Rich Internet Application testing: open issues, questions, proposals

Domenico Amalfitano

Anna Rita Fasolino

Porfirio Tramontana

Dipartimento di Informatica e Sistemistica

University of Naples Federico II, Italy

Context and Motivation

- Context:
  - Rich Internet Application Testing
- Motivation:
  - Identifying a set of open issues and questions regarding RIA testing
  - Outlining possible solutions for a joint discussion.

Outline

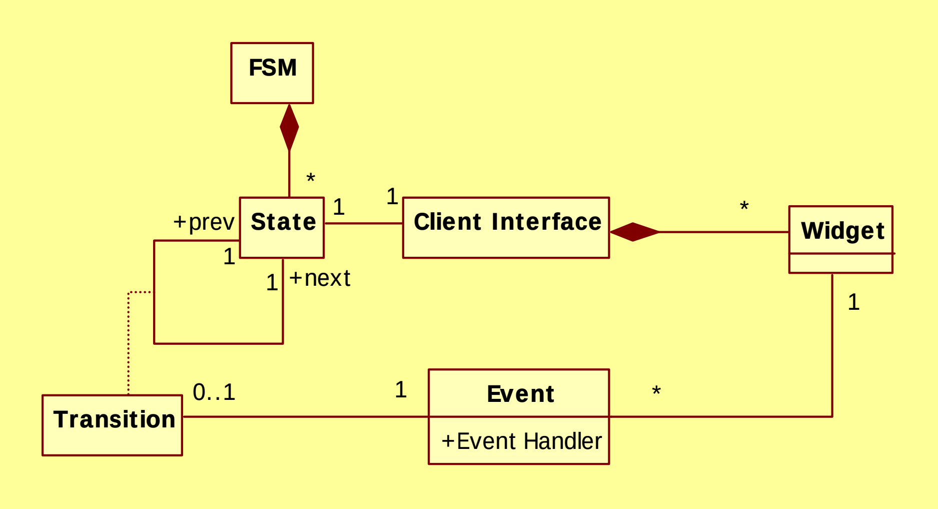
- RIA representation models and Reverse Engineering techniques
- RIA testing techniques based on execution traces
- Supporting tools

RIA user interface representation models

- Event-Interaction-Graph [A. M. Memon and Q. Xie. Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software. *IEEE TSE*, 2005]
  - Models the flow of events on the RIA UI
- Finite State Machine (FSM) Model
  - Provides an abstraction on the UI of the RIA made up of states and transitions

FSM Model of an RIA

1. FSM models the behaviour of an RIA.
2. FSM represents all the elaboration states where the RIA receives any input solicitation by its user.
3. Each state of the RIA is described by the client Interface shown to the user at that interaction time.
4. Each *client interface* is characterized only by the sub-set of its *widgets* that are 'clickable' or, more in general, that have a *registered event listener* and a corresponding *event handler*.
5. *Transitions* are associated with user events that trigger the RIA migration towards a new state.



FSM conceptual model

FSM Reverse Engineering techniques

- The FSM Reverse Engineering problem has been addressed in some recent papers
  - [1] D. Amalfitano, A.R. Fasolino, P. Tramontana: Reverse Engineering Finite State Machines from Rich Internet Applications. WCRE 2008: 69-73
  - [2] 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
  - [3] D. Amalfitano, A.R. Fasolino, P. Tramontana, A Tool-supported Process for Reliable Classification of Web Pages, accettato per la pubblicazione in International Conference on Advanced Software Engineering & Its Applications (ASEA 2009), Springer
  - [4] D. Amalfitano, A.R. Fasolino, P. Tramontana, An Iterative Approach for the Reverse Engineering of Rich Internet Applications User Interfaces, submitted for publication

- Papers [1, 2 , 3] describe a process based on the analysis of execution traces of the RIA, that generates a FSM model that needs to be validated manually at the end of the process.
- Paper [4] describes an iterative variant of this process, where the manual validation of the FSM is performed at each process iteration.
  - This process will be illustrated in the following slides.

The iterative Reverse Engineering Process [4]

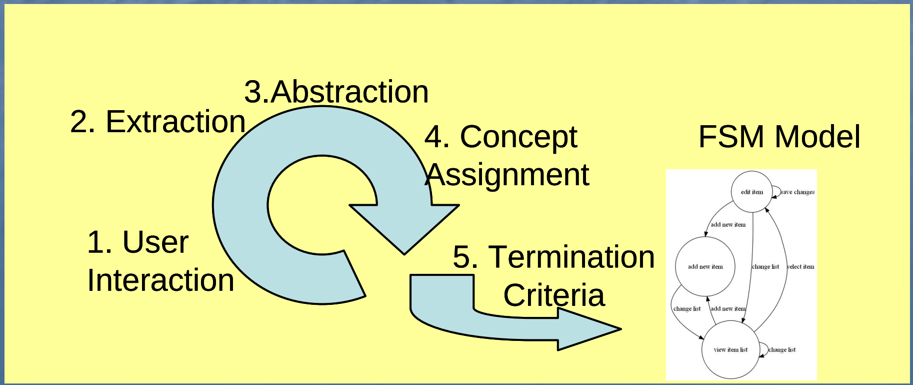
1. *User Interaction:* a user interacts with the RIA and triggers an event

2. *Extraction:* information about current interface, fired user event and the user interface that is obtained after the event processing are automatically retrieved and persistently stored

3. *Abstraction:* heuristic criteria evaluate the degree of similarity of the current user interface with the previously produced ones and propose a classification of the interface

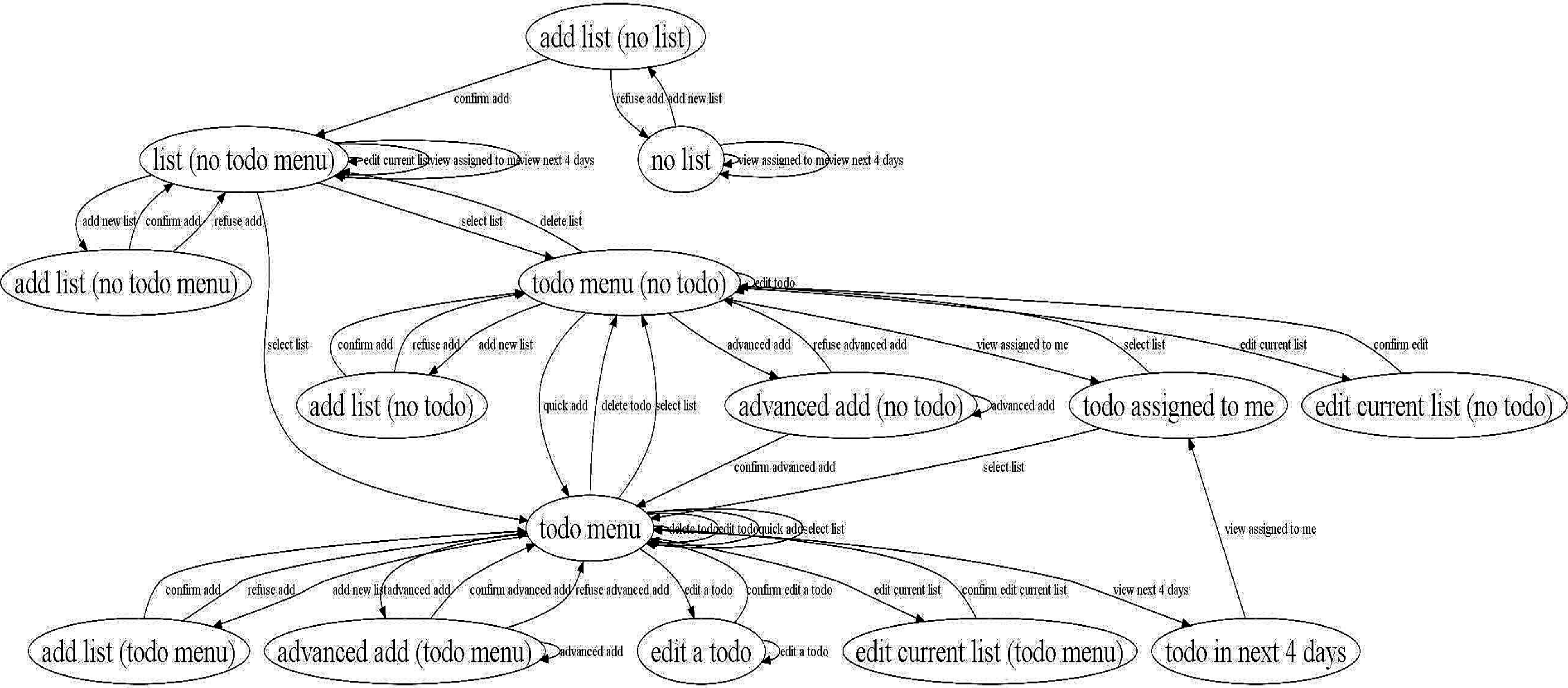
4. *Concept Assignment* : the software engineer validates the classifications proposed by the heuristic criteria and accepts or refuses them. If the classification is refused, he has to propose the correct concept to be assigned. In this way, the expert incrementally reconstructs a FSM modeling the behavior of the RIA GUI

5. *Termination criterion evaluation* :
   event coverage, or coverage of known scenarios, or the effort (e.g, the time spent) devoted to the whole process can be considered.



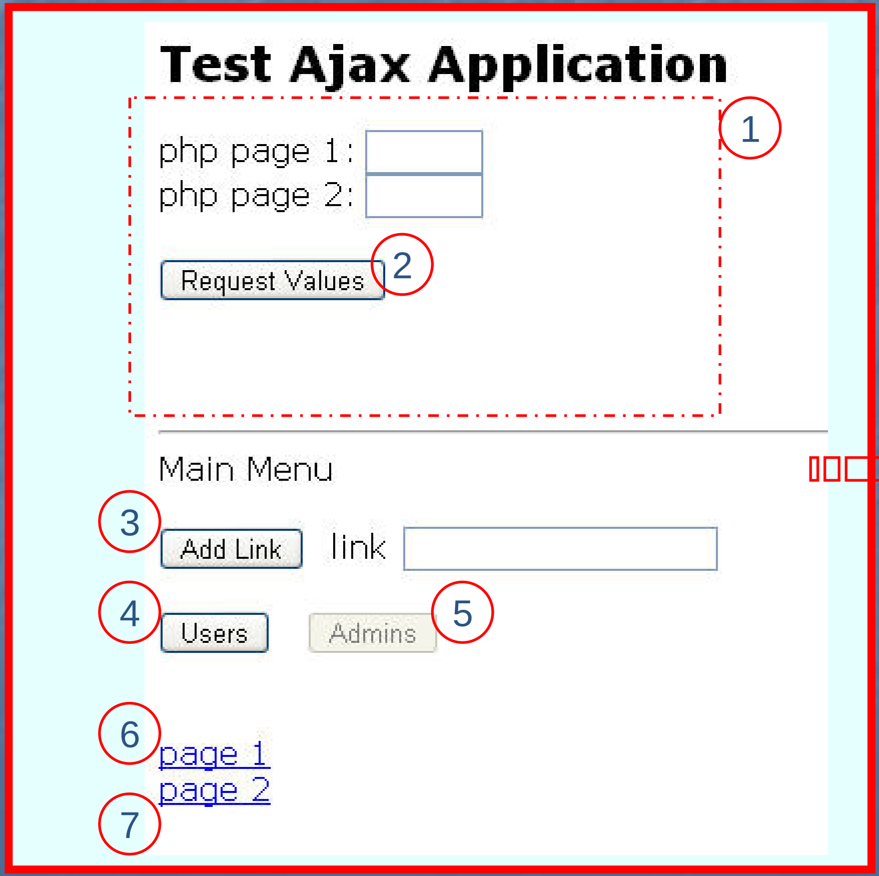
An example of obtained FSM

- The FSM Model of the application TuDu: http://app.ess.ch/tudu/welcome.action



Some details about the interface clustering approach

Examples of widgets extracted from a RIA interface and used for interface clustering:



An Example of Client Interface and the selected widgets:

1 – Form
2 – Button
3 – Button
4 – Button
5 – Button
6 – Link
7 – Link

Examples of widget attributes

	Id_widget	Attribute	Value	Xpath	Unindexed xpath	UnindexedxpathID	Active
1	5482	action	#	/html[2]/body[1]/form[1]	/html/body/form	/html/body/form	true
2	5475	type	button	/html[2]/body[1]/form[1]/input[3]	/html/body/form/input	html/body/form/input	true
	5475	onclick	"avvia()"	/html[2]/body[1]/form[1]/input[3]	/html/body/form/input	/html/body/form/input	true
3	5476	type	button	/html[2]/body[1]/div[1]/div[1]/input[1]	/html/body/div/div/input	/html/body/div/div[@id=‘container1’]/input	true
	5476	onclick	"new_link()"	/html[2]/body[1]/div[1]/div[1]/input[1]	/html/body/div/div/input	/html/body/div/div[@id=‘container1’]/input	true
4	5477	type	button	/html[2]/body[1]/div[1]/div[2]/input[1]	/html/body/div/div/input	/html/body/div/div/input	True
	5477	onclick	"gest ('users.xml')"	/html[2]/body[1]/div[1]/div[2]/input[1]	/html/body/div/div/input	/html/body/div/div/input	True
5	5478	type	button	/html[2]/body[1]/div[1]/div[2]/input[2]	/html/body/div/div/input	/html/body/div/div[@id=‘container2’]/input	False
	5478	Onclick	"gest (‘admins.xml’)"	/html[2]/body[1]/div[1]/div[2]/input[2]	/html/body/div/div/input	html/body/div/div[@id=‘container2’]/input	False
6	5479	Href	page1.html	/html[2]/body[1]/div[2]/a[1]	/html/body/div/a	/html/body/div/a	true
7	5480	href	page2.html	/html[2]/body[1]/div[2]/a[1]	/html/body/div/a	/html/body/div/a	true

Heuristic criteria for interface clustering

- **C1)** two interfaces are equivalent and can be clustered if they have the same number of widgets with the same subset of attributes.

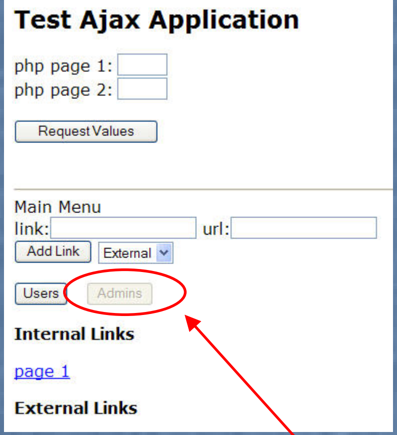
- **C2)** two interfaces are equivalent and can be clustered if they have the same number of ***enabled*** widgets with the same subset of attributes.

-**C3)** two interfaces are equivalent and can be clustered if they contain:
- ***the same set of containers having the*** ***same type of widgets***.

-**C4)** two interfaces are equivalent and can be clustered if they contain ***the same set of containers with the same ID, containing the*** ***same type of widgets***.

- A container is implemented by tags such as DIV, TABLE, etc…

Examples of heuristic clustering criteria applications



Test Ajax Application

php page 1:
php page 2:

Main Menu
link: url:

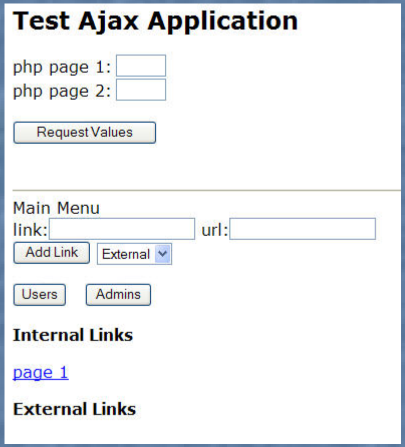
Internal Links

page 1

External Links

Interface I_1

This widget make the difference for C1



Test Ajax Application

php page 1:
php page 2:

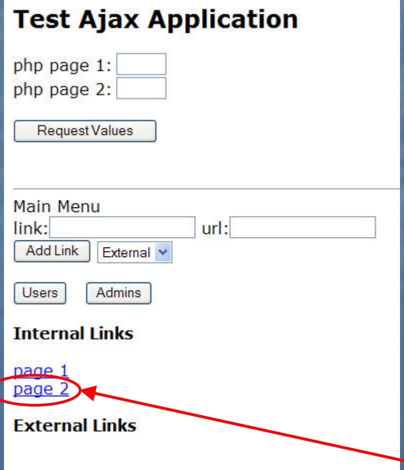
Main Menu
link: url:

Internal Links

page 1

External Links

Interface I_2



Test Ajax Application

php page 1:
php page 2:

Main Menu
link: url:

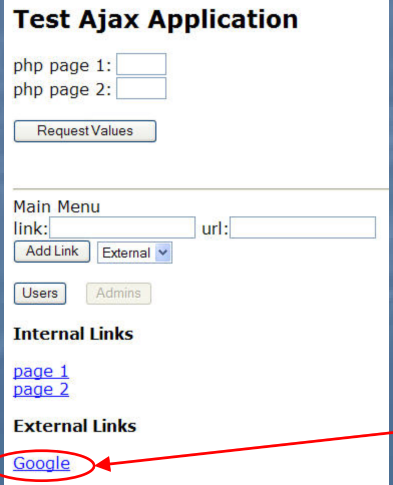
Internal Links

page 1
page 2

External Links

Interface I_3

This widget make the difference for C2



Test Ajax Application

php page 1:
php page 2:

Main Menu
link: url:

Internal Links

page 1
page 2

External Links

Google

Interface I_4

This widget make the difference for C4

- *Interface I_1 is equivalent to interface I_2 according to C1, while they are not equivalent according to C2*

- *Interface I_2 is equivalent to interface I_3 according to C3, while they are not equivalent according to C2*

- *Interface I_3 is equivalent to interface I_4 according to C3, while they are not equivalent according to C4*

A first observation

- The proposed reverse engineering process relies on a set of RIA execution traces which are manually collected.
- In order to improve the efficiency of the process, execution traces may be collected automatically. Our current work is addressing the following questions:
  - Question 1: is it possible to reconstruct the FSM model using a set of artificially built execution traces, such as those obtained using a RIA ripping (crawling) technique?
  - Question 2: how should the reverse engineering process be modified?

A possible variant of the Reverse Engineering Process based on RIA crawling techniques

- Similar to the iterative process, except for:
  - The Interaction with the RIA is performed by automatic firing of user events;
    - For each RIA user interface, the set of firable events are stored in a list and triggered according to a given navigation strategy (such as depth first/breadth first ones);
  - In the Abstraction step, just one heuristic criterion is used for the clustering (the criterion is a process input);
  - The Concept Assignment step may not be executed at all.

- The output of the process will include:
  - a FSM=(S, T, E), S={ GUI state}   T={ transition } E={event}
  - A set of automatically generated execution traces.

- Relevant Process factors:
    - The event navigation strategy (breadth first or depth first)
    - The criterion used to stop the exploration
    - The interface clustering criterion.

Details about the RIA Crawling based process

- The analysis starts from the home page of the RIA (with a reset db and restarted Web server)
- For each RIA user interface, the set of firable events are stored in a list and navigated according to a depth first/breadth first strategy
- An event triggering causes the reaching of a new interface
  - If the interface is equivalent to an already visited one, according to the chosen heuristic equivalence criteria
    - The navigation is stopped
    - The sequence of events triggered from the home page to the current one are saved
    - The navigation restarts from the home page and follows the direction of an event that has not already been triggered
      - Optionally, another trace termination criterion based on a maximum depth could be considered

Some questions

- Using the proposed Reverse Eng. processes, we are able to obtain a set of RIA execution traces .

- Question 3: may we use the execution traces (either generated by RIA crawling, or by manual exploration of the application) for the aims of testing?

- Question 3.1: which test case generation technique can be proposed?

A first test case generation technique

- Test generation technique: Transform each execution trace into a test case.

- Test Case=Sequence of fired events and assertions
  - Fired events are described as events on widgets of an interface
  - Assertions may be of two types:
    - based on the verification of generic properties of the RIA behavior, such as:
      - absence of Javascript crashes;
      - absence of server response timeouts (e.g. for missing server resources);
    - Based on the verification of equivalence conditions between expected GUI states and actual ones.

  - Test Case Set-up requires start operations such as: db reset to a known state, Web server restart, RIA home page reload

Testing problems associated with the proposed test generation technique

a) Test Suite Minimization problem

b) Assessing the effectiveness of test suite generation techniques

c) Definition of test oracles

A) Test Suite Minimization problem

- Test suites generated by the proposed technique may need to be reduced in order to improve testing process efficiency.
- Which reduction techniques are applicable?
- Can we use classical reduction techniques to get minimal test suites that assure pre-defined coverage criteria?
  - A reduction technique was proposed in Giuseppe A. Di Lucca, Anna Rita Fasolino, Porfirio Tramontana: A Technique for Reducing User Session Data Sets in Web Application Testing. WSE 2006: 7-13

Examples of coverage criteria usable for minimization aims

- With respect to the abstracted FSM:
  - FSM state coverage
  - FSM transition coverage
  - FSM event coverage
  - FSM K-length path coverage

- With respect to the application code:
  - Javascript function call coverage
  - Javascript function call sequence coverage [Memon et al. Call-Stack Coverage for GUI Test Suite Reduction, IEEE TSE 2008]
  - Javascript function code coverage

B) Assessing test suite effectiveness

- A possible approach:
  - assessing the effectiveness of test suites by evaluating their defect detection capability on a fault-seeded version of the RIA.
- Faults can be defined:
  - According to a RIA fault model, or
  - By analysing known defects of the RIA retrieved from its CVS.
- Testing Effectiveness evaluation can be based on the number of detected faults.

C) Testing Oracle definition

- Which types of testing oracles can be used?
- Which types of assertions are usable?

- Assertions may specify:
  - A) generic properties of the RIA behavior, such as:
    - Absence of Javascript crashes;
    - Absence of server side exceptions (such as response timeouts for missing server resources);

  - B) specific properties of the RIA behavior that can be deduced from client interface analysis, such as:
    - equivalence conditions between expected GUI states and actual ones.

Validation

- The outlined test generation approach, and possible solutions to the listed problems should be investigated by experiments.
- Experimental activities will have to be designed and executed with tool support.

Tool support (1/3)

- Most of the testing activities can be supported by our prototype tools:
  - a tool for recording user sessions (and generating the FSM of the RIA) (CReRIA) (already available)
  - A crawler tool for generating execution traces automatically (CRAWLRIA) (In Progress)
  - A tool for transforming execution traces into Selenium-executable test cases, and implementing a test suite minimization technique (TestRIA) (in progress)
  - A tool that replicates execution traces and performs several coverage analyses, and detects run-time Javascript errors and server exceptions (DynaRIA) (already available)

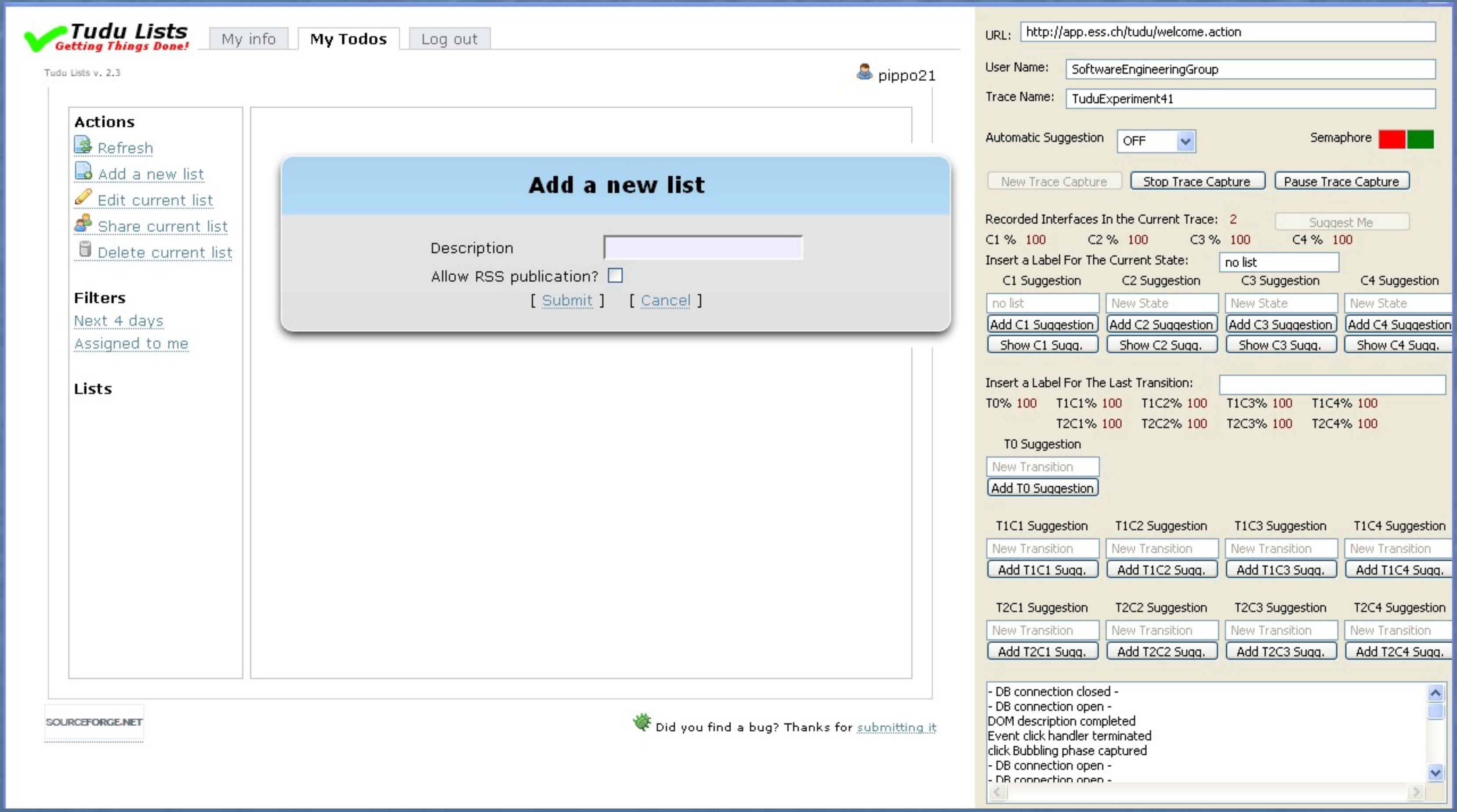
Tool support (2/3)

- CreRia
  - Supports the interactive and automatic reverse engineering processes
  - Collects user session traces and assesses their coverage degree as:
    - #user events fired on analysed interfaces/#user events that are firable
- CrawlRia
  - Crawls a RIA according to a given heuristic clustering criterion, and produces a set of session traces
- TestRia
  - Transforms traces (obtained either by the crawlRia or by CReRia) in executable test cases (as Selenium executable tests)
  - Minimise test suites according to coverage criteria

Tool support (3/3)

- DynaRIA
  - Executes collected traces in order to detect:
    - JS errors
    - Network exceptions
  - Extracts a log of the sequence of executed scripts and network requests (synchronous and/or asynchronous) during trace execution
  - Assesses the Javascript coverage of an execution trace as:
    - #JS function executed/# JS functions
    - #JS code line executes/#JS code lines
  - Generates interaction diagrams (e.g. sequence diagrams) of the traced executions
  - Generates a EFG of traced executions

A snapshot from CreRia TOOL



Add a new list

Description

Allow RSS publication? □

[Submit] [Cancel]

URL: http://app.ess.ch/tudu/welcome.action

User Name: SoftwareEngineeringGroup

Trace Name: TuduExperiment41

Automatic Suggestion: OFF

Semaphore 🟥 🟩

Recorded Interfaces In the Current Trace: 2

C1 % 100 C2 % 100 C3 % 100 C4 % 100

Insert a Label For The Current State: no list

C1 Suggestion	C2 Suggestion	C3 Suggestion	C4 Suggestion
no list	New State	New State	New State

Insert a Label For The Last Transition:

T0% 100 T1C1% 100 T1C2% 100 T1C3% 100 T1C4% 100

T2C1% 100 T2C2% 100 T2C3% 100 T2C4% 100

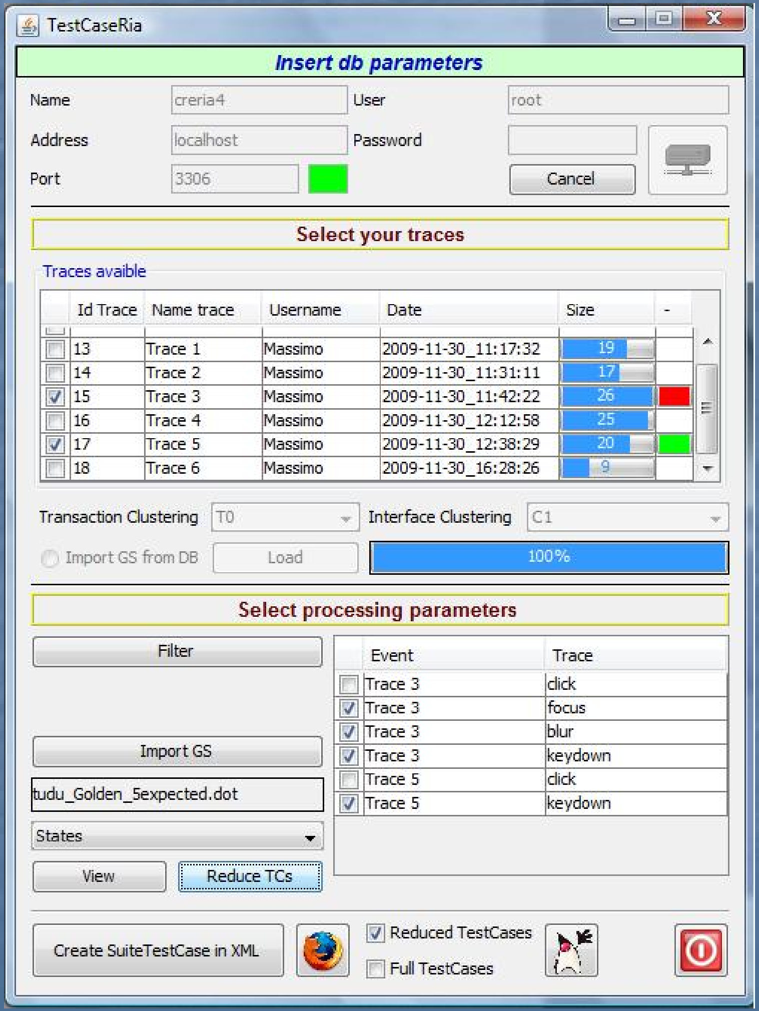
T0 Suggestion: New Transition

T1C1 Suggestion	T1C2 Suggestion	T1C3 Suggestion	T1C4 Suggestion
New Transition	New Transition	New Transition	New Transition

T2C1 Suggestion	T2C2 Suggestion	T2C3 Suggestion	T2C4 Suggestion
New Transition	New Transition	New Transition	New Transition

```
- DB connection closed -
- DB connection open -
DOM description completed
Event click handler terminated
click Bubbling phase captured
- DB connection open -
- DB connection open -
```

TestCaseRia GUI: A snapshot



TestCaseRia

Insert db parameters

Name	creria4	User	root
Address	localhost	Password	
Port	3306 🟩		Cancel

Select your traces

Traces avaible

	Id Trace	Name trace	Username	Date	Size	-
☐	13	Trace 1	Massimo	2009-11-30_11:17:32	19	
☐	14	Trace 2	Massimo	2009-11-30_11:31:11	17	
☑	15	Trace 3	Massimo	2009-11-30_11:42:22	26	🟥
☐	16	Trace 4	Massimo	2009-11-30_12:12:58	25	
☑	17	Trace 5	Massimo	2009-11-30_12:38:29	20	🟩
☐	18	Trace 6	Massimo	2009-11-30_16:28:26	9	

Transaction Clustering: T0
Interface Clustering: C1

○ Import GS from DB Load 100%

Select processing parameters

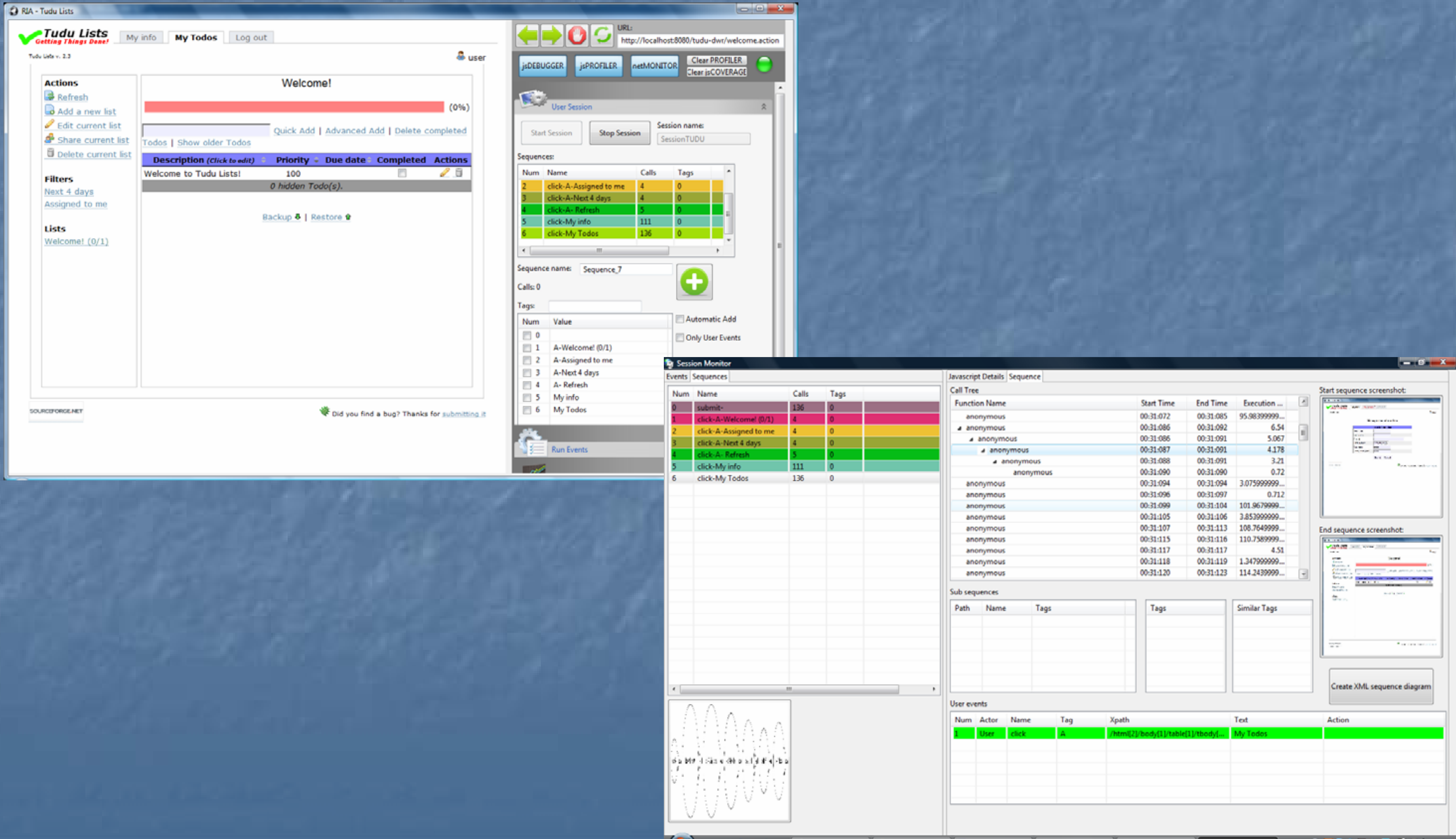
	Event	Trace
☐	Trace 3	click
☑	Trace 3	focus
☑	Trace 3	blur
☑	Trace 3	keydown
☐	Trace 5	click
☑	Trace 5	keydown

tudu_Golden_5expected.dot

States

☑ Reduced TestCases
☐ Full TestCases

DynaRia snapshots



Description (Click to edit)	Priority	Due date	Completed	Actions
Welcome to Tudu Lists!	100		□	
0 hidden Todo(s).				

URL: http://localhost:8080/tudu-dwr/welcome.action

Session name: SessionTUDU

Sequences:

Num	Name	Calls	Tags
2	click-A-Assigned to me	4	0
3	click-A-Next 4 days	4	0
4	click-A- Refresh	5	0
5	click-My info	111	0
6	click-My Todos	136	0

Sequence name: Sequence_7

Calls: 0

Tags:

Num	Value
□ 0	
□ 1	A-Welcome! (0/1)
□ 2	A-Assigned to me
□ 3	A-Next 4 days
□ 4	A- Refresh
□ 5	My info
□ 6	My Todos

□ Automatic Add

□ Only User Events

Num	Name	Calls	Tags
0	submit-	136	0
1	click-A-Welcome! (0/1)	4	0
2	click-A-Assigned to me	4	0
3	click-A-Next 4 days	4	0
4	click-A- Refresh	5	0
5	click-My info	111	0
6	click-My Todos	136	0

Call Tree

Function Name	Start Time	End Time	Execution ...
anonymous	00:31:072	00:31:085	95.98399999...
anonymous	00:31:086	00:31:092	6.54
anonymous	00:31:086	00:31:091	5.067
anonymous	00:31:087	00:31:091	4.178
anonymous	00:31:088	00:31:091	3.21
anonymous	00:31:090	00:31:090	0.72
anonymous	00:31:094	00:31:094	3.075999999...
anonymous	00:31:096	00:31:097	0.712
anonymous	00:31:099	00:31:104	101.9679999...
anonymous	00:31:105	00:31:106	3.853999999...
anonymous	00:31:107	00:31:113	108.7649999...
anonymous	00:31:115	00:31:116	110.7589999...
anonymous	00:31:117	00:31:117	4.51
anonymous	00:31:118	00:31:119	1.347999999...
anonymous	00:31:120	00:31:123	114.2439999...

Sub sequences

Path	Name	Tags

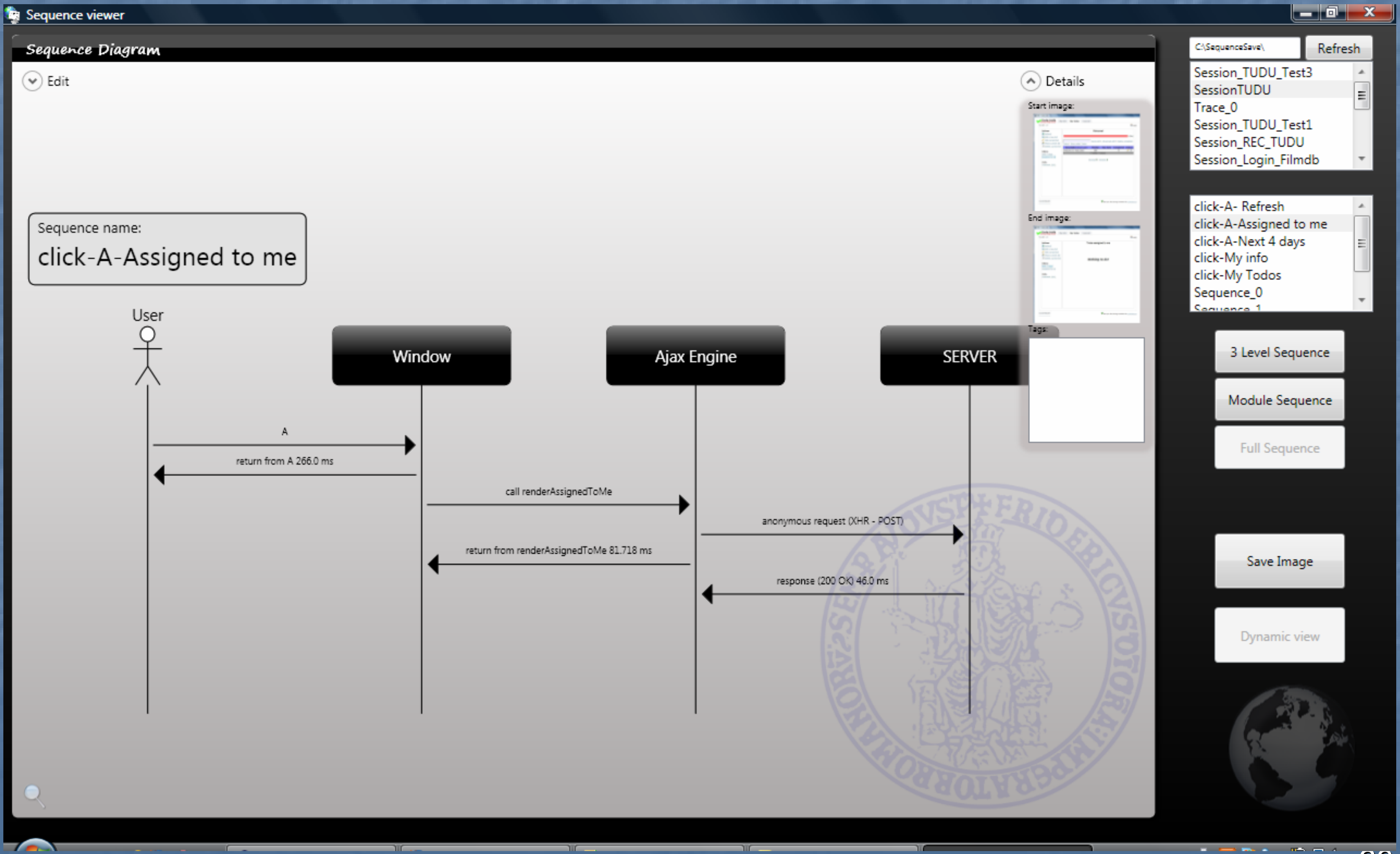
Tags

Similar Tags

User events

Num	Actor	Name	Tag	Xpath	Text	Action
1	User	click	A	/html[2]/body[1]/table[1]/tbody[...	My Todos	

JS function call-Network request in a sequence diagram model (3 – level view)



Sequence viewer

Sequence Diagram

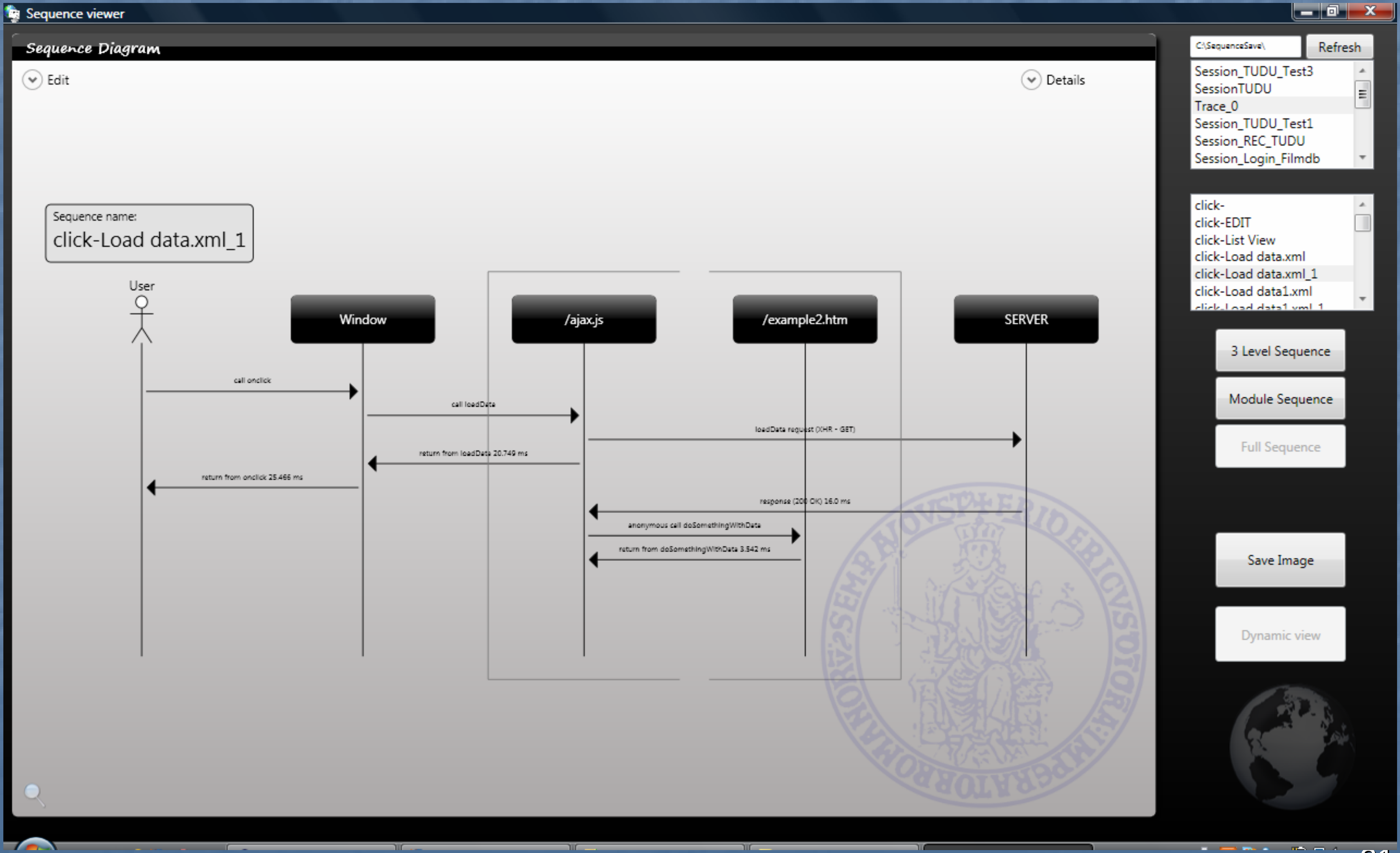
Sequence name: click-A-Assigned to me

C:\SequenceSave\

Session_TUDU_Test3
SessionTUDU
Trace_0
Session_TUDU_Test1
Session_REC_TUDU
Session_Login_Filmdb

click-A- Refresh
click-A-Assigned to me
click-A-Next 4 days
click-My info
click-My Todos
Sequence_0
Sequence_1

JS function call-Network request in a sequence diagram model (multi – level view)



Sequence name:
click-Load data.xml_1

Conclusions

- We wait for your feedback about these ideas.
- We would like to present some of these ideas in a position paper for TESTBEDS
- We are trying to obtain some preliminary experimental data to discuss at TESTBEDS.
- We may present some of our tools either in a tool demo in TESTBEDS, or in a slideshow to share via Web.