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

tesi di laurea

Testing di applicazioni flex: uso dello strumento FlexUnit

Anno Accademico 2010/2011

relatore
Ch.mo prof. Porfirio Tramontana

candidato
Gionatan Murru
Matr. 534001578

Problematiche affrontate:

1. Cos'e Flex
2. Come è strutturato
3. Come è possibile testare l'interfaccia di applicazioni Flex
4. Possibilità di uso della tecnica Fault Injection come supporto per il testing

Soluzioni proposte:

- Sviluppo di una particolare applicazione come esempio per realizzare le operazioni di testing
- Proposta dello strumento FlexUnit come supporto del testing per applicazioni Flex
- Uso in FlexUnit della tecnica Fault Injection
- Considerazioni finali

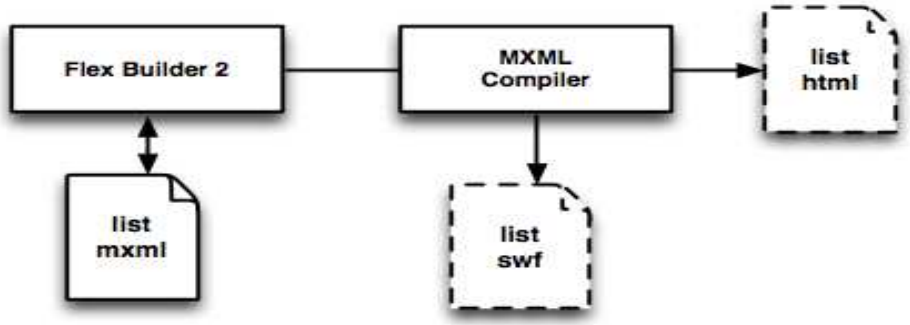
- **Flex è un framework per sviluppare applicazioni interattive e grafiche.**
- **È basato su Flash.**
- **Lo sviluppo prevede la sinergia tra due linguaggi: ActionScript e MXML**
- **L'obiettivo di Flex è quello di permettere un rapido e facile sviluppo di applicazioni Rich Internet Application, meglio conosciute come RIA**

Il linguaggio ActionScript:

- È un linguaggio ad alto livello
- È orientato agli oggetti
- Non è compilato, ma interpretato (o pseudocompilato)
- Si basa su ECMAScript e la sua sintassi è molto simile a JavaScript
- Istanzia un formato di file, chiamato SWF, adatto per applicazioni grafiche e interattive

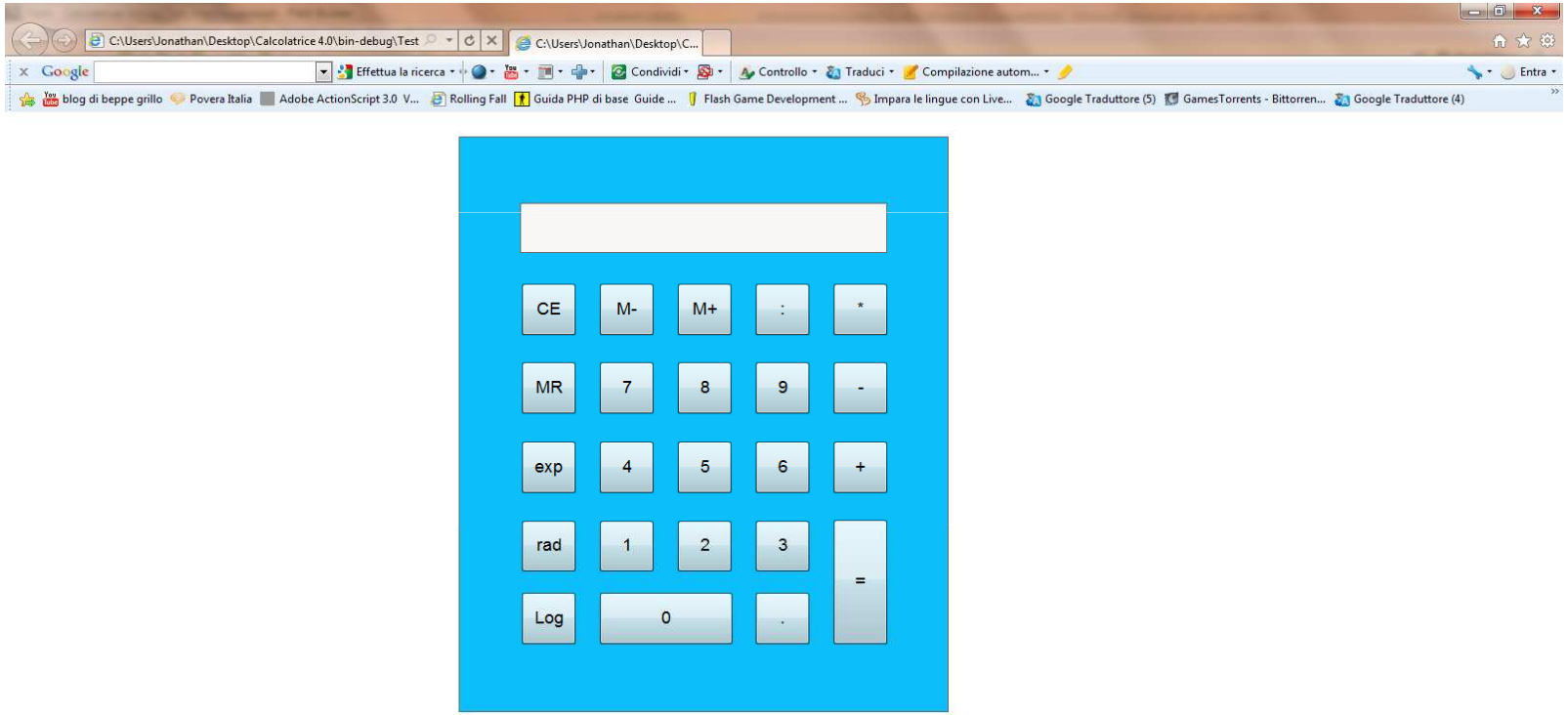
Il linguaggio MXML:

- È un linguaggio proprietario della Macromedia, basato su XML
- È strutturato in tag
- Il suo compito è quello di sviluppare la GUI (Grafical User Interface) della nostra interfaccia.
- Il suo compilatore unisce il file binario MXML con l'SWF, ovvero il file ActionScript già compilato, ed esegue l'applicazione su un browser web



- **La nostra applicazione sviluppata in Flex:**

  Una Calcolatrice scientifica memorizzata su server remoto, in cui solo la parte dell'applicazione che elabora i dati è trasferita a livello client.



Testing di applicazioni Flex

FlexUnit: è un'insieme di librerie open source distribuite dalla Adobe come supporto per il testing di applicazioni Flex.
Esso simula, per le applicazioni Flex, quelle che sono le stesse funzionalità di JUnit3 per le applicazioni Java.

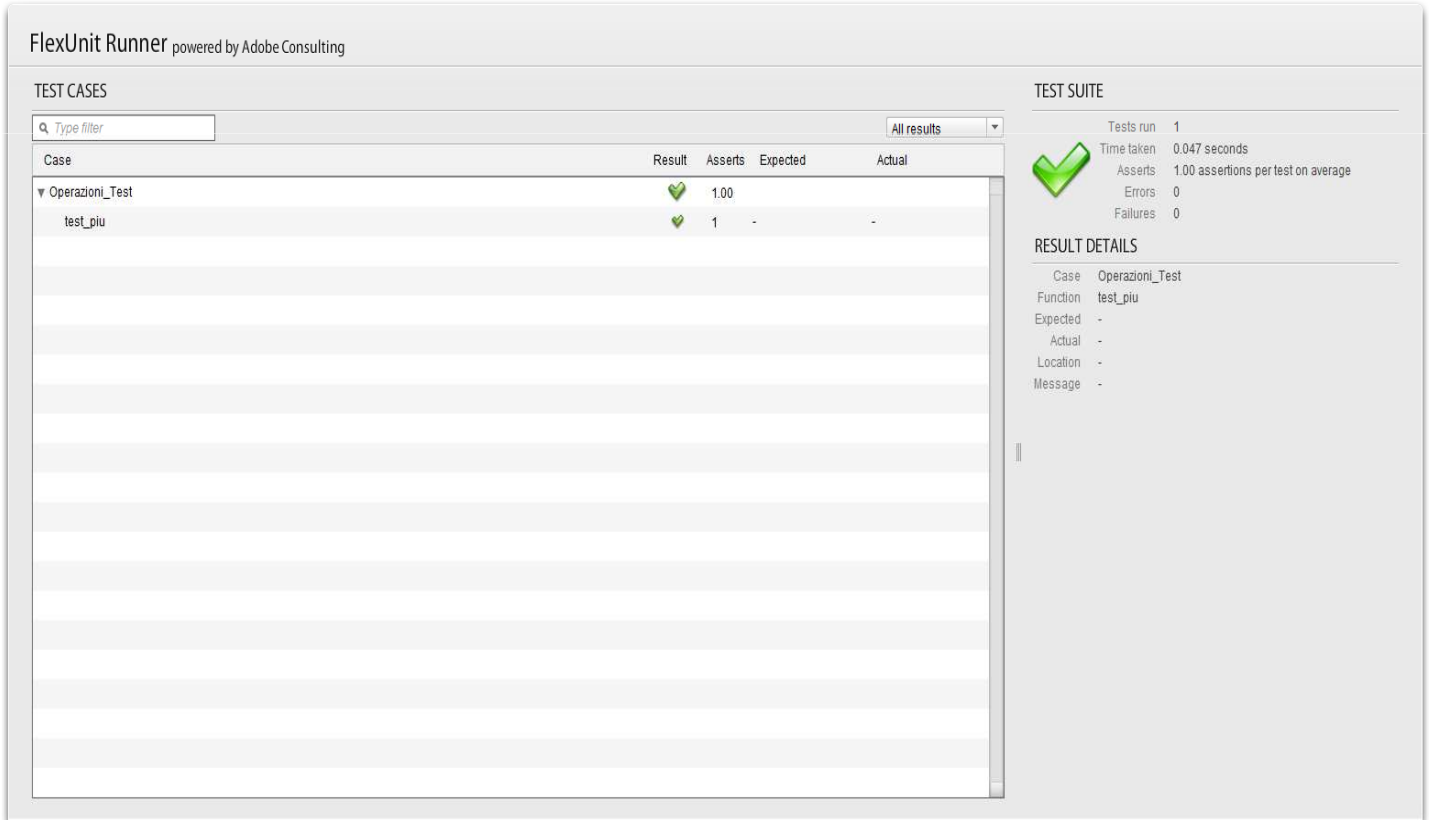
Per testare l'interfaccia con FlexUnit viene scritto del codice che simuli l'uso dell'applicazione. In questo modo si automatizza il testing in modo che l'operazione può essere ripetuta quante volte si voglia e di volta in volta confrontare il risultato ottenuto con quello atteso.

Esempio: vogliamo simulare la pressione dei pulsanti della nostra calcolatrice in modo che svolga l'operazione di somma (3+5).
Il risultato ottenuto verrà poi raffrontato con il valore atteso (8).

FlexUnit(2): Codice e report del risultato del test

```
FlexGlobals.topLevelApplication.tre.dispatchEvent(new MouseEvent(MouseEvent.CLICK));
FlexGlobals.topLevelApplication.p.dispatchEvent(new MouseEvent(MouseEvent.CLICK));
FlexGlobals.topLevelApplication.cinque.dispatchEvent(new MouseEvent(MouseEvent.CLICK));
FlexGlobals.topLevelApplication.u.dispatchEvent(new MouseEvent(MouseEvent.CLICK));

assertEquals(FlexGlobals.topLevelApplication.disp.text, '8');
```



Il test è fallito in quanto non è stato riscontrato nessun errore.
L'icona verde di validazione sta ad indicare che il valore atteso coincide con quello ottenuto in output dal test (8)

Fault Injection in Flex:

Viene iniettato volontariamente un errore nel codice dell'applicazione in modo da osservare il comportamento dell'intero sistema in base a questa singola perturbazione.

Nel nostro caso abbiamo prodotto delle versioni difettate, ognuna contenente un errore.

Sono stati simulati gli errori che più frequentemente possono presentarsi durante la scrittura del codice dell'applicazione

Di seguito si riporta un esempio.

```
public function piu():String{
            if (disp.text!=" "){          //funzione di somma del codice originale
            a=Number(disp.text);          //viene recuperato dal display il numero, il quale viene
                        }                 //poi associato all'operando piu (somma)
            op="piu";                     //viene poi pulito il display per poter inserire il secondo
            disp.text= " ";               //operando e quindi viene svolta l'operazione
                        return op;

            }
```

Lo sostituiamo con:

```
public function piu():String{             //ora se per sbaglio venissero digitati due o più operatori
            disp.text=" ";                //(es. prima 'per' e poi 'diviso') tra gi operandi, il
            a=Number(disp.text);          //risultato non sarebbe corretto secondo le aspettative
            op="piu";
            disp.text= " ";
                        return op;

            }
```

Per verificare il guasto automatizziamo l'operazione (5)(-)(+)(3). L'oracolo ha valore 8 perché per motivi di standardizzazione, viene considerato valido per l'operazione, l'ultimo operando immesso (+ nel nostro caso) tra i due valori (5 e 3). Il risultato ottenuto invece è 3. Analizzando i valori di stato delle variabili si comprende che il risultato ottenuto è 3 perché non imponendo nessun vincolo per quanto riguarda la pressione concatenata di più operatori, l'ultima operazione viene svolta come se fosse la prima (e unica) avendo come primo operatore 0 (di default). Di conseguenza l'applicazione svolge (0)(+)(3)=(3).

Conclusioni

FlexUnit: i pro e i contro

- **È gratuito**
- **È supportato nativamente da Flex**
- **Non è molto complesso da usare**
- **Usa lo stesso linguaggio di Flex (ActionScript)**

- **Utilizza solo criteri black box**
- **Nel testing dell'interfaccia, bisogna incapsulare l'intero codice dell'applicazione nel codice di testing appesantendo la sua esecuzione**