7th International Conference on the Quality of Information and Communications Technology

Oporto, Portugal
29 September - 2 October, 2010

Comprehending Ajax Web Applications by the DynaRIA Tool

University of Naples "Federico II", Italy

Dipartimento di Informatica e Sistemistica

- Domenico Amalfitano
- Anna Rita Fasolino
- Armando Polcaro
- Porfirio Tramontana

Rich Internet Applications (RIAs)

- "A heterogeneous family of solutions with the common goal of adding new capabilities to the conventional hypertext-based Web" [*Fraternali, Rossi, Sánchez-Figueroa, IEEE Internet Computing, May 2010*]
- RIAs combine the Web distributed architecture with desktop applications' interface interactivity and computation power.
- Resulting combination improves all the elements of a Web application (data, business logic, communication, and presentation).

- Most important RIA features:
  - Client-side storage and part of the computation is on the client.
  - Bidirectional communication between client and server.
    - Both the client and server can initiate communication.
  - Powerful presentation and interaction capabilities

Rich Internet Applications technologies

- Several technologies are available for RIAs:
  - Plug-in based (such as Flash and Flex),
  - Browser-based (such as Mozilla XUL),
  - Script based (such as Ajax).
- Ajax is a well know RIA implementation approach.
- It is based on a combination of Web technologies including XHTML, CSS, JavaScript , XML, and XMLHttpRequest.

Ajax-based Rich Internet Applications

- Key aspects of Web Applications developed in Ajax
  - The User Interface (UI) is implemented by one or more Web pages composed by widgets that are updated, deleted or added independently at run time.
  - Ajax Engine (AE):
    - is composed of JavaScript modules,
    - implements the client-side business logic of the RIA,
    - manipulates the UI components,
    - engine's JavaScript functions are driven by user events or other external events,
    - communicates with the server (exchanges few amounts of data, by asynchronous or synchronous requests).

Quality aspects of RIAs

- *Usability* of RIAs is actually improved.
- However, their *analyzability* and *comprehensibility* are strongly affected by:
  - the *heterogeneous* nature,
  - the *dynamic configuration*, defined at run-time,
  - the wide variety of *frameworks* used for implementing them.
- Maintenance and Testing activities require extra effort.
- There is a great need for effective techniques and tools supporting the analysis of RIAs!

Existing analysis approaches and tools

- Techniques for Ajax analysis usable in reverse engineering and testing contexts, have been proposed in the literature [see the paper for more references…]

- Several tools supporting both Ajax testing and run-time analysis are currently available.
  - JavaScript debuggers, Ajax profilers, and tools for automated testing.
  - Examples: Selenium, Firebug, etc…

Features for Ajax analysis offered by existing tools

- JS debugging
- DOM change inspecting
- Network monitoring
- User session tracing,
- User session replaying,
- Performance analysis,
- Code coverage
- UML diagrams abstraction

Coverage of features offered by the analysed tools

- Most relevant features of Ajax analysis offered by the analyzed tools and by the DynaRIA tool are:

	Firebug	A.T.F.	Venkman	DynaTrace	Selenium	DynaRIA
JS Debugging	Y	Y	Y	N	N	P
DOM change inspecting	Y	Y	N	N	N	Y
Network Monitor	Y	Y	N	Y	N	Y
User Session Tracing	N	N	N	Y	Y	Y
User Session Replaying	N	N	N	N	Y	Y
Performance Analysis	Y	N	P	Y	N	P
Code Coverage	N	N	N	N	N	Y
UML diagrams abstraction	N	N	N	N	N	Y

Table Legend: Y = Yes; N = No; P = Partially

The DynaRIA tool

- Purposes of DynaRIA:
  - To provide an integrated stand-alone environment based on dynamic analysis supporting:
    1. Comprehension
    2. Testing
    3. Quality assessment activities involving the client-side of Ajax applications.
  - To include most of the features (besides additional ones) offered by existing tools
- Implemented using Java technologies, it can be downloaded from:

  http://wpage.unina.it/ptramont/downloads.htm

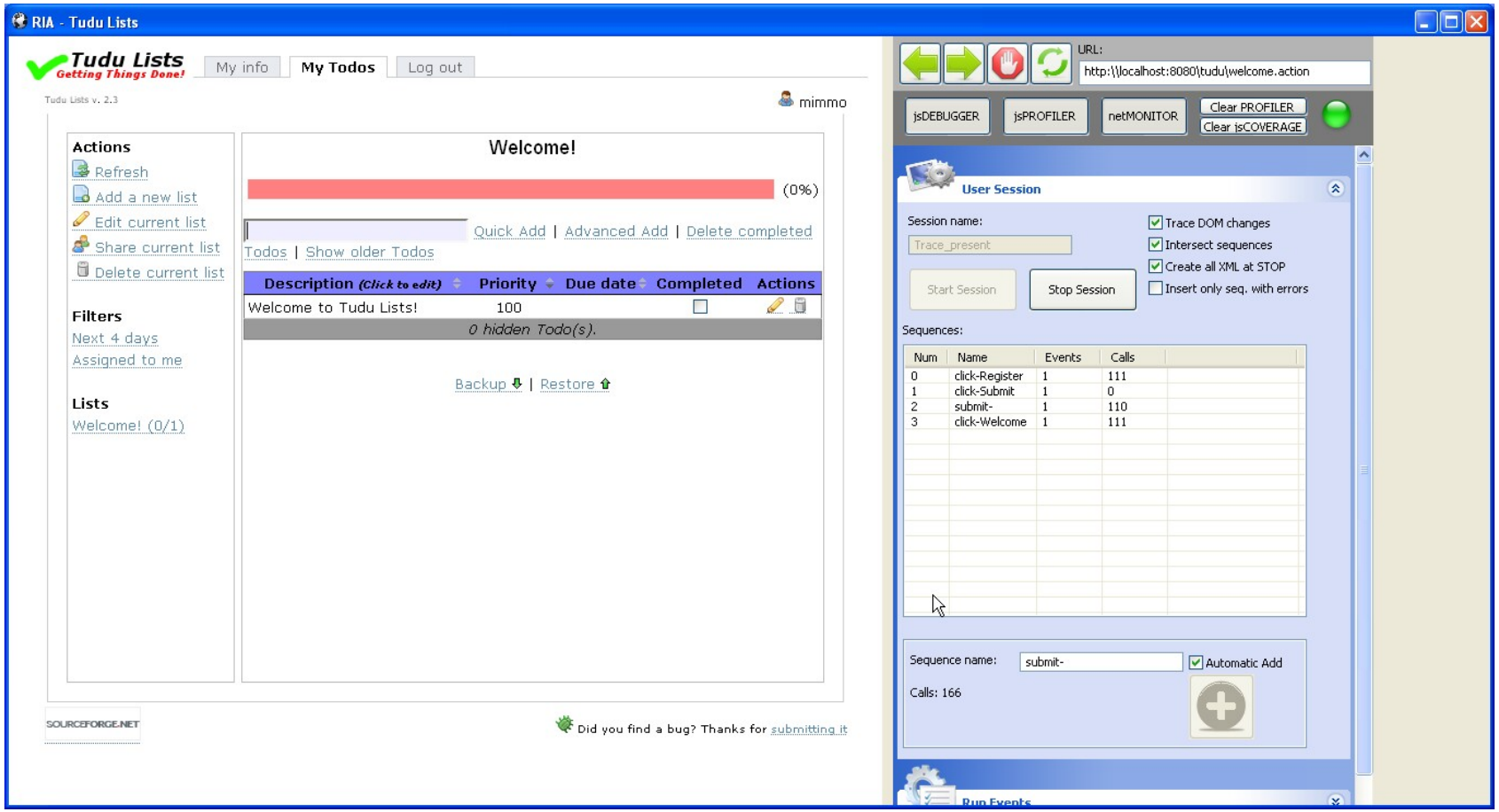
The DynaRIA tool- Comprehension features

- The tool offers functionalities for:
  - Extraction of data about the run-time behaviour of the application (by dynamic analysis of user sessions);
  - Analysis of user sessions with the aim of obtaining details and abstractions about the RIA behaviour, such as:
    - fired events, associated JS function call-tree, executed JS functions, server requests made by a JS function…
  - Visualization of data and abstractions such as:
    - UML sequence diagrams and Event-flow-graphs, views on JS function code, on JS executed lines of code, JS call tree, DOM changes, network traffic and exceptions, views on the UI…

Extraction, Analysis and Visualization by DynaRIA

- User sessions are traced by means of an integrated browser offered by the tool.
- Extracted data are shown in several Session Monitor Views provided by the tool.
- DynaRIA abstracts UML sequence diagrams at various levels of detail and abstraction from each user session or from its parts.
- The diagrams can be visualized by the ***dynaRIA Sequence Diagram Viewer.***

The Integrated browser and the User Session tracing panel



The DynaRIA tool- Testing features

- The tool supports *user-session testing* of the RIA, and provides features of:
  - Capture and Replay of user sessions
  - Error detection of run-time JS exceptions (such as JS syntax errors, array out of bound errors, etc.) and network warnings.
  - JS Code coverage evaluation, such as:
    - # executed JS functions / #JS functions
    - # executed LOC/#LOC
    - etc..

DynaRIA tool – Quality assessment features

- Dynamic analysis data are used for computing metrics about the JS code
  - Examples: # JS modules, JS module and function sizes, fan-in, fan-out of JS modules, server coupling, DOM coupling, …
- The metrics can be used to compute quality factors of the Ajax applications.

Experiment

- Goal: evaluating the effectiveness of the tool in realistic software comprehension scenarios.
- Experimental procedure:
  - Selected tasks that are typical of RIA analysis scenarios were executed with the tool support:
    - feature comprehension
    - error detection in testing and debugging processes
    - testing process evaluation
    - RIA quality assessment
  - Four case studies, involving two different RIAs, were executed.

First Case Study- Comprehension tasks

- *Subject application*: AjaxFilmDB.
- *Goal*: to understand how the functionality of adding a new film to the DB has been implemented.
- *Comprehension Tasks*:

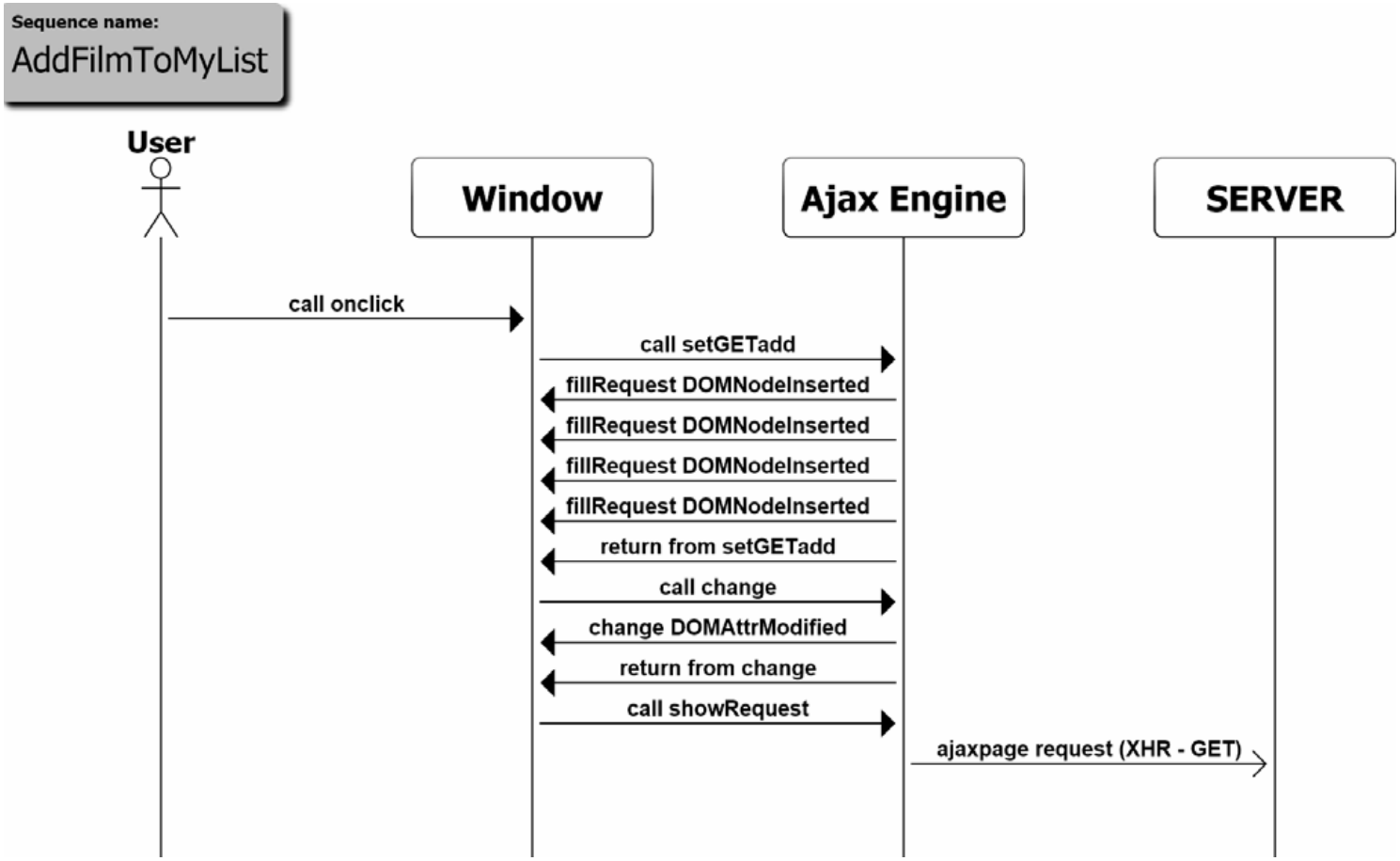
T 1.1	How do the high-level components of the application interact ?
T 1.2	What low-level components of the application interact?
T 1.3	How do the low-level components of the application interact?
T 1.4	What low-level components exchange messages with the server side of the application?
T 1.5	What are the internal elaboration details of the considered functionality?

First Case Study- results

- T1.1: Required the comprehension of the flow of interactions among the Browser, the Ajax Engine and the Server side of the application.

- The task was accomplished using the high level Sequence diagrams produced by the tool.

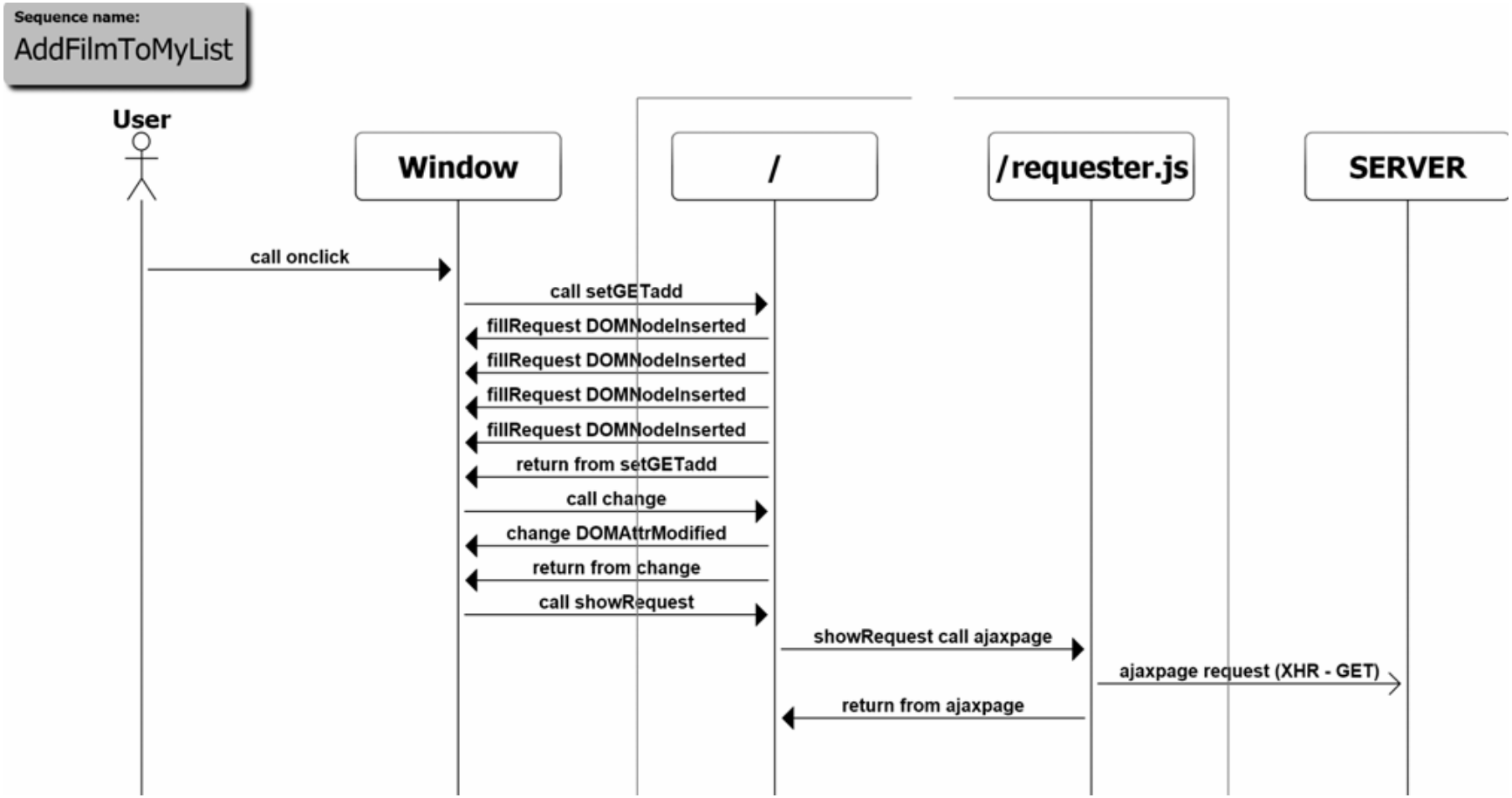
First Case Study- results



Comprehension tasks

- T1.2 – T1.4: Required the comprehension of the interactions between the JS inner modules, the server and the browser.
- These tasks were accomplished using the low level Sequence diagrams abstracted by the tool.

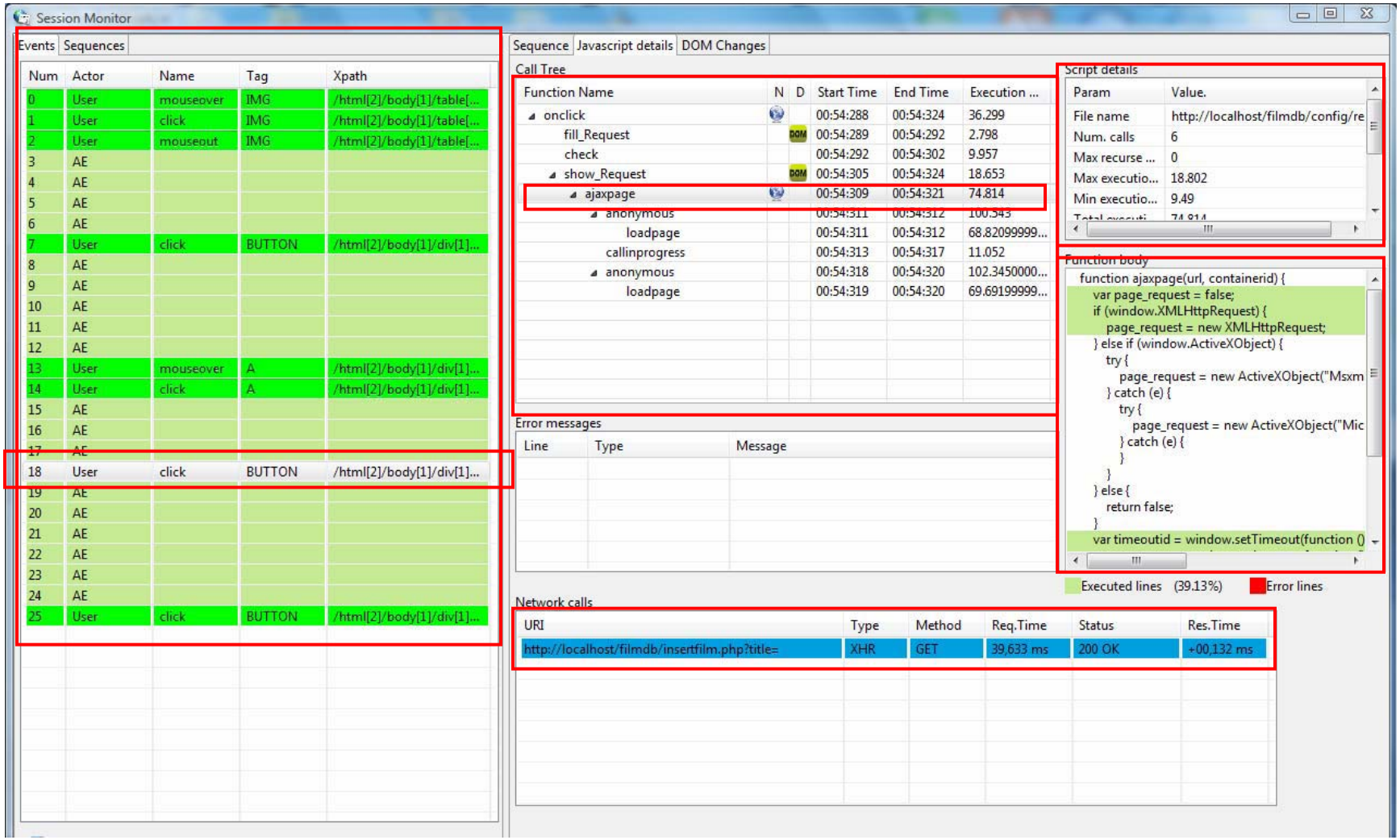
Comprehension tasks



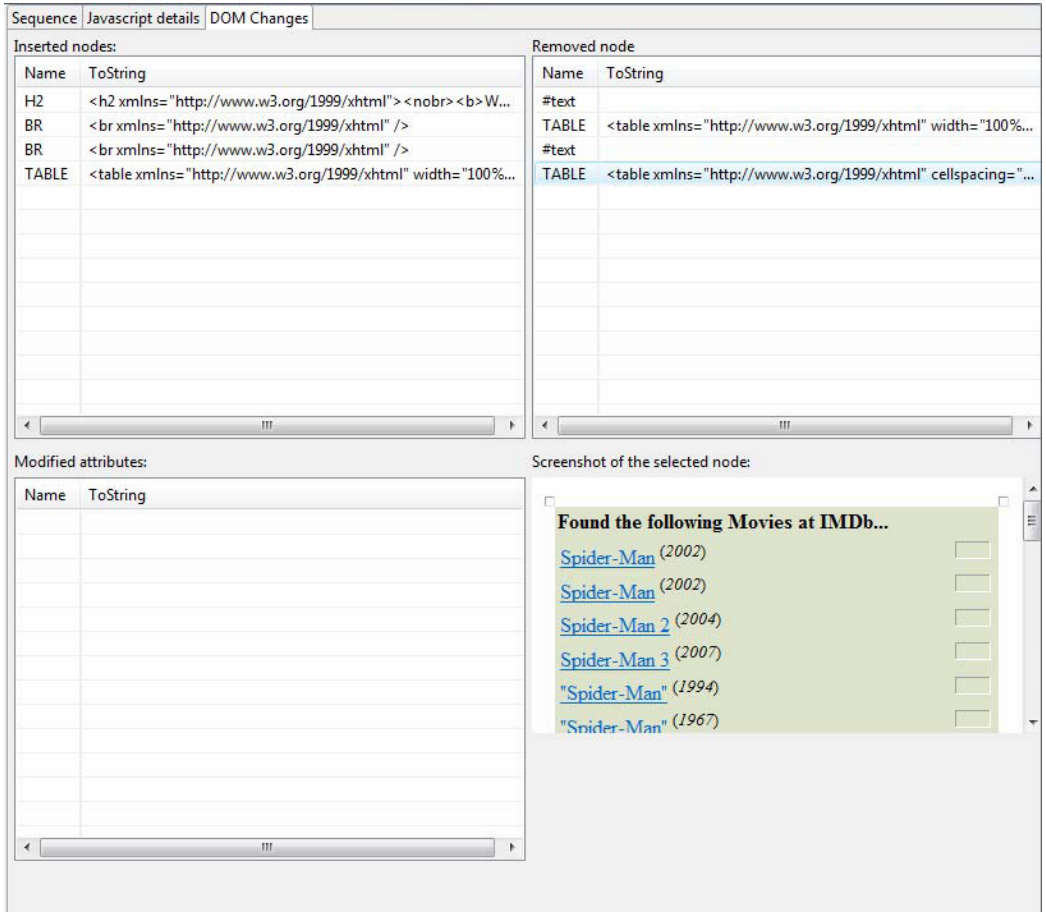
Comprehension tasks

- T 1.5: required the comprehension of internal details of the elaboration.
- The task was successfully performed using the data shown by several 'Session Monitor' views.
- The views focused on four different code perspectives:
  - Fired Events,
  - JS functions that carry out the elaboration,
  - Server that provides data or elaboration by communicating with the client,
  - User Interface where the effects of the elaboration are shown.

Session monitor views examples



Session monitor views examples



Summary data about the traced execution	
# user events	8
# JS modules	6
# JS function calls	97
# server requests	25
# server responses	25
# DOM change requests	42
# exceptions	0

First Case Study - Results

- All the considered comprehension tasks were accomplished thanks to
  - the high-level and the lower-level Sequence diagram views offered by the tool
  - the opportunity for a user of navigating through different views about the code components.

Second Case Study- Testing & Debugging tasks

- *Subject application*: AjaxFilmDb
- *Goal*: to find run-time exceptions of a functionality execution and the JS components that are responsible for them.
- *Comprehension Tasks*:

T 2.1	What run-time exceptions do occur during the functionality execution?
T 2.2	What JS functions (and lines of code) are responsible for run-time exceptions?

Second Case Study

- Experimental procedure:
  - One of the authors injected faults of different types in the JS code
  - Another author performed the testing & debugging tasks using DynaRIA.
- Results:
  - T 2.1: completed with success thanks to the tool capability of detecting run-time JS exceptions.
  - T 2.2: solved thanks to the tool feature of detecting the components that are involved in exceptional executions.

Third Case Study- Testing Processes

- *Subject application*: Tudu
- *Goal*: To test a RIA functionality using a user-session based technique, and to assess the testing effectiveness by evaluating the code coverage and fault detection capability.
- *Tasks*:

T 3.1	Generation of a test suite from user sessions
T 3.2	Test suite coverage assessment
T 3.3	Generation of several application faulty versions by fault injection
T 3.4	Replay of test suites on the faulty versions of the application
T 3.5	Test suite fault detection capability assessment

Testing tasks

- The testing process tasks were almost all supported by the tool:
  - T3.1 Generation of a test suite from user sessions
  - T3.2 Test suite coverage assessment
  - T3.4 Replay of test suites on the faulty versions of the application
  - T3.5 Test suite fault detection capability assessment
- The tool provided a valid aid for client-side automated testing of Ajax applications.

Third Case Study

Fault Type	Number of injected faults
References to not defined objects/ methods/ attributes	7
JS function call instructions with undefined, incorrect, or missing parameters	5
JS syntax errors	2
Array out of bound errors	2
Server requests of missing resources or JS files	3

Fourth Case Study- RIA Quality assessment

- *Subject application*: Tudu
- *Goal*: to use the metrics computed by the tool in order to estimate the internal quality of JS modules.
  - Which are the *larger modules*, *in size*, involved in a given functionality execution?
  - How much is a module *coupled* to other modules, or to the server?

RIA internal quality assessment

- Experimental procedure
  - The application has been exercised according to a predefined sequence of actions
    - (es.User registration- Login- Adding a todo list- Adding a todo- Logout).
  - The set of JS modules loaded at run-time has been obtained
  - JS modules were characterized with respect to their size and coupling levels.

Fourth Case Study-results

Module	#JS func.	LOC	Fan-in	Fan-out	#Server Req.	#DOM changes
logout.action	2	2	1	1	0	0
scriptaculous.js	395	2693	27	15	5	0
utils.js	65	1321	17	28	4	140
showTodos.action	54	338	17	17	1	3
todos.js	45	90	4	6	0	0
welcome.action	2	2	1	1	0	0
register.action	3	3	2	2	0	0
scriptaculous/effects.js	143	1134	21	12	0	0
engine.js	62	908	21	25	5	0
tabs.js	9	92	5	8	0	3
Todo_lists.js	35	70	1	2	0	0
prototype.js	328	1961	34	34	0	0

Fourth Case Study - Results

- The considered metrics provided a useful starting point for making hypotheses about the quality of the modules involved in given executions of the applications.
- These metrics do not definitely characterize the JS modules, but are just valid with respect to the considered execution traces of the application
- Dealing with Ajax applications whose source code can be dynamically loaded at run-time, this one is the only feasible approach for obtaining the code of the application and analyzing it.

Conclusions

- In this paper we presented DynaRIA a tool that provides a user-friendly environment for analysing the dynamic behaviour of Rich Internet applications implemented in Ajax.
- DynaRIA provides an integrated environment offering features needed for supporting program comprehension, testing, debugging and quality assessment activities.
- Some case studies preliminarily showed the usefulness of these features and the effectiveness of the tool.

Future Work

- We plan to extend the experimentation in order to evaluate the actual cognitive support provided by the tool.
- To improve and extend the analysis and visualization features offered by the tool.