Visualizzazione dei palinsesti televisivi

Codename: JTVGuide

Progetto di Informatica III

Prof. Angelo Gargantini

Michele BOLOGNA – michele.bologna@gmail.com – 56108

Sebastiano ROTA – sebastiano.rota@gmail.com - 57166

Sommario

Introduzione

Spesso vi è la necessità di conoscere il palinsesto di uno (o più) canali televisivi senza la *fatica* di dover accendere la televisione e consultare il televideo, o la pagina del giornale contenente "*i programmi di oggi*". Alcuni strumenti potrebbero essere quelli di utilizzare i siti Internet degli appositi canali (ad esempio: Il palinsesto di Rai Uno).

Ma si consideri il processo:

- apertura del browser
- clic sul bookmark o ricerca su Google del palinsesto del canale
- consultare l'orologio e scegliere tutti i programmi il cui orario d'inizio è maggiore o uguale all'orario corrente.

È chiaro che tale processo è un'attività noiosa e ripetitiva (consideriamo una persona che vuole conoscere ogni giorno i programmi televisivi della serata), e la perdita di tempo dovuta alla ricerca dell'informazione è improponibile (spesso i siti contengono banner, advertising a tutto schermo, etc.). Senza contare che tale processo va ripetuto per tutti i canali di cui si vuole conoscere il palinsesto. Ci si potrebbe attrezzare con siti che raccolgono i palinsesti di più canali (Il palinsesto di satellite.it è il miglior esempio): anche questi siti, però, sono caratterizzati da una lentezza di caricamento notevole (il palinsesto considerato utilizza addirittura la tecnologia Flash, che causa una lentezza considerevole nel caricamento e un'allocazione di memoria per il browser non indifferente), ricadendo quindi nel caso di uno spreco di tempo e risorse per ottenere una semplice informazione.

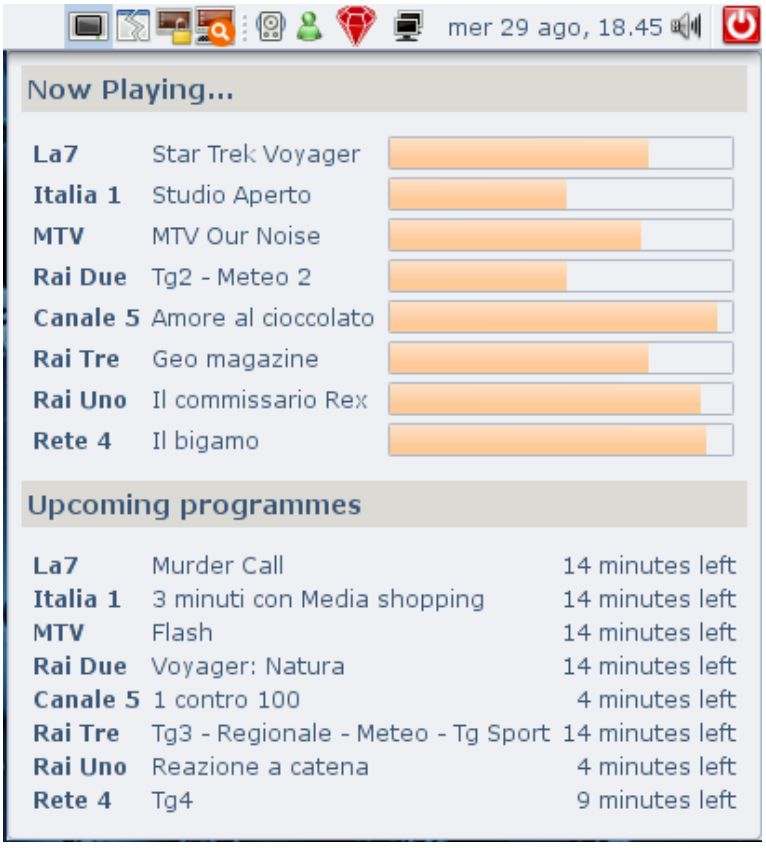
Frequentemente si sa che il proprio programma preferito sarà in onda in un determinato momento: basti pensare, ad esempio, alla frase "so che nei prossimi giorni ci sarà in onda il nuovo episodio di "Prison Break", ma non so su che canale né a che ora…": il processo di ricerca dell'informazione passa per la ricerca nei palinsesti dei canali del programma (in questo caso il film) desiderato. Come spiegato poco più sopra, tale processo è dispendioso per l'utente sia in termini di tempo che di risorse.

Infine, si ha poi un altro grande problema: spesso ci si dimentica che un programma che si vuole guardare sta per essere trasmesso in televisione; ad esempio: si pensi alla situazione in cui l'utente ha visionato il palinsesto futuro di un canale e ha deciso che il programma della serata è degno di nota. Se, durante la serata, l'utente *non si ricorda* di voler guardare tale programma, avrà perso definitivamente la trasmissione desiderata.

La soluzione del problema è *semplice*: visto che il processo è standard ed è sufficiente automatizzarlo, perché non avere un'applicazione che si occupi di fare il download dei palinsesti dei vari canali che interessano all'utente e in grado di mostrargli i programmi che sono in onda al momento? L'utente può anche ricercare dei programmi nel palinsesto, e selezionare un allarme per determinati programmi: in questo modo l'applicativo informa l'utente (tramite un reminder) che il programma sta per iniziare, fungendo da "agenda" dei programmi preferiti. In questo modo l'overhead di ricerca e selezione dell'informazione televisiva è commissionato all'applicazione, e l'utente può concentrarsi sull'attività "operativa" vera e propria: guardare i propri programmi di interesse.

Valutazione delle alternative

Attualmente esistono dei programmi che soddisfano alcune (ma non tutte) le necessità descritte nell'analisi dei requisiti: l'esempio più completo è sicuramente OnTV, un applet del desktop environment Gnome. Tuttavia, la maggior parte di questi programmi è sviluppata per i sistemi operativi UNIX-like.



Per Windows non esistono alternative gratuite: come si può leggere qui, http://www.levysoft.it/archivio/2007/09/13/guida-tv-online-per-windows-mac-e-linux-alla-ricerca-del-miglior-programma-per-visualizzare-i-palinsesti-delle-emittenti-televisive-fino-ad-approdare-al-progetto-open-source-per-gnome-ontv/:

"In realtà, però, io ero alla ricerca di qualcosa di integrato col mio sistema operativo e disponibile anche offline. Come accennato prima, per Windows non ho trovato nulla di facilmente fruibile e freeware anche se un programma ben fatto esiste e si chiama, ovviamente GuidaTV. Questo supporta **171 palinsesti, tra reti tv terrestri e satellitari e ed emittenti radiofoniche** e i palinsesti sono disponibili da sette a venti giorni antecedenti all'andata in onda delle trasmissioni. Il problema è che richiede un **abbonamento annuale di 10 euro**."

Ecco perché JTVGuide ha le potenzialità per diventare un prodotto di successo, essendo **gratuito** ed **open-source**.

XMLTV

Inoltre, sempre in ambiente UNIX-like, si nota che *tutti* i programmi che consento di visualizzare il palinsesto si appoggiano ad un'applicativo esterno, XMLTV. XMLTV è un programma a linea di comando che consente di disaccoppiare il problema di download dei palinsesti (backend) da quello della visualizzazione (frontend); infatti, la libreria è composta da una serie di grabber (esiste un grabber di palinsesti per ogni nazione, tra cui l'Italia). I grabber rispondono ad un file di configurazione che consente di selezionare i canali

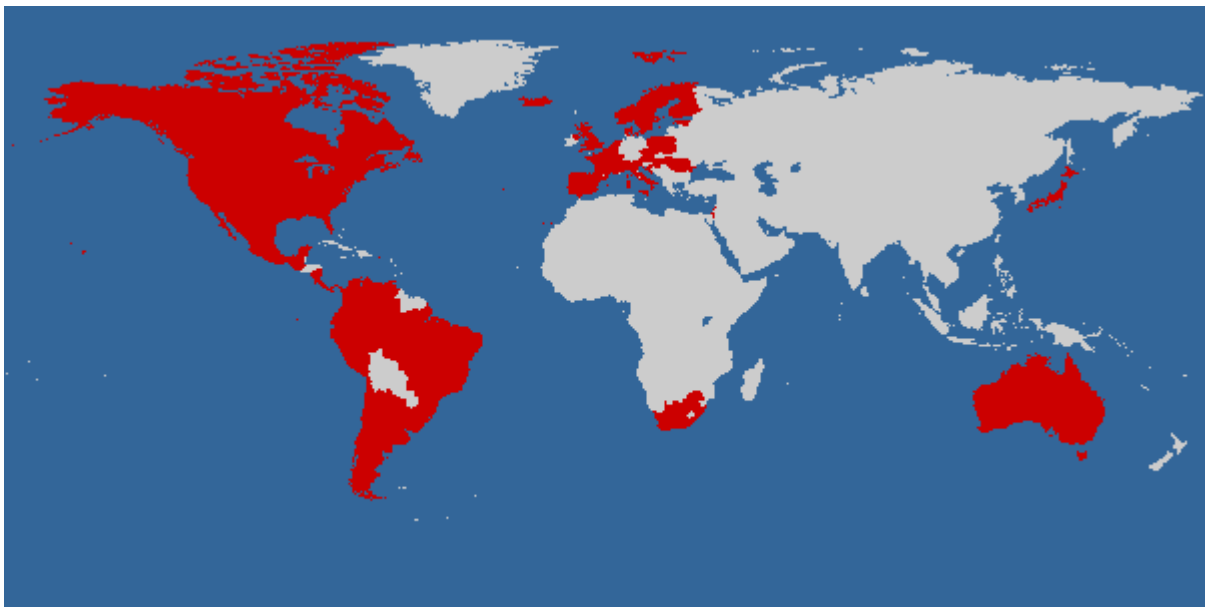
di interesse. Avendo selezionato i canali di interesse dal file di configurazione, si può lanciare il grabber per la specifica nazione: il grabber si occuperà del download dei palinsesti dei canali selezionati e produrrà un file XML con la descrizione dei palinsesti dei canali scelti.

Si è dovuto scegliere tra due alternative:

- Scrivere (da zero) un download-parser del palinsesto per ogni specifico canale e integrarlo nel programma. Nessuna separazione tra backend e frontend. Lo sforzo per lo sviluppo del download parser di un solo canale non è indifferente, e quindi il numero di canali televisivi integrabili nel programma sarebbe stato molto ridotto.
- Utilizzare XMLTV come programma esterno: il progetto costituisce il frontend di visualizzazione (parsing del file XML prodotto da XMLTV) dei programmi e integra un wrapper per utilizzare XMLTV. La lista dei canali è numerosa e aggiornata, l'algoritmo di download e di parsing è il migliore attualmente presente sul mercato. In questo modo si riusa del codice; infine, XMLTV è open-source.

Si è deciso di utilizzare l'idea di disaccoppiamento suggerita da XMLTV, sviluppando un'applicazione che contenesse un wrapper per il grabber di XMLTV. Il grabber è generico, in quanto XMLTV raccoglie un grabber per ogni nazione. In questo modo, l'applicazione sviluppata può appoggiarsi ad un grabber non italiano ed essere utilizzata anche da utenti non italiani! Infatti, è sufficiente che ci sia il grabber per la nazione selezionata: al momento esistono grabbers per: "Australia, Belgium and Luxembourg, Brazil, Argentina, Britain and Ireland, Croatia, Denmark, Estonia, Finland, France, Hungary and Romania, Iceland, Italy, Japan, Netherlands, North America, Norway, Portugal, Reunion Island (France), South Africa, Spain, Sweden, and Switzerland" – si veda XmltvWorldDomination).

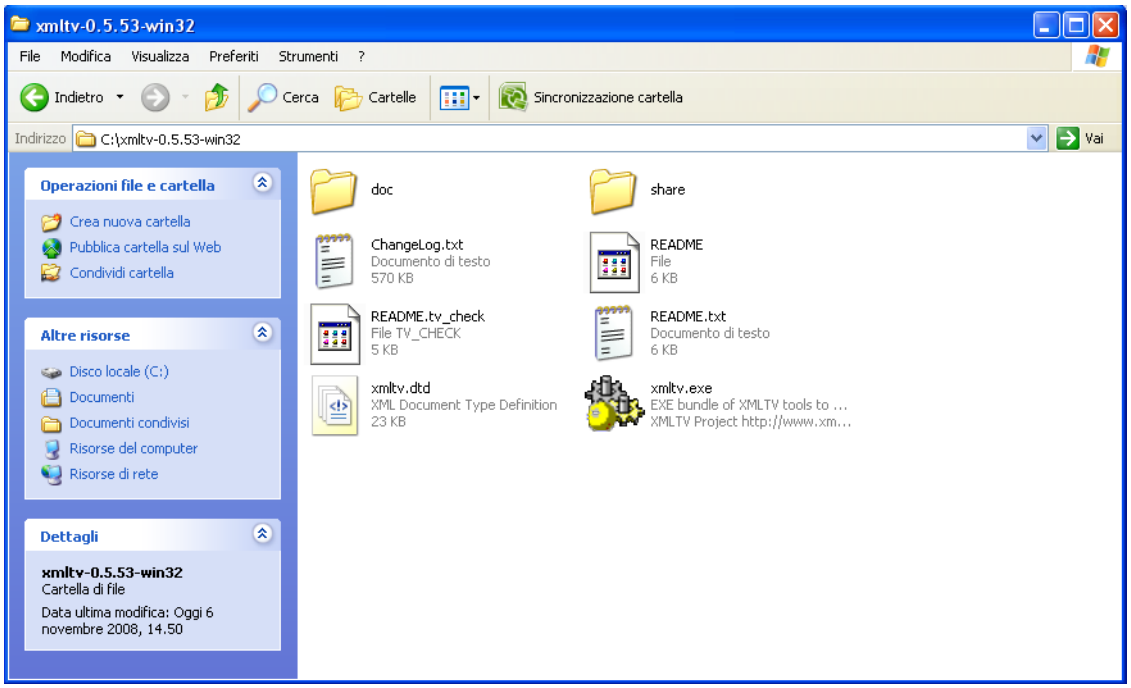
Nota sui tempi di XMLTV: il tempo di download di uno schedule è abbastanza variabile e dipende da molti fattori (velocità della linea, condizioni di traffico, numero di canali selezionati, etc.) ed è solitamente un'attività abbastanza onerosa: si consiglia di non selezionare troppi canali (>= 20) per non innescare un processo di download dello schedule che potrebbe durare parecchio tempo.

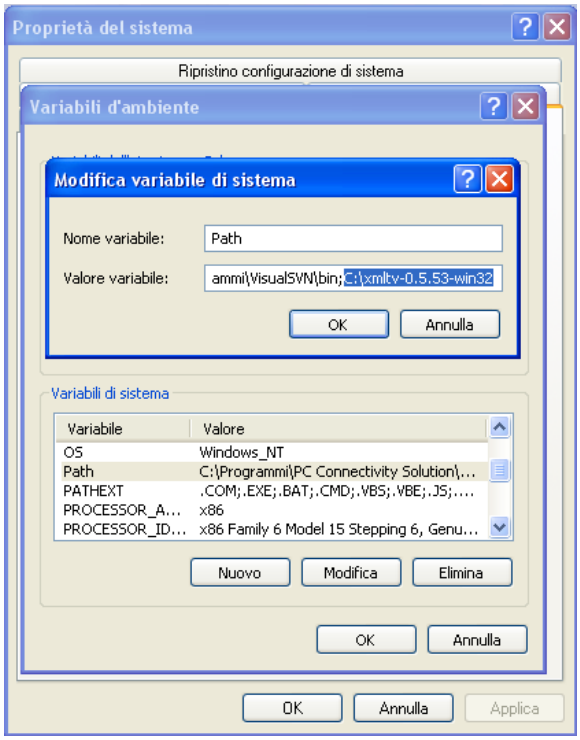


Installazione di XMLTV

Windows

Come già illustrato in precedenza XMLTV è un software in grado di effettuare il download dei palinsesti televisivi con le modalità desiderate dall'utente (canali di interesse e numero di giorni di cui si vuole ottenere la programmazione). L'applicativo può essere reperito da sourceforge.net, esso è infatti open source. Si possono trovare on-line tutte le revisioni ma è sempre consigliabile rimanere allineati all'ultima disponibile in quanto solitamente più aggiornata e veloce. Una volta in possesso dell'ultima versione disponibile del software occorre decomprimerla in una cartella apposita nel proprio file system (ad esempio: C:\xmltv-0.5.53-win32) all'interno della quale saranno presenti tutti i files necessari al funzionamento del software.





Modificare la variabile d'ambiente "Path" (start -> Pannello di controllo -> Sistema -> Avanzate -> Variabili d'ambiente -> Variabili di sistema) e aggiungere il path della cartella in cui precedentemente è stato decompresso l'archivio di XMLTV (nell'esempio C:\xmltv-0.5.53-win32).

Linux

Se xmltv è già pre-packaged nella propria distribuzione, è sufficiente installare il pacchetto corrispondente (tipicamente xmltv). Su Debian e Ubuntu (tutte le versioni stable/testing/unstable):

```
root@servertwo:~# apt-cache show xmltv

Package: xmltv

Section: universe/interpreters

Architecture: all

Version: 0.5.51-2ubuntu2

Depends: libxmltv-perl (= 0.5.51-2ubuntu2), xmltv-gui (= 0.5.51-
2ubuntu2), xmltv-util (= 0.5.51-2ubuntu2)

Filename: pool/universe/x/xmltv/xmltv_0.5.51-2ubuntu2_all.deb

Description: Functionality related to the XMLTV file format for TV li-
stings

 Gather television listings, process them and organize your viewing.

 XMLTV is a file format for storing TV listings, defined in xmltv.dtd.

 This is a meta-package that installs all of the XMLTV pieces.

This package is intended mainly for end-users who will run XMLTV

 programs directly.  Maintainers of other packages that rely on XMLTV

 functionality should consider depending on one or more of the related

 packages (libxmltv-perl, xmltv-util, xmltv-gui) as needed instead of

 this meta-package.
```

Di conseguenza l'installazione può essere effettuata semplicemente lanciando il comando:

```
apt-get install xmltv
```

Specifica dei requisiti funzionali

Si vuole realizzare un'applicazione che consenta di visualizzare il palinsesto di uno o più canali televisivi. In particolare, si vuole che l'applicazione sia dotata delle seguenti funzionalità:

- *Interfacciamento trasparente all'utente con il programma XMLTV:* l'utente attraverso l'applicativo comunica con il programma XMLTV che si occupa del download dei palinsesti desiderati.

- *Archiviazione dei canali, programmi e palinsesti di interesse:* l'applicativo si occupa di archiviare tutti i dati forniti dal programma XMLTV in opportune strutture dati al fine di riorganizzarli a seconda della preferenze indicate dall'utente in fase di visualizzazione.
- *Specifica dei canali e del numero di giorni dei quali si vuole conoscere il palinsesto:* è data la possibilità all'utente di poter decidere in ogni momento le opzioni che preferisce (indicazione dei canali di interesse e numero di giorni di cui vuole conoscere la programmazione televisiva).
- *Visualizzazione dei programmi in onda nel momento in cui viene interrogato:* l'utente è in grado di conoscere (per i canali di interesse da lui selezionati come indicato nel punto precedente) quali sono i programmi attualmente in onda e quelli che andranno in onda appena successivamente.
- *Visualizzazione dei programmi di ogni canale di interesse:* l'utente è in grado di consultare la programmazione di ogni canale di interesse da lui selezionato in modo indipendente dalla programmazione degli altri canali.
- *Visualizzare il palinsesto di tutti i canali di interesse in modo continuo:* l'utente può visualizzare la programmazione di tutti i suoi canali di interesse contemporaneamente e in modo continuo.
- *Ricerca, all'interno dei palinsesti, dei programmi:* è data la possibilità all'utente di poter ricercare all'interno dei palinsesti dei suoi canali di interesse, per i giorni di interesse, il/i programmi che desidera in modo da conoscere se e quando essi andranno in onda.
- *Promemoria per ricordare l'inizio dei programmi favoriti*

Specifica dei requisiti non funzionali

- *Affidabilità:* l'applicativo deve poter mantenere un buon livello prestazionale nel tempo. A tal scopo è stato utilizzato il programma XMLTV il cui algoritmo di download e parsing di palinsesti risulta essere attualmente il migliore sul mercato. Tale software è continuamente sottoposto a revisioni al fine di migliorarne le caratteristiche prestazionali.
- *Portabilità:* il programma deve essere multipiattaforma: si utilizza, a tal scopo, il linguaggio di programmazione Java. XMLTV è open source, e disponibile per le piattaforme UNIX-like e Windows
- *Usabilità:* il programma deve gestire XMLTV (tipicamente il grabber va lanciato dalla linea di comando) "nascondendo" all'utente la struttura backend-frontend: l'utente deve utilizzare l'interfaccia grafica solo per visualizzare e ricercare i palinsesti (relativi ai canali e ai giorni di suo interesse).
- *Efficienza:* utilizzando la versione CVS di XMLTV, contenente le ultime migliorie ai vari grabber, il tempo di download di 3 giorni di programmazione per 6 canali televisivi non supera i 3 minuti, per 5 giorni di programmazione e 10 canali televisivi il tempo necessario si aggira intorno ai 5 minuti.
- *Manutenibilità:* il programma deve essere sviluppato secondo l'architettura MVC (Model-View-Controller) e deve garantire successive evoluzioni nel tempo. Il core sarà indipendente dall'interfaccia grafica la quale farà uso delle primitive messe a disposizione dal cuore del sistema che potrà così essere mantenuto e aggiornato in modo indolore.

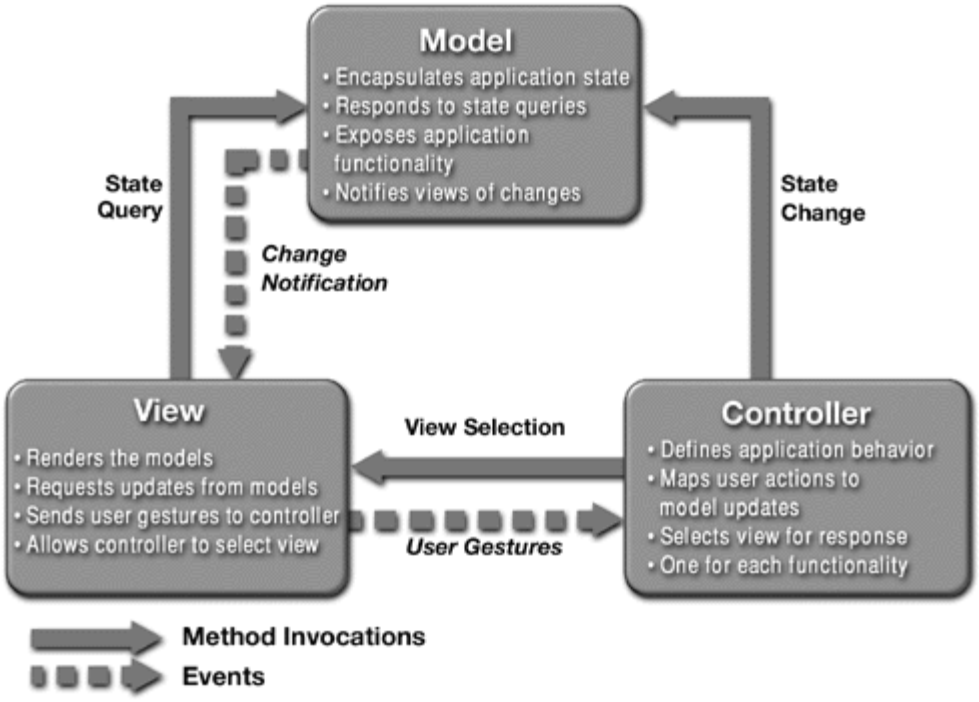
Pattern MVC

Il progetto è stato creato seguendo il modello Model View Controller, un design pattern molto famoso.

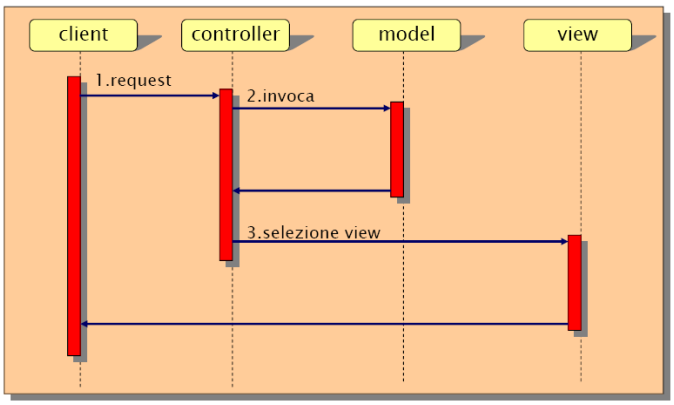
Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il model contiene i dati e fornisce i metodi per accedervi;
- il view visualizza i dati contenuti nel model;

- il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti



Questo schema, fra l'altro, implica anche la tradizionale separazione fra la logica applicativa (in questo contesto spesso chiamata logica di business, a carico del *model*, e l'interfaccia utente, a carico del *view* e del *controller*.



Strumenti utilizzati

Apache Ant

Apache Ant è un software per l'automazione del processo di build. È simile a make ma scritto in Java ed è principalmente orientato allo sviluppo Java. Ant è un progetto Apache, open source, ed è rilasciato sotto licenza Apache.

Subversion

Subversion (noto anche come svn, che è il nome del suo client a riga di comando) è un sistema di controllo versione progettato da CollabNet Inc. con lo scopo di essere il naturale successore di CVS, oramai considerato superato.

Log4j

Log4J è una libreria Java sviluppata dalla Apache Software Foundation che permette di mettere a punto un ottimo sistema di logging per tenere sotto controllo il comportamento di una applicazione, sia in fase di sviluppo che in fase di test e messa in opera del prodotto finale. L'ultima versione stabile disponibile è la 1.2.15.

Il modo migliore per configurare la libreria, ed utilizzarla in un'applicazione, è scrivere un file di properties. Il file può anche essere scritto in formato xml.

Il file di configurazione è costituito da due componenti principali: **Logger** e **Appender**.

Ciascun Logger viene associato ad un livello di log. I livelli disponibili, in ordine gerarchico, sono i seguenti: DEBUG, INFO, WARN, ERROR, FATAL.

Nell'esempio il Logger viene impostato con livello DEBUG e gli vengono associati due Appender: APPENDER_OUT e APPENDER_FILE. Ciascun Appender definisce un indirizzamento del flusso.

Log4j mette a disposizione diversi Appender. I più utilizzati sono i seguenti:

- **ConsoleAppender** che permette di scrivere sulla console dell'applicazione;
- **FileAppender** che permette di scrivere su file;
- **RollingFileAppender** che permette di scrivere su un file di testo definendone la lunghezza massima. Quando la lunghezza massima è raggiunta, il file è rinominato aggiungendo un numero progressivo al nome del file;
- **DailyRollingFileAppender** che permette di scrivere su un file di testo definendone un intervallo di tempo massimo. Quando il tempo scade, viene creato un altro file;
- **SocketAppender** che permette di scrivere su un socket utilizzando il protocollo TCP/IP;
- **JMSAppender** che permette di scrivere su una coda JMS;
- **SMTPAppender** che permette di inviare mail utilizzando il protocollo SMTP e JavaMail.

Ciascun Appender naturalmente ha bisogno di alcuni parametri di configurazione specifici. Ad esempio, il FileAppender ha bisogno della directory e del nome del file di log sul quale scrivere, mentre l'SMTPAppender ha bisogno dell'indirizzo del server SMTP.

A ciascun Appender è possibile associare un Layout mediante il quale è possibile specificare il modo in cui le informazioni devono essere formattate.

Log4J mette a disposizione diverse tipologie di Layout predefinite. Le principali sono le seguenti:

- **SimpleLayout** che produce stringhe di testo semplice;
- **PatternLayout** che produce stringhe di testo formattate secondo un pattern definito nel file di configurazione;
- **HTMLLayout** che produce un layout in formato HTML;
- **XMLLayout** che produce un layout in formato XML.

Se il layout non viene specificato, log4j utilizza il SimpleLayout.

Con il metodo statico `getLogger` della classe Logger otteniamo un'istanza della classe mediante la quale possiamo invocare uno dei metodi disponibili corrispondenti ai diversi livelli di log predefiniti.

In entrambi i casi log4j utilizza il pattern `Singleton` per cui, se in vari punti della applicazione si invoca un Logger con lo stesso nome, si otterrà sempre la stessa istanza.

Ad esempio scrivendo:

```
Logger log1 = Logger.getLogger("myLog");
Logger log2 = Logger.getLogger("myLog");
```

otteniamo due variabili che puntano alla stessa istanza.

Se eseguiamo l'esempio notiamo che sia sulla console, sia sul file di log, vengono stampate le 5 voci perché il livello definito nel file di configurazione è DEBUG che si trova al gradino più basso della gerarchia. Se impostiamo il livello ad ERROR, ad esempio, noteremo come vengano stampati esclusivamente i messaggi di ERROR e FATAL.

Potremo prevedere log di livello DEBUG per tenere traccia di alcune informazioni utili durante lo sviluppo, come il valore di particolari variabili, e log di livello ERROR per tenere traccia di informazioni utili quando l'applicazione è finita.

Nel momento in cui l'applicazione è pronta per essere utilizzata basta soltanto modificare il livello di log (sostituire DEBUG con ERROR) nel file di configurazione e tutto continua a funzionare senza dover modificare i sorgenti e ricompilarli.

JDOM

jdom e' una libreria java open source per la creazione e manipolazione di file xml molto potente e semplice da usare.
Per scaricare la libreria, la documentazione ufficiale e qualche esempio potete visitare il sito ufficiale di jdom, http://www.jdom.org.

JUNIT

JUnit è un unit test framework per il linguaggio di programmazione Java.

JUnit è stato creato da Kent Beck insieme ad Erich Gamma. Da allora ha ispirato ed è stato modello guida per altri unit test frameworks per altri linguaggi.

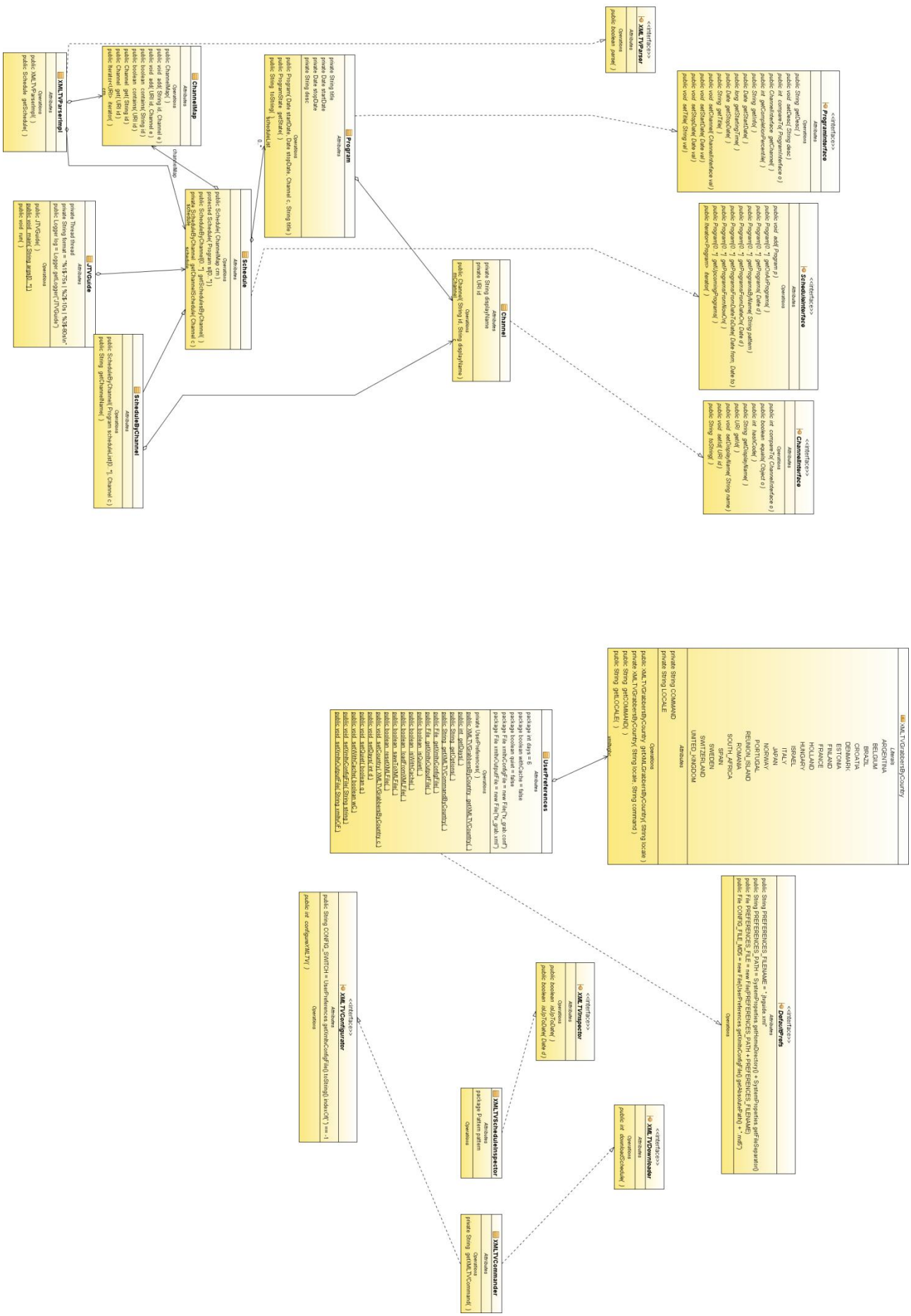
L'esperienza avuta con *JUnit* è stata importante nella crescita dell'idea di TDD, test driven development, e ha portato come risultato che alcune conoscenze di *JUnit* siano presenti nelle discussioni sullo sviluppo guidato da test.

JSmooth

JSmooth è un software opensource che permette di costruire un exe, file eseguibile su piattaforma Windows, in maniera semplice ed intuitiva, a partire da un file jar. Quando l'utente avvia il file eseguibile, l'exe creato scaricherà automaticamente la JRE se questa non è presente sul sistema.

Progettazione

Michele BOLOGNA – Sebastiano ROTA



Pattern MVC

Dividiamo il progetto JTVGuide in due progetti:

- JTVGuide contiene il "core" del programma. Contiene solo la business model e una classe main(), che fa funzionare il programma in modalità command line. Contiene quindi il MODEL
- JTVGuideUI, contiene il visualizzatore di programmi televisivi e la gestione dell'interfaccia grafica e la gestione dell'interazione con l'utente. Contienere quindi VIEW e CONTROLLER.

D'ora in poi ci soffermeremo su JTVGuide in quanto è la parte più critica del programma. Il visualizzatore non è una componente fondamentale del programma, e, quindi, non necessita di una spiegazione approfondita.

In questo modo abbiamo a disposizione un'applicativo (JTVGuide) che può funzionare sia come programma autonomo (usando il main()), sia come libreria (per JTVGuideUI).

MODEL: modelliamo la tipologia dei dati che abbiamo a disposizione. Possiamo avere:

- Channel: descrive il generico canale di trasmissione con nome, id, logo, etc.
- Program: descrive il generico programma televisivo, ed è caratterizzato da orari di inizio e di fine, un titolo e una descrizione. Inoltre, contiene un riferimento (navigable association → mChannel) al canale su cui è trasmesso
- Schedule: rappresenta il palinsesto televisivo contenente tutti i programmi. È costituito da un insieme di canali. Il palinsesto può essere aggiornato tramite XMLTV

Come si può capire se il palinsesto è aggiornato? Descriviamo le operazioni in sequenza:

1. (controllo esistenza) Il file XML contenente l'output di XMLTV esiste? Se esiste, salta al 2. Se non esiste salta al 3.
2. (controllo upToDate()) Il file XML contiene programmi il cui inizio è la data di oggi (= palinsesto aggiornato)? Se è aggiornato, salta al 4. Se non è aggiornato, salta al 3.
3. Richiama il programma esterno XMLTV con le opzioni specificate dall'utente (contenute in UserPreferences)
4. Fine

Come funziona XMLTV e la relativa gestione?

Innanzitutto, XMLTV è un grabber di palinsesti. Ad ogni nazione corrisponde un grabber (ad esempio: per l'Italia si richiama tv_grab_it, per il Giappone tv_grab_jp). Per associare ogni nazione al grabber corretto, usiamo una classe Enumeration (XMLTVGrabbersByCountry). Per operare con XMLTV, è necessario specificare:

- Un file di configurazione, contenente i canali di cui si vuole scaricare il palinsesto (ad esempio specifico che voglio il palinsesto di Canale 5 ma non quello di Italia 1). Come si produce tale file? XMLTV viene distribuito con un file di configurazione già pronto per ogni grabber. Tale file contiene la lista di tutti i canali di cui è possibile scaricare il palinsesto mediante il grabber selezionato. Su ogni riga

è presente un canale. Se la riga non è commentata, il grabber si preoccupa di scaricare il palinsesto di tale canale. La classe XMLTVCommander (implementation di XMLTVConfigurator) si preoccupa di scrivere (aiutando l'utente nella scelta) il file di configurazione.

- Un file XML di output, che conterrà il palinsesto scaricato da XMLTV
- La locazione (nel sistema) dell'eseguibile di XMLTV (su Linux, si deve chiamare direttamente il grabber, mentre su Windows si deve richiamare xmltv.exe seguito dal nome del grabber)
- Una serie di parametri (opzionali, ad esempio il numero di giorni di cui si vuole scaricare il palinsesto). Tali parametri sono contenuti nella classe UserPreferences

Per scaricare il palinsesto, usiamo la classe XMLTVCommander (implementation di XMLTVDownloader): richiama il comando usando il file di configurazione e scrive l'output sul file specificato, utilizzando le opzioni eventualmente specificate dall'utente.

Una volta ottenuto il palinsesto, si deve fare il parsing XML del file prodotto da XMLTV. Questo è il compito della classe XMLTVParser: fa il parsing del file XML e alloca le strutture dati descritte precedentemente: Channel, Program e Schedule.

Ritorniamo ora allo Schedule: quando un palinsesto non esiste oppure non è aggiornato (XMLTVScheduleInspector si occupa di determinare se uno schedule è aggiornato o meno), si deve richiamare la procedura di aggiornamento.

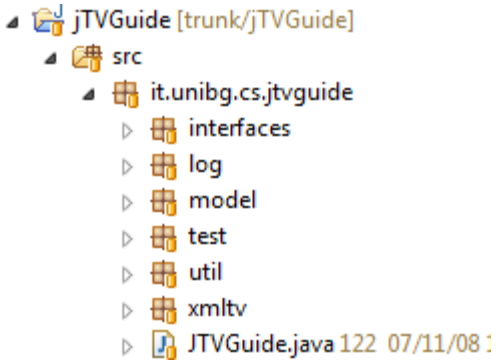
Qual è il criterio per cui si testa se uno schedule è aggiornato o no?

XMLTVScheduleInspector legge il file XML e cerca dei programmi televisivi la cui data di inizio sia oggi E, in aggiunta, domani: se una di queste due condizioni non è soddisfatta, lo schedule è da riscaricare.

VIEW: interfaccia grafica che mostra varie viste dello Schedule, mostrando per ogni Channel i programmi in onda.

CONTROLLER: mediante il mouse posso spostarmi tra la finestra principale contenente il palinsesto e le viste secondarie, come il configuratore (XMLTVConfigurator) e le opzioni (UserPreferences).

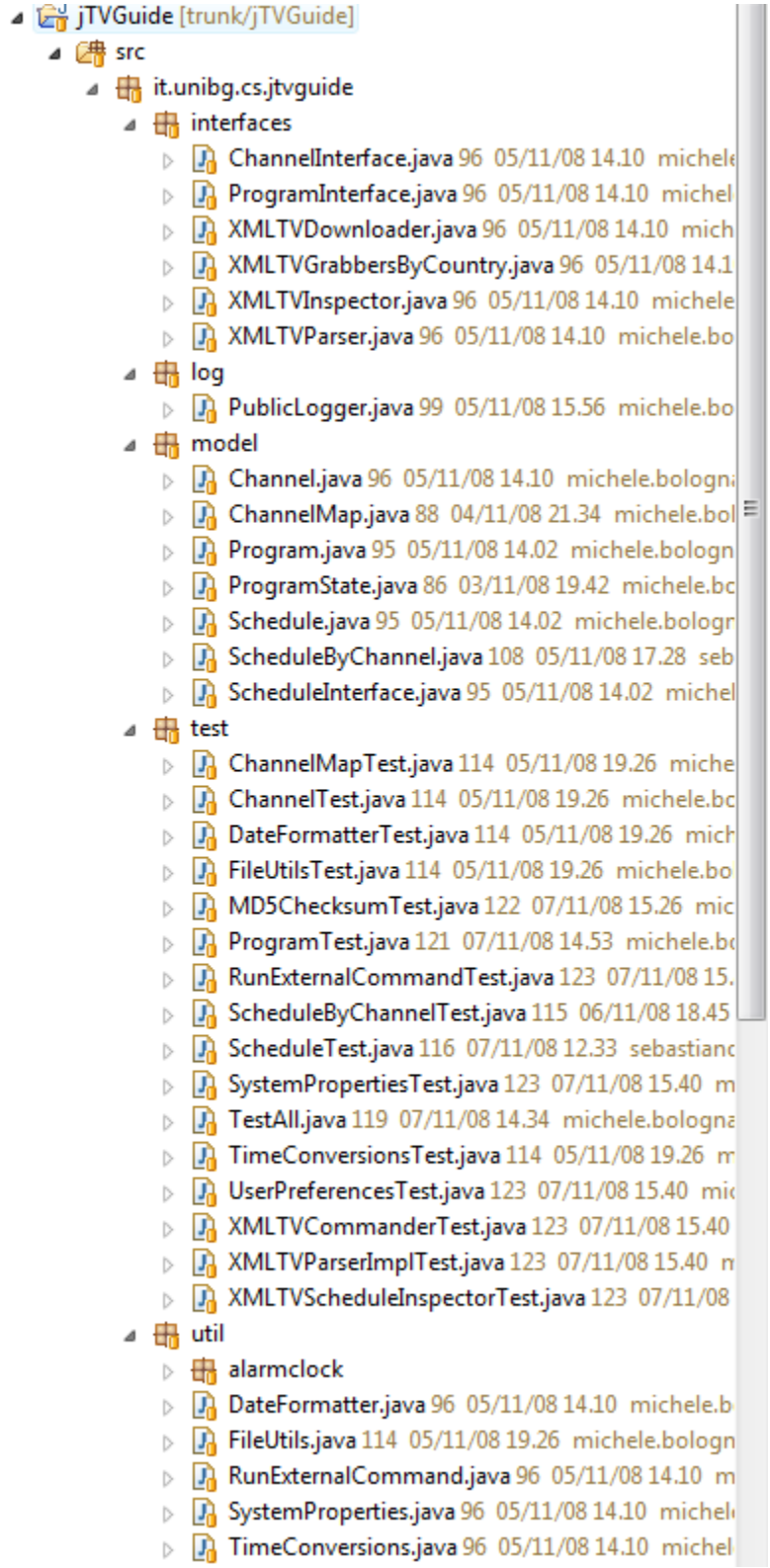
Divisione in package (dettagli su JTVGuide CORE)

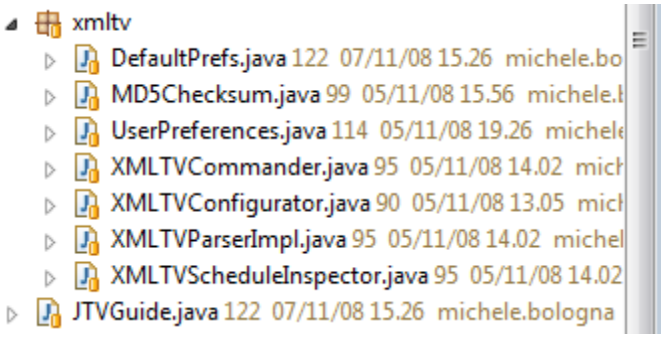


Il CORE è diviso in alcuni package:

- interfaces: contiene le interfacce, cioè classi astratte usate per meglio gestire l'astrazione del dominio
- log: contiene gli strumenti per la facilitazione del log

- model: contiene le classi del dominio, ovvero che descrivono la business logic del programma: Program, Channel, Schedule, etc
- test: contiene le classi di test, descritte più avanti nel capitolo
- util: contiene classi di utilità, come MD5, conversioni di date/tempi, etc.
- xmltv: contiene le classi di gestione xmltv





Package interfaces

- ChannelInterface: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità di tipo canale televisivo
- ProgramInterface: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità di tipo programma televisivo
- XMLTVDownloader: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità che scarica il palinsesto televisivo
- XMLTVInspector: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità che ispeziona il palinsesto televisivo per testare se il palinsesto è aggiornato
- XMLTVParser: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità che fa il parsing del file XML prodotto da XMLTV
- XMLTVGrabbersByCountry: è una classe enum che esplicita l'associazione tra locale (es. "it", "en") e il grabber XMLTV corrispondente (es. tv_grab_it, tv_grab_en)

Package log

- PublicLogger: inizializza il logger con una clausola static, ed espone alle classi che lo richiedono un'istanza specifica del logger configurato (singleton)

Package model

- Channel: contiene la descrizione delle operazioni per descrivere correttamente un'entità di tipo canale televisivo
- ChannelMap: è una struttura dati per accedere velocemente ai canali memorizzati specificando un ID univoco
- Program: contiene la descrizione delle operazioni per descrivere correttamente un'entità di tipo programma televisivo
- Schedule: contiene la descrizione delle operazioni per descrivere correttamente un'entità di tipo palinsesto televisivo
- ScheduleByChannel: *si comporta come uno Schedule*, tranne per il fatto che i programmi inseriti sono caratterizzati dallo stesso canale
- ProgramState: è una classe enum che elenca i possibili stati in cui il programma può trovarsi

Package test

Contiene le classi di test, descritte approfonditamente più avanti in questa relazione

Package util

- DateFormatter: contiene un parser di date/tempi per passare facilmente dal dominio delle stringhe a quello delle date (e viceversa)
- FileUtils: contiene utilità per operare sui file, ad esempio per copiare un file da un path all'altro, per cercare pattern all'interno di file testuali (grep), per contare le righe non commentate in un file, etc
- RunExternalCommand: permette di richiamare un programma esterno (es. dir). Sarà usata per richiamare il programma xmltv
- SystemProperties: permette di ricavare informazioni sul sistema operativo su cui sta girando il programma (es. versione, architettura, separatore delle directory (\, /), la home dell'utente…). Queste informazioni saranno utilizzate per richiamare correttamente xmltv con RunExternalCommand e per salvare il file di configurazione nella home dell'utente
- TimeConversions: permette di convertire facilmente tempi in secondi, millisecondi, e così via

Package util.alarmclock

Contiene le classi per implementare una sveglia all'orario programmato. Per le motivazioni spiegate nelle conclusioni, abbiamo deciso di introdurre comunque il codice nel progetto, ma di non utilizzarlo a causa del mancato testing.

Package xmltv

- DefaultPrefs: raggruppa le costanti di default
- XMLTVConfigurator: contiene la descrizione delle operazioni che una classe deve implementare per descrivere correttamente un'entità che configura il programma xmltv
- MD5Checksum: questa classe fornisce dei metodi per leggere da file, scrivere su file confrontare e calcolare un hash MD5. Utilizziamo questa funzionalità per capire se uno schedule è da aggiornare o meno; infatti, ogni volta che si scarica uno schedule, si salva l'hash md5 del file di configurazione. Quando si controlla se uno schedule è aggiornato, si verifica che l'hash md5 non sia cambiato: nel caso in cui sia cambiato, il file di configurazione è cambiato, e di conseguenza si deve riscaricare lo schedule
- UserPreferences: raccoglie le preferenze dell'utente. La classe è costituita da metodi static, in modo che ci sia solo un'istanza di questa classe e si possa accedere velocemente ai suoi dati da tutte le classi del progetto
- XMLTVCommander: questa classe funge da commander di xmltv e si occupa di richiamare la versione di xmltv corretta con i parametri specificati dall'utente. Tale classe riesce sia a configurare xmltv che a scaricare lo schedule (infatti implementa sia XMLTVConfigurator che XMLTVDownloader)
- XMLTVParserImpl: costituisce il parser dei file prodotti da XML
- XMLTVScheduleInspector: implementa le condizioni di verifica dell'aggiornamento di uno schedule descritte precedentemente

Metodologia di sviluppo

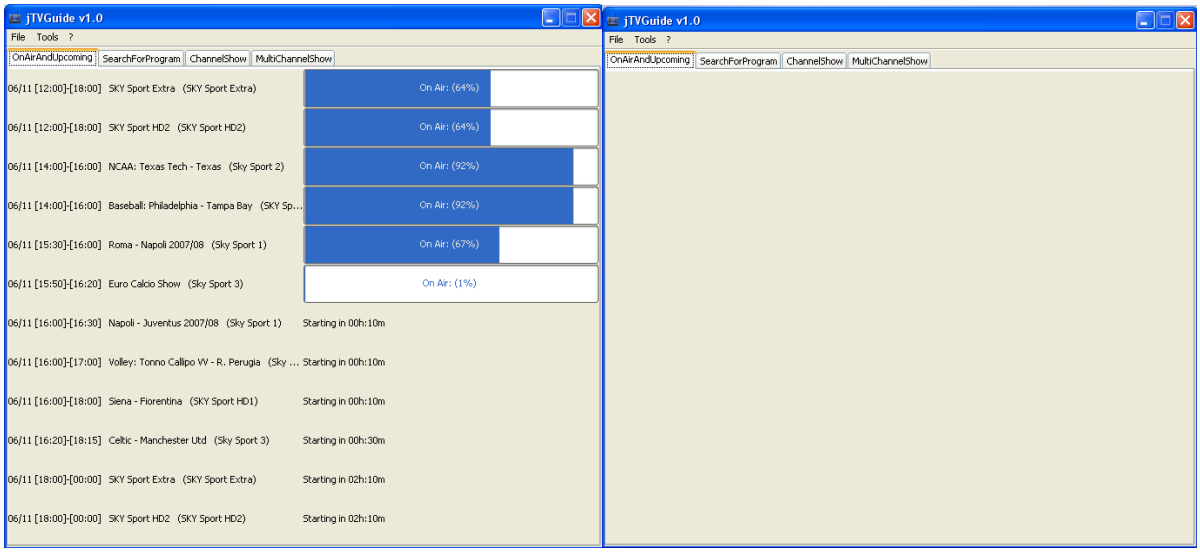
Il software sarà stato sviluppato secondo la metodologia TDD, che consiste nello scrivere il proprio codice ripetendo ciclicamente questi passi:

- scrivo il test prima ancora che esista il codice da testare
- il test (ovviamente) fallisce
- scrivo il codice minimo che faccia passare il test
- il test passa
- eseguo il refactoring del codice

Casi d'uso

Use case #1 - Avvio

1. Quando l'applicativo viene avviato, controlla la presenza del file XML contenente il palinsesto dei canali. Se esso viene trovato le quattro schede di visualizzazione vengono popolate con i dati presenti nel file, altrimenti vengono lasciate vuote (si osservino le due immagini esplicative).



Nota: utilizzo dei thread

Durante la visualizzazione, i programmi in onda tendono a "riempire la barra di avanzamento" (in quanto la loro durata aumenta). Man mano che il tempo passa, l'applicazione modifica l'avanzamento dei vari programmi. Inoltre, i programmi con grado di avanzamento > 101% sono chiaramente terminati, e vengono eliminati dalla vista dell'utente, facendo posto ai nuovi programmi in onda (quelli che precedentemente erano negli upcoming).

Tutto questo aggiornamento è svolto tramite un thread di aggiornamento.

I thread, rappresentano il mezzo mediante il quale Java fa eseguire un'applicazione da più Virtual Machine contemporaneamente, allo scopo di ottimizzare i tempi del runtime.

Un thread è un singolo flusso sequenziale di controllo all'interno di un processo. Sono oggetti particolari, ai quali si richiede un servizio corrispondente al lancio di una attività, che procede in concorrenza con chi l'ha richiesto.

Esistono due modalità per implementare thread in Java:

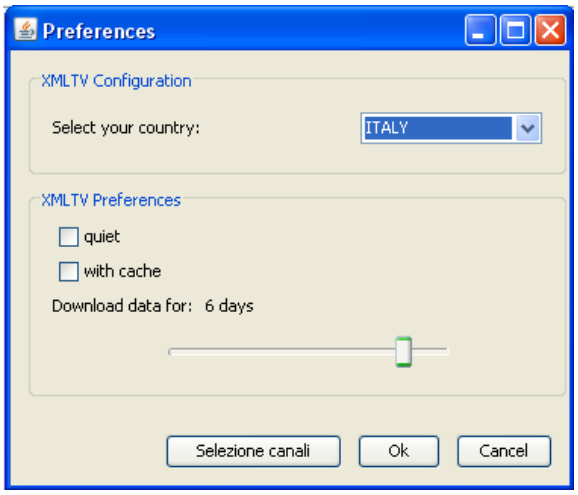
1. come sottoclasse della classe Thread

2. come classe che implementa l'interfaccia Runnable

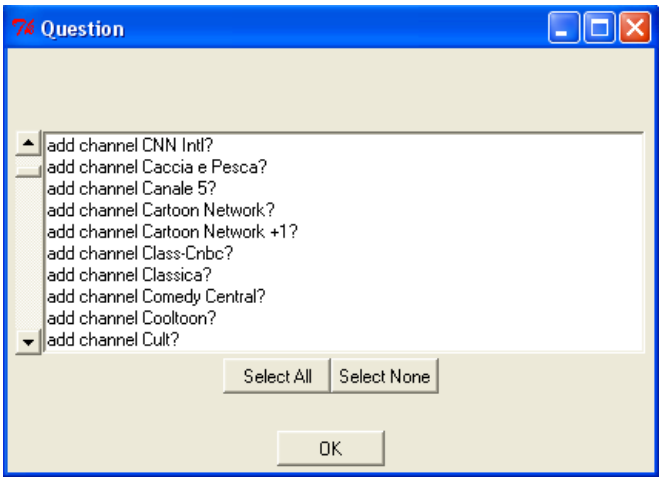
Per non limitare l'estendibilità presente e futura del programma, si è scelto di utilizzare la seconda soluzione, dato che in Java è permessa l'implementazione di più classi astratte (ma non è permessa l'estensione di più classe concrete).

Use case #2 – Tools -> Preferences

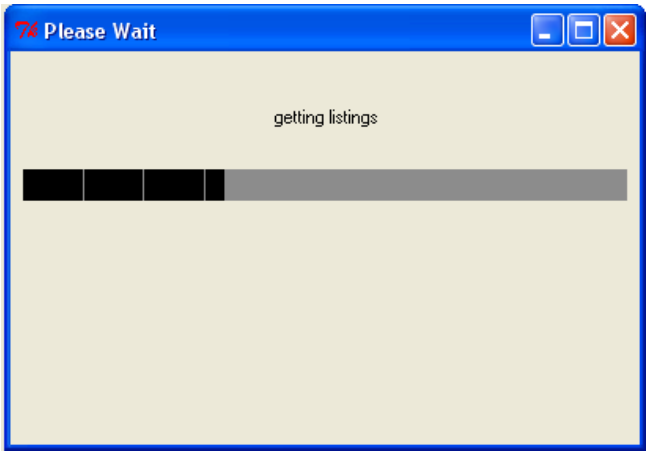
1. Come già illustrato precedentemente, quando l'applicativo viene avviato, controlla che non sia già presente il file XML con la descrizione del palinsesto.
2. L'utente può a questo punto decidere di voler cambiare le proprie preferenze (Tools -> Preferences, o Ctrl+P):



3. L'utente può selezionare un paese diverso da quello evidenziato (se è interessato a conoscere il palinsesto di canali di altri paesi), può selezionare una o entrambe le checkbox presenti (esse velocizzano sensibilmente il download dei palinsesti) e selezionare il numero di giorni dei quali vuole conoscere la programmazione.
4. Se l'utente clicca successivamente su "Selezione canali" gli viene mostrata una window nella quale può selezionare i canali di cui è interessato ad ottenere la programmazione:



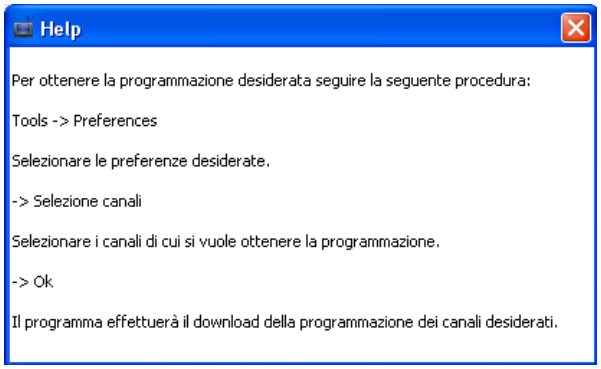
5. Una volta selezionati tutti i canali di interesse e aver cliccato su "OK" il programma memorizza le preferenze appena espresse, segnala la fine della configurazione ("Finished configuration") e torna alla window precedente.
6. Quando l'utente clicca nuovamente su "Ok" il programma effettua il download dei canali selezionati:



7. Nel caso in cui al punto 3 l'utente dovesse cliccare su "Ok" invece che su "Selezione canali" il programma si occuperà di reperire il palinsesto dei canali che l'utente aveva selezionato l'ultima volta che aveva cliccato su "Seleziona canali" (se per l'utente fosse la prima volta in assoluto e non avesse mai specificato alcuna preferenza riguardo ai canali il programma passerebbe automaticamente alla visualizzazione della scheda di selezione dei canali).

Use case #3 – ? -> Help

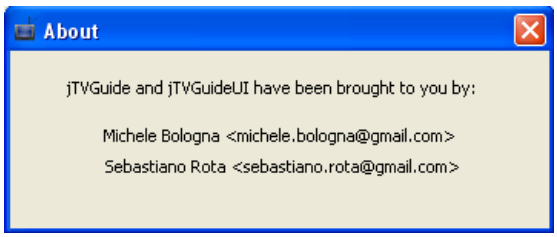
1. L'utente può decidere di consultare l'Help dell'applicazione (? -> Help o Ctrl+H):



2. Il programma visualizzerà una finestra con una breve descrizione della procedura di reperimento dei palinsesti.

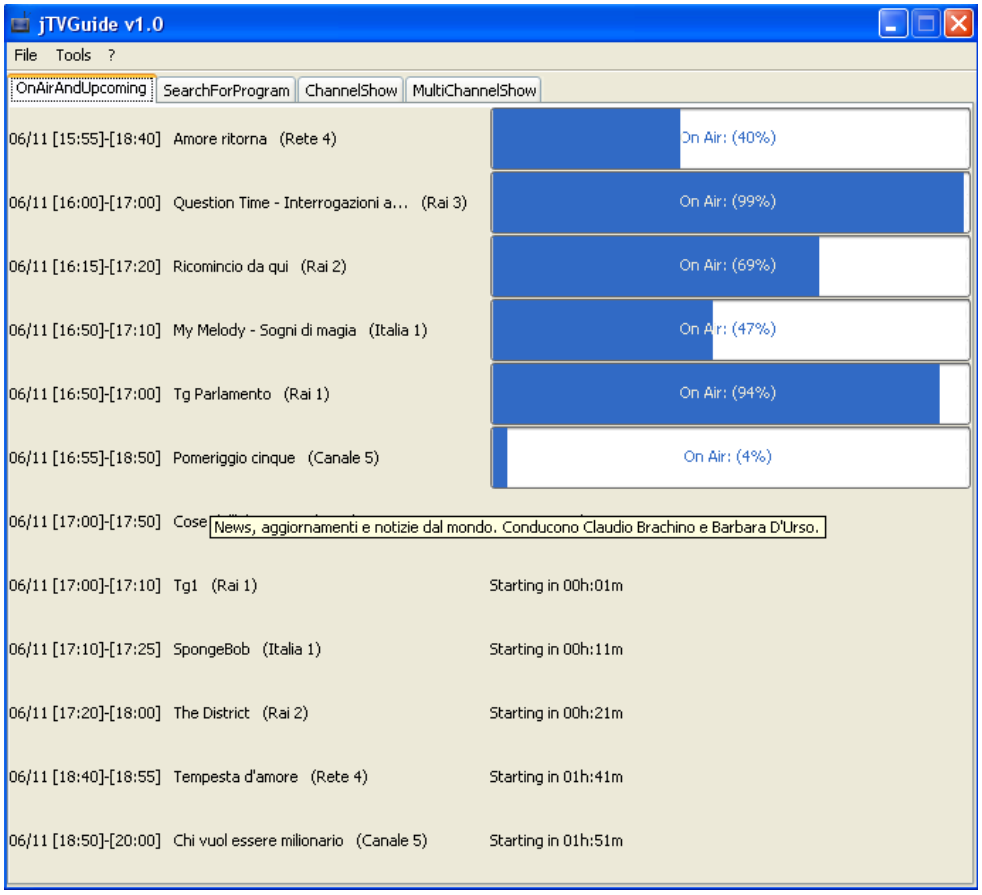
Use case #4 – ? - > About

1. Nel caso in cui l'utente segua il percorso ? -> About (o Ctrl+A) il programma visualizzerà la seguente finestra:

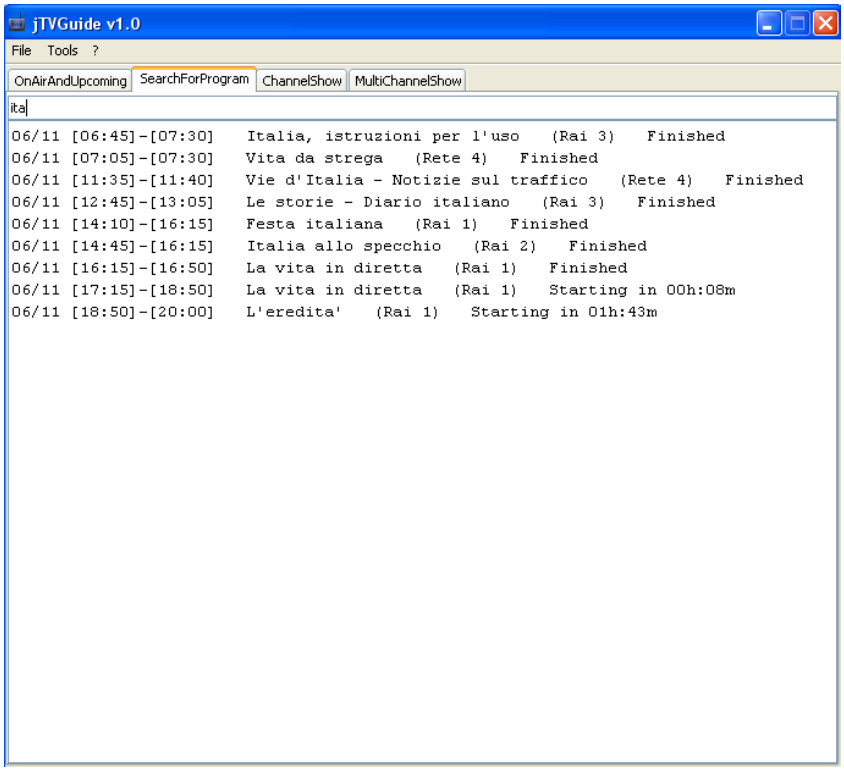


Use case #5 – Visualizzazione dei palinsesti

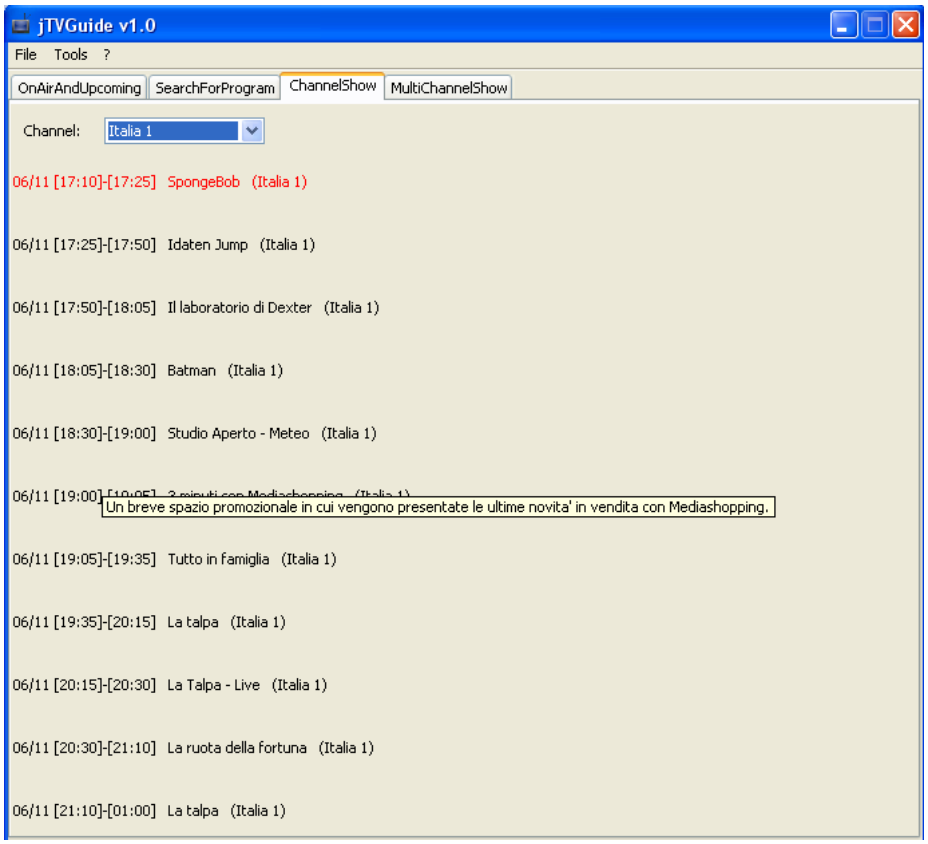
1. L'utente dopo aver ottenuto il palinsesto dei canali di interesse come illustrato precedentemente può selezionare una delle quattro schede di visualizzazione: OnAirAndUpcoming, SearchForProgram, ChannelShow o MultiChannelShow.
2. OnAirAndUpcoming visualizza i programmi attualmente in onda sui canali di interesse e i programmi immediatamente successivi. Ogni programma visualizzato porta con sé delle informazioni: giorno corrente, orario di inizio e di fine della trasmissione, titolo, canale e percentuale di avanzamento (in continuo aggiornamento). Inoltre attraversando con il puntatore del mouse l'area occupata dalla progress bar o dalle informazioni relative al programma, si può osservare comparire un tooltip che contiene la descrizione del programma in questione:



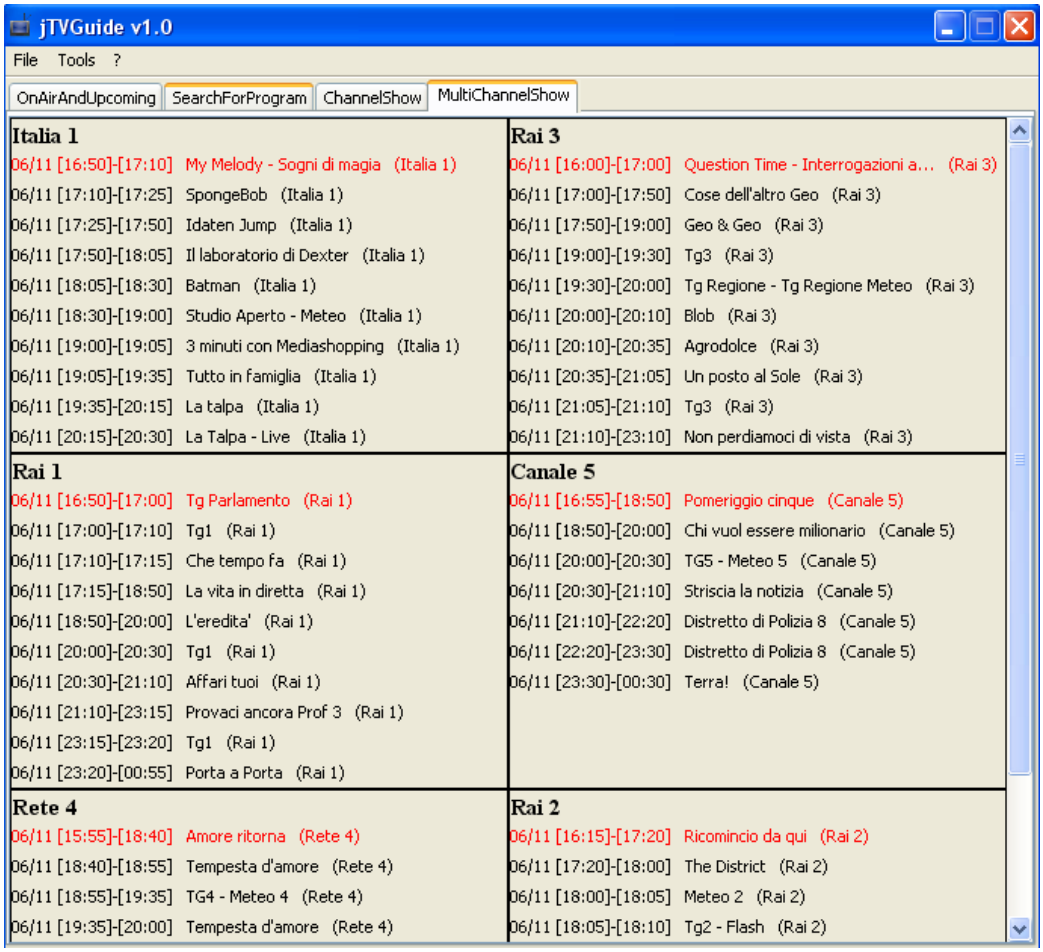
3. SearchForProgram permette di ricercare all'interno dei palinsesti dei canali di interesse dell'utente il/i programmi che desidera, il sistema visualizzerà tutti i programmi il cui titolo contiene la stringa immessa dall'utente nell'apposito textfield.



4. ChannelShow, dopo aver selezionato dalla listbox il canale di interesse dell'utente (la lista contiene tutti i canali cui l'utente ha indicato al sistema di ottenere il palinsesto) visualizza i programmi di quel determinato canale in ordine temporale di messa in onda, a partire da quello attualmente in onda. Anche in questo tipo di visualizzazione attraversando con il puntatore del mouse l'area occupata da un programma, si può osservare comparire un tooltip contenente la descrizione del programma in questione.



5. MultiChannelShow permette all'utente di avere contemporaneamente sotto controllo tutti i palinsesti dei canali di suo interesse. Ogni palinsesto visualizzato consta del programma in onda al momento della consultazione e di alcuni programmi immediatamente successivi. Il discorso del tooltip informativo vale anche per questa visualizzazione:



Test Junit

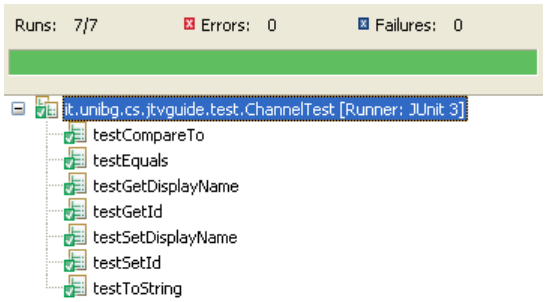
JUnit è un frame work che permette all'utente di facilitare lo sviluppo di applicazioni per l'automazione del testing in ambiente Java.
Una classe di test deve estendere la classe junit.framework.TestCase e deve avere un nome significativo, ad esempio nomeClasseDaTestareTest.java. Ogni metodo della classe di test è una procedura il cui nome inizia generalmente con test seguito dal nome del metodo da testare della classe considerata.
Ogni caso di test permette di controllare che il codice che si sta eseguendo si comporti esattamente come ci si aspetta, a tal scopo ogni test deve contenere le asserzioni che il metodo sottoesame non contenga difetti. Le asserzioni verificano quindi che il risultato dell'esecuzione di un metodo sia quello desiderato, in caso contrario viene sollevata un'eccezione e il test non va a buon fine. A titolo dimostrativo viene riportato di seguito il codice della classe ChannelTest adibita al testing della classe Channel.

```
package it.unibg.cs.jtvguide.test;

import it.unibg.cs.jtvguide.log.PublicLogger;
import it.unibg.cs.jtvguide.model.Channel;

import java.net.URI;
```

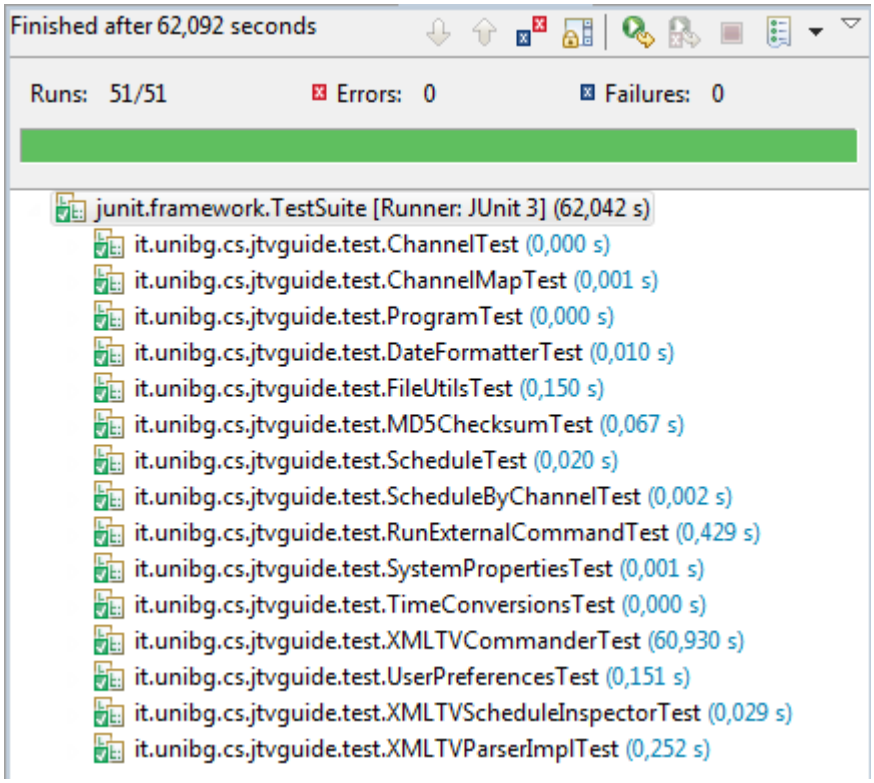
```
import java.net.URISyntaxException;
import java.text.ParseException;

import junit.framework.TestCase;

/**
 * Classe di test per la classe Channel
 *
 * @author Michele Bologna, Sebastiano Rota
 *
 */
public class ChannelTest extends TestCase {

	private Channel channelTest;

	/**
	 * Si istanzia un nuovo Channel utilizzato nei successivi metodi
	 */
	public void setUp() throws Exception {
		channelTest = new Channel("www.canale5.it", "Canale5");
	}

	/**
	 * Rimozione del Channel istanziato
	 */
	public void tearDown() throws Exception {
		channelTest = null;
	}

	/**
	 * Test del metodo compareTo della classe Channel
	 */
	public void testCompareTo() {
		try {
			assertEquals(0,channelTest.compareTo(new Chan-
nel("www.canale5.it", "Canale5")));
		} catch (ParseException e) {
			PublicLogger.getLogger().error(e);
		}
	}

	/**
	 * Test del metodo equals della classe Channel
	 */
	public void testEquals() {
		Object o = channelTest;
		Object obj = null;
		try {
			obj = new Channel("www.italia1.it", "Italia1");
		} catch (ParseException e) {
			PublicLogger.getLogger().error(e);
		}
		assertTrue(channelTest.equals(o));
		assertFalse(channelTest.equals(obj));
	}

	/**
	 * Test del metodo getDisplayName della classe Channel
	 */
	public void testGetDisplayName() {
		assertEquals("Canale5", channelTest.getDisplayName());
	}
```

```
	/**
	 * Test del metodo getId della classe Channel
	 */
	public void testGetId() {
		assertEquals("http://www.canale5.it", channelT-
est.getId().toString());
	}

	/**
	 * Test del metodo setDisplayName della classe Channel
	 */
	public void testSetDisplayName() {
		channelTest.setDisplayName("Italia1");
		assertEquals("Italia1",channelTest.getDisplayName());
	}

	/**
	 * Test del metodo setId della classe Channel
	 */
	public void testSetId() {
		try{
			channelTest.setId(new URI("http://www.italia1.it"));
			assertEquals("http://www.italia1.it", channelT-
est.getId().toString());
		}
		catch(URISyntaxException e){
			fail("Exception");
		}
	}

	/**
	 * Test del metodo toString della classe Channel
	 */
	public void testToString() {
		assertEquals("Canale5", channelTest.toString());
	}

}
```

La classe di test può quindi essere utilizzata grazie a JUnit al fine di testare la classe Channel. Ogni metodo della classe di test viene verificato:



Nell'esempio ogni test di metodo è andato a buon fine.
Ogni classe di test può essere compilata e eseguita singolarmente o raccolta in una suite di test.
La classe TestAll.java estende junit.framework.TestSuite, la sua esecuzione permette di effettuare tutta la batteria di test programmata, cioè di compilare ed eseguire tutte le classi di test realizzate.



Di seguito viene riportata la documentazione di ogni metodo appartenente alle varie classi di test.
Ogni classe ha un metodo setUp() con il compito di istanziare l'oggetto principale utilizzato successivamente nei vari metodi di test e un metodo tearDown() con il compito di rimuovere l'oggetto istanziato nel metodo di setUp():

Classe: ChannelMapTest Classe di test per ChannelMap del package it.unibg.cs.jtvguide.model	
void testAddAndContainsStringChannel()	Test congiunto dei metodi add(String, Channel) e contains(String) della classe ChannelMap
void testAddAndContainsURIChannel()	Test congiunto dei metodi add(URI, Channel) e contains(URI) della classe ChannelMap
void testGetString()	Test del metodo get(String) della classe ChannelMap
void testGetURI()	Test del metodo get(URI) della classe ChannelMap

Classe: ChannelTest Classe di test per Channel del package it.unibg.cs.jtvguide.model	
void testCompareTo()	Test del metodo compareTo della classe Channel
void testEquals()	Test del metodo equals della classe Channel
void testGetDisplayName()	Test del metodo getDisplayName della classe Channel
void testGetId()	Test del metodo getId della classe Channel
void testSetDisplayName()	Test del metodo setDisplayName della classe Channel
void testSetId()	Test del metodo setId della classe Channel
void testToString()	Test del metodo toString della classe Channel

Classe: DateFormatterTest Classe di test per DateFormatter del package it.unibg.cs.jtvguide.util	
void testFormatDate()	Test del metodo formatDate della classe DateFormatter
void testFormatDate2Time()	Test del metodo formatDate2Time della classe DateFormatter
void testFormatDate2TimeWithDay()	Test del metodo formatDate2TimeWithDay della classe DateFormatter
void testFormatString()	Test del metodo formatString della classe DateFormatter

Classe: FileUtilsTest Classe di test per FileUtils del package it.unibg.cs.jtvguide.util	
void testUncommentedLinesCount()	Test del metodo uncommentedLinesCount della classe FileUtils
void testGrep()	Test del metodo grep della classe FileUtils

Classe: MD5ChecksumTest Classe di test per MD5Checksum del package it.unibg.cs.jtvguide.xmltv	
void testCheckMD5()	Test del metodo checkMD5 della classe MD5Checksum
void testGetMD5Checksum()	Test del metodo getMD5Checksum della classe MD5Checksum
void testWriteMD5ToFile()	Test del metodo writeMD5ToFile della classe MD5Checksum

Classe: ProgramTest Classe di test per Program del package it.unibg.cs.jtvguide.model	
void testGetAndSetDesc()	Test congiunto dei metodi getDesc e setDesc(String) della classe Program
void testCompareTo()	Test del metodo compareTo della classe Program
void testGetChannel()	Test del metodo getChannel della classe Program
void testGetCompletionPercentile()	Test del metodo getCompletionPercentile della classe Program
void testGetInfo()	Test del metodo getInfo della classe Program
void testGetStartDate()	Test del metodo getStartDate della classe Program
void testGetStartingTime()	Test del metodo getStartingTime della classe Program
void testGetState()	Test del metodo getState della classe Program
void testGetStopDate()	Test del metodo getStopDate della classe Program
void testGetTitle()	Test del metodo getTitle della classe Program
void testSetChannel()	Test del metodo setChannel della classe Program
void testSetStartDate()	Test del metodo setStartDate della classe Program
void testSetStopDate()	Test del metodo setStopDate della classe Program
void testSetTitle()	Test del metodo setTitle della classe Program
void testToString()	Test del metodo toString della classe Program

Classe: RunExternalCommandTest

Classe di test per RunExternalCommand del package it.unibg.cs.jtvguide.util	
void testCommand()	Test del metodo runCommand della classe RunExternalCommand

Classe: ScheduleByChannelTest Classe di test per ScheduleByChannel del package it.unibg.cs.jtvguide.model	
void testAdd()	Test del metodo add della classe ScheduleByChannel
void testGetChannelName()	Test del metodo getChannelName della classe ScheduleByChannel

Classe: ScheduleTest Classe di test per Schedule del package it.unibg.cs.jtvguide.model	
void testAddAndGetPrograms()	Test congiunto dei metodi add e getPrograms della classe Schedule
void testGetOnAirPrograms()	Test del metodo getOnAirPrograms della classe Schedule
void testGetProgramsByName()	Test del metodo getProgramsByName della classe Schedule
void testGetProgramsFromDateToDate()	Test del metodo getProgramsFromDateToDate della classe Schedule
void testGetSchedulesByChannel()	Test del metodo getSchedulesByChannel della classe Schedule
void testGetUpcomingPrograms()	Test del metodo getUpcomingPrograms della classe Schedule

Classe: SystemPropertiesTest Classe di test per SystemProperties del package it.unibg.cs.jtvguide.util	
void testDetect()	Test del metodo detectOS della classe SystemProperties

Classe: TimeConversionsTest Classe di test per TimeConversions del package it.unibg.cs.jtvguide.util	
void testMillisecs2Time()	Test del metodo millisecs2Time della classe TimeConversions

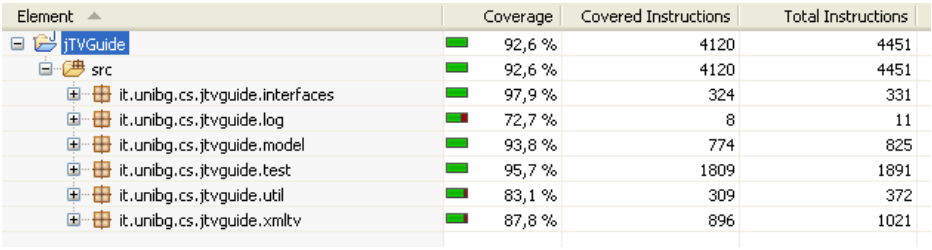
Classe: UserPreferencesTest Classe di test per UserPreferences del package it.unibg.cs.jtvguide.xmltv	
void testOptions()	Test congiunto dei metodi setDays, setQuiet, setWithCache, isQuiet, isWithCache, getDays, getXmltvConfigFile, getXmltvOutputFile e getXMLGrabbersByCountry della classe UserPreferences
void testIO()	Test congiunto dei metodi loadFromXMLFile e saveToXMLFile della classe UserPreferences

Classe: XMLTVCommanderTest Classe di test per XMLTVCommander del package it.unibg.cs.jtvguide.xmltv	
void testXMLTVCommands()	Test congiunto dei metodi configureXMLTV e downloadSchedule della classe XMLTVCommander

Classe: XMLTVParserImplTest Classe di test per XMLTVParserImpl del package it.unibg.cs.jtvguide.xmltv	
void testParsing()	Test del metodo parse della classe XMLTVParserImpl

Classe: XMLTVScheduleInspectorTest Classe di test per XMLTVScheduleInspector del package it.unibg.cs.jtvguide.xmltv	
void testIsUpdate()	Test del metodo isUpToDate della classe XMLTVScheduleInspector

Escludendo dal Build Path la classe JTVGuide.java (utilizzata soltanto durante lo sviluppo del software per eseguire delle prove sul codice in via di sviluppo) e il package it.unibg.cs.jtvguide.util.alarmclock e le sue classi (implementate nella parte core in prospettiva di eventuali sviluppi futuri dell'applicativo) la copertura del codice ottenuta eseguendo la suite di test risulta essere del 92,6% (come documentato nel file jTVGuideTestReport.html).



Element	Coverage	Covered Instructions	Total Instructions
jTVGuide	92,6 %	4120	4451
src	92,6 %	4120	4451
it.unibg.cs.jtvguide.interfaces	97,9 %	324	331
it.unibg.cs.jtvguide.log	72,7 %	8	11
it.unibg.cs.jtvguide.model	93,8 %	774	825
it.unibg.cs.jtvguide.test	95,7 %	1809	1891
it.unibg.cs.jtvguide.util	83,1 %	309	372
it.unibg.cs.jtvguide.xmltv	87,8 %	896	1021

EMMA Coverage Report (generated Fri Nov 07 15:15:20 CET 2008)

[all classes]

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %	line, %
all classes	100% (34/34)	93% (207/222)	93% (4120/4451)	87% (902,5/1032)

OVERALL STATS SUMMARY

total packages: 6

total executable files: 34

total classes: 34

total methods: 222

Michele BOLOGNA – Sebastiano ROTA

total executable lines: 1032

COVERAGE BREAKDOWN BY PACKAGE

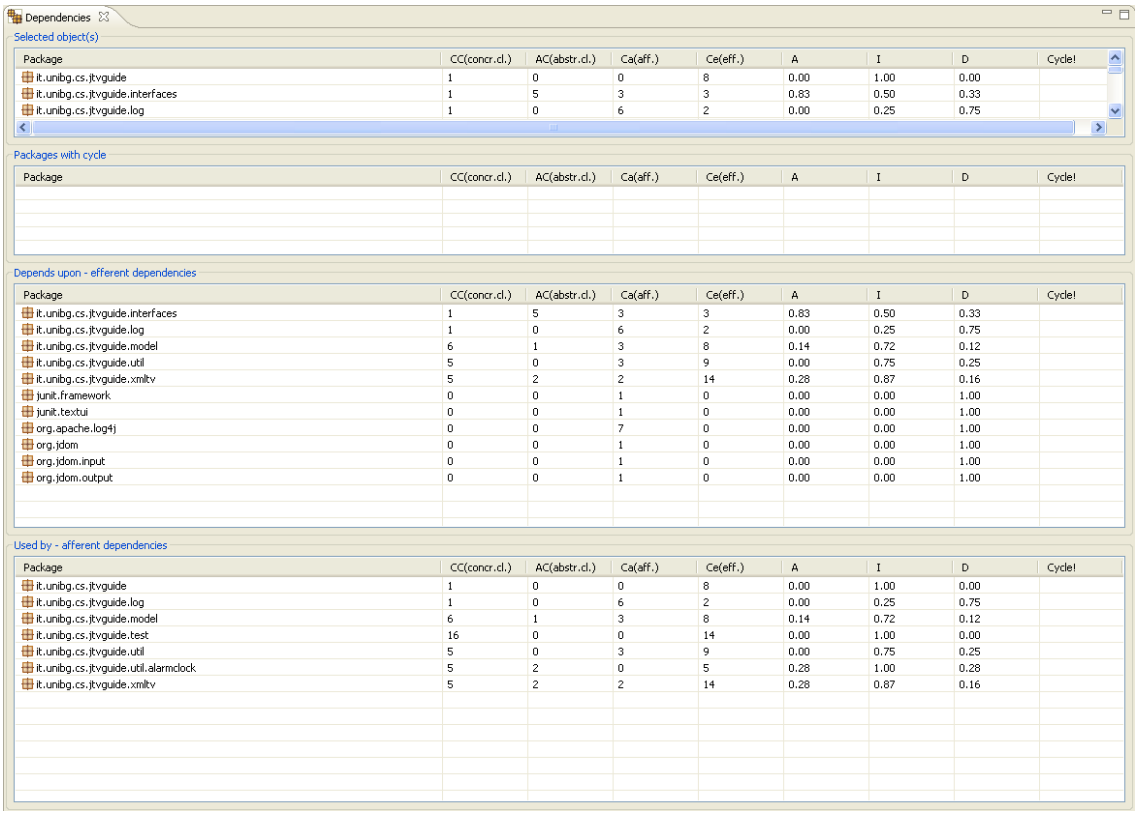
name	class, %	method, %	block, %	line, %
it.unibg.cs.jtvguide.log	100% (1/1)	67% (2/3)	73% (8/11)	80% (4/5)
it.unibg.cs.jtvguide.util	100% (5/5)	77% (17/22)	83% (309/372)	73% (73,7/101)
it.unibg.cs.jtvguide.xmltv	100% (5/5)	91% (32/35)	88% (896/1021)	84% (224,1/266)
it.unibg.cs.jtvguide.model	100% (6/6)	96% (53/55)	94% (774/825)	89% (154,7/174)
it.unibg.cs.jtvguide.test	100% (16/16)	97% (97/100)	96% (1809/1891)	91% (413,2/452)
it.unibg.cs.jtvguide.interfaces	100% (1/1)	86% (6/7)	98% (324/331)	96% (32,8/34)

[all classes]

EMMA 2.0.5312 EclEmma Fix 1 (C) Vladimir Roubtsov

Analisi con JDepend

JDepend è uno strumento che permette di analizzare le relazioni tra package Java. Lo scopo dell'analisi è cercare di verificare la modularità e la qualità del sistema. Questi due parametri sono strettamente legati alla manutenibilità e estensibilità dell'applicativo. Dall'analisi con JDepend del sistema realizzato si può evincere l'assenza di cicli tra i diversi pacchetti del progetto:



Dependencies

Selected object(s)

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
it.unibg.cs.jtvguide	1	0	0	8	0.00	1.00	0.00	
it.unibg.cs.jtvguide.interfaces	1	5	3	3	0.83	0.50	0.33	
it.unibg.cs.jtvguide.log	1	0	6	2	0.00	0.25	0.75	

Packages with cycle

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!

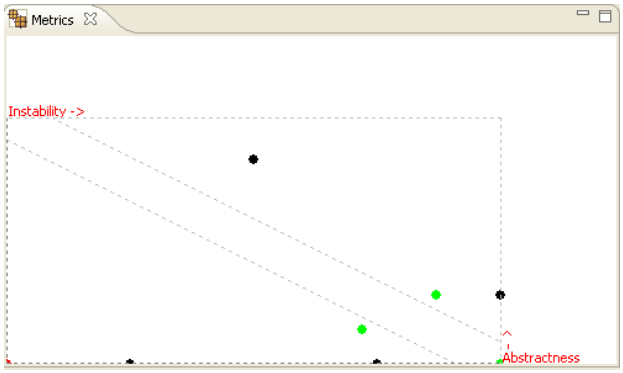
Depends upon - efferent dependencies

Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
it.unibg.cs.jtvguide.interfaces	1	5	3	3	0.83	0.50	0.33	
it.unibg.cs.jtvguide.log	1	0	6	2	0.00	0.25	0.75	
it.unibg.cs.jtvguide.model	6	1	3	8	0.14	0.72	0.12	
it.unibg.cs.jtvguide.util	5	0	3	9	0.00	0.75	0.25	
it.unibg.cs.jtvguide.xmltv	5	2	2	14	0.28	0.87	0.16	
junit.framework	0	0	1	0	0.00	0.00	1.00	
junit.textui	0	0	1	0	0.00	0.00	1.00	
org.apache.log4j	0	0	7	0	0.00	0.00	1.00	
org.jdom	0	0	1	0	0.00	0.00	1.00	
org.jdom.input	0	0	1	0	0.00	0.00	1.00	
org.jdom.output	0	0	1	0	0.00	0.00	1.00	

Used by - afferent dependencies

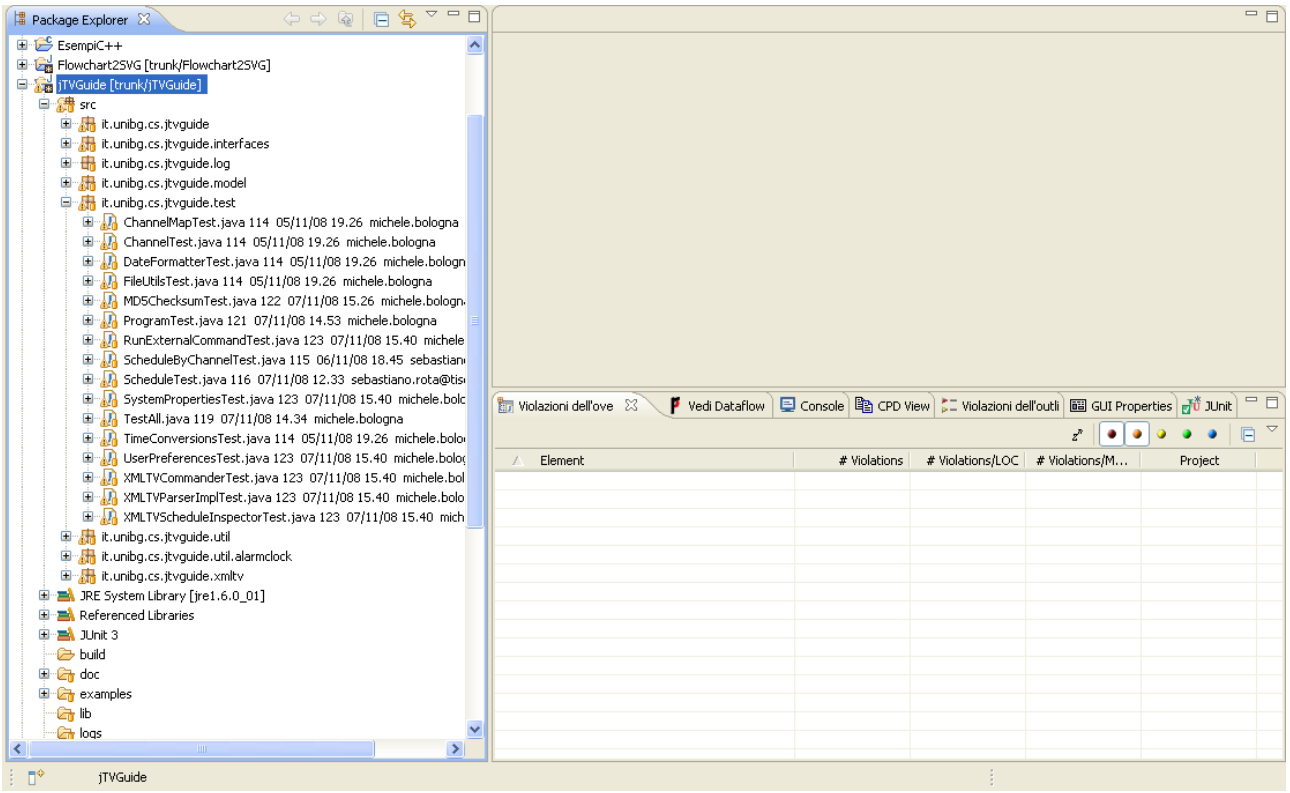
Package	CC(concr.cl.)	AC(abstr.cl.)	Ca(aff.)	Ce(eff.)	A	I	D	Cycle!
it.unibg.cs.jtvguide	1	0	0	8	0.00	1.00	0.00	
it.unibg.cs.jtvguide.log	1	0	6	2	0.00	0.25	0.75	
it.unibg.cs.jtvguide.model	6	1	3	8	0.14	0.72	0.12	
it.unibg.cs.jtvguide.test	16	0	0	14	0.00	1.00	0.00	
it.unibg.cs.jtvguide.util	5	0	3	9	0.00	0.75	0.25	
it.unibg.cs.jtvguide.util.alarmclock	5	2	0	5	0.28	1.00	0.28	
it.unibg.cs.jtvguide.xmltv	5	2	2	14	0.28	0.87	0.16	

Per completezza viene riportato anche la rappresentazione grafica riportante le metriche di JDepend riguardanti l'applicazione:



Analisi con PMD

PMD è uno strumento capace di compiere un'analisi statica del codice, in parole povere si occupa di analizzare il codice senza eseguirlo allo scopo di individuare errori durante la sua scrittura. PMD organizza le violazioni riscontrate in cinque categorie diverse, dalla più grave alla meno grave. Si può notare la totale assenza di violazioni di primo e di secondo livello all'interno del progetto:



Conclusioni

Durante il nostro percorso universitario, ci siamo imbattuti in pochi progetti pratici (a nostro avviso). Questo progetto, però, rappresenta una piacevole eccezione che ci ha permesso di migliorare la nostra conoscenza di molti fattori chiave che ci torneranno senz'altro utili nella nostra futura vita lavorativa.

Un aspetto fondamentale riguarda le direttive di progetto: ci hanno concesso ampia libertà, grazie alla quale abbiamo potuto sperimentare alcuni aspetti che ci incuriosivano e che sono poi confluiti nel nostro progetto.

Abbiamo toccato con mano i metodi, le tecnologie e gli strumenti che abbiamo visto principalmente nei nostri corsi di studio, come ad esempio ingegneria del software e i corsi di informatica.

Abbiamo imparato (di nostra volontà) a coordinare da zero un progetto di sviluppo del software, mediante la progettazione, la sincronizzazione tramite SVN, la metologia TDD (test-driven development).

Non sono certo mancate le difficoltà, ma tutto sommato possiamo ora vantare tutte quest'esperienze che abbiamo acquisito.

Abbiamo aggiunto il supporto 'alarmclock', nel package util, che funziona come una sveglia: si programma un timer, e, all'orario specificato, viene innescato un evento. Alarmclock dovrebbe essere utilizzata per notificare l'inizio di un programma quando l'utente specifica che tale programma è interessante.
Per una mera questione di tempi, e dato che la classe non era stata sufficientemente testata, abbiamo deciso di includere lo stesso il sorgente nel progetto, ma di non utilizzarlo nelle nostre classi; come descriveremo negli sviluppi futuri, la futura versione di JTVGuide utilizzerà tale classe.

Sviluppi futuri

Crediamo molto nelle potenzialità del software che abbiamo sviluppato.

Il software è ben progettato: ci ha permesso infatti un'alta manutenibilità e soprattutto un'alta copertura dei test. Un'altra prova delle potenzialità del progetto è il fatto che abbiamo rilasciato i progetti JTVGuide e JTVGuideUI sui server di GoogleCode sotto licenza GNU open-source.

Il codice, la documentazione ed i test sono stati sviluppato (ove possibile) in inglese.

Sui siti ufficiali di JTVGuide e JTVGuideUI si possono trovare le informazioni per accedere a SVN e consultare tutte le revisioni SVN. Ogni revisione SVN è stata corredata da un messaggio che ne indica le differenze tra le varie revisioni. In questo modo, chiunque può contribuire e inviare le proprie modifiche al programma.
Nei prossimi mesi rimetteremo mano al codice e includeremo altre funzionalità che ci sono venute in mente.

Siti di riferimento

http://jtvguide.googlecode.com

http://jtvguideui.googlecode.com