



Tesi di Laurea

Reverse Engineering e testing di interfacce utente applicazioni Flex

Anno Accademico 2010 / 2011

relatore

Ch.mo prof. Porfirio Tramontana

candidato

Fabio Voccia

Matr. 576/93



Problema affrontato

Reverse Engineering ed Automazione del Testing delle interfacce utente di applicazioni Flex di tipo Rich Internet Application.

Obiettivo

Realizzare un'applicazione che sia in grado di effettuare la riflessione e l'automazione del testing delle interfacce utente di applicazioni Flex.

➤ Automazione del testing :

Esplorare le possibili esecuzioni dell'applicazione, individuando eccezioni sincrone non gestite le quali danno luogo a stati di crash dell'applicazione.

➤ Riflessione :

Analizzare a run-time la struttura delle classi ActionScript implementate e le relative istanze. Per ogni istanza individuata visualizzare i valori assunti dagli attributi e richiamare i metodi implementati.



Rich Internet Application - RIA

Le RIA sono applicazioni Web in evoluzione ed hanno le seguenti caratteristiche:

- **Interfaccia utente ricca di contenuti simili alle applicazioni desktop.**
- **Prevalente elaborazione lato client.**

Algoritmo automazione del testing

- **L'obiettivo del testing è quello di scoprire gli errori delle applicazioni dall'osservazione dei malfunzionamenti.**
- **Sappiamo che la correttezza di un programma è un problema indecidibile; infatti la tesi di Dijkstra afferma che il testing, può solo dimostrare la presenza dei difetti ma non la loro assenza.**
- **La strategia adottata per l'algoritmo di testing è quella della visita in profondità di un grafo connesso ed aciclico.**
 - **I nodi rappresentano le interfacce dell'applicazione Flex.**
 - **Gli archi rappresentano gli eventi delle interfacce.**

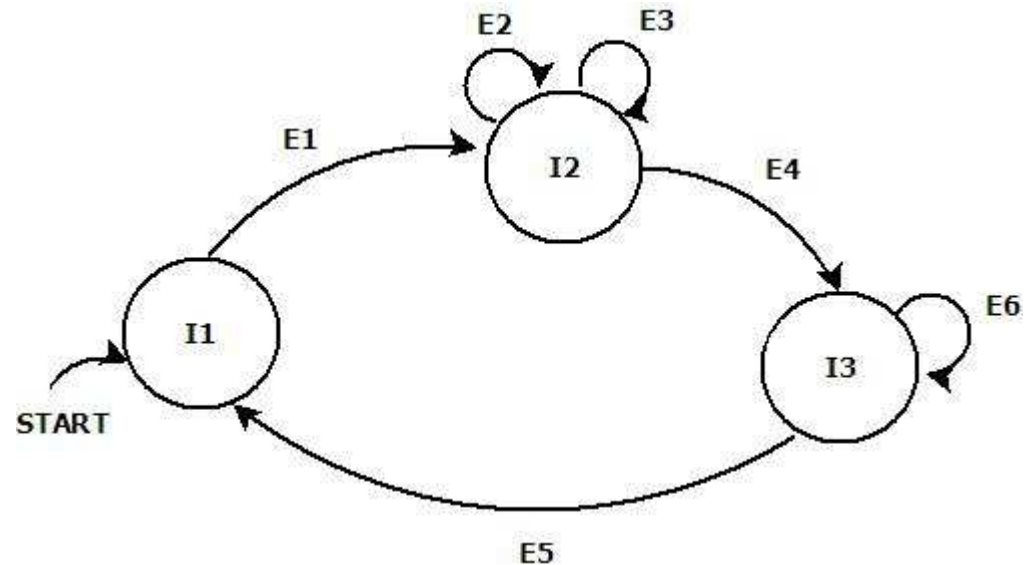


Per la definizione dell'algoritmo di testing è stata applicata la seguente euristica:

- **Un evento dell'interfaccia comprende la valorizzazione di un insieme di componenti di testo i quali possono essere tutti vuoti, tutti pieni, uno solo pieno, uno solo vuoto, e da un evento del componente Button.**
- **La sessione di test termina quando tutti gli eventi delle interfacce individuate sono stati eseguiti.**
- **Due o più interfacce si definiscono equivalenti quando l'identificativo univoco delle interfacce è lo stesso e gli identificativi univoci dei componenti di testo e dei componenti Button delle interfacce sono uguali.**
- **Un caso di test è composto da un insieme finito di eventi delle interfacce navigate. Il caso di test si arresta a fronte di un evento dell'interfaccia che non genera nessun cambiamento di stato dell'applicazione.**

Algoritmo automazione del testing: Visita in profondità del grafo

- La visita in profondità comincia sempre dalla prima interfaccia con il vincolo di terminare la navigazione appena viene incontrata un'interfaccia già nota.



- Esempio di applicazione utente composta da tre interfacce quali I1, I2, I3.
 - Prima interfaccia: I1.
 - Eventi Interfaccia I1: E1.
 - Eventi Interfaccia I2: E2, E3, E4.
 - Eventi Interfaccia I3: E5, E6.

Esempio di navigazione del grafo:

- (Start) E1 → E2 (Stop)
- (Start) E1 → E3 (Stop)
- (Start) E1 → E4 → E5 (Stop)
- (Start) E1 → E4 → E6 (Stop)

Algoritmo per il Reverse engineering: Riflessione

- **L'algoritmo analizza il bytecode relativo all'applicazione SWF caricata e restituisce tutte le classi analizzate.**
Selezionando la generica classe, l'algoritmo mostra la struttura completa della classe ovvero le classi eventualmente estese, membri pubblici e privati, metodi e i valori di default per i parametri opzionali.
Inoltre l'algoritmo analizza tutte le istanze appartenenti al tipo selezionato, e mostra tutti i valori assunti in quell'istante di tempo dagli attributi, richiama i metodi e visualizza i valori restituiti.

Componenti fondamentali

- **Classe ByteCodeType del package AS3Commons-bytecode: Essa analizza il bytecode relativo all'applicazione SWF in test e restituisce la struttura completa della classe ovvero le classi estese, membri pubblici e privati, metodi e i valori di default per i parametri opzionali.**
- **Classe Type del package AS3Commons-reflect: L'istanza di tale classe, consente l'accesso alle costanti, variabili, funzioni di accesso (getter / setter), metodi e attributi definiti dal tipo.**

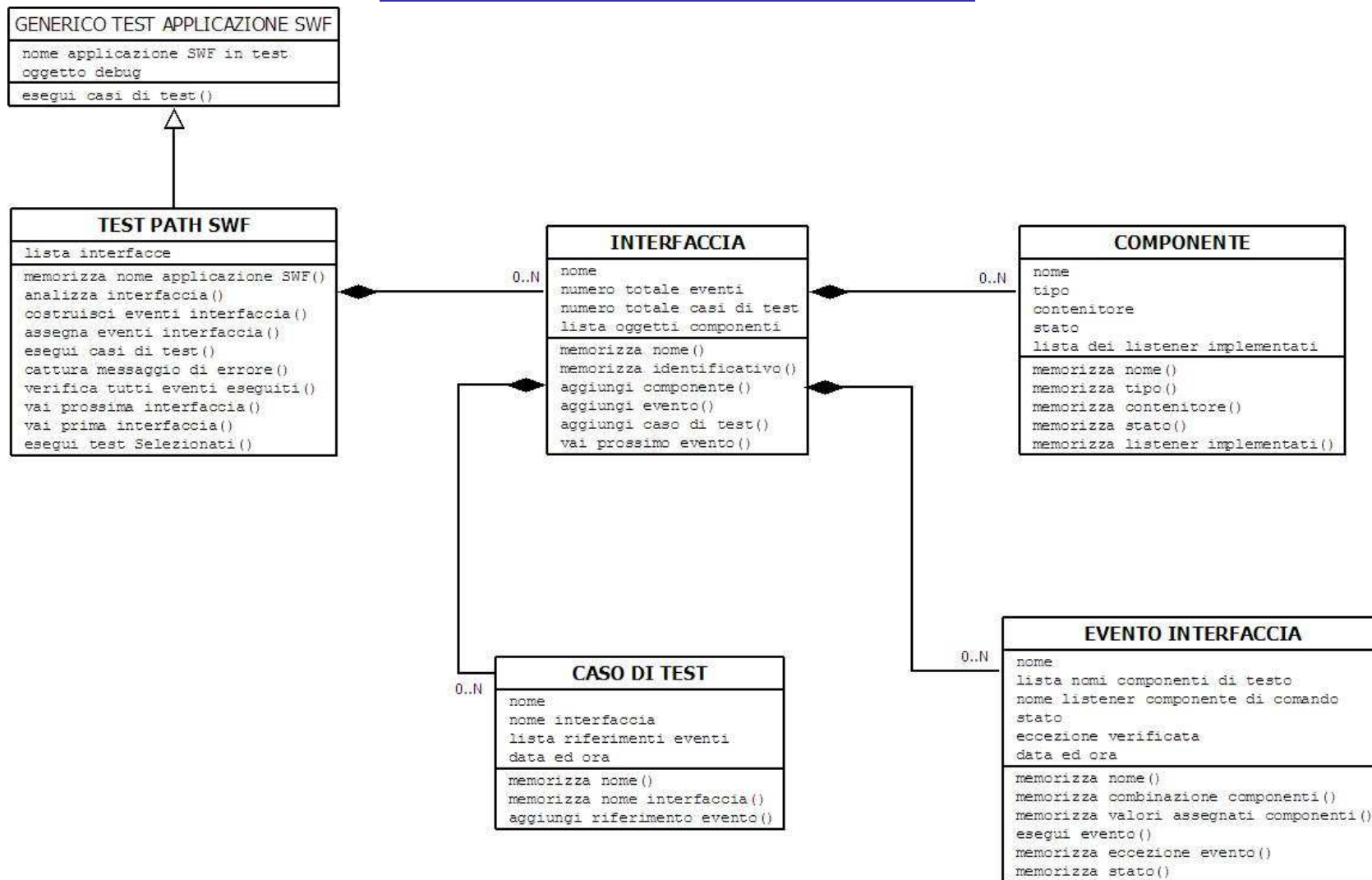
Applicazione SWFAnalyzer

- L'applicazione progettata è stata realizzata in Adobe Flash Builder 4.0 ed è stata chiamata SWFAnalyzer.
- Prima interfaccia: consente il caricamento del file <nomefile>.swf da sottoporre a reverse engineering e testing.
- Seconda interfaccia: Esegue la funzionalità reverse engineering- riflessione .
- Terza interfaccia: Esegue il testing del file <nomefile>.swf caricato.

The screenshot displays the SWFAnalyzer application window. At the top, there are menu options: 'Load SWF file', 'Reflect', and 'SWF Testing'. Below the menu is a table with the following columns: Num, NomeTest, NomeEvento, Parent, ParentInt, Interf Start, Comando, Input, and Value. The table contains six rows of test data. Below the table are buttons for 'Run All Test', 'Run Selected Test', 'Load Test', 'Copy Test', 'DeleteDB', and 'Delete Test'. At the bottom, there are two panels: 'Display operations' and 'SWFAnalyzer debug'. The 'Display operations' panel shows a log of an event execution, including details like 'Evento Numero: 1', 'Interfaccia Start: interf1', 'Nome: Event_interf1_cmdApriPers_click_1', 'Stato: Evento da scatenare', 'Configurazione Evento', 'Comando: cmdApriPers Evento click', 'Input Text', 'Run', 'Input Text', 'Input Text Value', 'Error', 'Eseguito: Evento scatenato OK', and 'Data: 2011-07-25 19:35:09'. The 'SWFAnalyzer debug' panel is currently empty.

Num	NomeTest	NomeEvento	Parent	ParentInt	Interf Start	Comando	Input	Value
1	TestSWF_interf2_1	Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
2	TestSWF_interf2_2	Event_interf2_cmdApriPers_click_1	Event_interf1_cn	interf1	interf2	cmdApriPers Ev		
		Event_interf2_cmdAvanti_click_2	Event_interf1_cn	interf1	interf2	cmdAvanti Event		
		Event_interf3_cmdApriPers_click_1	Event_interf2_cn	interf2	interf3	cmdApriPers Ev		
		Event_interf2_cmdItemPersona_click_3	Event_interf3_cn	interf3	interf2	cmdItemPerson		
3	TestSWF_interf2_3	Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
		Event_interf2_cmdChiudi_click_4	Event_interf1_cn	interf1	interf2	cmdChiudi Ever		
		Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
		Event_interf2_cmdApriPers_click_1_Text_8_5	Event_interf1_cn	interf1	interf2	cmdApriPers Ev	btIato-btCittadir	bt0wDXS4 - C
4	TestSWF_interf2_4	Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
		Event_interf2_cmdApriPers_click_1_Text_7_6	Event_interf1_cn	interf1	interf2	cmdApriPers Ev	btCittadinanza-bx	F9lz - G - aVf
5	TestSWF_interf2_5	Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
		Event_interf2_cmdApriPers_click_1_Text_7_7	Event_interf1_cn	interf1	interf2	cmdApriPers Ev	btIato-btCognon	htad0 - Bchw
6	TestSWF_interf2_6	Event_interf1_cmdApriPers_click_1			interf1	cmdApriPers Ev		
		Event_interf2_cmdApriPers_click_1_Text_7_8	Event_interf1_cn	interf1	interf2	cmdApriPers Ev	btIato-btCittadir	2938 - wBZT

Diagramma delle classi: schema concettuale



Sperimentazione reverse engineering ed automazione del testing

- Si utilizza per l'attività di sperimentazione l'applicazione **RegistraStudiante.swf**.
Essa è composta da quattro interfacce per l'inserimento dati e memorizzazione.

Sperimentazione 1: fase di reverse engineering – Riflessione.

- Il software SWFAnalyzer analizza e visualizza le classi utente implementate nell'applicazione **RegistraStudiante.swf**.
 - Classi analizzate: **Persona.as**, **Studiante.as**
- Il software SWFAnalyzer ricerca per ogni interfaccia dell'applicazione **RegistraStudiante.swf** tutte le istanze della classe **Persona.as** e **Studiante.as**.
Istanze individuate: **objStudiante**
 - Per l'istanza **objStudiante**, visualizza:
 - Il valore assunto dagli attributi.
 - Richiama i metodi e visualizza i valori restituiti.

The screenshot shows the SWFAnalyzer application window. The main area is divided into four panes: Classi, Classi estese, Metodi, and Attributi. The 'Classi' pane lists various classes, with 'Studiante' selected. The 'Classi estese' pane shows 'Persona' as the superclass. The 'Metodi' pane lists methods like 'getCAP()', 'getCitta()', etc. The 'Attributi' pane lists attributes like 'CAP', 'citta', 'cittadinanza', etc. Below the main panes is a 'Reflect' button. At the bottom, there are two smaller panes: 'Display operations' and 'SWFAnalyzer debug'. The 'Display operations' pane shows a log of method invocations and their return values, such as 'Metodo Invocato : public getSesso():String' and 'Valore Restituito : 6m'.

Classi	Classi estese	Metodi	Attributi
spark.skins.SparkSkin	Persona	public getCAP():String	CAP : String
spark.utils.FTETextUtil		public getCitta():String	citta : String
spark.utils.MaskUtil		public getCittadinanza():String	cittadinanza : String
spark.utils.TextUtil		public getCodiceFiscale():String	codicefiscale : String
SparkDownloadProgressBar.as\$6948.Error		public getCognome():String	cognome : String
Studiante		public getIdata(Nascita):String	data(Nascita) : String
StyleManagerImpl.as\$815.StyleEventDisp		public getFacolta():String	facolta : String
StyleManagerImpl.as\$815.StyleModuleInf		public getMatricola():String	indirizzo : String
SWFToTest		public getIato():String	matricola : String
TextClipboard.as\$238.TextClipboardSingl		public getNazione():String	nato : String
TextContainerManager.as\$186.Remappe		public getNome():String	nazione : String
TextContainerManager.as\$186.TMContai		public getProvincia():String	nome : String
TextFlow.as\$24.HostFormatHelper		public getProvinciaRes():String	provincia : String
TextFlowLine.as\$159.SelectionCache		public getSesso():String	provinciaRes : String
TextLineFactoryBase.as\$181.FactoryBack		public getTotEsami():String	seesso : String
TextLineFactoryBase.as\$181.FactoryDisp		public getVial():String	totesami : String

```

Display operations
-- Metodo Invocato : public getSesso():String
-- Valore Restituito : 6m
- Metodo Invocato : public getCittadinanza():String
-- Valore Restituito :
- Metodo Invocato : public getNato():String
-- Valore Restituito : nREKWT
- Metodo Invocato : public getProvincia():String
-- Valore Restituito : UunO1bm7
- Metodo Invocato : public getCodiceFiscale():String
-- Valore Restituito : L
- Metodo Invocato : public getNazione():String
-- Valore Restituito :
- Metodo Invocato : public getProvinciaRes():String
-- Valore Restituito :
    
```

Sperimentazione reverse engineering ed automazione del testing

Sperimentazione 1: fase di Testing

- **Fault injection:**
È stato inserito un fault nell'evento click del Button cmdMemPersona dell'interfaccia Dati Personali.
- Istruzione prima inserimento fault:
`objStudiante.setCognome(txtCognome.text);`
- Istruzione dopo inserimento fault:
`objStudiante.setCognome();`
- Numero di eventi da testare per l'evento click del componente Button cmdMemPersona:

 $e = 2 * n + 2$, n = numero delle caselle di testo.
 $e = 2 * 8 + 2 = 18$ eventi.

Registrazione Studente

Dati Personali

Nome

Cognome

Sesso

Nato/a

Provincia

Data Nascita

Cittadinanza

Codice Fiscale

- **Risultati della sessione di test:**
- Numero totale eventi generati:
299 eventi
 - Numero di malfunzionamenti rilevati:
18 malfunzionamenti
 - Tempo totale impiegato:
53[s]

Conclusioni

- **L'attività di sperimentazione ha dimostrato che il processo di testing definito e il software realizzato quale SWFAnalyzer, sono stati efficaci per l'analisi dinamica della struttura dell'applicazione e per l'individuazione e la cattura degli stati di crash dell'applicazione.**

Sviluppi Futuri

- **Nella fase di analisi della struttura dell'interfaccia, considerare componenti utili al test oltre ai componenti "TextInput" e "Button" altri componenti standard, per esempio "List", "ComboBox", "CheckBox".**
- **Progettare nuove funzionalità per il processo di Riflessione:**
 - **Funzionalità Proxy utile per intercettare le chiamate ai membri degli oggetti individuati dal processo di introspezione. Grazie a tale funzionalità sarà possibile analizzare la collaborazione che avviene tra gli oggetti al fine di svolgere un'attività.**