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

tesi di laurea Ingegneria del Software

Sperimentazione di processi di testing automatico di applicazioni AJAX

Anno Accademico 2010/2011

relatore
Ing. prof. Porfirio Tramontana

correlatore
Ing. Domenico Amalfitano

candidato
Carlo Scherma
Matr. 534/3097

Contesto

Testing di Rich Internet Applications (RIA) basate su AJAX

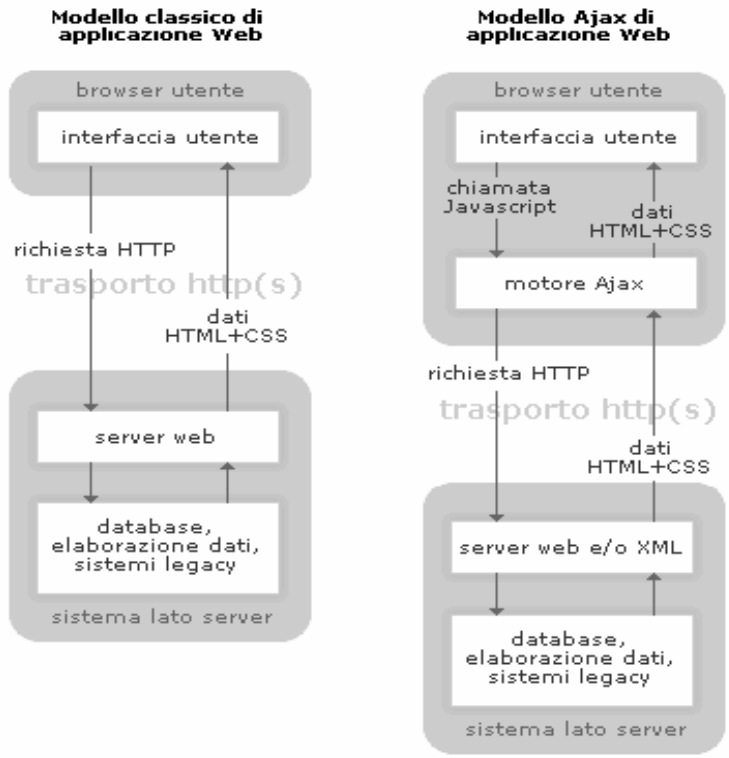
Obiettivo

- **Provare un processo di testing per Rich Internet Applications che preveda l'utilizzo di alcuni tools sviluppati nel dipartimento di Informatica e Sistemistica dell'università di Napoli.**
- **Verificare l'efficacia del processo proposto e dei tools utilizzati attraverso esperimenti su una applicazione nota nella quale sono stati iniettati opportuni difetti.**

Rich Internet Applications (RIA)

Le RIA sono un modello emergente di applicazione Web con le seguenti caratteristiche:

- Prevalente elaborazione sul lato client;
- Interfacce utente (GUI) più complesse e dinamiche rispetto a quelle delle applicazioni Web tradizionali;
- Ricca interazione dell'utente con la GUI;
- Un motore Ajax, scritto in Javascript, tra l'interfaccia utente ed il server che è responsabile:
  - ➢ della comunicazione sincrona e/o ascincrona con il web server;
  - ➢ della gestione dell'interfaccia grafica.



Tecniche di Testing di RIA

Le caratteristiche delle RIA fanno si che gli approcci applicati al testing di applicazioni Web tradizionali non sono adatti a testare efficacemente le RIA;

- il codice dell'Ajax Engine può variare dinamicamente.

Le tecniche di testing per RIA possono essere classificate in base alle seguenti categorie:

- Obiettivo del Testing;
- Tecnica utilizzata per generare i casi di test;
- Oracolo;
- Tipi di strumenti a supporto del processo di testing.

Tecnica di Testing proposta

• Generazione di casi di test a partire da esecuzioni reali dell'applicazione raccolte mediante il tool CreRia e corrispondenti alle esecuzioni degli scenari dell'applicazione sotto diverse possibili precondizioni.

• Generazione automatica di una FSM.

• Minimizzazione delle tracce raccolte mediante il tool TestRia:

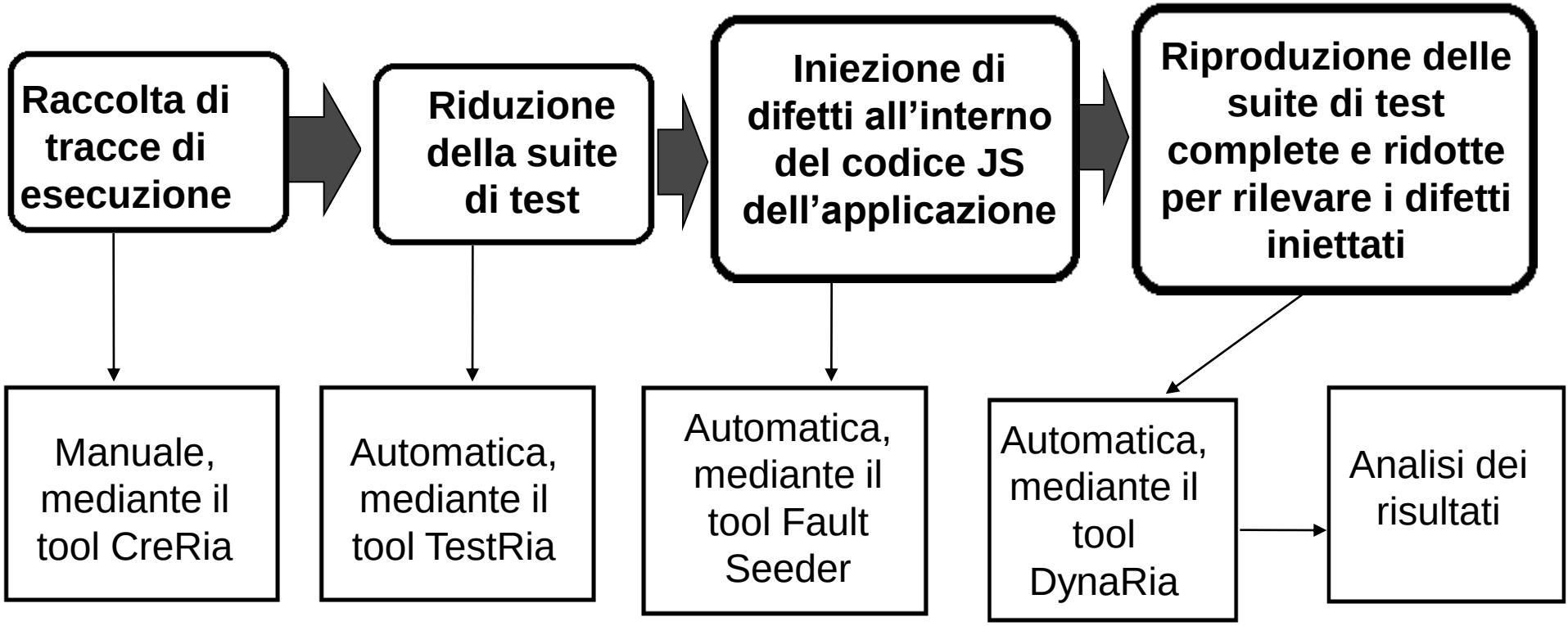
- Copertura degli stati;
- Copertura delle transizioni;
- Estrazione di una suite di test in grado di coprire gli stati o le transizioni esplorate durante la raccolta delle tracce.

• Iniezione di diversi difetti nella nostra applicazione sottoposta a test;

• Valutazione di :

- L'efficacia dei casi di test di individuare malfunzionamenti (crash dell'applicazione) causati da difetti iniettati, rieseguendo la suite di test;
- L'efficienza in termini di tempo impiegato per rieseguire la suite di test.

Processo Sperimentale



Tool utilizzati

I tools utilizzati per supportare il processo di testing proposto sono :

- **CreRia**, per la raccolta, l'analisi e la classificazione di tracce di esecuzione. Supporta anche l'attività di astrazione e validazione dell' FSM risultante;
- **Fault Seeder**, per l'iniezione automatica di difetti all'interno del codice javascript in modo da generare versioni difettate dell'applicazione sotto test;
- **DynaRia,** per la riesecuzione dei casi di test e valutazione dei risultati;
- **TestRia,** per la riduzione della suite di test.

Fault Seeding

Per valutare l'efficacia e la capacità di individuare i difetti è necessario avere a disposizione il codice sorgente con difetti reali conosciuti.

Problema: non sempre sono disponibili o conosciuti per un'applicazione i relativi difetti reali.

Soluzione: iniettare fault javascript nell' Ajax Engine, all'interno del codice dell'applicazione sotto test, in modo da creare versioni difettate dell'applicazione di cui conosciamo i difetti che cercheremo poi di rilevare attraverso il processo descritto.

- Sono state create 175 versioni difettate ognuna contenente un solo difetto e, si è cercato di simulare quegli errori che più frequentemente si verificano durante la scrittura del codice JavaScript come ad esempio operatore booleano errato, operatore di confronto errato, riferimento ad un oggetto null ecc.

Risultati Sperimentali

	Test Suite	
	Numero di eventi	Errori rilevati da ogni traccia
Traccia 1_8_1	21	140/175
Traccia 1_10_1	24	169/175
Traccia 1_11_1	33	164/175
Traccia 3_6_6	20	8/175
Traccia 3_10_1	25	129/175
Traccia 3_11_1	25	21/175
Traccia 4_6_6	30	130/175
Traccia 4_10_1	13	87/175
Traccia 4_11_1	24	71/175
Traccia 5_6_6	19	53/175
Traccia 6_6_6	19	92/175
Traccia 7_6_6	16	51/175
Traccia 8_6_6	24	61/175
Traccia 9_10_1	30	11/175
Traccia 10_10_1	21	63/175
Traccia 10_11_1	31	64/175
Traccia 14_11_1	29	38/175
Traccia 15_11_1	29	163/175
Traccia 16_11_1	26	170/175
Traccia 17_11_1	28	162/175
Media eventi	24,35	
Tempo Medio	21,49	

La nostra suite di test:

- è composta da 20 tracce di esecuzione e ha rilevato tutti i difetti iniettati nel codice javascript dell'applicazione sotto test.
- le tracce sono composte in media da 24 eventi.
- Sommando i tempi medi di esecuzione di ogni traccia la suite di test richiede un tempo di esecuzione in media di 21 minuti per ciascun difetto iniettato nell'applicazione.

Risultati Sperimentali

	Riduzione stati			Riduzione transizioni							
	Traccia 1_11_1	Traccia 7_6_6	Traccia 14_11_1	Traccia 1_8_1	Traccia 1_10_1	Traccia 3_11_1	Traccia 4_6_6	Traccia 9_10_1	Traccia 10_10_1	Traccia 1_11_1	Traccia 6_6_6
errori rilevati da ogni traccia	164/175	51/175	38/150	140/175	169/175	21/175	130/175	11/175	63/175	164/175	92/175
Tempo medio	3 minuti			7 minuti							

Riduzione della test suite completa e generazione di due test suite tramite TestRia:

- si riducono il numero totale di tracce che passano da 20 a 3 (riduzione degli stati) oppure a 8 (riduzione delle transizioni) ;
- si riducono i tempi di esecuzione;
- le test suite ridotte sono in grado di rilevare tutti i 175 difetti iniettati

Conclusioni e sviluppi futuri

Conclusioni: La sperimentazione ha dimostrato che il processo di testing proposto è sicuramente efficace e valido e i tool utilizzati hanno dato risultati più che soddisfacenti.

Sviluppi Futuri:

- Automatizzare il processo di generazione dei casi di test nella fase di raccolta delle tracce di esecuzione cercando di non peggiorare sia l'efficacia che l'efficienza riducendo però i tempi di progettazione e scrittura della test suite.
- Aumentare il numero di esperimenti su applicazioni AJAX differenti in modo da validare sia il processo di testing proposto sia il corretto funzionamento dei tool.
- Ampliare le funzionalità dei tool utilizzati.