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

Tesi di laurea

Un analizzatore dinamico per il Reverse Engineering di Rich Internet Applications implementate con tecnologia Ajax a supporto di processi di comprensione e testing

Anno Accademico: 2008/2009

relatore
Ch.ma Prof.ssa
Anna Rita Fasolino

correlatore
Ing. Domenico Amalfitano

candidato
Armando Polcaro
Matr. 41 / 2858

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

Scopo del lavoro di tesi

- **Contesto**: Reverse Engineering di Rich Internet Applications (RIA)

- **Scopo della tesi**: proporre processi e strumenti di Reverse Engineering a supporto di comprensione e testing di RIA

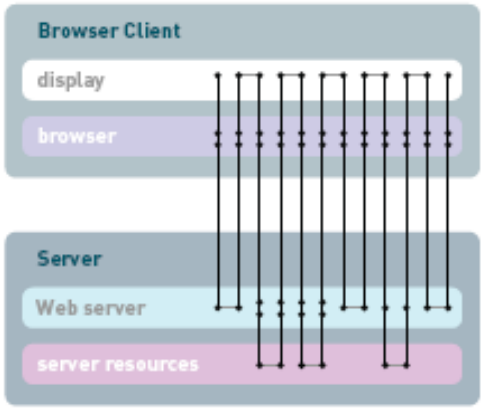
- **Contributo**: proposte di modelli per la descrizione del comportamento di una RIA e di uno strumento per automatizzare l'analisi dinamica di RIA Ajax-based

Rich Internet Applications (RIA)

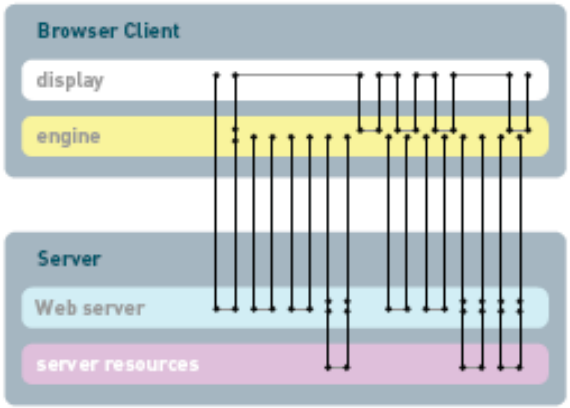
- Elevata interattività con l'utente finale
- Elimina la natura click - and - wait
- Introducono un layer logico detto ***Client-Side engine***
  - Responsabile di quasi tutta l'elaborazione eseguita sul client
  - Questo motore può avere implementazioni differenti
  - Permette di effettuare richieste asincrone, non bloccanti
  - Permette lo scambio di piccole quantità di dati con il server per l'aggiornamento delle interfacce utente.

- Interfaccia modulare le cui parti cambiano dinamicamente e in maniera indipendente
- Caratteristiche e funzionalità simili alle applicazioni desktop

- Tecnologie realizzative:
  - Script based
    - Ajax
  - Plug-in based
    - Adobe Flex
    - Microsoft Silverlight



Modello di comunicazione di una Web application tradizionale

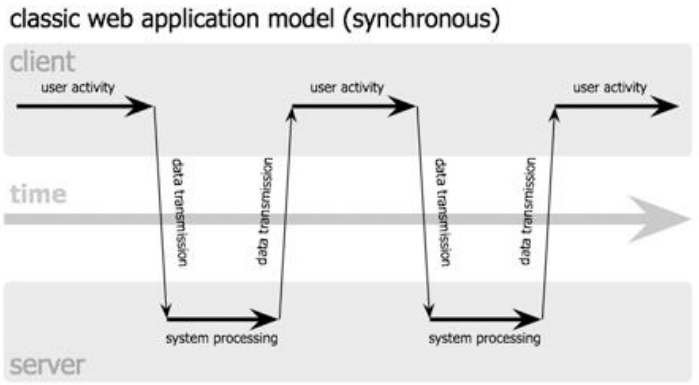


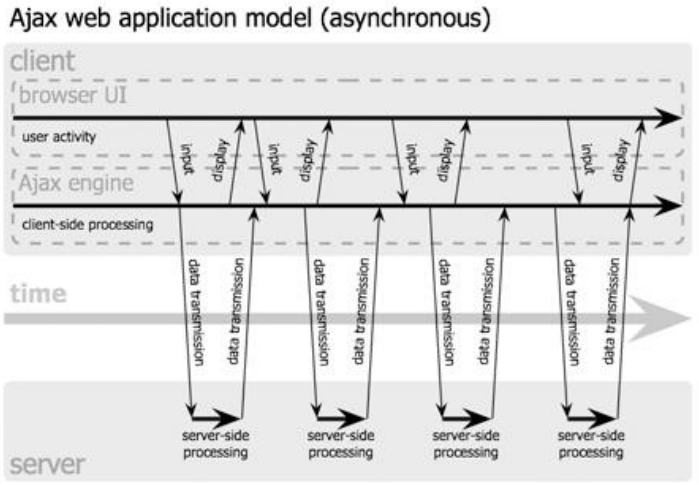
Modello di comunicazione di una RIA

Ajax

Asyncronous Javascript and XML

- **Un insieme di tecnologie:**
  - XHTML e CSS per la presentazione
  - DOM (Document Object Model) aggiornato in modo dinamico
  - Scambio di dati tramite XML oppure HTML preformattato
  - Un'istanza della classe *XMLHttpRequest*, che consente al browser di dialogare in modo asincrono con il server
  - Javascript, che gestisce il tutto





- Il *client – engine* delle RIA basate su Ajax è l'***Ajax Engine***
  - Composto da moduli di codice Javascript

Problemi di comprensione di RIAs

- **Caratteristiche delle RIAs:**
  - Utilizzano tecnologie eterogenee con conseguente aumento della complessità
  - Implementate utilizzando *frameworks* che ne facilitano lo sviluppo, ma ne aumentano la complessità della logica interna
  - Evolvono dinamicamente a run-time
  - Sviluppate in breve tempo e/o con risorse insufficienti
  - Sviluppate senza prendere in considerazione i principi dell'ingegneria del software
  - Documentazione insufficiente o inesistente.

- **Conseguenze:**
  - Difficoltà di analisi
  - Difficoltà di comprensione
  - Difficoltà di manutenzione
  - Difficoltà di testing

Strumenti per la comprensione di RIAs

- Esistono alcuni strumenti per l'analisi delle RIAs
  - Firebug
  - Venkman
  - DynaTrace Ajax Edition
- Utilizzati durante lo sviluppo offrono funzionalità di:
  - Debugging del codice Javascript
  - Ispezione del DOM
  - Monitoraggio della comunicazione tra client e server
- Effettuano l'analisi dinamica della sessione utente

- **Problemi**:
  - Offrono viste spesso non correlate fra di loro
  - Non offrono meccanismi di astrazione
  - Non sono utili a supporto di processi di comprensione

- **Obiettivo**:
  - Introdurre uno strumento per ottenere in automatico modelli e viste a diversi livelli di astrazione utili a comprendere il funzionamento di una RIA

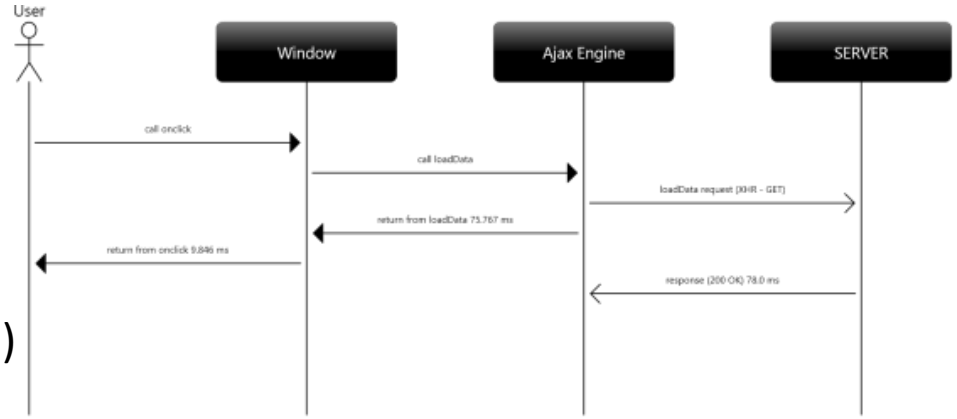
Diagrammi di sequenza

Un primo modello è un diagramma di sequenza a tre livelli che si rifà al modello comunicazionale di Ajax visto in precedenza

- **Attore**: utente che scatena eventi sull'interfaccia della RIA

- **Le entità coinvolte nell'interazione sono:**
  - **Window:** rappresenta l'istanza del DOM all'interno del Browser Web
  - **Ajax Engine:** rappresenta la business logic della RIA
  - **Server:** responsabile dell'elaborazione server-side e della comunicazione con il client

- **Messaggi scambiati:**
  - Chiamata di funzione Javascript
  - Modifica dell'interfaccia in termini di inserimento, rimozione e modifica attributi di nodi del DOM
  - Richieste al Server (con relative risposte)

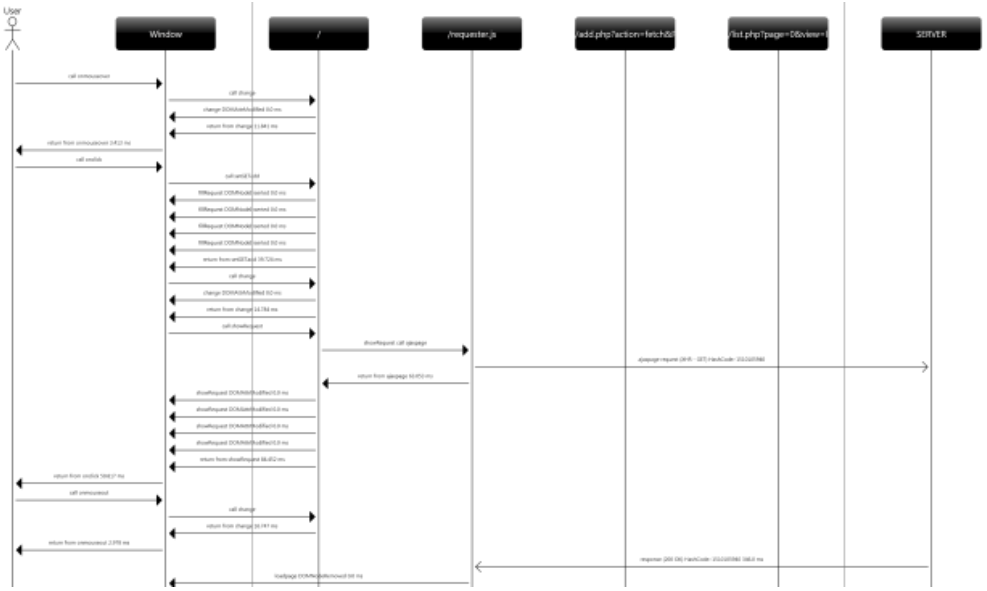


Diagrammi di sequenza - 2

Un secondo modello è un diagramma di sequenza che mostra ad un maggior livello di dettaglio le interazioni che avvengono internamente all'Ajax Engine tra i moduli Javascript che lo compongono

- **Attore**: utente che scatena eventi sull'interfaccia della RIA

- **Le entità coinvolte nell'interazione sono:**
  - **Window**
  - **Moduli Javascript interni all'Ajax Engine**
  - **Server**



- **Messaggi scambiati:**
  - Chiamata di funzione Javascript all'interno dell'Ajax Engine
  - Modifiche all'interfaccia, in termini di inserimento, rimozione e modifica di attributi di nodi del DOM, da parte di moduli Javascript
  - Richieste al Server (con relative risposte) effettuate dai moduli Javascript

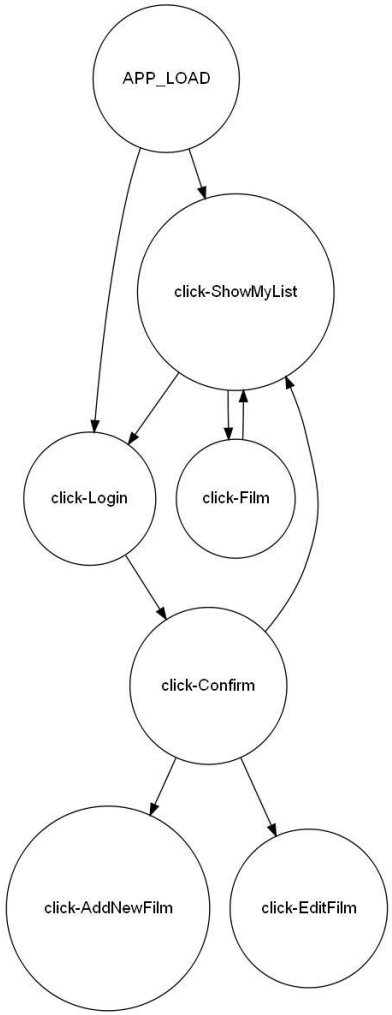
Event Flow Graph

- **Altro modello proposto è un Grafo del Flusso degli Eventi (EFG)**
  - Esso modella l'interazione dell'utente con l'interfaccia grafica (GUI) della RIA in termini di eventi utente scatenati sulla sua interfaccia

- **È un grafo in cui:**
  - **Nodi:** rappresentano gli eventi scatenati dall'utente
  - **Archi:** rappresentano l'ordine e la dipendenza tra eventi

- **Mostra un flusso ordinato di eventi**

- **Esempio**:
  un arco tra il nodo "click-*Login"* ed il nodo "*click-Confirm*" sta ad indicare che l'evento "*click-Confirm*" può essere eseguito solo successivamente all'evento "click-*Login"* .



Viste Proposte

Per comprendere nel dettaglio il funzionamento dell'Ajax Engine sono state introdotte diverse viste ad un più basso livello di astrazione che rappresentano:

- Interazione tra eventi e funzioni Javascript eseguite
- Interazione tra eventi e richieste effettuate al Server
- Albero delle chiamate Javascript creato nella gestione dell'evento
- Interazione tra funzioni Javascript ed il DOM
- Vista sulle modifiche fatte al DOM (nodi inseriti, rimossi o modificati)
- Interazione tra funzioni Javascript e richieste al Server
- Vista sul codice Javascript eseguito (LOC)
- Vista sugli errori Javascript rilevati (LOC)
- Viste sulla copertura di funzioni Javascript e LOC
  - Caricate nell'Ajax Engine
  - Eseguite nell'Ajax Engine

dynaRIA Tool

- **Tool software capace di supportare sia processi di comprensione che di testing di RIA Ajax-based**

  Link per il download: http://wpage.unina.it/ptramont/downloads.htm

- **Supporto al processo di comprensione**

  - **Fornisce un ambiente in cui è possibile navigare all'interno dell'interfaccia di una RIA e durante l'esecuzione raccogliere dinamicamente:**
    - Eventi scatenati sugli oggetti dell'interfaccia della RIA
      - Permette di etichettare in maniera automatica gli eventi scatenati
      - Permette di etichettare manualmente una sequenza di eventi per associare ad essa un particolare significato
    - Funzioni Javascript eseguite nella gestione dell'evento scatenato
      - LOC eseguite
      - Errori eseguiti
    - Le modifiche avvenute all'interfaccia della RIA (inserimento, rimozione e modifica di nodi del DOM)
    - Richieste al server con relative risposte

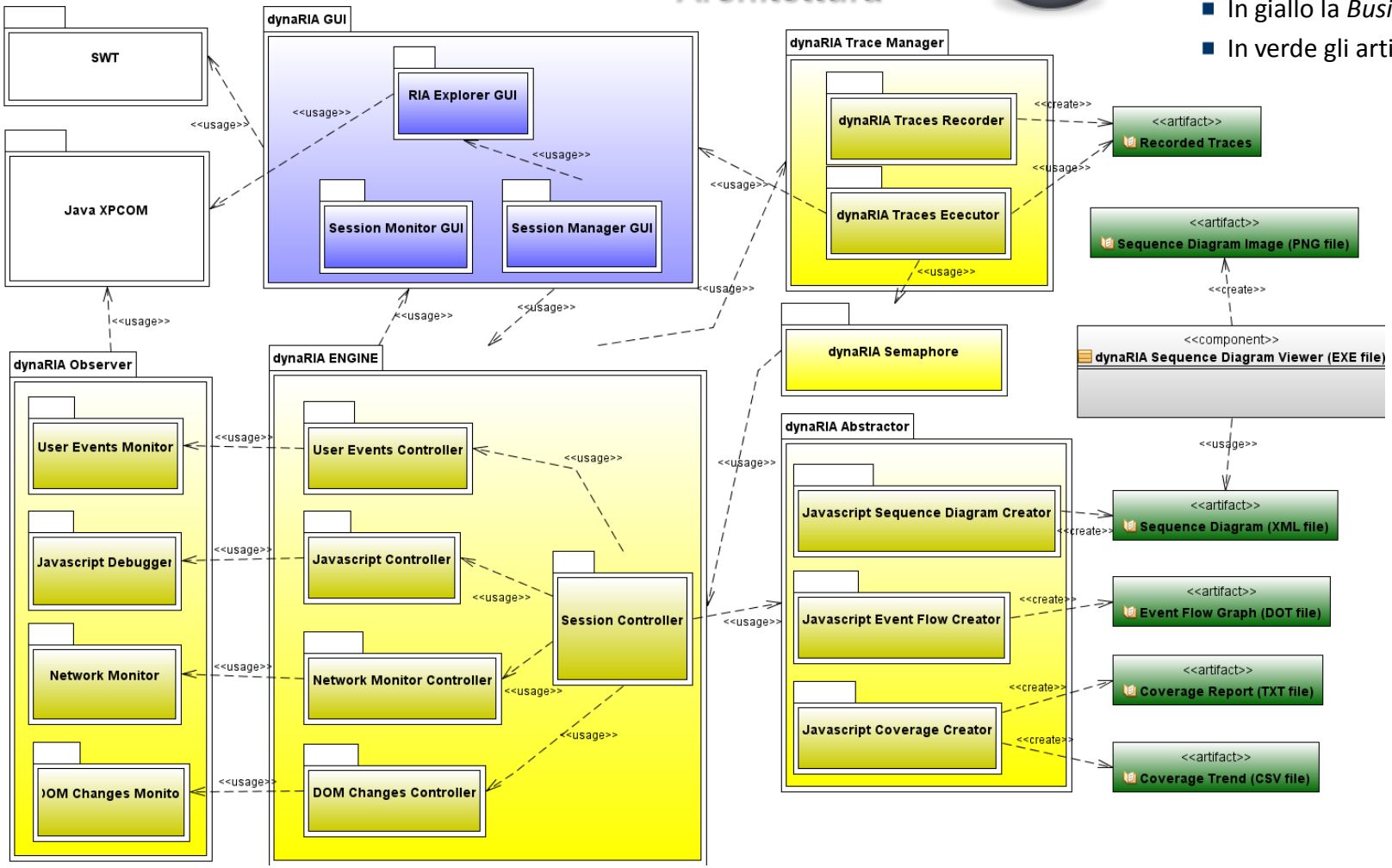
  - **Effettua astrazione sui dati raccolti per ottenere i modelli e le viste presentati in precedenza**

dynaRIA Tool

- **Supporto al processo di testing**
  - **Registrazione di una traccia di esecuzione**
    - Registrazione di eventi scatenati sulla GUI della RIA
  - **Riesecuzione ed analisi di tracce registrate**
    - Eventuale compilazione di campi di INPUT
    - Scatenamento dell'evento
    - Verifica che non ci siano stati errori Javascript
    - Verifica che le eventuali richieste al server abbiamo avuto una risposta regolare
  - **Produce varie metriche di copertura relative alle tracce eseguite**:
    - Il numero di funzioni Javascript eseguite sul totale dell'Ajax Engine
    - Il numero totale di LOC di funzioni Javascript eseguite sulle LOC totali dell'Ajax Engine
    - Il numero di funzioni eseguite per ogni modulo Javascript dell'Ajax Engine
    - Il numero di LOC eseguite per ogni modulo Javascript dell'Ajax Engine
    - Il numero di LOC eseguite per ogni funzione dell'Ajax Engine

dynaRIA Tool
Architettura

- In azzurro la *GUI*
- In giallo la *Business Logic*
- In verde gli artifatti

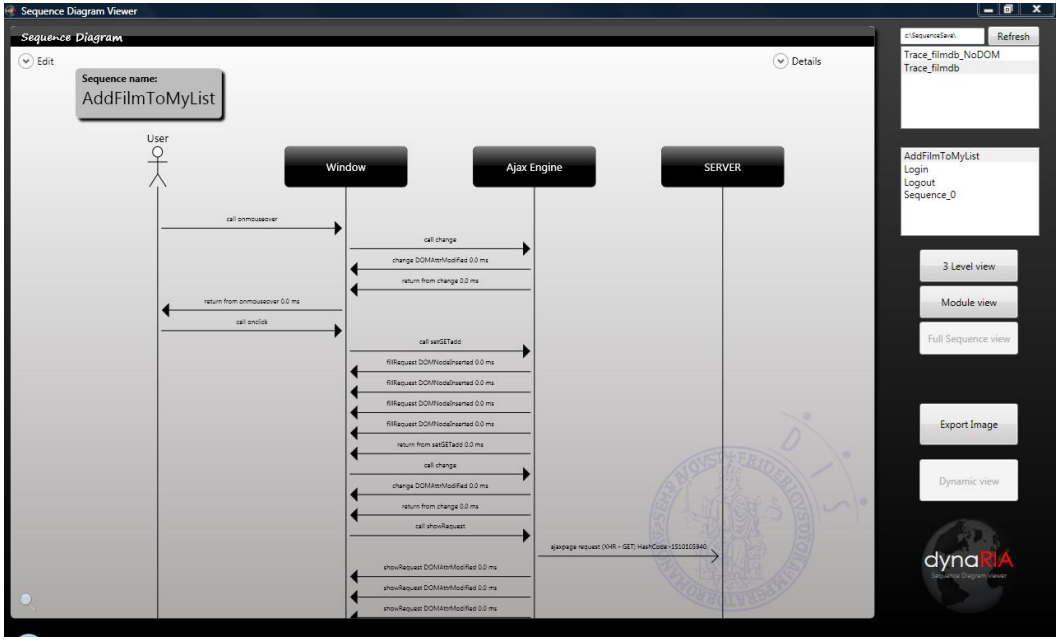


- **Tecnologie utilizzate:**
  - Java
  - Libreria SWT
  - Libreria JavaXPCOM
- **Ambiente di sviluppo utilizzato:**
  - NetBeans IDE 6.7

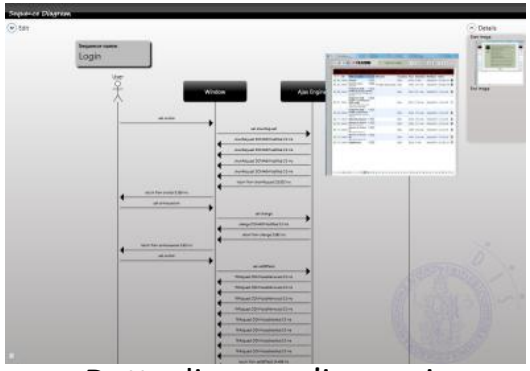
dynaRIA Sequence Diagram Viewer Tool

Tool software realizzato per visualizzare i diagrammi di sequenza prodotti dal *tool dynaRIA*

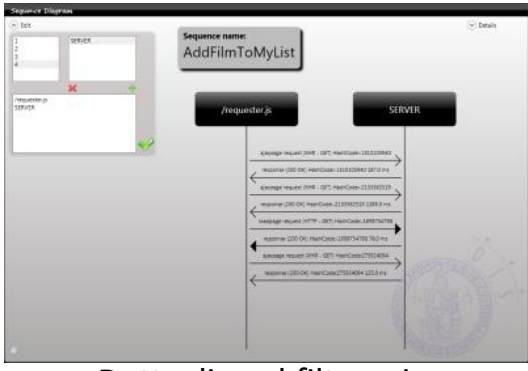
- **Funzionalità**:
  - Visualizzazione dei diagrammi di sequenza a tre livelli o a livello dei moduli javascript dell'Ajax Engine
  - Possibilità di filtraggio e zoom per ingrandire una zona del diagramma
  - Possibilità di visualizzazione delle immagini dell'interfaccia grafica della RIA all'inizio ed alla fine dell'elaborazione rappresentata in sequenza



Sviluppato in Visual Basic .NET 2008 sfruttando le potenzialità della libreria WPF (Windows Presentation Foundation)



Dettaglio su un'immagine dell'interfaccia della RIA



Dettaglio sul filtraggio

Esperimenti

- **GOAL:** valutare l'efficacia e l'utilità del *tool dynaRIA* nel supportare processi di comprensione e testing attraverso l'uso di un framework di valutazione
  - Il framework definisce un insieme di task tipici di analisi e comprensione
  - Si eseguiranno tali task attraverso l'uso di *dynaRIA*

- **Primo caso di studio**
  - Comprendere diversi aspetti di una funzionalità di una RIA usando *dynaRIA*

- **Secondo caso di studio**
  - Utilizzare le capacità di analisi di *dynaRIA* nell'ambito di un processo di testing

Primo caso di studio

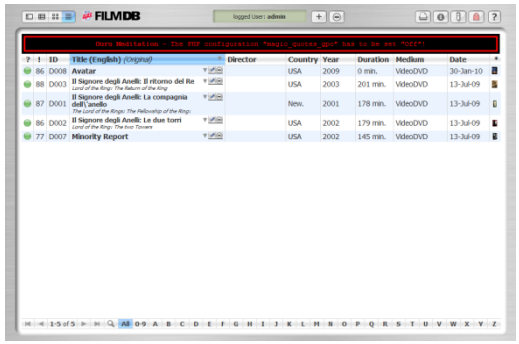
Comprensione

- **Obiettivo: conoscere come è implementata una funzionalità offerta da una RIA esistente**
  - **Task:**
    - **Comprendere le interazioni tra le entità che collaborano nello svolgimento della funzionalità**
      - **Alto livello (Window – Ajax Engine – Server)**
      - **Basso livello (Window – Moduli interni Ajax Engine – Server)**
    - **Comprendere il flusso dell'elaborazione Javascript necessario ad implementare la funzionalità**
      - **Albero delle chiamate Javascript creato nella gestione dell'evento**
    - **Comprendere le modifiche effettuate all'interfaccia durante l'esecuzione della funzionalità**
      - **Interazione tra funzioni Javascript ed il DOM (inserimento, cancellazione o modifica di attributi di un nodo)**
    - **Comprendere l'intervento del server durante l'esecuzione della funzionalità**
      - **Interazione tra funzioni Javascript e il Server**

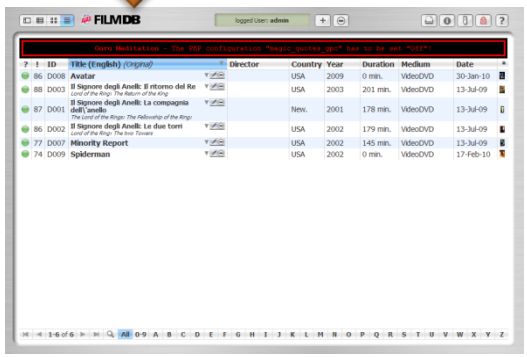
Primo caso di studio - 2

Comprensione

- **RIA: AJAX-FilmDB**
- **Funzionalità analizzata:**
  - Aggiunta Film (*AddFilmToMyList*)

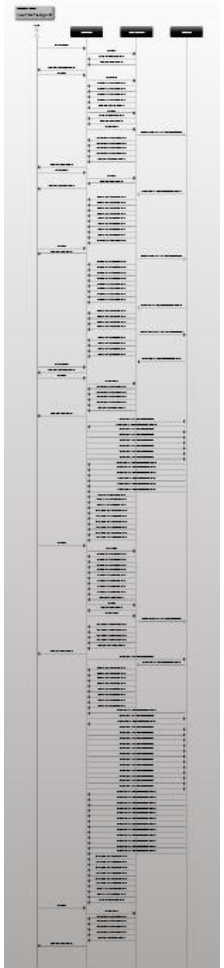


Elaborazione

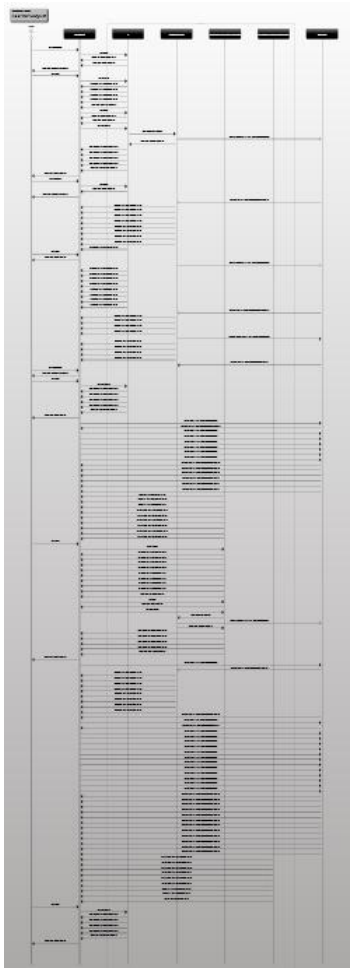


Interazioni tra le entità che collaborano nello svolgimento della funzionalità "*addFilmToMyList*"

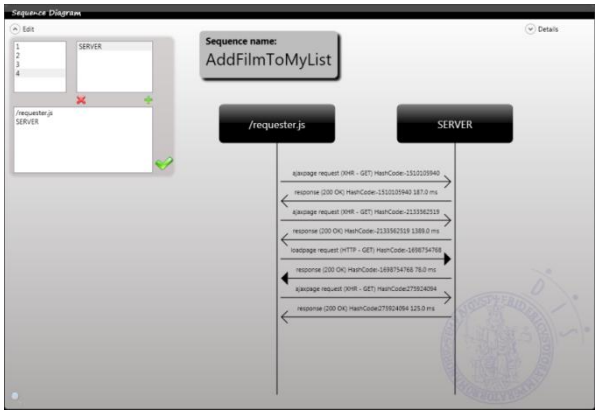
Alto Livello



Basso Livello



Dettaglio della comunicazione tra un modulo *requester.js* ed il Server

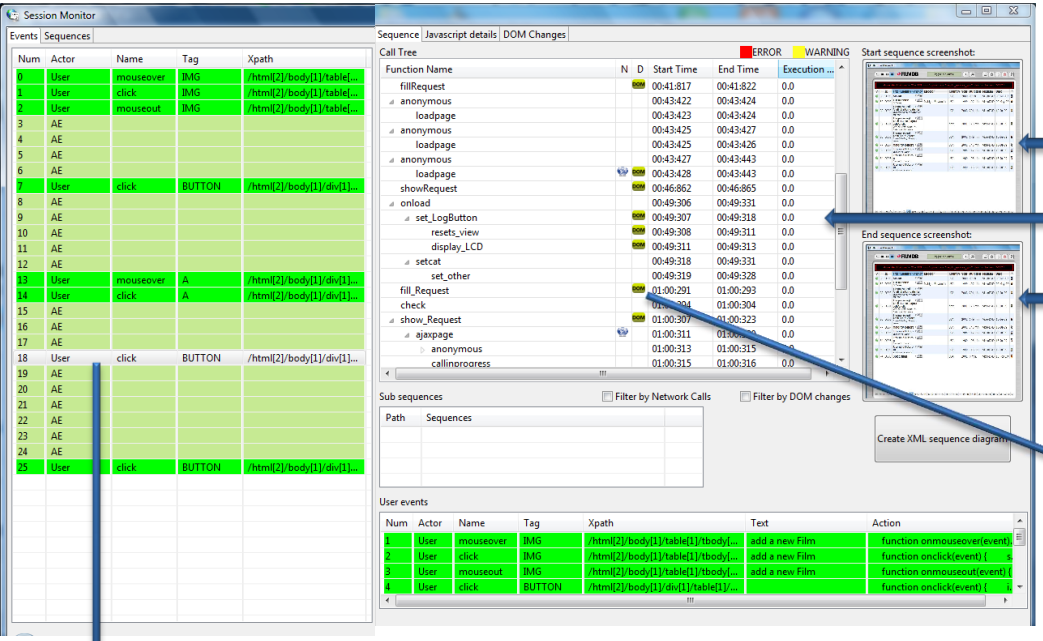


Sintesi elaborazione

AddFilmToMyList	
Totale eventi utente	8
Totale chiamate funzioni Javascript	97
Totale richieste al Server	25
Totale risposte dal Server	25
Totale modifiche al DOM	42
Errori Javascript	NO
Warning su rete	NO

Primo caso di studio - 3

Comprensione

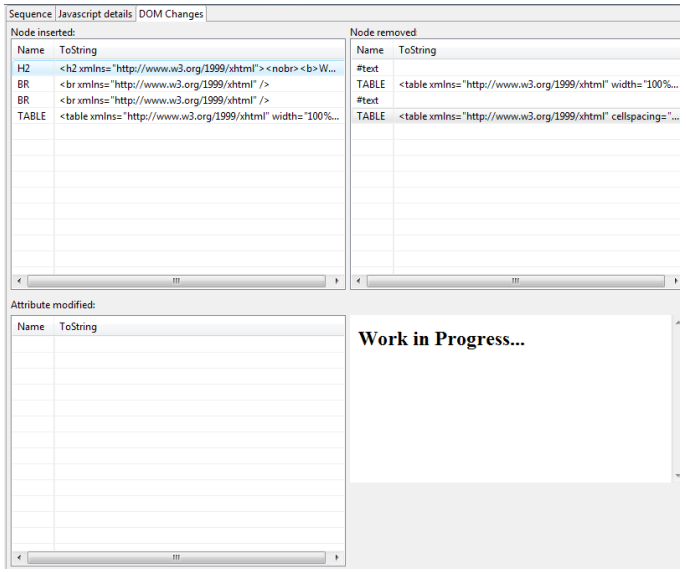


Interfaccia della RIA prima dell'aggiunta film

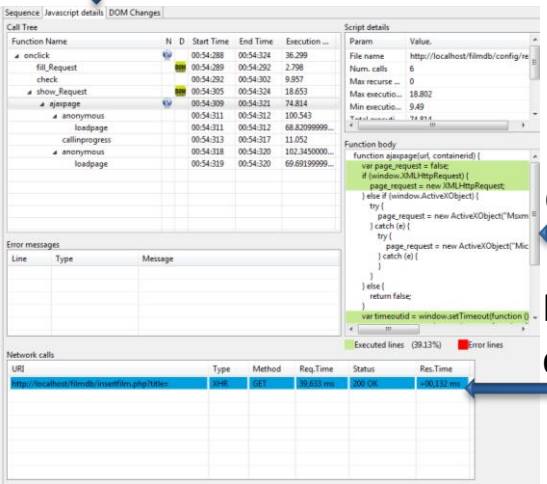
Flusso dell'elaborazione Javascript eseguita

Interfaccia della RIA dopo l'aggiunta del film

Modifiche effettuate all'interfaccia



Flusso dell'elaborazione Javascript scatenata da un singolo evento

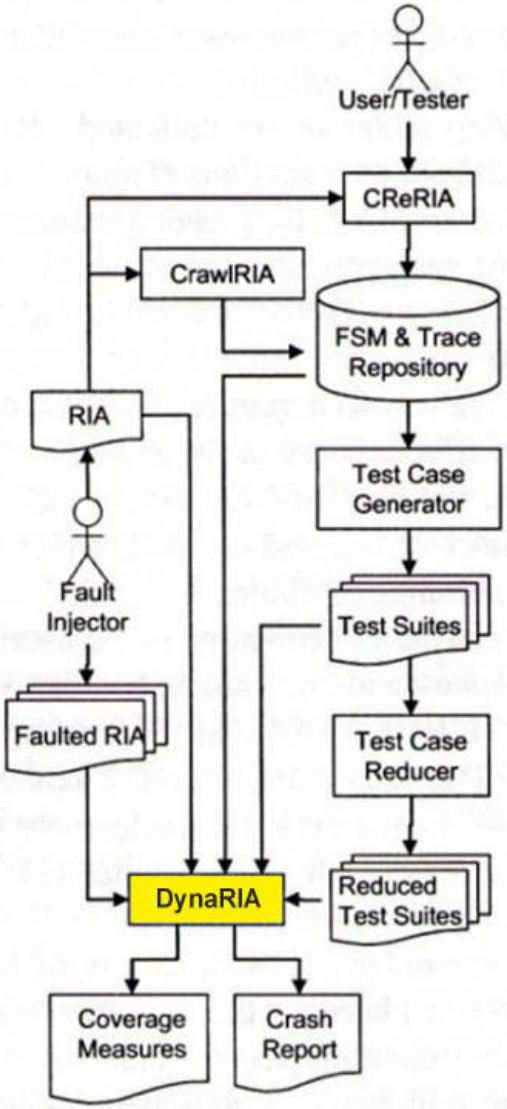


LOC eseguite della funzione selezionata

Richieste al server della funzione selezionata

Secondo caso di studio

Testing

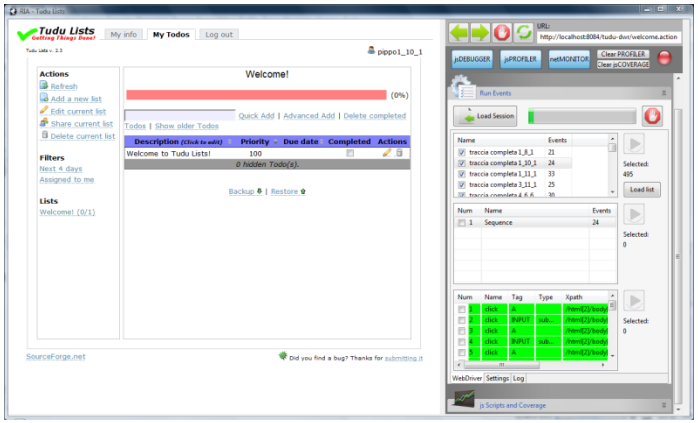


- **Per l'automazione dei processi di testing due aspetti chiave sono:**
  - **Riconoscimento automatico di errori che si verificano durante l'esecuzione di una test suite**
  - **La valutazione della copertura del codice raggiunta dalla test suite**
- **Possibile approccio al testing di RIA usa test suites definite a partire da sessioni utente o tracce di esecuzione ottenute automaticamente mediante tecniche di crawling**
- ***dynaRIA* è stato utilizzato per l'esecuzione automatica di test suites su differenti versioni difettate di una RIA**
- **Con l'obiettivo di:**
  - **Rilevare errori**
    - **Errori durante l'esecuzione del codice Javascript**
    - **Errori di comunicazione irregolare con il server**
      - **Esempio: errore 404 (risorsa non trovata)**
  - **Valutare la copertura del codice Javascript ottenuta dalle test suites**

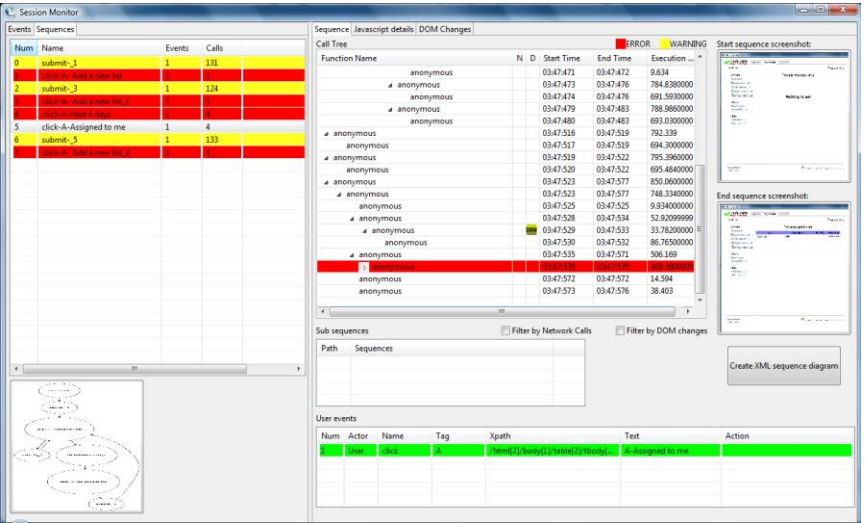
Secondo caso di studio - 2

Testing

- RIA: Tudu Lists
- Esecuzione test suites

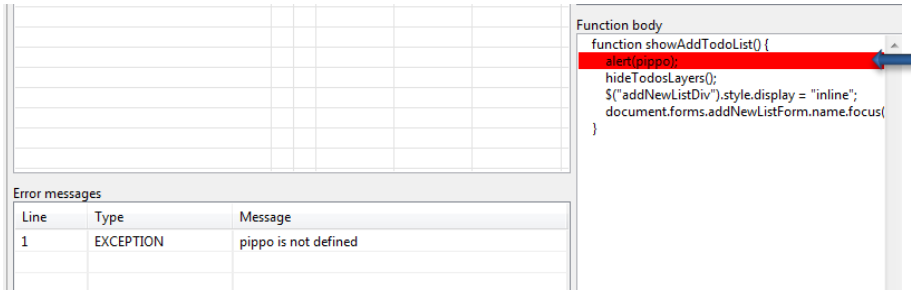


- Errori rilevati



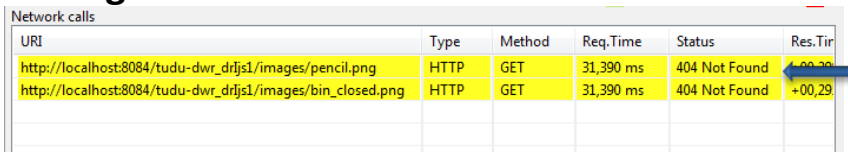
Dettaglio di un errore Javascript rilevato

LOC errore Javascript



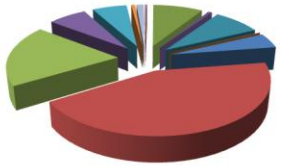
Dettaglio di un errore su rete

Risorsa non trovata



- Copertura del codice Javascript

Totale funzioni Javascript caricate: 1017



Totale funzioni Javascript eseguite: 172

Conclusioni e sviluppi futuri

- **OBIETTIVI RAGGIUNTI:**
  - È stato realizzato uno strumento che ci permette di:
    - Ottenere modelli e viste per comprendere una RIA
    - Testare automaticamente una RIA
  - L'efficacia dello strumento è stata dimostrata attraverso casi di studio

- **SVILUPPI FUTURI:**
  - *Comprensione:*
    - Proporre tecniche per la ricostruzione di altri modelli:
      - Architetturali della RIA
      - Pattern Ajax utilizzati
    - Altre viste che siano di supporto per la comprensione
    - Integrazione con altri *tools* che modellano la RIA attraverso l'utilizzo di FSM (*CReRIA*)
  - *Testing*
    - Impiego di *dynaRIA* a supporto di altri processi di testing basati su tecniche diverse:
      - sul grafo del flusso degli eventi
      - su analisi combinata con lo stato dell'interfaccia della RIA.