Techniques and Tools for Rich Internet Applications Testing

Domenico Amalfitano

Anna Rita Fasolino

Porfirio Tramontana

Dipartimento di Informatica e Sistemistica

University of Naples Federico II, Italy

RIA interface testing

- RIAs represent the new generation of Web applications, providing richer, more interactive, dynamic and usable user interfaces than traditional ones.
  - AJAX (Asynchronous JavaScript and XML) is a set of technologies (JavaScript, XML, XMLHttpRequest objects) providing one of the most diffused approach for implementing RIAs.

- But the enhancements provided by RIAs have a price in terms of interface complexity and testability

- Finding processes and techniques for an efficient and effective testing of RIA interfaces is a very challenging task

From hypertexts to RIAs

- Hypertexts consist of a set of static HTML pages
- Web 1.0 Applications (in particular Thin Client Web Applications) are client-server applications in which the server side manages all the application logic
- In RIAs, a large portion of the application logic is managed on the client side and an important amount of code is devoted to the user interface management

RIA User Interface

- It is implemented by Web pages composed by individual components, which can be updated, deleted or added at run time independently.
  - The manipulation of the page components is performed by an Ajax engine written in JavaScript
- Its status changes due to Javascript event handlers elaborations.
  - Event Handlers are triggered by user events or other external events (such as timeout events or asynchronous responses from the server).

- The RIA User Interface can be considered like a dynamically changing event-driven software system (EDS).

Some RIA Testing Open issues

- How the artifacts to be tested can be modelled?
  - Artifacts can be dynamically implemented and modified, so the model cannot be statically defined
- How test cases can be automatically generated?
  - How oracles can be defined?
- How test cases can be automatically replayed?
  - How oracles can be evaluated?
- What kind of tools can supporting testing processes?
- What kind of defects can be automatically tested?

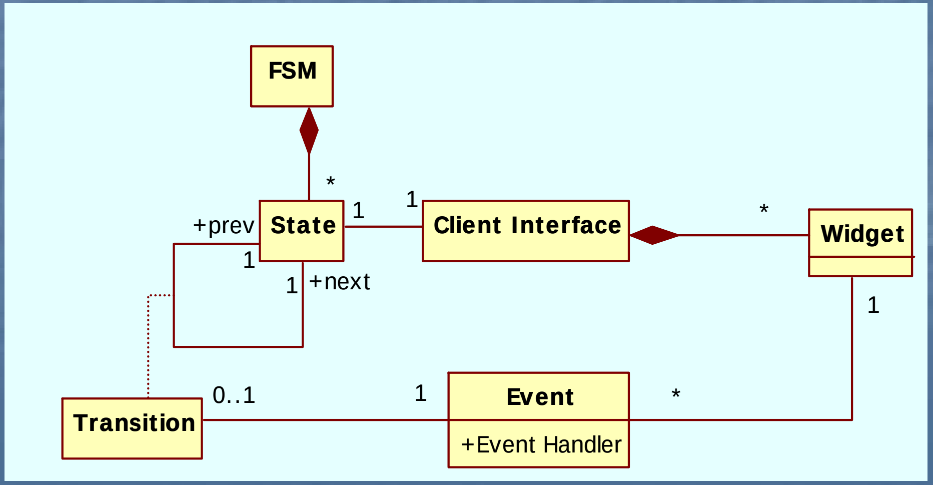
- A set of models, techniques, tools and processes will be presented and discussed in order to categorize possible solutions to this open issues

RIA Interface

- Event based testing techniques of GUI of desktop applications are based on models such as:
  - Event Flow Graphs.
  - Event Interaction Graphs.
  - Finite State Machines.
- Due to the similarities between RIAs and GUIs, all these models may be used to model the behaviour of an RIA.

Finite State Machine Model

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



- In a first approximation, different Client Interfaces correspond to different States
  - The number of possible states may be unlimited !
- A more useful model will consider 'equivalent' Client Interfaces as the same State

FSM Reverse Engineering Technique

- In the past, we have proposed a reverse engineering approach for obtaining the FSM that:
  - Is based on the analysis of the execution traces of the RIA;
  - Solves the problem of FSM states and transitions explosion by heuristic criteria that aim at clustering equivalent states and transitions of the FSM.
    - In example, two client interfaces may be considered equivalent if they contain the same set of containers with the same ID attribute value and containing the same type of widgets

1. D. Amalfitano, A. R. Fasolino, P. Tramontana, Reverse Engineering Finite State Machines from Rich Internet Applications, (WCRE 2008).
2. D. Amalfitano, A. R. Fasolino, P. Tramontana, Experimenting a Reverse Engineering Technique for Modelling the Behaviour of Rich Internet Applications, (ICSM 2009).
3. D. Amalfitano, A. R. Fasolino, P. Tramontana, An Iterative Approach for the Reverse Engineering of Rich Internet Applications, (IARIA ICIW 2010).

Test case generation techniques

- A possible approach for test cases generation based on dynamic analysis consists in the collection of execution traces and in their transformation to test cases
- Execution traces may be collected:
  - By a manual approach based on the analysis of the interactions with an RIA of real users or testers;
  - By an automatic approach based on Crawling techniques;
  - By mixing manual and automatic approaches

Manual vs Automatic Test Case Generation

- Manual approaches can produce test cases that actually reproduces the real user behaviours from execution traces
- The automatic approaches can be executed in a completely automated way

- Both approaches produces very huge test suites
  - But they should be the only practicable approaches in absence of documentation and/or the server side source code
  - Techniques can be proposed for stopping the execution traces collection and for reducing the test suite size on the basis of model/code coverage criteria
    - State coverage, Transition coverage, Path coverage
    - LOC coverage, Function coverage, Script coverage

D. Amalfitano, A. R. Fasolino and P. Tramontana, "Rich Internet Application Testing Using Execution Trace Data," Third International Conference on Software Testing, Verification, and Validation Workshops, 2010 (TESTBEDS 2010), IEEE CS Press, pp.274-283

Testing Oracle

- The automatic definition of oracles is possible only in some cases:
  - Invariant Testing
    - The oracle can be expressed by means of queries on the interface
  - Crash Testing
    - The oracle consists of the absence of raised exceptions
  - Regression Testing
    - The oracle is the equivalence with the interface obtained by the original version of the mutated application
- Otherwise, oracles must be manually defined for each test case

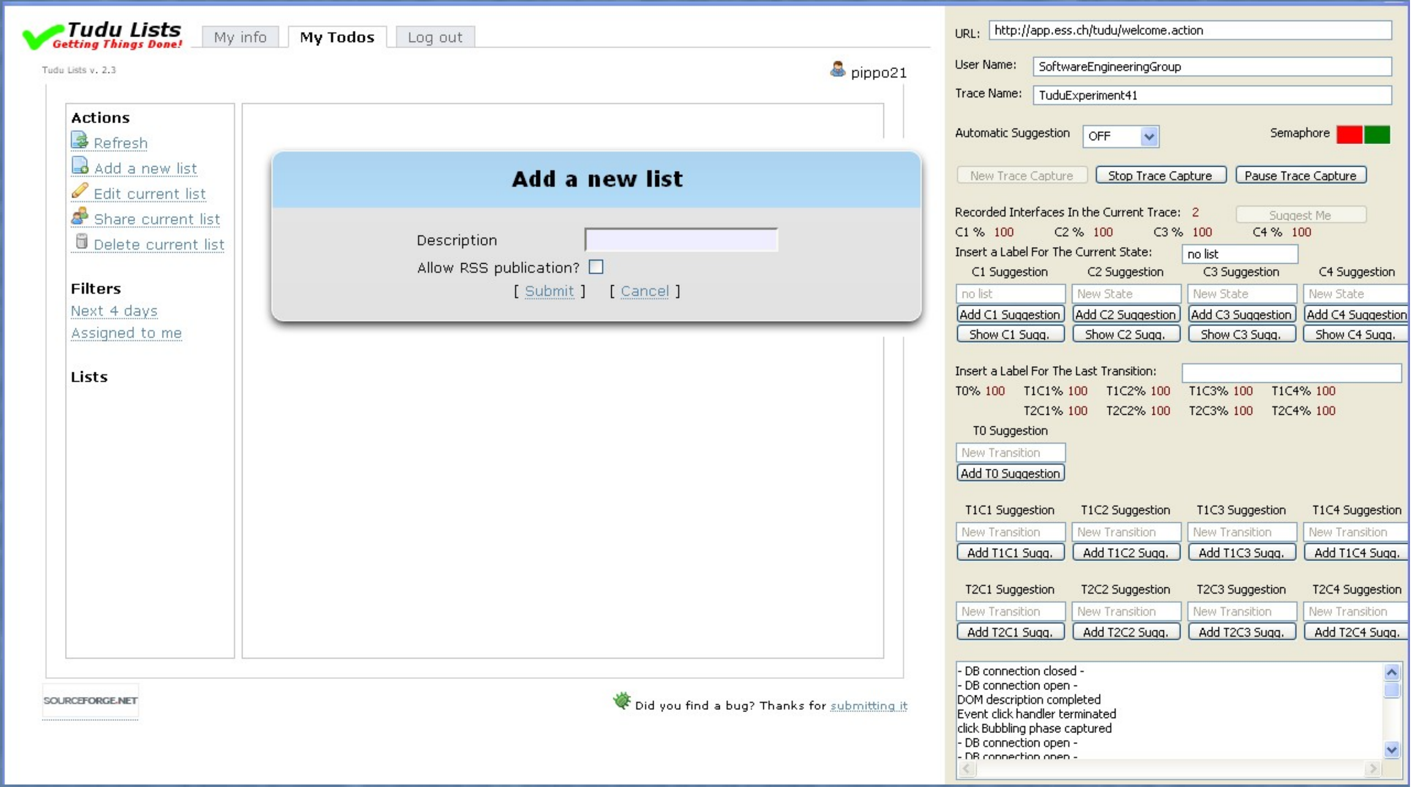
- Objective oracles can be automatically evaluated
  - E.g. oracles defined in terms of XPath queries
- Subjective oracles must be manually evaluated
  - E.g.: some usability or accessibility issues are usually manually evaluated

Testing Automation Tools Categories

- Crawler
  - That automatically navigates RIA interfaces
- Capturer
  - That collects execution traces by observing the user navigation
- Test Case Generator
  - That extracts and encodes test cases from execution traces
- Test Suite Reducer
  - That reduces the size of a test suite trying to not reduce its effectiveness
- Test Case Replayer
  - That automatically replays encoded test cases
- Assertion Generator
  - That automatically generates oracles
- Assertion Verifier
  - That automatically verifies oracle assertions

CReRIA

- Supports the user during the execution traces collection
- Produces the FSM on the basis of the collected execution traces and of the chosen clustering heuristics



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: no list; C2 Suggestion: New State; C3 Suggestion: New State; C4 Suggestion: 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, T1C2, T1C3, T1C4 Suggestion: New Transition

T2C1, T2C2, T2C3, T2C4 Suggestion: 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 -
```

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

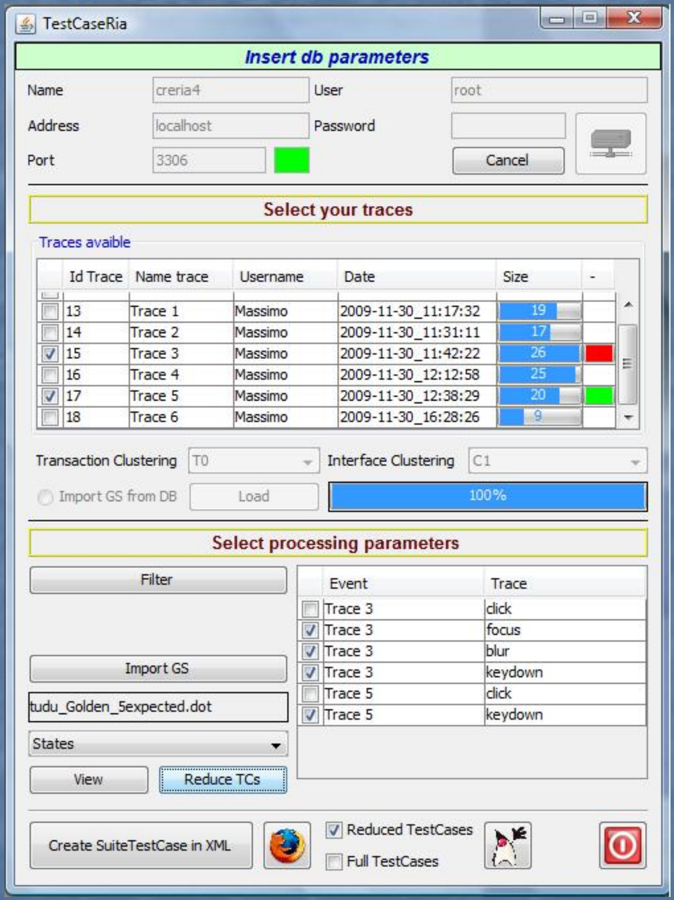
CrawlRIA

- Explores the states of the interface of a RIA by triggering the events related to the widgets of the interface, with a depth first or a breadth first strategy
  - Tree branches are cut when an interface equivalent to another already visited one is encountered
    - Heuristic clustering criteria are used to evaluate the equivalence between the interfaces
- Produces a set of execution traces, all starting from the same starting state

CrawlRIA is currently under restructuring and it will be online soon!

TestRIA

- Transforms traces in executable test cases (as Selenium executable tests)
- Reduces test suites according to some FSM model coverage criteria
- Supports assertion generation
- Replays encoded test cases


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

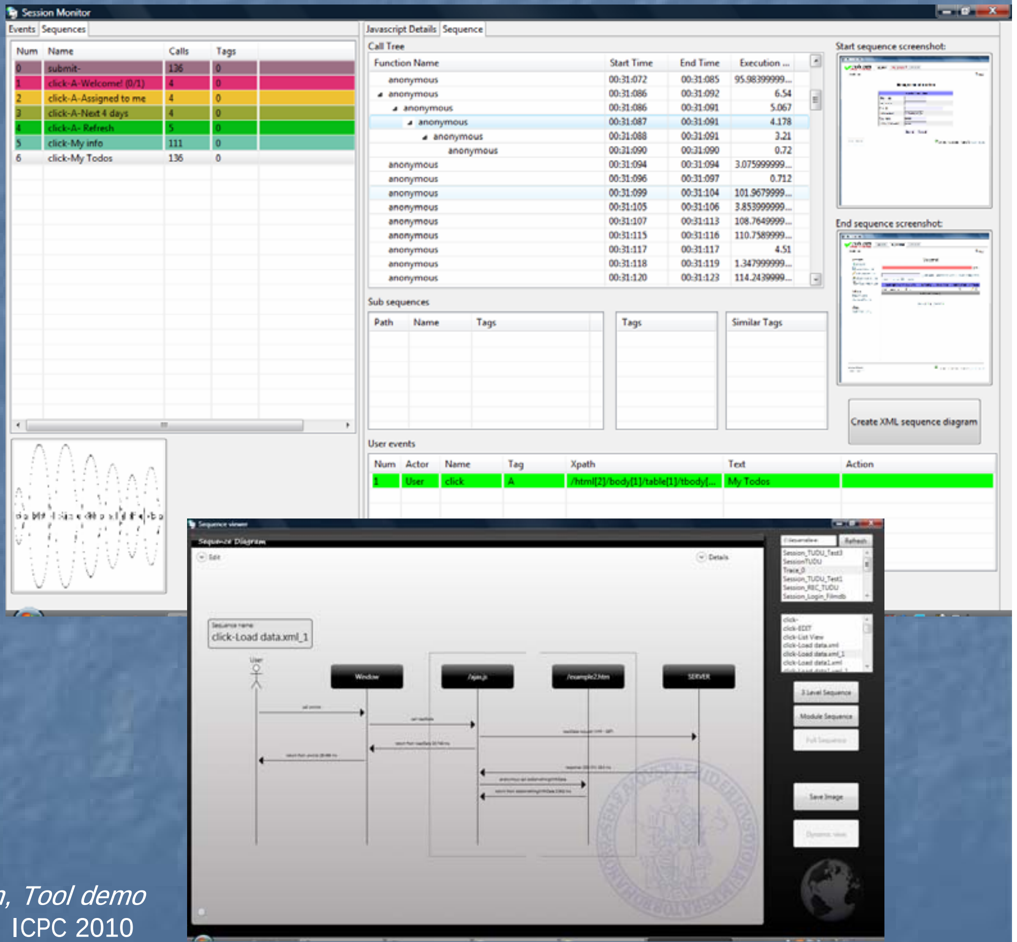
☐ Full TestCases

D. Amalfitano, A. R. Fasolino and P. Tramontana, "Rich Internet Application Testing Using Execution Trace Data," Third International Conference on Software Testing, Verification, and Validation Workshops, 2010 (TESTBEDS 2010), IEEE CS Press, pp.274-283

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

DynaRIA

- Dynamically captures information about the execution sessions of a RIA
- Produces metrics, static and dynamic views of the RIA interfaces
- Monitors JS crashes, network exception and other fault causes
- Replays execution sessions



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

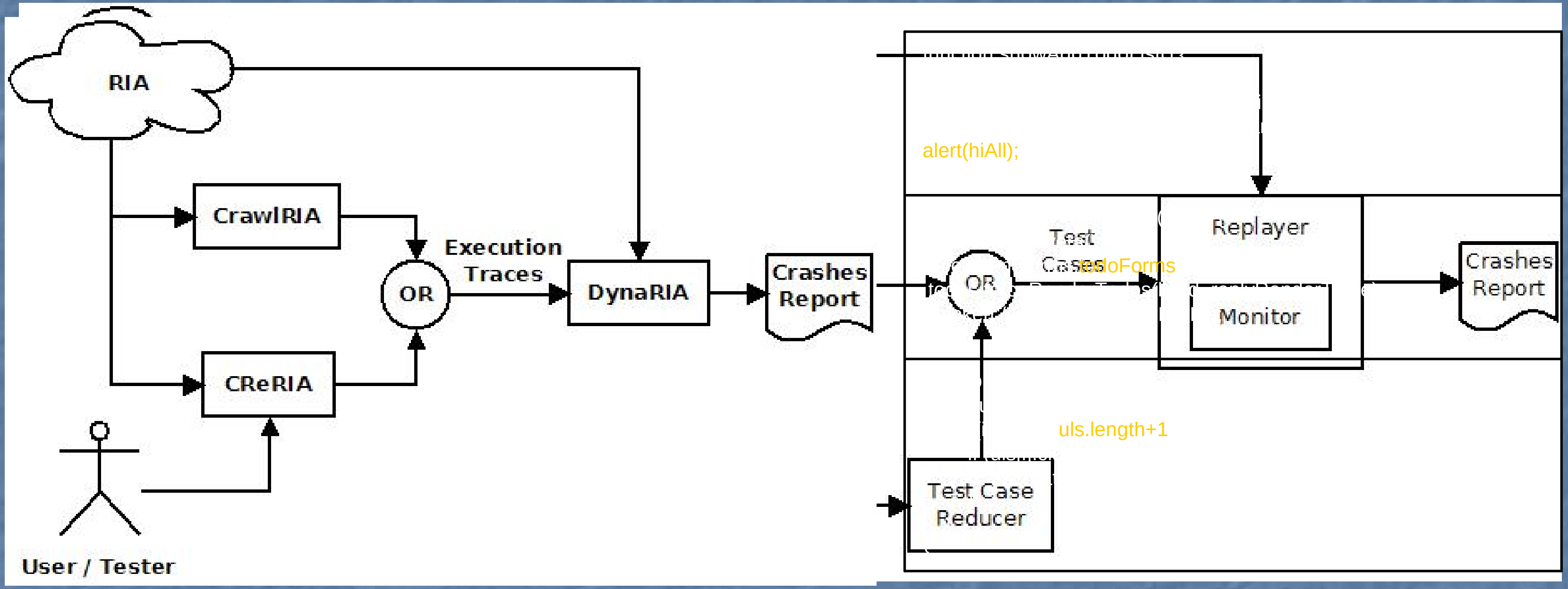
User events

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

Domenico Amalfitano, Anna Rita Fasolino, Armando Polcaro, Porfirio Tramontana, *DynaRIA: a Tool for Ajax Web Application Comprehension, Tool demo at the 18th IEEE International Conference on Program Comprehension*, ICPC 2010

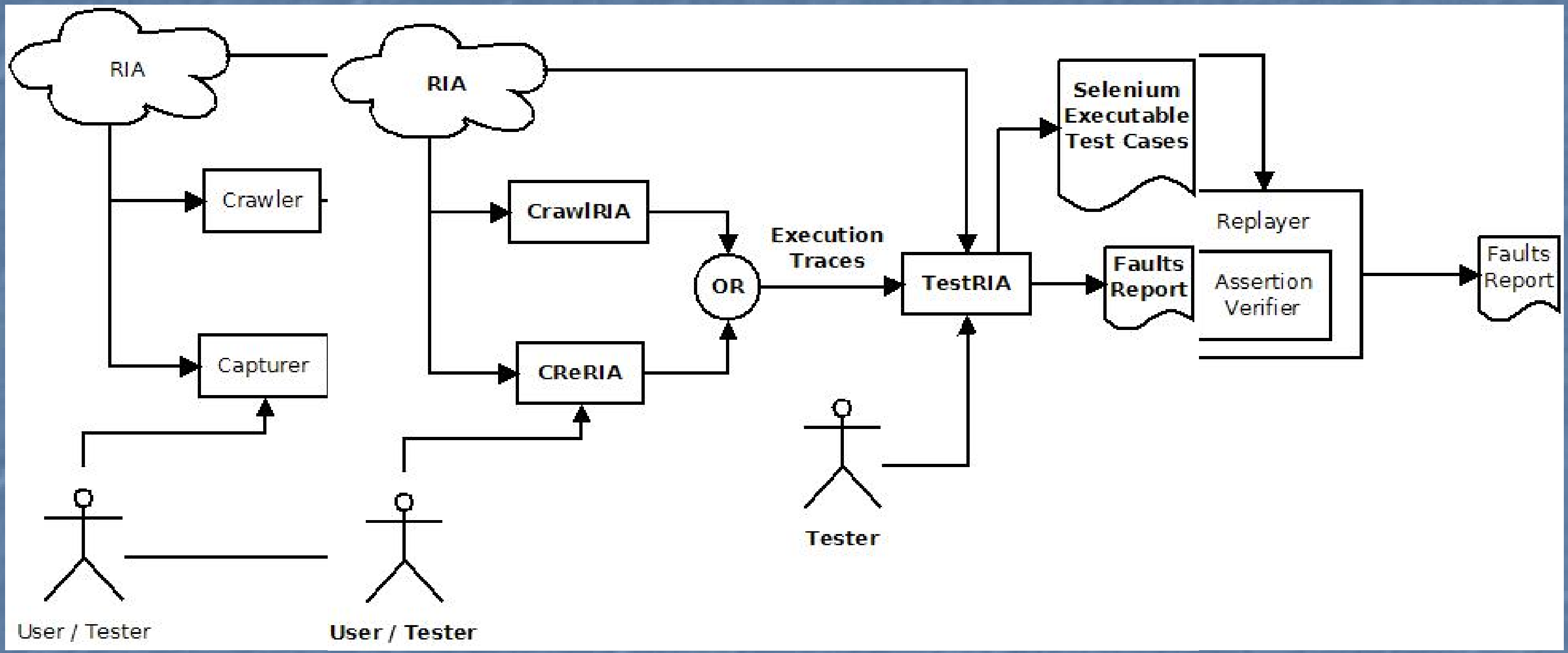
Available from http://wpage.unina.it/ptramont/downloads.htm

Putting all together: Crash Testing Process



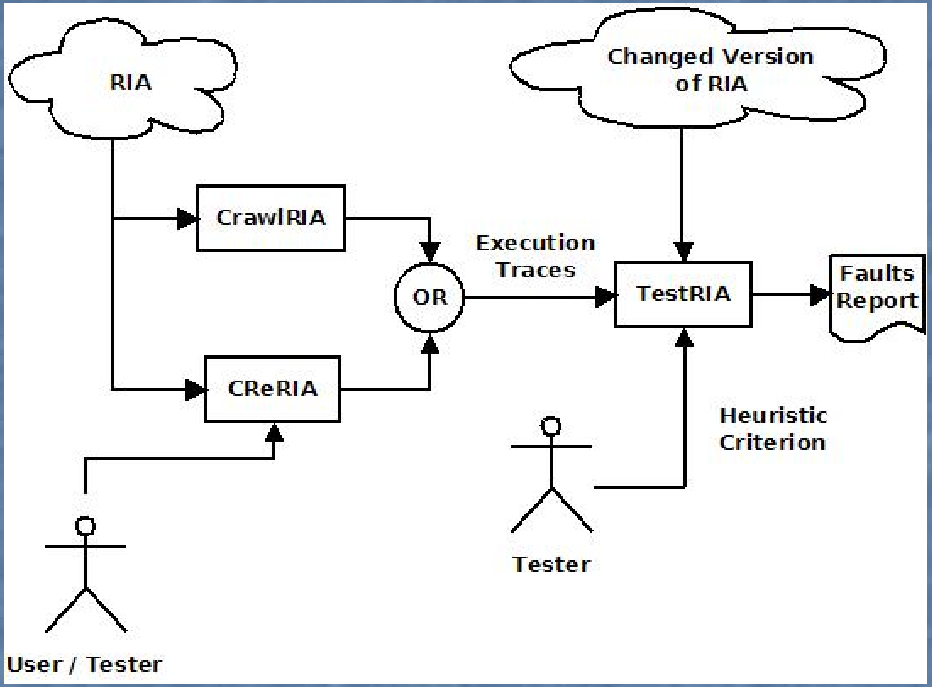
- Examples of faults that can be discovered:
  - JS unreferenced objects
  - Array out of bound

Putting all together: User Visible Fault Testing Process



- Examples:
  - Invariant Testing
    - Accessibility testing
    - Error page detection
  - Application Invariant Testing
  - Oracle testing
  - …

Putting all together: Regression Testing Process



- The oracle is automatically defined in terms of equivalence with the interface obtained by executing the same test case on the original RIA version

Conclusions

- The RIA testing is a relatively new but terribly challenging field
- We have proposed some processes supporting the testing automation in a client side reverse engineering scenario without no access to server side code and documentation
  - The processes are supported by a set of tools that we are developing and that are freely available on my website at http://wpage.unina.it/ptramont

Future works

- Execution of larger case studies and/or empirical studies in order to evaluate effectiveness and efficiency of testing processes by varying:
  - Number and type of users
  - Applications
  - Reduction techniques
  - The type of defects (by using a suitable fault model and fault injection techniques)
- Comparison with other emerging approaches/tools, such as the ones of the SERG group in Delft and of IRST in Trento

Any questions?