UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

Tesi di laurea in Ingegneria Informatica

# Uno strumento a supporto del Reverse Engineering di applicazioni Flash

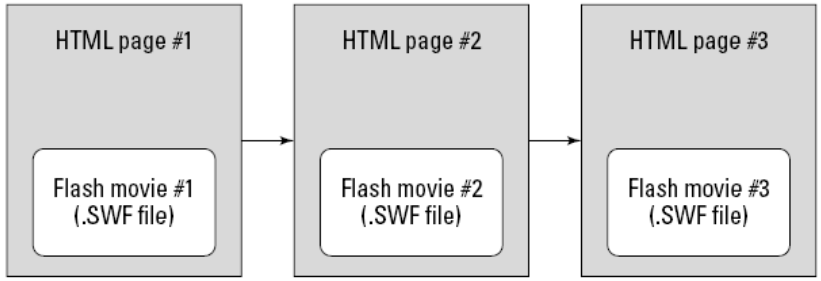
Anno accademico 2007-2008

Relatore
Ch.mo prof.
**Porfirio Tramontana**

Candidato
**Esposito Luigi**
Matr.041-003067

## Contesto : Macromedia Flash e le web applications

- **Strumento per la creazione di applicazioni dall'ampio contenuto grafico e multimediale.**
- **Sfrutta le potenzialità della grafica vettoriale e della compressione multimediale.**
- **Si basa su quattro elementi: Lo stage, la libreria, la linea temporale, il codice actionscript.**
- **E' particolarmente adatto per la realizzazione di RIA o di interi siti web.**




- **L'actionscript è responsabile del controllo dell'applicazione ed evolve verso la programmazione a oggetti**

**Sintassi per le classi**
**Variabili fortemente tipate**
**Restituzione di valori tipati dalle funzioni**

UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

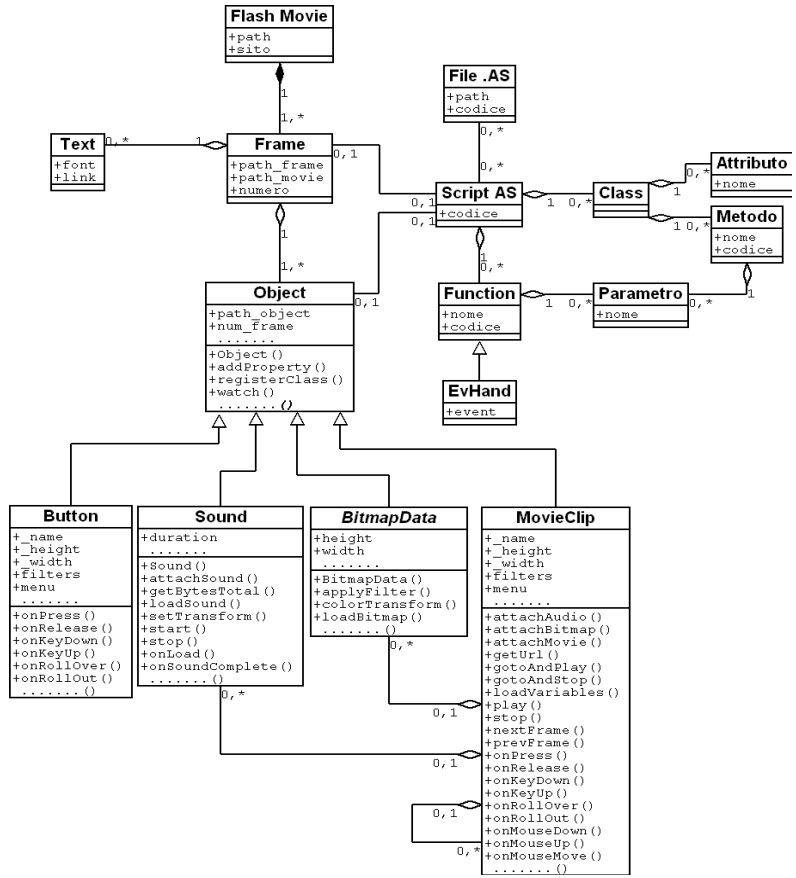


## Flash movie e gestore di eventi

**Una flash movie può essere descritta dal modello di figura. Agli oggetti che la compongono può essere associato uno script che modifica la linearità temporale dell'esecuzione.**

**L'esecuzione dipende dagli eventi generati dalla riproduzione o dall'utente dell'applicazione.**

**Il comportamento in seguito al verificarsi di un evento è stabilito dal gestore di eventi.**




## Gestori di eventi

1. **Metodi del gestore.**
2. **Oggetto Listener e Broadcaster.**
3. **Gestore dei bottoni e dei movieclip.**

# Scopi preposti

**L'applicazione realizzata ha lo scopo principale di estrapolare informazioni contenute negli scripts delle applicazioni flash riconoscendo le istruzioni presenti all'interno, memorizzando informazioni prelevate da queste ultime e legami tra di esse in un database. L'applicazione rappresenta dunque un parser per il linguaggio Actionscript 2.0 realizzato mediante il generatore di parser Javacc.**

**JavaCC è un generatore di Parser Top-Down ricorsivi. Esso controlla la correttezza di una stringa di token data, costruendo l'albero della derivazione partendo dal simbolo iniziale e scendendo fino alle foglie. Il JavaCC appartiene alla classe dei parser LL(K) ( Left-to-right, Left-most derivation, k-symbol lookahead). Tali parser leggono il file di input da sinistra verso destra, creando una derivazione sinistra della stringa da analizzare, osservando k simboli per volta per decidere quale regola della grammatica applicare.**

**La grammatica da utilizzare deve essere ricavata dalla sintassi Actionscript e viene utilizzata per la creazione delle regole di produzione dei tokens. I tokens saranno descritti mediante l'uso di espressioni che indicano quali combinazioni di stringhe in ingresso corrispondono a parole della grammatica. Una volta descritta la grammatica Il Javacc consente di ottenere il codice che implementa il parser descritto.**

# Tecnologie utilizzate.

- ***Flash Decompiler*** **come strumento di reverse per il recupero degli oggetti che compongono le animazioni, e in particolare degli scripts ad esse associati.**
- **Il DBMS Open Source *MySQL* e vari tool grafici, per la costruzione del database che consente di memorizzare le informazioni ricavate dagli scripts.**
- **Il driver nativo JDBC per l'interazione tra l'applicazione e il DBMS.**
- **Il framework *Junit* come strumento a supporto del testing e della verifica dei risultati.**

# L'ambiente di sviluppo Eclipse

**Eclipse è un ambiente di sviluppo Open Source ideato per lo sviluppo di applicazioni Java, ma dotato di numerosi plugin che lo rendono un IDE di tipo RAD.**

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

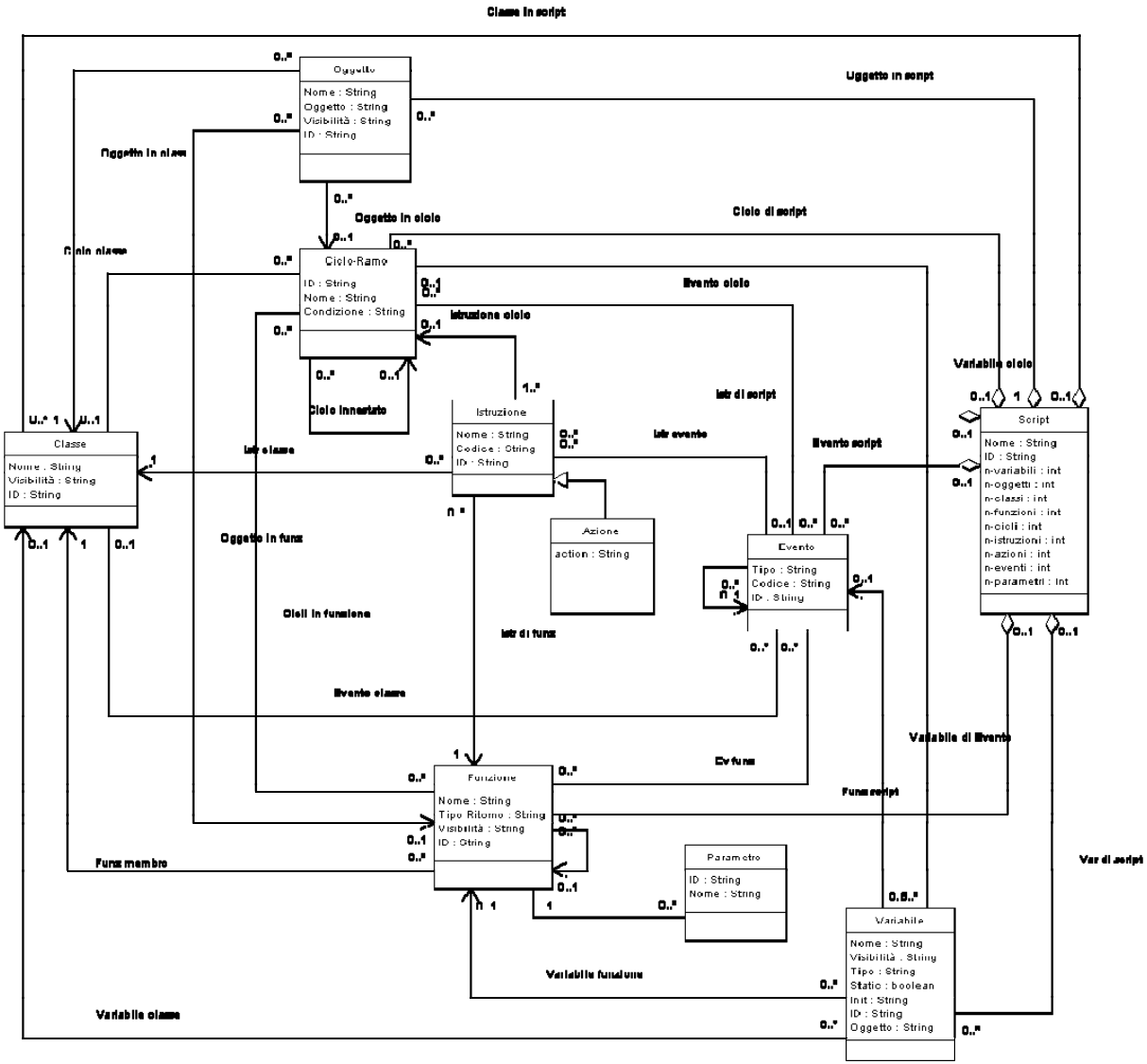


## Modello degli scripts

**Al fine di capire quali informazioni estrapolare è stato creato un modello E-R degli scripts da analizzare.**

**Esso consente di individuare quali sono gli elementi principali di cui si compone uno script e i legami che esistono tra i diversi costrutti.**

**Il modello E-R tradotto costituirà la struttura del database realizzato per memorizzare i risultati dell'analisi.**



## Le sezioni del parser

**La *sezione delle opzioni* provvede ad inizializzare alcuni parametri della compilazione e la memorizzazione delle classi prodotte dal Javacc, nonché alcune regole di parsing come il LookAhead.**

```
OPTION
{
      STATIC = false; // default TRUE
      LOOKAHEAD = 2; // default 1
      OUTPUT_DIRECTORY = "path"; // default dir corrente
}
```

**L'*unità di compilazione Java* contiene la dichiarazione della classe parser e il codice per l'inserimento dei parametri di connessione con il database e della cartella da analizzare.**

```
PARSER_BEGIN(ParseScript)
.                 public class ParseScript {
.                 }
PARSER_END(ParseScript)
```

**La *sezione dei tokens* contiene la specifica dell'analizzatore lessicale e la definizione delle parole del linguaggio, ossia la definizione della grammatica formale. In questa sezione vengono specificate quindi le regole di produzione dei tokens.**

```
SKIP : {
  <SINGOLA_RIGA_COMMENTO : "//" (~["\n","\r"])* ("\n" | "
                                        \r" | "\r\n")>
  <COMMENTO_PIU_RIGHE   : "/*" (~["*"])* "*" ("*" | ~["*","/"]
                                        (~["*"])* "*")* "/">
}
SPECIAL_TOKEN : {}
TOKEN : {
      <SET_PROPERTY :
      ("setProperty")(<SPAZIO>)*(<TONDE_INTERNE>)>
}
MORE : {}
```

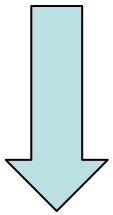
**La *sezione delle rules* contiene la specifica del parser, ossia l'insieme delle sequenze di tokens considerate legittime in ingresso. Essa costituisce quella che viene definita Bakus Normal Form.**

```
Tipo_rit  nome_regola (parametri_di_scambio):
{}
{}
```

## Alcune regole di produzione delle azioni

**Il cuore dell'applicazione è l'analizzatore lessicale che definisce le regole di produzione dei tokens e quindi indica quali sono le stringhe presenti in ingresso ritenute valide secondo la grammatica considerata. A titolo di esempio riportiamo alcune delle regole per la produzione dei token delle azioni e degli eventi.**

Stringa in ingresso



Token prodotto

```
//Tokens per azioni varie
<GOTO_AND_STOP  : ("gotoAndStop")(<SPAZIO>)*(<TONDE_INTERNE>)>
<GOTO_AND_PLAY  : ("gotoAndPlay")(<SPAZIO>)*(<TONDE_INTERNE>)>
<PLAY           : ("play")(<SPAZIO>)*(<TONDE_INTERNE>)>
<STOP           : ("stop")(<SPAZIO>)*(<TONDE_INTERNE>)>
<GET_PROPERTY   : ("getProperty")(<SPAZIO>)*(<TONDE_INTERNE>)>

//Tokens per gli eventi dei movieClip
<CLIP_EVENT_LOAD        : "onClipEvent( load )">
<CLIP_EVENT_ENTER_FRAME : "onClipEvent( EnterFrame )">
<CLIP_EVENT_UNLOAD      : "onClipEvent( unload )">
<CLIP_EVENT_MOUSE_UP    : "onClipEvent( mouseUp )">
<CLIP_EVENT_MOUSE_DOWN  : "onClipEvent( mouseDown )">

//Tokens per gli eventi dei bottoni
<ON_PRESS         : "on( press )">
<ON_RELEASE       : "on( release )">
<ON_RELEASE_OUTSIDE : "on( releaseOutside )">
<ON_KEYPRESS      : "on( keyPress ">
<ON_KEYDOWN       : "on( keyDown )">
<ON_KEYUP         : "on( keyUp )">
```

## L'applicazione

**L'applicazione si compone di nove parti di cui sei generate dal javaCC e tre implementate dallo sviluppatore:**

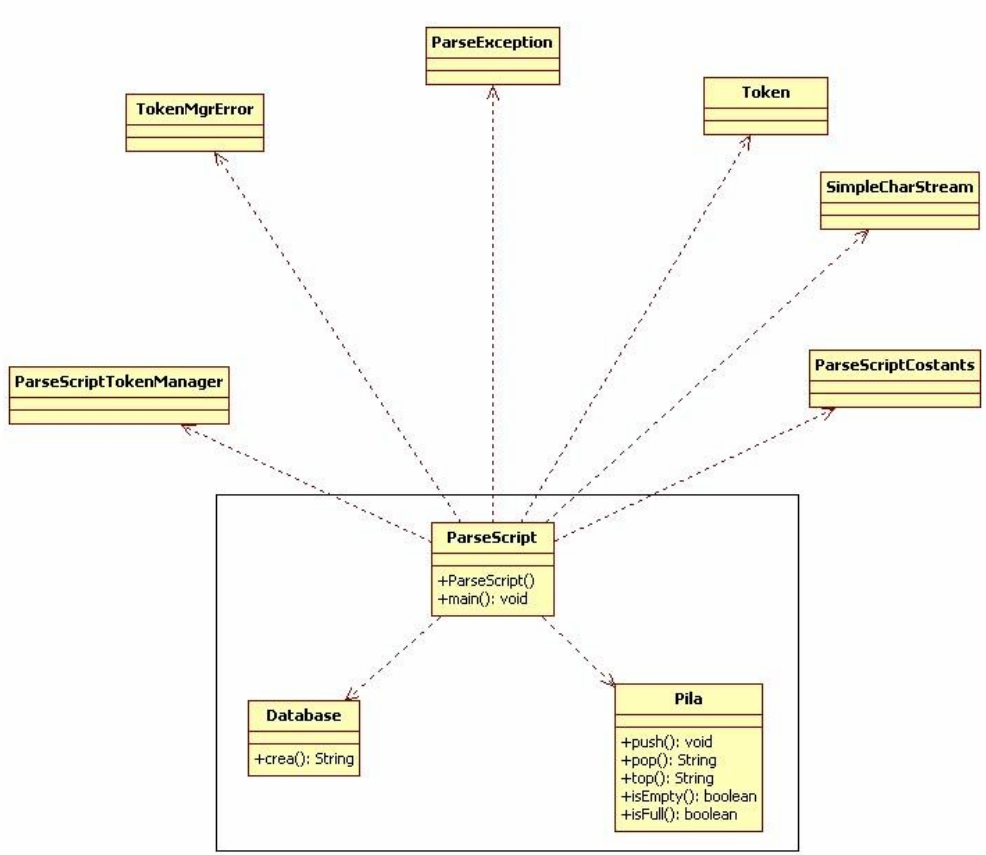
- **Il parser;**
- **La pila di stato;**
- **Il database;**

**La classe parser realizza l'analisi degli scripts Interpretandone i contenuti e si compone di quattro sezioni:**

- **Sezione delle opzioni;**
- **Unità di compilazione java;**
- **Sezione dei tokens;**
- **Sezione delle rules;**

**La pila di stato tiene traccia della profondità dell'istruzione corrente rispetto alla radice dell'albero corrispondente allo script.**

**La classe database si occupa di creare il database in cui memorizzare le informazioni trasmesse dal parser durante l'analisi.**



# Classi generate dal parser

***TokenMgrError*** **è una classe che rileva la presenza di errori; Questa classe viene usata per gli errori rilevati dall'analizzatore lessicale ed è una sottoclasse di Throwable.**

***ParseException*** **è un'altra classe per la rilevazione degli errori; Questa classe è usata per gli errori rilevati dal parser ed è una sottoclasse di Exception e di Throwable.**

***Token*** **è una classe contenente i tokens. Ogni oggetto token ha un campo di tipo intero che rappresenta il tipo di token (Plus, Number, or EOF) e un campo stringa che rappresenta la sequenza di caratteri del file d'ingresso che corrispondono al token.**

***SimpleCharStream*** **è una classe che fornisce i caratteri all'analizzatore lessicale.**

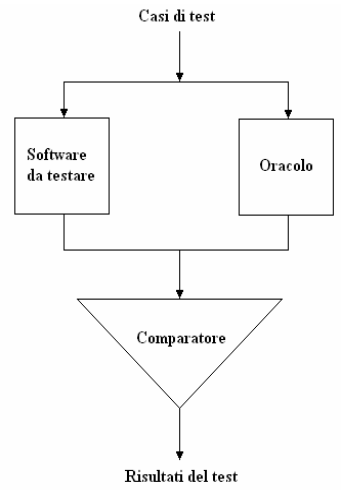
***ParseScriptConstants*** **è un'interfaccia che definisce un numero di classi usate sia nel parser che nell'analizzatore lessicale.**

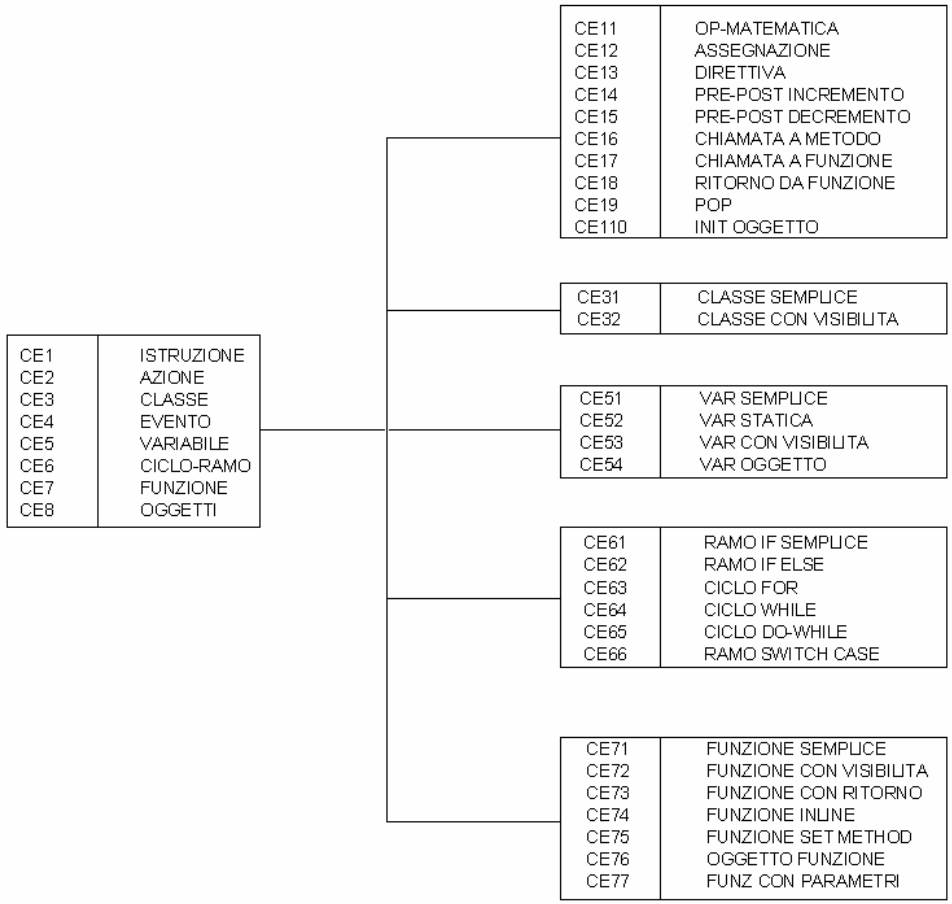
***ParseScriptTokenManager*** **è l'analizzatore lessicale.**

***ParseScript*** **è il parser.**

## Testing

**Nella fase di testing si tenta di rilevare malfunzionamenti nel software e quindi di migliorarne la qualità. Durante lo sviluppo si effettua il test di tipo *White Box*, mentre una volta terminato lo sviluppo si verifica che il software rispetti le specifiche con un testing di tipo *Black Box*. Avendo utilizzato degli strumenti automatici per la generazione delle classi è possibile evitare il testing di unità e concentrarsi sulle specifiche. E' stata sviluppata una *test suite* contenente scripts corrispondenti alle classi di equivalenza ed è stata sollecitata l'applicazione fornendo i test in ingresso. La successiva fase è la validazione dei risultati ed è stata automatizzata con il framework Junit.**



## Classe di test

- **Mediante il framework Junit è stata sviluppata la classe di test di cui se ne riporta un frammento del metodo che esegue effettivamente la verifica del risultato.**
- **Il metodo di test sfrutta gli stream INquery e INris per leggere da file le query da effettuare e i risultati attesi (L'oracolo è memorizzato in un file).**
- **Il metodo confronta mediante la funzione assertEquals il risultato reale ottenuto mediente la query, con il risultato atteso.**
- **La assert fallisce se i due parametri sono diversi.**

```
//Ciclo finchè ci sono query
do{

    //Leggo da file la query e il risultato atteso
    this.query  = this.INquery.readLine();
    this.atteso = this.INris.readLine();

    if (this.query != null){

       //Incremento il contatore delle assert
       count++;

       //Prelevo il risultato reale dal DB
       this.rslt  = this.connessione.executeQuery(this.query);
       this.rsmd  = this.rslt.getMetaData();

       //Ottengo il nome dell'unica colonna prelevata nella query
      String Colonna = new String(this.rsmd.getColumnName(1));

      //Prelevo la prima riga del result set
      this.rslt.next();
      this.reale = this.rslt.getString(Colonna);

      //Stampo il controllo effettuato
      System.out.println("");
      System.out.println("Query eseguita : "+this.query);
      System.out.println("Valore atteso  : "+this.atteso);
      System.out.println("Valore reale   : "+this.reale);

      //Testo il risultato
     assertEquals(this.atteso,this.reale);
    }
}
while(this.query != null);
```

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

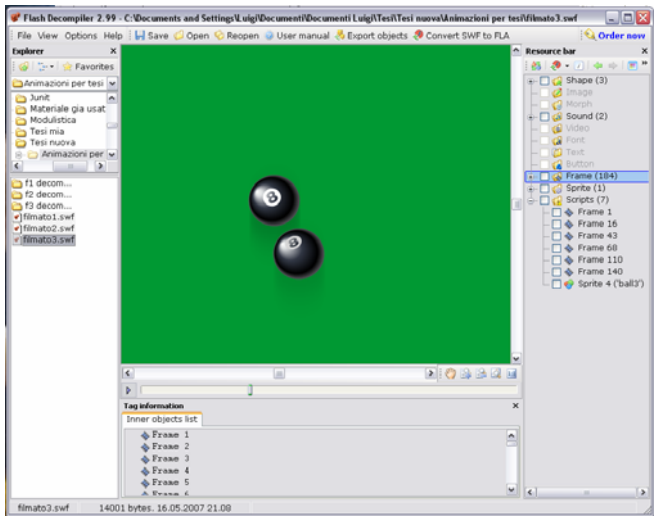


## Esempi di utilizzo

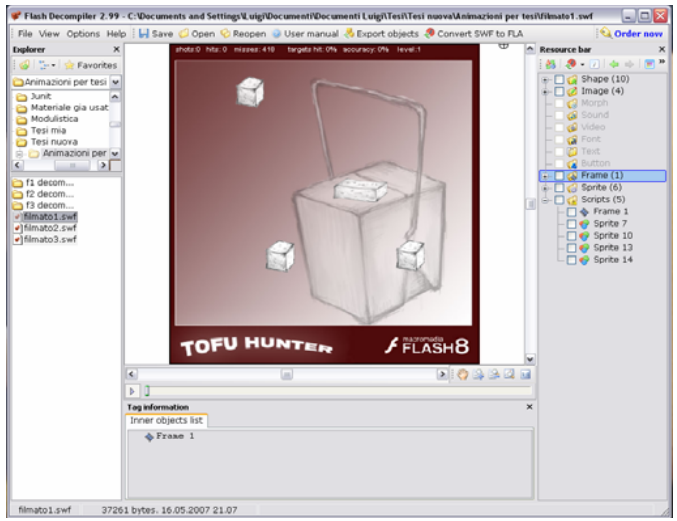
- **Il testing *Black Box* consente di verificare che l'applicazione risponda alle specifiche tecniche.**

- **E' utile verificare anche il comportamento dell'applicazione nei casi reali, a tal fine sono state scelte due applicazioni flash da esaminare : *Tavolo da biliardo e Tofu Hunter*.**

***Tavolo da biliardo***



***Tofu Hunter***



## Risultati ottenuti

**L'analisi mostra che le due applicazioni hanno caratteristiche differenti in termini di composizione e di scripts rilevati. L'esempio più rilevante è l'applicazione *Tofu Hunter* che è dotata di scripts abbastanza complessi. Per capire il modo in cui il parser tratta le istruzioni si riporta un frammento della tabella *Istruzione* del database ottenuto analizzando uno script dell'applicazione.**

| ID | NOME | CODICE | AZIONE | EVENTO | FUNZIONE | CLASSE | SCRIPT | CICLO |
|---|---|---|---|---|---|---|---|---|
| File1 IST25 | Assegnazione | stats_txt.tabStops = 100 | 0 | null | null | null | File1 | null |
| File1 IST20 | Assegnazione | my_fmt.size = 12 | 0 | null | null | null | File1 | null |
| File1 ACT1 | Azione | delete(_global.onPress) | 1 | null | File1 FUN7 | null | null | null |
| File5 ACT1 | Azione | stop() | 1 | null | null | null | File5 | null |
| File1 IST9 | Assegnazione | _global.shots = 0 | 0 | null | null | null | File1 | null |
| File4 ACT1 | Azione | stop() | 1 | null | null | null | File4 | null |
| File1 IST37 | Chiamata a funzione | ._alpha = randRange (80,100) | 0 | null | null | null | null | File1 CIC4 |
| File1 IST35 | Chiamata a funzione | ._yscale = randRange (80,100) | 0 | null | null | null | null | File1 CIC4 |
| File1 IST5 | Assegnazione | stats_txt.text = "shots:" | 0 | null | File1 FUN1 | null | null | null |
| File3 IST1 | Direttiva di initclip | #initclip | 0 | null | null | null | File3 | null |
| File1 IST43 | Operazione matema... | _global._x = _global._x + _global.speed | 0 | null | File1 FUN6 | null | null | null |
| File1 IST10 | Assegnazione | _global.cDepth = 100 | 0 | null | null | null | File1 | null |
| File1 IST11 | Assegnazione | _global.level = 1 | 0 | null | null | null | File1 | null |
| File1 IST12 | Chiamata a metodo | this.attachMovie("crosshair_mc","crosshair_... | 0 | null | null | null | File1 | null |
| File1 IST29 | Post-incremento | this.cDepth++ | 0 | null | null | null | null | File1 CIC4 |
| File1 IST6 | ritorno da funzione | return Math.floor(Math.random() * ((maxNum)... | 0 | null | File1 FUN2 | null | null | null |

**La tabella mostra il riconoscimento di alcuni tipi di istruzioni e le informazioni relative al costrutto in cui sono incluse. Dall'esempio si può capire che effettuando delle query è possibile risalire alle istruzioni che compongono una data funzione o un dato ciclo e quindi si può risalire all'albero associato ad un dato script.**

## Conclusioni e sviluppi futuri

- ***Adobe flash*** **è una delle tecnologie più utilizzate per la produzione di applicazioni grafiche dotate di contenuti multimediali, effetti speciali e dimensioni ridotte.**
- **L'evoluzione del linguaggio di programmazione interno *Actionscript* ha favorito lo sviluppo di applicazioni interattive la cui riproduzione in un contesto web è affidata al *Flash Player* che ingloba *l'Actionscript Virtual Machine* ed è presente nella maggior parte dei browser.**
- **Negli ultimi anni aumenta la tendenza allo sviluppo di *Rich Internet Application* e si è resa necessaria la cooperazione della tecnologia flash con tecnologie alternative e concorrenti come *AJAX***
- **Per far fronte a queste necessità sono stati rilasciati dalla *Adobe* dei kit per richiamare funzionalità relative ad altre tecnologie come *Javascript* all'interno del codice *Actionscript* rendendo così le RIA sviluppate in flash degli ibridi in cui parte del codice di programmazione appartiene ad un linguaggio e parte ad altri.**
- **Il parser risulta valido in un contesto puramente legato all'Actionscript, ma necessita di essere adeguato per l'analisi delle parti degli scripts che non contengono codice Actionscript.**
- **Possibili sviluppi futuri riguardano l'inclusione dell'applicazione in un contesto più ampio, in cui è presente un software simile al *Flash Decompiler* in grado di decompilare gli eseguibili delle applicazioni flash, oppure l'utilizzo dell'applicazione come traduttore di linguaggio.**