The tools of the *MATERIA* Project

Dipartimento di Informatica e Sistemistica

University of Naples "Federico II", Italy

Anna Rita Fasolino
Porfirio Tramontana
Domenico Amalfitano

Outline

- Introduction
  - Models and Reverse Engineering techniques.
  - Testing techniques.
- The Tools
  - CReRIA
  - CrawlRIA
  - TestRIA
  - DynaRIA
- Future Works

Introduction

- The MATERIA Project
  - **MATERIA** is the acronym of **M**odelling **A**nd **TE**sting of **R**ich **I**nternet **A**pplications.

- Goals of the MATERIA project:
  - 1) Defining representation models suitable for comprehending and testing existing RIAs.
    - Proposing Reverse Engineering techniques for obtaining these models.
  - 2) Investigating RIA testing techniques.

Goal 1) Defining RIA representation models

- Finite State Machines
- Event Flow Graphs
- Sequence Diagrams

Finite State Machine

- A FSM provides an abstraction of the behaviour of UI of the RIA, made up of *states* and *transitions*.

- Each state of the FSM is associated with the client Interface shown to the user at the interaction time.
  - Each client interface is characterized by a sub-set of its widgets.
- Each transition is associated with a user event that causes the RIA transition towards another user interface state.

Event Flow Graph

- The EFG provides an abstraction about the flow of user events that are triggered on the UI of a RIA.
- The EFG is a Directed Graph made up of nodes and directed edges.
  - Each node represents a user event triggered on the UI, or a sequence of user events clustered together.
  - The directed edges represent the execution order and the dependencies among user events.

UML Sequence Diagram

- UML sequence diagrams are used to represent the inner interactions between the modules of a RIA at various levels of detail and abstraction.
  - High-level sequence diagrams show the interactions among three main layers of the application:
    - Web Browser – Ajax Engine – Server.
  - Low-level sequence diagrams report the interactions among :
    - Web browser, JavaScript modules making up the Ajax engine, and Server.

Obtaining the representation models by Reverse Engineering (RE)

- We have presented RE techniques (based on the analysis of user execution traces) to obtain the proposed models.
- To solve the problem of FSM state and transition explosion, several clustering criteria have been used.
- Specific Tools implement these RE techniques:
  - CreRIA
  - CrawlRIA
  - DynaRIA

CReRIA Tool- offered functionalities

- User session tracing by means of a Web browser included in the UI of the tool;
- Extraction and storing of relevant data about user interfaces and events occurred during the navigation;
- FSM abstraction;
- Implementation of interface and event clustering techniques (according to different abstraction criteria);
- Recording of user session traces (sequences of interfaces and events) and of the corresponding paths on the abstracted FSM (sequences of states and transitions)
  - these data are stored in a “FSM & Trace Repository” implemented by a MySQL database.
- Transformation of the user session traces into executable traces
  - the execution can be performed by the DynaRIA tool.

CrawlRIA Tool- Offered functionalities

- Automatic Crawling of the RIA user interfaces
  - The crawler automatically triggers the events found on RIA interfaces;
  - Depth first or breadth first visiting strategies can be applied;
  - The obtained execution traces are stored in the “FSM & Trace Repository”;
- UIs and events encountered during the crawling process are clustered together using the same techniques implemented by the CReRIA tool to generate an FSM model;
  - FSM models are stored in the “FSM & Trace Repository”;

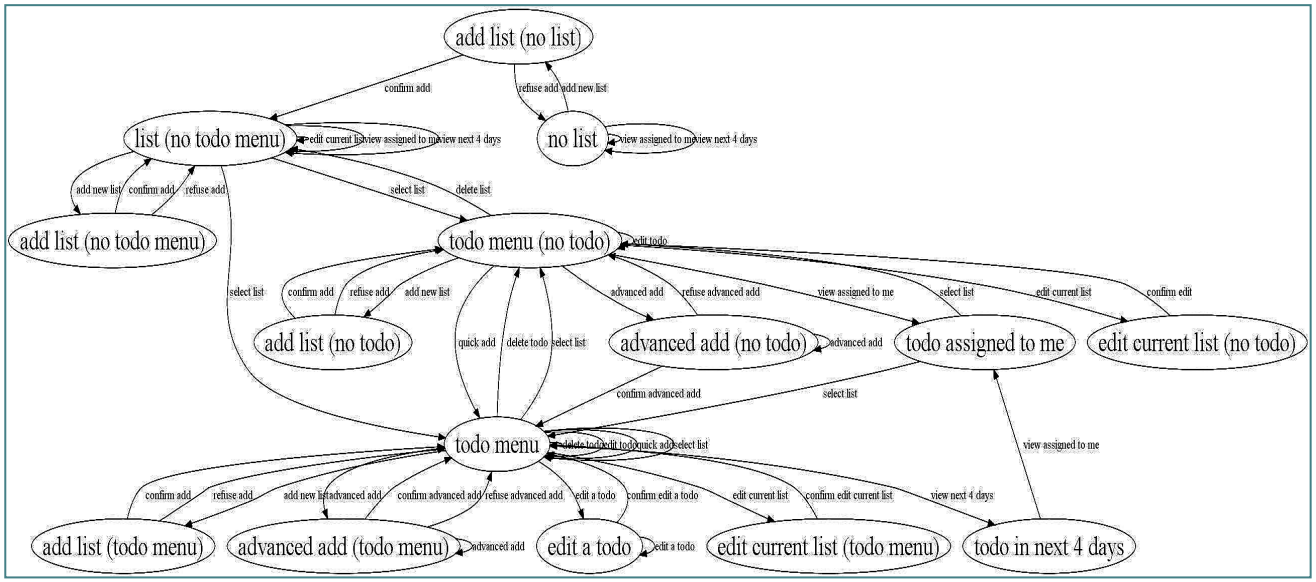
DynaRIA Tool : extraction functionalities

- Provides an integrated Web browser where a user can interact with a RIA while relevant data about the user session are captured and stored.
- Captured data include:
  - the sequence of user events fired on DOM objects of the user interface;
  - the JavaScript functions that are activated by user event handlers;
  - the executed lines of code of JS functions;
  - exceptions and errors occurred at run time;
  - changes (such as add, delete, or change) on DOM objects resulting from a given event management;
  - message exchanges between client and server.

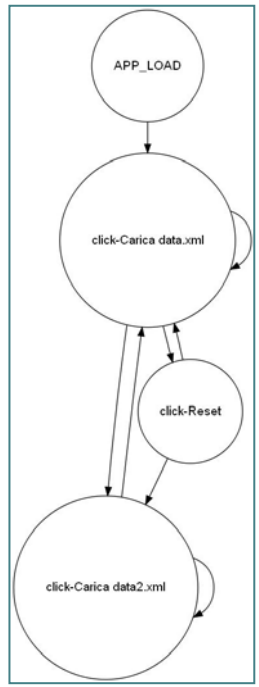
DynaRIA tool: Analysis and visualization functionalities

- The tool provides several abstractions and views about the RIA run-time behaviour, such as;
  - UML sequence diagrams;
    - They can be visualized by means of another tool, "DynaRIA sequence diagram viewer";
  - Event-flow-graphs that report the flow of events fired along a user session;
  - JS Source code views, both at the session level and at the JS function level;
- It also provides cross-referencing functions for switching between views.

Examples of FSM and EFG

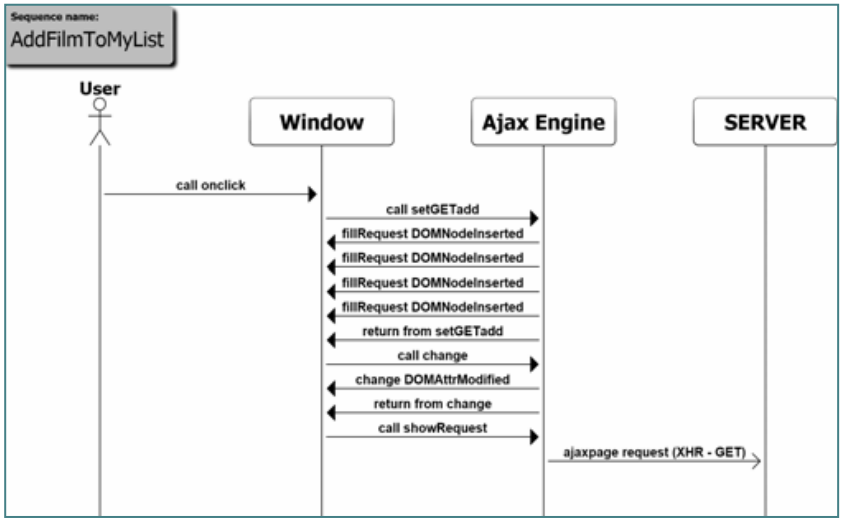


Finite State Machine

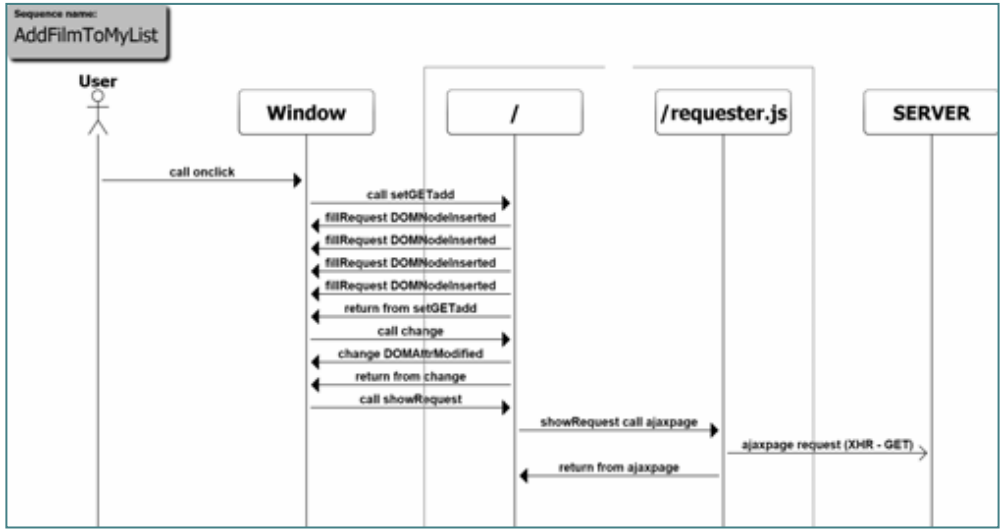


Event Flow Graph

Examples of sequence diagrams



An excerpt of an high level UML sequence diagram



An excerpt of a detailed UML sequence diagram

Goal 2: RIA testing

- We decided to investigate user-session based testing in the context of RIAs.
- In "TestBeds 2010" we proposed a testing technique based on the following steps:
  - Collection of a set of execution traces of the application;
    - Manual collection by CreRIA.
    - Automatic collection by CrawlRIA.
  - Test suite generation;
  - Test suite reduction.

Test Suites Generation

- A test suite is generated by transforming each execution trace into a test case.
  - In general the behaviour of an RIA depends on the current state of the application data, as well as by its environment and session data.
  - before recording each execution trace, we set the RIA in pre-defined states. These states will provide the pre-conditions of the related test cases.

The Test Oracles

- Test oracles are needed to define the PASS/FAIL result of a test case execution.
- We decided to evaluate test case results by:
  - Checking the occurrence of JavaScript crashes.
  - Checking if the reached interfaces coincide with the expected ones.
    - This evaluation is performed by checking if specific widgets with given attribute values are rendered on the reached interface.

Test Suite Reduction

- Given a test suite, the reduction technique computes a minimal set of test cases assuring the same coverage of the original test suite.
- Three **reduction techniques M1, M2, and M3** have been proposed, that consider different types of coverage:
  - M1 covers the same set of **FSM states** covered by the original suite;
  - M2 covers the same set of **FSM transitions** (or events) covered by the original suite;
  - M3 covers the same set of **JS code components** (such as functions) as the original test suite.

TestRIA Tool

- Offered functionalities:
  - reduces a test suite Ts into a smaller one that satisfies the same Ts coverage requirements;
  - transforms the execution traces stored in the FSM & Trace Repository in a test suite composed of executable test cases in Selenium format;
  - executes the test cases in Selenium RC.

…DynaRIA Tool

- Testing functionalities:
  - executes the test cases generated by CreRIA and monitors their execution in the browser environment;
  - traces the JS code execution, keeps track of performed network traffic, and detects any JS error or network warning occurred at runtime, during user session replay;
  - computes several code coverage metrics with respect to a replayed user session.

Experiments

- We have experimented the RE process on the following RIAs:
  - Tudu List (http://app.ess.ch/tudu/welcome.action).
  - The List (http://www.agavegroup.com/?p=51).
  - Ajax FilmDb (http://ajaxfilmdb.sourceforge.net/).
  - Pikipimp (http://www.pikipimp.com/).
  - Buttonator (http://www.buttonator.com/).

- We have experimented the testing techniques using the following RIAs:
  - Tudu List
  - Ajax FilmDb

Next step

- Experimenting our RE and testing techniques using further RIAs.

- A problem:
  - Obtaining RIAs implemented in Ajax .
  - We need the source code of these RIAs in order to inject faults and to set the RIA's initial state (before test execution).
- A question for Xun:
  - May you help us in finding some exemplar RIAs to be used in the next case studies?

References

- D. Amalfitano, A.R. Fasolino, P. Tramontana: *Reverse Engineering Finite State Machines from Rich Internet Applications*. WCRE 2008: 69-73.
- D. Amalfitano, A.R. Fasolino, P. Tramontana, *Experimenting a Reverse Engineering Technique for Modelling the Behaviour of Rich Internet Applications*. International Conference on Software Maintenance, ICSM 2009: 571-574.
- D. Amalfitano, A.R. Fasolino, P. Tramontana, *A Tool-supported Process for Reliable Classification of Web Pages*. International Conference on Advanced Software Engineering & Its Applications (ASEA 2009), Springer.
- D. Amalfitano, A.R. Fasolino, P. Tramontana, *An Iterative Approach for the Reverse Engineering of Rich Internet Application User Interfaces*. ICIW 2010.
- D. Amalfitano, A.R. Fasolino, P. Tramontana, *Rich Internet Application Testing Using Execution Trace Data*. TESTBEDS 2010.
- D. Amalfitano, A.R. Fasolino, P. Tramontana, *DynaRIA: a Tool for Ajax Web Application Comprehension*. IEEE I.C.P.C. 2010