radio button* e la selezione multipla mediante *check box*. È infatti di difficile utilizzo un *list box* sulla TV.

Dal passo di redesign è possibile ottenere l'interfaccia concreta per il dispositivo target e quindi generare l'interfaccia utente finale per tale dispositivo.

Il processo termina proprio con la generazione dell'interfaccia utente del dispositivo verso il quale si è migrato.

5.5 Estensione del server

La modularità con la quale è stato progettato il migration/proxy server ha permesso di poterlo estendere per la nuova piattaforma in modo abbastanza semplice. Come è stato descritto nel capitolo precedente, TERESA è in grado

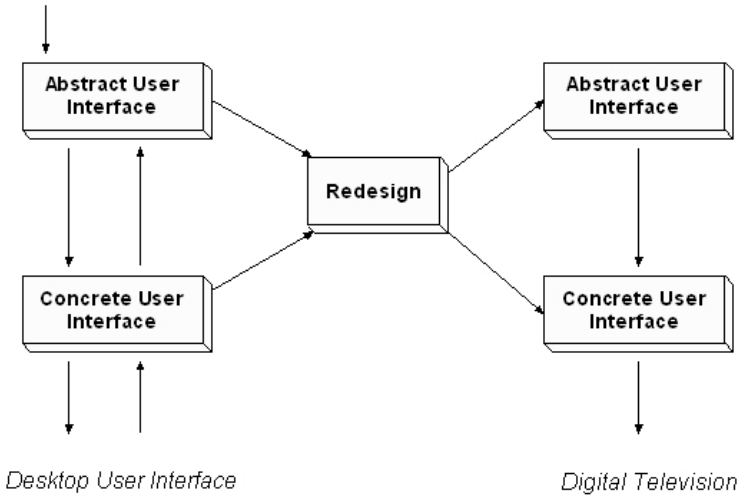

Figura 5.2: Passi per la generazione dell'interfaccia finale

di generare interfacce utenti finali (FUI) a partire dall'interfaccia concreta (CUI) o da quella astratta (AUI).

Il migration/proxy server invece è in grado di effettuare il reverse engineering della pagina web, risalendo alla sua interfaccia concreta (CUI). Un altro file xml viene utilizzato per salvare il contesto dell'applicazione, ovvero lo stato di tutti gli elementi della pagina. Quindi se ad esempio nella pagina era stato selezionato un *check box*, questo dovà essere salvato nel contesto in modo che tale selezione possa comparire nell'interfaccia utente finale della nuova piattaforma.

È stato aggiunto un modulo (redesign) che permettesse la trasformazione di una CUI per interfaccia Desktop in una CUI per TV digitale.

Tale modulo è un parser XML che, leggendo la CUI per piattaforma Desktop, la modifica riadattandola per la TV digitale. Il risultato è la CUI per TV digitale, dalla quale può essere generata l'interfaccia utente finale.

Durante la generazione dell'interfaccia utente finale, è necessario eseguire delle operazioni per preservare la continuità. Il contesto relativo all'interfaccia utente sorgente dovrà essere mantenuto nell'interfaccia utente finale.

Ogni oggetto dell'interfaccia utente possiede un *id* univoco. Chiaramente tale

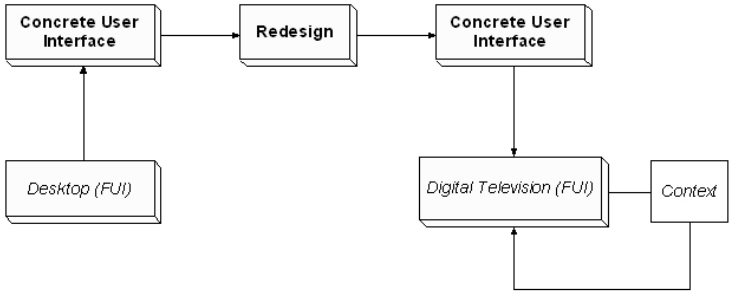

Figura 5.3: Generazione dettagliata dell'interfaccia finale

identificatore viene mantenuto nell'interfaccia utente finale. In tal modo è possibile identificare gli oggetti e attribuire i valori contenuti nel contesto. In linea di massima è possibile immaginare il contesto come una sequenza di coppie (id, valore). Infatti viene utilizzata una tabella hash per mantenere il contesto in memoria. Ogni volta che un nuovo oggetto dell'interfaccia utente finale viene creato, si accede alla tabella hash, si recupera il valore relativo all'oggetto, e si crea il nuovo oggetto avente il valore appena ottenuto.

5.6 Scenari

Per comprendere al meglio gli ambienti di utilizzo del processo di migrazione, verranno analizzati alcuni possibili scenari di utilizzo.

Scenario 1 - Mancanza del PC
Luca lavora come impiegato presso una filiale della banca Antonveneta. A fine giornata, prima di tornare a casa, decide di rimanere qualche minuto in più per potere usufruire di internet, dato che non ha una connessione disponibile a casa sua. In vista del viaggio a Barcellona programmato per la settimana seguente con la sua fidanzata, decide di dare un'occhiata ad un portale dedicato alla città catalana per potere programmare le visite turistiche nella città (musei, mostre, spettacoli). La giornata lavorativa è stata però faticosa, Luca è stanco e non vorrebbe trattenersi in ufficio ancora per molto. Preferirebbe inoltre

accedere a queste informazioni con la sua fidanzata, in modo da organizzare insieme a lei le giornate a Barcellona. Luca allora decide di migrare sulla sua TV Digitale per continuare a vedere a casa tale sito. Il servizio di migrazione genererà automaticamente un codice relativo alla migrazione, in modo che, quando deciderà di proseguire la visualizzazione del portale da casa, davanti la TV, gli basterà immettere tale codice.

Scenario 2 - Tra una pubblicità e l'altra

È lunedì sera e Simone, studente universitario, sta dando un'occhiata al sito della sua facoltà per organizzare il suo piano di studi del secondo semestre. Su canale 5 però, sta per iniziare uno dei film preferiti, il primo episodio di Guerre Stellari. Decide allora di migrare sulla sua TV digitale in modo da poter continuare la compilazione del piano di studi online durante le lunghe pause pubblicitarie, e senza doversi alzare ed andare nell'altra stanza al suo PC.

Scenario 3 - Rifiuto di utilizzo del PC

Roberto e Sara sono una coppia di pensionati della provincia di Lucca. Hanno deciso di andare a trovare il figlio che lavora come sistemista, presso una azienda di Roma. Purtroppo non hanno una cartina e non vorrebbero avventurarsi nel viaggio senza avere idea della strada da percorrere. Inoltre non hanno nemmeno il computer, non hanno mai voluto imparare ad usarlo. Allora chiedono aiuto al figlio che decide di andare su internet e consultare il percorso su viamichelin.it. Il figlio effettua quindi una migrazione sulla Tv digitale dei genitori. In questo modo Roberto e Sara, pur non sapendo neanche in cosa consista internet, potranno consultare la mappa comodamente sul loro televisore.

Scenario 4 - Servizio offerto dal Comune di Pisa

Il comune di Pisa ha realizzato un portale informativo che contiene informazioni di vario genere riguardanti la città, come: orari degli autobus, viabilità, orari degli uffici del comune, ecc... . Nonostante il grande servizio offerto da

questo portale il comune ha deciso di estendere il servizio a tutti i cittadini e non solo, quindi, a coloro che possono fruire di una connessione ad internet. Il servizio sfrutta la piattaforma televisiva digitale. Ad ogni modifica effettuata sul portale web verrà avviato automaticamente un processo di migrazione (uno a molti) su iTV. Tale processo genererà l'interfaccia per la nuova piattaforma scaricabile via http (o via broadcasting) sul Set-Top-Box.

Scenario 5 - Servizio offerto dalla COOP
Laura è una madre di famiglia che solitamente si serve della COOP per fare la spesa. Grazie al nuovo servizio offerto, quando Laura va a fare la spesa, viene generata una nuova interfaccia per piattaforma digitale associata al suo codice cliente contenente le informazioni sulla spesa effettuata. In tal modo, Laura, quando torna a casa, può controllare sulla sua TV digitale lo storico delle spese effettuate alla COOP e quindi può comodamente confrontare i prezzi che compaiono sui volantini dei vari supermercati con i prezzi della COOP. Inoltre la COOP permette di visualizzare le offerte settimanali.

Scenario 6 - Questione di comodità
Marco è un medico e si trova in un albergo di Roma per un convegno a cui egli partecipa attivamente. Prima di mettersi a dormire vuole riguardare (sul suo palmare) per l'ultima volta ciò che dovrà dire il giorno seguente al convegno. È però stanco e non vuole affaticare la vista leggendo sul piccolo display del suo palmare. Allora decide di effettuare una migrazione sulla TV digitale e di rileggere comodamente disteso sul letto le informazioni che lo riguardano.

Scenario 7 - Supporto agli anziani
Luisa è una nonna di 78 anni che per vari problemi si trova in una casa di riposo in Garfagnana. Luisa ha due nipoti, Francesco e Chiara che, essendo studenti universitari, si trovano a diversi chilometri di distanza dalla nonna. Per questi motivi non possono andarla a trovare molto spesso. Per non far sentire alla nonna la loro mancanza, Francesco e Chiara hanno realizzato delle pagine

contenenti le loro fotografie e di tanto in tanto le aggiornano ed effettuano una migrazione su TV digitale. In tal modo Luisa, nei pomeriggi di solitudine, si reca nel salone della casa di riposo dove si trovano le TV e può osservare i propri nipoti leggere i pensieri che hanno scritto per lei.

Scenario 8 - Supporto ai pazienti di una clinica
Francesco è un giocatore di calcio che a causa di uno scontro ha subito una frattura del piede. Egli si è dovuto recare in una clinica privata di Roma per sottoporsi ad un intervento. Per ogni aggiornamento sul database dello stato di salute di Francesco viene generata una versione visualizzabile su TV digitale. In tal modo i familiari di Francesco possono monitorare dalla TV di casa lo stato del figlio, anche dopo il suo rientro. Infatti sono in grado di ottenere informazioni come i medicinali da comprare, la cura da effettuare, la date delle visite future all'intervento. Inoltre il medico che visita Francesco può controllare la sua cartella clinica direttamente dalla TV della stanza dove si trova ricoverato, senza il bisogno di avere con sé le cartelle di tutti i pazienti che visita in giornata.

Scenario 9 - Prenotazione on-line
Piero è un idraulico di Mazara del Vallo, lavora 12 ore al giorno e torna ogni sera a casa alle 21. Egli ha sentito dire che su certi siti si trovano voli scontati per molte città europee e allora telefona al suo amico Marco chiedendo spiegazioni a riguardo, considerando che ha poca dimestichezza con internet. Allora gli chiede di potergli prenotare un volo per Londra per il fine settimana successivo per lui e la sua ragazza, alla quale vuole fare una sorpresa. Marco allora effettua la ricerca, trova il volo e inizia la prenotazione fino a quando deve inserire i dati riguardanti la carta di credito per effettuare il pagamento. Marco non ha questi dati e Piero, per questioni di sicurezza, preferisce non comunicarglieli telefonicamente. Quindi Marco effettua una migrazione sulla TV digitale di Piero in modo da potergli far terminare la transazione con i dati corretti.

Scenario 10 - Censimento
In occasione del prossimo censimento si è deciso utilizzare la piattaforma televisiva interattiva per effettuare le interviste, visto che per quella data sarà obbligatorio l'uso del decoder digitale terrestre e che la stragrande maggioranza della popolazione possiede la TV.

Scenario 11 - Informazioni scolastiche sul figlio
Il Preside dell'ITC di Pisa ha deciso di istituire degli incontri tempestivi con i genitori per tenere maggiormente sotto controllo gli studenti ed evitare sorprese ai genitori. Maria, parrucchiera di Pisa, si trova in una difficile situazione perché dovrebbe lasciare diverse volte il lavoro per andare a scuola. Fortunatamente il preside della scuola ha deciso di utilizzare internet per la diffusione delle informazioni sugli studenti o, in alternativa, la TV digitale per quelle famiglie che non hanno la disponibilità di una connessione. Maria ha quindi richiesto il servizio di migrazione su TV digitale e può visualizzare tali informazioni (presenze, voti, ecc...) anche quando si trova a lavoro.

5.7 Esempi

In questa sezione verrano analizzati dei possibili scenari.

5.7.1 Esempio 1

Consideriamo il seguente scenario: Luca si trova sull'autobus e sta tornando a casa. In genere non ha tempo per fare la spesa e quindi gli capita molto spesso di effettuare la spesa online. Visto che manca ancora qualche minuto prima di arrivare a casa, decide di utilizzare il suo PDA ed accedere al sito internet che gli permette di poter effettuare la spesa (fig. 5.4).

Figura 5.4: Accesso al sito per effettuare la spesa

Luca quindi comincia a ricercare i prodotti che vuole acquistare e li aggiunge al carrello (fig. 5.5).
Nel frattempo l'autobus è quasi arrivato davanti casa e allora Luca decide di migrare sulla TV digitale (fig 5.6) di casa sua in modo da poter continuare la spesa successivamente, seduto comodamente sul suo divano.

Figura 5.5: Aggiunta dei prodotto al carrello

Figura 5.6: Applicazione client per la migrazione

Luca quindi scende dall'autobus, entra in casa e si mette comodo sul divano, davanti la TV, e continua il lavoro che aveva interrotto precedentemente (fig. 5.7).

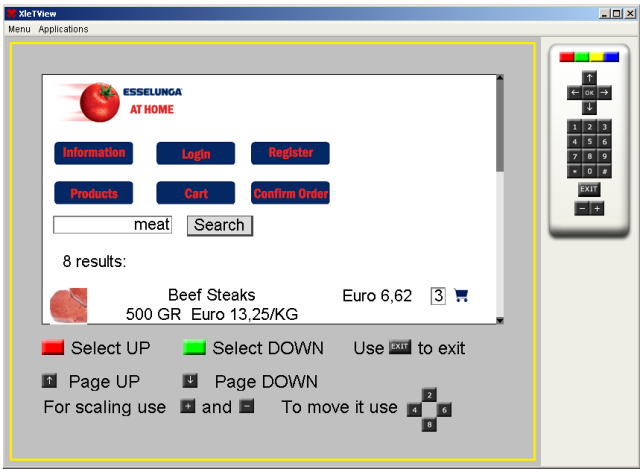

Figura 5.7: Interfaccia utente dopo la migrazione

Luca quindi continua ad effettuare gli acquisti (fig. 5.8, fig. 5.9).

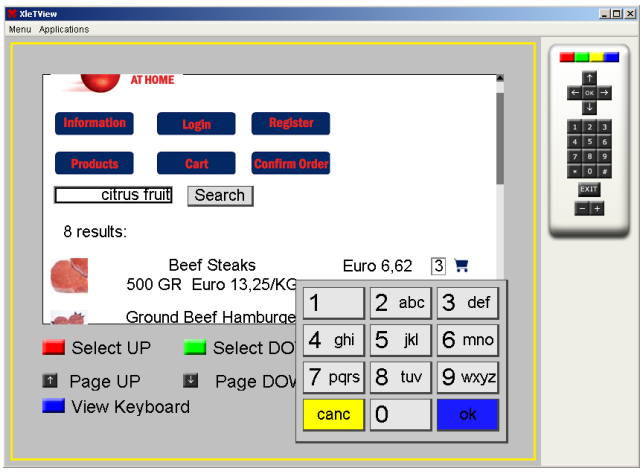

Figura 5.8: Ricerca di nuovi prodotti

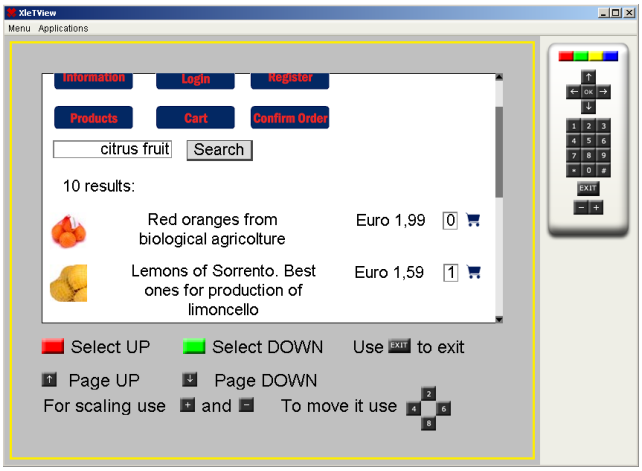

Figura 5.9: Analisi dei risultati e aggiunta di altri prodotti al carrello

Infine, dopo aver aggiunto al carrello tutti i prodotti di cui ha bisogno, conclude l'acquisto (fig. 5.10, 5.11, 5.12).

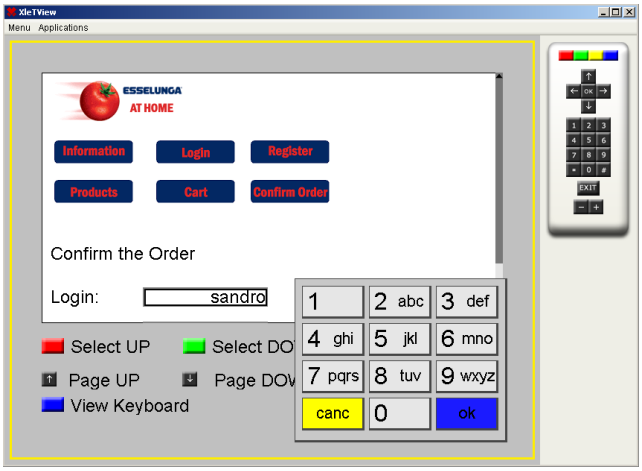

Figura 5.10: Immissione di login e password per effettuare il pagamento

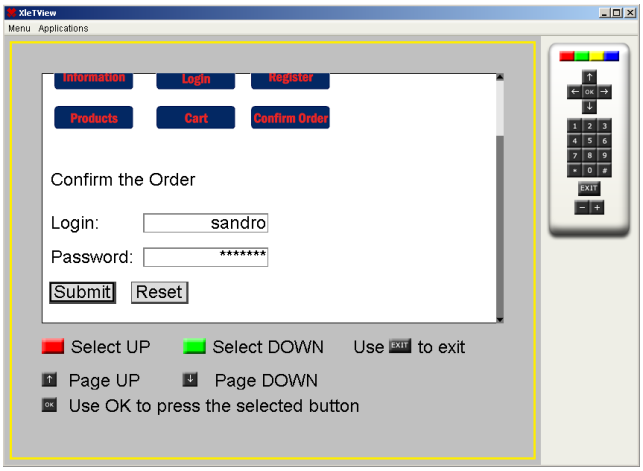

Figura 5.11: Sottomissione dei dati

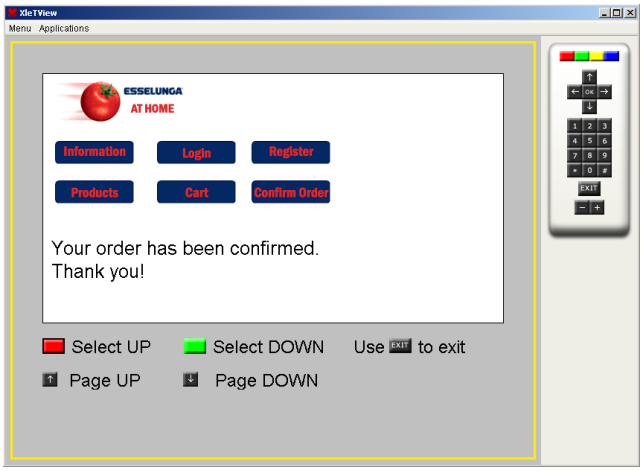

Figura 5.12: Conferma dell'avvenuto acquisto

Dopo aver terminato l'acuisto, Luca decide di uscire dall'applicazione con l'apposito comando EXIT (fig. 5.13).

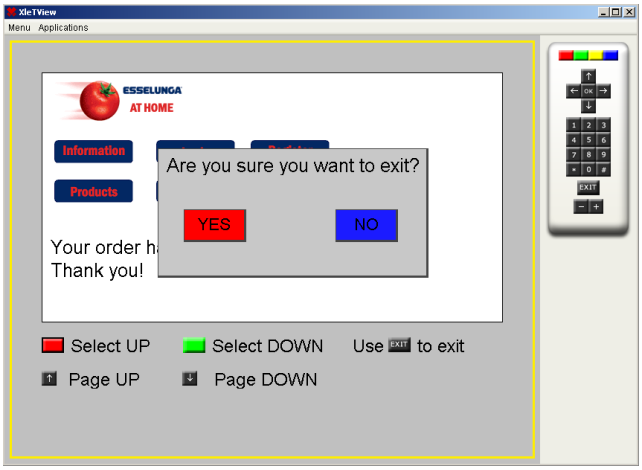

Figura 5.13: Uscita dall'applicazione

5.7.2 Esempio 2

Consideriamo un altro possibile scenario: prenotazione on-line di un ristorante. L'utente sta utilizzando il suo PC, e ad un certo punto, durante la prenotazione, decide di voler continuare la prenotazione sulla sua TV digitale, comodamente seduto sul divano. In tal modo, tra una pubblicità e l'altra del suo film preferito potrà concludere la prenotazione del ristorante. Lo stato dell'interfaccia utente al momento della migrazione è quello mostrato in figura.

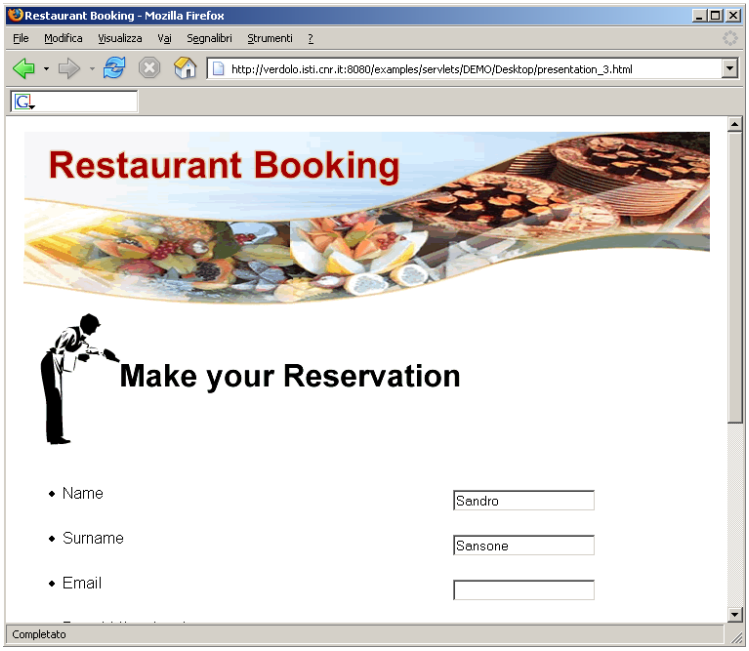

Figura 5.14: L'utente naviga utilizzando un browser X/HTML

A questo punto l'utente può effettuare la migrazione (utilizzando il client per la migrazione). Al momento della migrazione, come detto nella sezione precedente, il migration/proxy server effettua il reverse engineering della pagina web, recupera lo stato dal DOM della pagina e genera l'interfaccia utente per la piattaforma TV digitale.

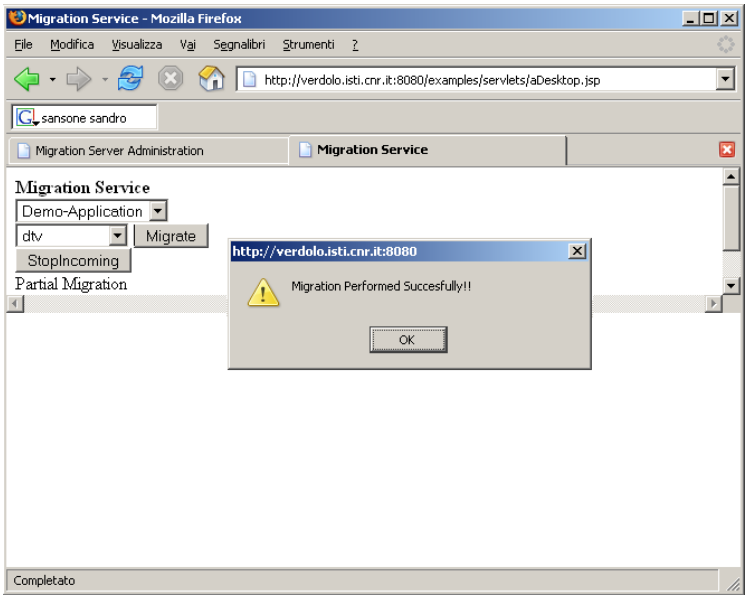

Figura 5.15: L'utente decide di migrare facendo richiesta al Migration Server

Quindi la nuova pagina generata può essere eseguita sulla nuova piattaforma.

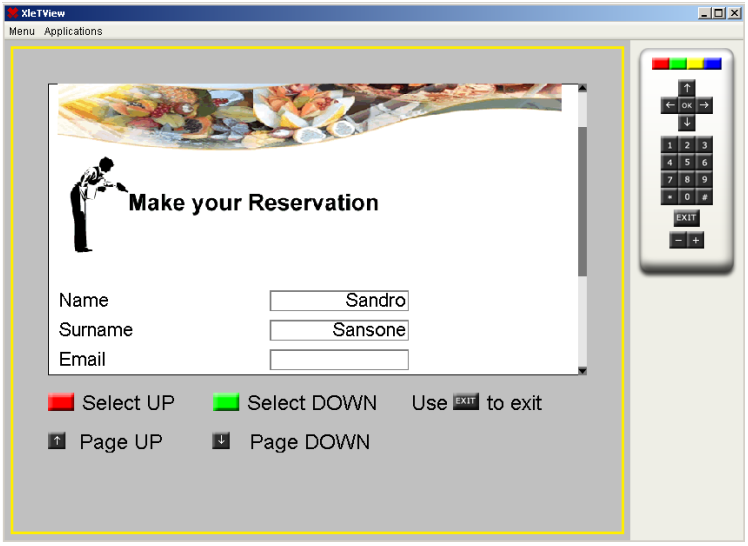

Figura 5.16: L'utente continua il suo lavoro sulla nuova piattaforma

Capitolo 6

Conclusioni

6.1 Conclusioni ed estensioni future

Il lavoro svolto in questa tesi ha portato alla realizzazione di un editor per interfacce utenti per TV digitale e di un ambiente per la migrazione a *run-time* di applicazioni web da varie piattaforme (quali Desktop o PDA) verso la piattaforma televisiva digitale.

Un primo approccio è servito per acquisire le conoscenze adatte, relative alla piattaforma televisiva. È stato effettuato uno studio di fattibilità che ci ha permesso di conoscere gli strumenti che avevamo a disposizione. In particolare è stata studiata la piattaforma MHP con le problematiche relative allo sviluppo di applicazioni per tale piattaforma.

In seconda battuta si è deciso di analizzare a fondo le caratteristiche di questo nuovo mezzo di comunicazione, per costruire interfacce utenti che fossero davvero usabili. Si è tenuto conto delle peculiarità della TV digitale in relazione alla gestione dell'input, alla visualizzazione dei contenuti e all'interazione con l'utente. Questo studio è stato fondamentale perchè, fino ad oggi, la TV è stata utilizzata solamente in modo passivo. Con questa nuova rivoluzione, l'utente partecipa attivamente, interagendo con essa. È quindi necessario che l'applicazione progettata sia intuitiva e semplice. Altri tipi di studi, relativi ai contenuti, alla disposizione degli oggetti all'interno della pagina e alla

semplicità, sono stati portati avanti per un secondo motivo, ma non meno importante: il target di utenti. La TV digitale eredita tutti gli utenti della TV analogica e quindi anche persone non informatizzate.

Successivamente il lavoro si è concentrato sull'editor per interfacce utenti di TERESA (TERESA UI Editor). Tale editor è stato esteso in modo che si potessero progettare interfacce utenti per TV digitale. È possibile salvare le interfacce utenti realizzate utilizzando le descrizioni logiche (un il formato XML: in particolare si tratta di una particolare astrazione, il TERESA XML Language). Dall'interfaccia progettata è possibile generare l'interfaccia utente finale, eseguibile sui Set-Top-Box. L'interfaccia utente finale corrisponde ad una applicazione Java. Tali applicazioni vengono chiamate Xlet e per certi aspetti sono simili alle applet, solo che quest'ultime vengono eseguite sui browser. Per testare effettivamente il reale funzionamento delle applicazioni generate, è stato utilizzato un emulatore per TV digitale, XletView.

Infine, la tesi si è conclusa integrando il processo di generazione delle Xlet in un ambiente per il supporto alla migrazione. Tale ambiente si basa su un migration/proxy server che riceve richieste di migrazione ed effettua a *runtime* l'adattamento dell'interfaccia per il dispositivo sul quale è stato deciso di migrare. Ad esempio, se un utente sta navigando utilizzando un dispositivo mobile, come un PDA, ad un certo punto può decidere di migrare verso la TV digitale continuando il suo lavoro su tale dispositivo. A partire dall'interfaccia sorgente viene generata quella per la TV digitale e il contesto relativo a quell'utente, come ad esempio dei dati immessi in dei campi di editing, vengono mantenuti nella nuova interfaccia.

La mancanza di mezzi non ci ha permesso di poter testare questo sistema in un ambiente reale, ma in un ambiente emulato.
Non escludo che questo possa aver contribuito a creare delle imperfezioni nelle applicazioni generate. Il miglioramento, la correzione e l'arricchimento del lavoro da noi svolto potrebbe quindi essere argomento di ulteriori studi, utilizzando un ambiente reale per trarne tutti i benefici che ne derivano e mettere in risalto la nostra attività.

Appendice A

TERESA XML Language

A.1 Abstract User Interface

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT interface (presentation+)>

<!ELEMENT presentation (connection*,
                        (interactor | interactor_composition))>
<!ATTLIST presentation
          name ID #REQUIRED>

<!ELEMENT connection (conn_type)>
<!ATTLIST connection
          presentation_name IDREF #REQUIRED>

<!ELEMENT conn_type (elementary_conn | complex_conn)>

<!ELEMENT elementary_conn EMPTY>
<!ATTLIST elementary_conn interactor_id IDREF #REQUIRED>

<!ELEMENT complex_conn (bool_operator, conn_type+)>
```

```
<!ELEMENT bool_operator EMPTY>
<!ATTLIST bool_operator name (and | or) #REQUIRED >

<!ELEMENT interactor_composition (operator,
                                  first_expression+,
                                  second_expression?)>

<!ELEMENT operator EMPTY>
<!ATTLIST operator
           name (grouping | ordering |
                 relation | hierarchy) #REQUIRED >

<!ELEMENT first_expression (interactor | interactor_composition)>

<!ELEMENT second_expression (interactor | interactor_composition)>

<!ELEMENT interactor (interaction | only_output)>
<!ATTLIST interactor
          id ID #REQUIRED>

<!ELEMENT  interaction (selection  | edit | control )>
<!ATTLIST  interaction
           category CDATA #FIXED "interaction">

<!ELEMENT selection (single_choice | multiple_choice)>
<!ATTLIST selection
          type CDATA #FIXED "selection">

<!ELEMENT single_choice (singlechoice_low_card |
                         singlechoice_medium_card |
```

```
                          singlechoice_high_card)>
<!ATTLIST single_choice
          type CDATA #FIXED "single choice">

<!ELEMENT singlechoice_low_card EMPTY>
<!ATTLIST singlechoice_low_card
          type CDATA #FIXED "low card">

<!ELEMENT singlechoice_medium_card EMPTY>
<!ATTLIST singlechoice_medium_card
   type CDATA #FIXED "medium card">

<!ELEMENT singlechoice_high_card EMPTY>
<!ATTLIST singlechoice_high_card
          type CDATA #FIXED "high card">

<!ELEMENT multiple_choice (multiplechoice_low_card |
                           multiplechoice_medium_card |
                           multiplechoice_high_card)>
<!ATTLIST multiple_choice
          type CDATA #FIXED "multiple choice">

<!ELEMENT multiplechoice_low_card EMPTY>
<!ATTLIST multiplechoice_low_card
          type CDATA #FIXED "low card">

<!ELEMENT multiplechoice_medium_card EMPTY >
<!ATTLIST multiplechoice_medium_card
          type CDATA #FIXED "medium card">

<!ELEMENT multiplechoice_high_card EMPTY>
```

```
<!ATTLIST multiplechoice_high_card
          type CDATA #FIXED "high card">

<!ELEMENT edit (text_edit | object_edit |
                numerical_edit | position_edit)>
<!ATTLIST edit
          type CDATA #FIXED "edit">

<!ELEMENT text_edit EMPTY>
<!ATTLIST text_edit
          object CDATA #FIXED "alphanumeric">

<!ELEMENT object_edit EMPTY>
<!ATTLIST object_edit
          object CDATA #FIXED "object">

<!ELEMENT numerical_edit EMPTY>
<!ATTLIST numerical_edit
          object CDATA #FIXED "quantity">

<!ELEMENT position_edit EMPTY>
<!ATTLIST position_edit
          object CDATA #FIXED   "position">

<!ELEMENT control (navigator | activator)>
<!ATTLIST control
          type CDATA #FIXED "control">

<!ELEMENT navigator EMPTY>
<!ATTLIST navigator
          object CDATA #FIXED "navigator">
```

```
<!ELEMENT activator EMPTY>
<!ATTLIST activator
          object CDATA #FIXED "activator">

<!ELEMENT only_output (text | object | description | feedback)>
<!ATTLIST only_output
          category CDATA #FIXED "onlyoutput">

<!ELEMENT text EMPTY>
<!ATTLIST text
          object CDATA #FIXED "text">

<!ELEMENT object EMPTY>
<!ATTLIST object
          object CDATA #FIXED "object">

<!ELEMENT description EMPTY>
<!ATTLIST description
          object CDATA #FIXED "description">

<!ELEMENT feedback EMPTY>
<!ATTLIST feedback
          object CDATA #FIXED "feedback">
```

A.2 Concrete User Interface

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<!-- ENTITIES DECLARATION - - - - - - - - - - - - - - - - - - - -->

<!ENTITY % length_value "8 | 9 | 10 | 11 | 12 |13 | 14 | 15 |
                         16 | 17 | 18 |19 | 20">

<!ENTITY % font_value "Tiresias">

<!ENTITY % font_size "24_pt | 26_pt | 28_pt |
                      31_pt | 34_pt | 36_pt">

<!ENTITY % font_align "Center | Left | Right">

<!ENTITY % bool_op "and | or">

<!ENTITY % cardinality_value "low_card | medium_card |
                              high_card">

<!ENTITY % position "column | row">

<!ENTITY % element_selection_alignment "horizontal |
                                        vertical">

<!ENTITY % option "yes | no">


<!-- DATA DECLARATION - - - - - - - - - - - - - - - - - - - - - -->

<!ELEMENT concrete_dtv_interface (default_settings,
                                  presentation+)>

<!ELEMENT default_settings (background, font_settings,
```

```
                        operators_settings,
                        interactors_settings)>

<!ELEMENT background (background_color | background_image)>

<!ELEMENT background_color EMPTY>
<!ATTLIST background_color
          value CDATA #REQUIRED>

<!ELEMENT background_image EMPTY>
<!ATTLIST background_image
          src CDATA #REQUIRED>

<!ELEMENT font_settings (font, color, size, align)>

<!ELEMENT font EMPTY>
<!ATTLIST font
          font (%font_value;) #REQUIRED>

<!ELEMENT color EMPTY>
<!ATTLIST color
          color CDATA #REQUIRED>

<!ELEMENT size EMPTY>
<!ATTLIST size
          size (%font_size;) #REQUIRED>

<!ELEMENT align EMPTY>
<!ATTLIST align
          align (%font_align;) #REQUIRED>
```

```
<!ELEMENT operators_settings (grouping, ordering,
                              hierarchy, relation)>

<!ELEMENT interactors_settings (control, description,
                                text_edit, numerical_edit)>

<!ELEMENT presentation (presentation_properties, connection*,
                        (interactor | interactor_composition))>
<!ATTLIST presentation
          name ID #REQUIRED>

<!ELEMENT presentation_properties (title, background,
                                   font_settings, top)>

<!ELEMENT title EMPTY>
<!ATTLIST title
          value CDATA #REQUIRED>

<!ELEMENT top (input_text | image)>

<!ELEMENT input_text EMPTY>
<!ATTLIST input_text
          value CDATA #REQUIRED>

<!ELEMENT image EMPTY>
<!ATTLIST image
          alt CDATA #REQUIRED
          src CDATA #REQUIRED>

<!ELEMENT connection (conn_type)>
<!ATTLIST connection
```

```
          presentation_name IDREF #REQUIRED>

<!ELEMENT conn_type (elementary_conn | complex_conn)>

<!ELEMENT elementary_conn EMPTY>
<!ATTLIST elementary_conn
          interactor_id IDREF #REQUIRED>

<!ELEMENT complex_conn (bool_operator, conn_type+)>

<!ELEMENT bool_operator EMPTY>
<!ATTLIST bool_operator
          name (%bool_op;)  #REQUIRED >

<!ELEMENT interactor (interaction | only_output)>
<!ATTLIST interactor
          id ID #REQUIRED>

<!ELEMENT interaction (selection  | editing | control |
                       interactive_description)>

<!ELEMENT selection (single | multiple)>

<!ELEMENT single (radio_button)>
<!ATTLIST single
          cardinality (%cardinality_value;) #REQUIRED>

<!ELEMENT radio_button (choice_element+)>
<!ATTLIST radio_button
          label CDATA #REQUIRED
          alignment (%element_selection_alignment;) #REQUIRED>
```

```
<!ELEMENT choice_element EMPTY>
<!ATTLIST choice_element
          label CDATA #REQUIRED
          value CDATA #REQUIRED>

<!ELEMENT multiple (check_box)>
<!ATTLIST multiple
          cardinality (%cardinality_value;) #REQUIRED>

<!ELEMENT check_box (choice_element+)>
<!ATTLIST check_box
          label CDATA #REQUIRED>

<!ELEMENT editing (text_edit | object_edit |
                   numerical_edit | position_edit)>

<!ELEMENT text_edit (textfield)>

<!ELEMENT textfield EMPTY>
<!ATTLIST textfield
          label CDATA #REQUIRED
          length (%length_value;) #REQUIRED
          password (%option;) #REQUIRED>

<!ELEMENT object_edit EMPTY>

<!ELEMENT numerical_edit (textfield)>

<!ELEMENT position_edit EMPTY>
```

```
<!ELEMENT control (navigator | activator)>

<!ELEMENT navigator (text_link | image_link | button)>
<!ATTLIST navigator
          target CDATA #IMPLIED>

<!ELEMENT text_link EMPTY>
<!ATTLIST text_link
          label CDATA #REQUIRED>

<!ELEMENT image_link EMPTY>
<!ATTLIST image_link
          src CDATA #REQUIRED>

<!ELEMENT button EMPTY>
<!ATTLIST button
          label CDATA #REQUIRED>

<!ELEMENT activator (reset_button | mailto)>

<!ELEMENT reset_button EMPTY>
<!ATTLIST reset_button
          label CDATA #REQUIRED>

<!ELEMENT mailto EMPTY>
<!ATTLIST mailto
          address CDATA #REQUIRED>

<!ELEMENT interactive_description
               (textual | navigator | activator)+>
```

```
<!ELEMENT only_output (textual | object |
                       description | feedback)>

<!ELEMENT textual (text)>

<!ELEMENT text ((input_text | text_file), font_settings?)>

<!ELEMENT text_file EMPTY>
<!ATTLIST text_file
          src CDATA #REQUIRED>

<!ELEMENT object (image)>

<!ELEMENT description ((textual?, image?) | (image?, textual?))>

<!ELEMENT feedback EMPTY>

<!ELEMENT interactor_composition (operator, first_expression+,
                                  second_expression?)>

<!ELEMENT operator (grouping | ordering | hierarchy | relation)>
<!ATTLIST operator
          id ID #REQUIRED>

<!ELEMENT grouping (fieldset?, bullet?,
                    background_color?, position)>

<!ELEMENT fieldset EMPTY>

<!ELEMENT bullet EMPTY>
```

```
<!ELEMENT position EMPTY>
<!ATTLIST position
          value (%position;) #REQUIRED>

<!ELEMENT ordering ((ordered_list | bullet)?, position)>

<!ELEMENT ordered_list EMPTY>

<!ELEMENT hierarchy (bigger_font)>

<!ELEMENT bigger_font EMPTY>

<!ELEMENT relation (form)>

<!ELEMENT form EMPTY>

<!ELEMENT first_expression (interactor | interactor_composition)>

<!ELEMENT second_expression (interactor | interactor_composition)>
```

Bibliografia

[1] G. Mori, F. Paternò, C. Santoro: Design and Development of Multidevice User Interfaces through Multiple Logical Description, *IEEE Transactions on Software Engineering*, August 2004, Vol 30, No 8, IEEE Press, pp.507-520

[2] R. Bandelloni, G. Mori, F. Paternò: Dynamic Generation of Migratory Interfaces, *Mobile HCI 2005*, ACM Press, pp.83-90, Salzburg, September 2005

[3] R. Bandelloni, G. Mori, F. Paternò: Automatic User Interface Generation and Migration in Multi-Device Web Environments, *Technical Report*

[4] S. Berti, F. Correani, F. Paternò, C. Santoro: The TERESA XML Language for the Description of Interactive Systems at Multiple Abstraction Levels, *Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages*, pp.103-110, May 2004

[5] G. Mori, F. Paternò: Automatic Semantic Platform-dependent Redesign, *Smart Objects and Ambient Intelligence 2005*, pp.177-182, Grenoble, October 2005,

[6] S. Berti, F. Paternò, C. Santoro: A Taxonomy for Migratory User Interfaces, *DSV-IS 2005*, Lecture Notes Computer Science, Springer Verlag, Newcastle, July 2005

[7] S. Berti, F. Paternò, C. Santoro: Concepts, Dimensions and Scenarios, *Technical Report*

[8] Steve Morris and Anthony Smith-Chaignea: Interactive TV standards: A Guide to MHP, OCAP, and JavaTV, Focal Press, April 2005

[9] R. Bernhaupt, B. Ploderer, M. Tscheligi: Relevance of Prior Experience in MHP based Interactive TV Services, *INTERACT 2005*, pp.1150-1153

[10] J. Jones: DVB-MHP/Java TV Data Transport Mechanisms, *IFortieth International Conference on Tools Pacific: Objects for internet, mobile and embedded applications*, Volume 10 CRPITS '02, February 2002

[11] P. Perrot: DVB-HTML An Optional declarative language within MHP 1.1,
http://www.mhp.org/documents/mhp_perrot-dvb-html.pdf

[12] F. Formica: DVBT La nuova tecnologia digitale del Broadcasting,
http://web.unife.it/progetti/swimm/tiki-download_file.php?fileId=5

[13] L. Pallara, A. Venturi: Televisione Digitale Terrestre: la televisione diventa interattiva,
http://www.cineca.org/csd/files/PDF/DTT.pdf

[14] ETSI: TS 201 812 V1.1.1: *Digital Video Broadcasting,* Multimedia Home Platform Specification 1.0.3, http://www.etsi.org

[15] Fondazione Ugo Bordoni: Raccomandazioni per le interfacce dei servizi interattivi della televisione digitale

[16] Java Specification Request 242:
http://www.jcp.org/en/jsr/detail?id=242

[17] John Piesing:
Introduction to MHP 1.1.2, http://www.mhp.org/mhp_technology

[18] Steve Morris: Mhp vs. Java, The Differences,
http://www.interactivetvweb.org/resources/presentations/mhp-javadifferences.ppt

[19] E. Brunelli: *Televisione Digitale Interattiva*,
http://www.mokabyte.it/2004/05/jtdi-2.htm

[20] DVB: Digital Video Broadcasting, http://www.dvb.org

[21] MHP: Multimedia Home Platform, http://www.mhp.org

[22] ETSI: European Telecommunication Standards Istitute, http://www.etsi.org

[23] USABILE.IT: usabilità, accessibilità e interaction design per il web, http://www.usabile.it

[24] HAVi: Home Audio/Video Interoperability,
http://www.havi.org/

[25] Java TV: http://java.sun.com/products/javatv/

[26] Personal Basis Profile:
http://java.sun.com/products/personalbasis/

[27] VoiceXML: Voice Extensible Markup Language,
http://www.w3.org/TR/voicexml20/

[28] XletView: http://xletview.sourceforge.net/

[29] OpenMHP: http://www.openmhp.org

[30] Recensioni sui principali ambienti di authoring per piattaforma MHP: http://www.tvdigitaleterrestre.net/?p=19

[31] Icareus iTV suite: http://www.icareus.com/

[32] Pontrega HTML browser: http://www.nionex.com/

[33] My DTT: La piattaforma ideata da MyTV, http://www.my-tv.it/

[34] SetMedia: Suite per la gestione di contenuti digitali, http://www.interattiva.tv/

[35] The Interactive TV Web: http://www.interactivetvweb.org/

[36] OCAP: Open Cable Specification, http://www.opencable.com/ocap/

[37] JTIDY: Java port of HTML Tidy, http://jtidy.sourceforge.net/