Using Dynamic Analysis for Generating End User Documentation for Web 2.0 Applications

Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli Federico II
Napoli, Italia
domenico.amalfitano@unina.it, anna.fasolino@unina.it, porfirio.tramontana@unina.it

***Abstract*— The relevance of end user documentation for improving usability, learnability and operability of software applications is well known. However, software processes often devote little effort to the production of end user documentation due to budget and time constraints, or leave it not up-to-date as new versions of the application are produced. In particular, in the field of Web applications, due to their quick release time and the rapid evolution, end user documentation is often lacking, or it is incomplete and of poor quality. In this paper a semi-automatic approach for user documentation generation of Web 2.0 applications is presented. The approach exploits dynamic analysis techniques for capturing the user visible behaviour of a web application and, hence, producing end user documentation compliant with known standards and guidelines for software user documentation. A suite of tools support the approach by providing facilities for collecting user session traces associated with use case scenarios offered by the Web application, for abstracting a Navigation Graph of the application, and for generating tutorials and procedure descriptions. The obtained documentation is provided in textual and hypertextual formats. In order to show the feasibility and usefulness of the approach, an example of generating the user documentation for an existing Web application is presented in the paper.**

Keywords: *Rich Internet Applications; Reverse Engineering; Dynamic Analysis; Finite State Machine; Iterative Comprehension Processes; User Documentation Generation*

I. INTRODUCTION

According to the ISO/IEC 9126 Standard on Software Quality [19], software usability depends on several sub-characteristics of software, such as its learnability, i.e. the capability of the software product to enable the user to learn its application, and its operability, i.e. the product capability to enable the user to operate and control it.

The relevance of end user documentation for improving learnability and operability of software applications is well known [27]. Differently from technical software documentation that is intended to software developers, testers or maintainers and describes software from its internals, end user documentation shows how to use a software application and may include user guides, reference guides, help files, tutorials and walkthroughs which explain how to accomplish certain tasks. The IEEE Standard 1063 for Software User Documentation [18] describes minimum requirements for the structure, information content, and format of user documentation.

According to the IEEE 1063 Standard, user documentation should be *complete* and describe all the critical use cases offered by the application, as well as all the associated interaction scenarios. At the same time, documentation shall be *accurate* and reflect the functions and results of the applicable software version. Moreover, the standard recommends including explanations about all the known problems in using the software in sufficient detail such that the users can either recover from the problems themselves or clearly report the problem to technical support personnel. Moreover, reference mode documentation shall include each error message with an identification of the problem, probable cause, and corrective actions that the user should take. Eventually, the standard provides possible structures and formats of user documentation.

It can be deduced that obtaining complete, accurate, and effective user documentation is not a trivial task. Real software development processes often devote little effort to the production of end user documentation due to budget and time constraints, or leave it not up-to-date as new versions of the application are produced. In particular, in the field of Web applications, due to their quick release time and the rapid evolution, end user documentation is often lacking, or it is incomplete and of poor quality. Nowadays, due to the fast and growing diffusion of the Web 2.0, this problem is particularly true with Rich Internet Applications (RIAs). RIAs, indeed, with the enhanced dynamicity, responsiveness, and interactivity of their user interfaces are more and more similar to desktop applications, being able to offer more complex and richer functionalities [16]. As a consequence, richer and accurate end user documentation is absolutely needed for RIAs too.

To save on time and costs for developing this documentation, end user documentation tools that provide facilities to automate some or all of the often laborious tasks associated with creating an application's documentation can be used. End user documentation tools include screen casting software and authoring tools. Screen casting tools [30] are used to record activities on the computer screen, mouse movement and are suitable for recording demonstrations, remote technical assistance, sales presentations, and training. Authoring tools are computer based systems that allow a general group (including non-programmers) to create (i.e., author) content for intelligent tutoring systems.

Unfortunately, these tools are just able to record the workflow needed for accomplishing given tasks (presenting the sequence of screens shown to users and the actions that must be performed by users on these screens) and to transform it into procedure descriptions by means of editing functions, but are not able to provide any other information about the overall application behavior. As an example, with respect to rich Web applications, these tools have no facilities for generating site maps, navigational trees of the site, overview descriptions, or any other information that vice-versa may be obtained by dynamic analysis of the application. Reverse engineering techniques based on dynamic analysis of RIAs have been recently proposed in the literature for obtaining a Finite State Machine model of the user interface of the Web application [1, 2, 4]. These techniques have been exploited in the context of comprehension [5] and testing processes [3, 6], but the FSM model they produce can be considered a suitable model for describing the navigation of the application too.

In this paper, we propose of using the reverse engineering techniques proposed in [4] with the aim of generating end-user documentation for RIAs. In particular, we present a documentation process of RIAs that is based on a reverse engineering technique that extends the one proposed in [4], and a supporting tool. The tool provides facilities for collecting user session traces associated with use case scenarios offered by the Web application, for abstracting a Navigation Graph of the application, and for generating tutorials and procedure descriptions. The obtained end user documentation is provided in textual and hypertextual formats and is compliant with some indications provided by the IEEE Standard for Software User Documentation [18].

The paper is organized as it follows. Section II presents related work on documentation processes, end user documentation and reverse engineering of RIAs. Section III presents a model of end user documentation of RIAs that can be generated by the documentation generation process presented in section IV. Section V describes the tool supporting this process. In Section VI an example of generating the user documentation for an existing Web application is shown, while Section VII provides conclusions and discusses future work.

II. Related Work

Software Documentation is a relevant part of a software product [17] but it is often neglected in software development processes. Usually, software engineers operate under the pressure of strict schedules and deadlines and do not devote much time to the production of documentation. In these conditions, tools for the automatic generation of technical documentation about the code, like JavaDoc [20] or Doxygen [13] that create online documents by extracting text from specially formatted comments can be used, or reverse engineering [11] techniques and tools can be exploited for post-generating technical documentation after the development.

End user documentation is usually even more overlooked, being usually produced manually or at least using some tool of user documentation generation based on screen-casting or authoring techniques. As an example, Zhang et al. [31] propose SmartTutor an environment for creating IDE-based interactive tutorials via editable replay. This tool is proposed to support programmers in the task of learning software IDEs and its features are similar to the ones offered by Jtutor [21], a tool designed to create and replay code-based tutorials in Eclipse. Other similar solutions are provided by DocWizards [8], a follow-me documentation wizard system in which the procedures are authored through demonstration as well as by manual editing, and by EpiDocx [14], a commercial tool for tutorial generation and maintenance of Windows-based applications.

However, all these tools just limit themselves to capture and record the procedures needed for accomplishing user tasks, but they are not able to do any data mining from the observed application behaviour. As an example, they have no feature for automatic classification of shown user interfaces (or user events) based on their similarity and for associating them with a unique meaningful description. In addition, they have no feature for cross-referencing similar interfaces or events belonging to different user tasks.

As to the field of traditional Web applications, while several reverse engineering techniques and tools have been proposed for obtaining technical documentation about the applications [7, 9, 12, 26, 29] (such as high level and low level design models), to the best of our knowledge no specific reverse engineering solution has been defined for obtaining end-user documentation.

With respect to Web 2.0 applications, some feasible solutions have been proposed in the literature for obtaining Finite State Machines that model user interactions with Rich Internet Applications [2, 22, 24]. At the moment, these models have been used for the aims of comprehending the behaviour of a RIA from the user point of view [1], crawling the application [24], or testing it [3, 23]. However, these models may provide a suitable starting point for obtaining end user documentation of the Web application too.

III. End User Documentation of Web 2.0 Applications

Sommerville states that end user documentation should be prepared for different classes of users and different levels of user expertise, and suggests five types of documents (or five chapters) with different audience and levels of detail to be delivered with the software system. These documents include: Functional descriptions of services provided, Installation document, Introductory manual for getting started with the system, Reference manual including details of all system facilities and System administrators guide [27].

Analogously, IEEE Standard 1063 for Software User Documentation [18] recommends including both instructional mode (to learn about software) and reference mode documentation (to refresh the user memory about it). *Instructional* mode documentation should include procedures structured according to user's task, while *Reference mode* documentation should be arranged to facilitate random access to individual units of information. The Standard also suggests ways of organizing chapters and topics to facilitate learning.

According to these suggestions, we have decided to organize the end user documentation of a Rich Internet Application in three main parts: 1) an Introductory manual for getting started with the system, that provides an overall description of user tasks; 2) a Tutorial showing detailed descriptions of single user tasks; 3) a Reference guide showing explicative materials about screens shown to users and user actions to be performed during task executions.

In particular, the Introductory manual will be based on a Navigation Graph showing how the application allows its user to access all its functions. The Tutorial is a set of more detailed views and descriptions about each user function execution. There will be also traceability relationships between the various parts of the documentation that will be implemented by means of hyper-textual files.

In the following we provide further details about the documentation items.

A. *Introductory Manual*

In traditional Web applications the *navigation model* is one of the most important aspects of the application to be communicated to application users. This model is usually given by a navigation graph with nodes representing Web pages and edges representing direct transitions between Web pages. A navigation graph of static Web applications can be obtained in a straightforward manner by means of spiders or link checkers, while obtaining this graph for dynamic Web applications requires more sophisticated approaches for dealing with the problem of page explosion and the request generation problem [29].

With respect to Rich Internet Applications that can be considered as a hybrid between a Web application and a desktop application [16], obtaining a navigational model is more complicate. Indeed, the user interface of RIAs is not implemented by traditional Web pages having different URIs and interconnected by hyperlinks, but it is usually associated with a single Web page whose state changes depending both on events triggered on the user interface of the application and on various types of external or asynchronous event. As a consequence, a suitable navigational model of the application is given by Finite State Machines (FSMs) that represent the various states of the user interface and the transitions between them.

In particular, in the proposed Navigation Graph we assume that user interface states with similar structure are represented as a single node, while edges between nodes represent transitions due to user events that caused the user interface state to change. Moreover, selected paths belonging to this graph will show possible execution scenarios of use cases offered by the RIA.

Figure 1 shows an excerpt of a Navigation Graph for an example Web application. The graph shows five nodes and nine edges associated with transitions between user interface states.

We propose the Introductory Manual of end user documentation of a RIA to include the Navigation Graph and an index of all the user functions offered to various classes of actors of the application. Moreover, each user function will be cross-referenced to the Navigation Graph paths that describe the corresponding user interactions with the application, representing both normal and exceptional execution scenarios.

To obtain this graph, the reverse engineering technique proposed by Amalfitano et al. in [4] will be used. The technique is based on dynamic analysis of the application and is supported by the CReRIA tool that exploits data collected from user sessions for abstracting this model. Further details about the technique are provided in Section IV.

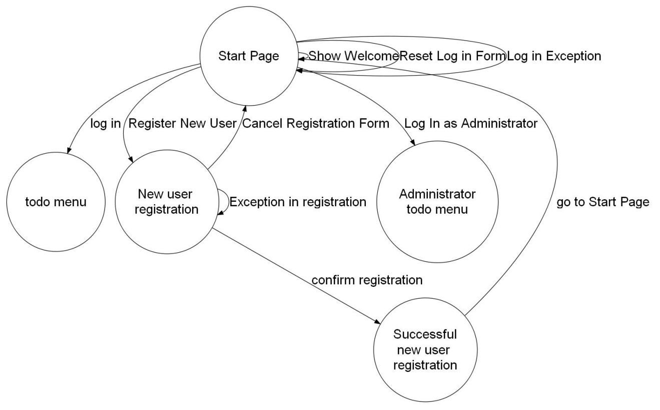


Figure 1. An example of Navigation Graph corresponding to the Login and Registration use cases of a Web Application

B. *Tutorial Documentation and Reference Guide*

To show the procedures needed for accomplishing user tasks, the proposed end user documentation will include a Tutorial section reporting operational descriptions of the scenarios of each use case offered by the Web application. The descriptions will be grouped on the basis of the actors involved in the use cases. For each actor's use case, both textual descriptions of the scenarios and the Navigation Graph paths associated with them will be reported, as well as the sequence of screen shots shown by the application during the scenarios execution will be illustrated. Screen shots will have labels reporting the meaning of the corresponding state of the execution and there will be explanations of input values and user events that need to be fired on the corresponding user interface.

As to the Reference Guide, it will be composed of detailed descriptions of all user interactions and user interfaces encountered during the execution of user tasks.

IV. THE DOCUMENTATION GENERATION APPROACH

The documentation generation process we propose in this paper exploits both user knowledge about the application, and information extracted and abstracted about the Web application by reverse engineering. The process will be based on three main steps: 1) Dynamic analysis of the Web application, 2) Navigation Graph Generation, 3) User Documentation Generation. The proposed process is supported by a reverse engineering tool and relies on a repository that stores both information obtained by reverse engineering and data annotations provided manually by the software engineer during the generation process. The process can be used to generate new user documentation incrementally, as well as to up-to-date existing documentation as the Web application evolves.

Figure 2 shows the proposed process, while the process steps are illustrated in the following subsections.

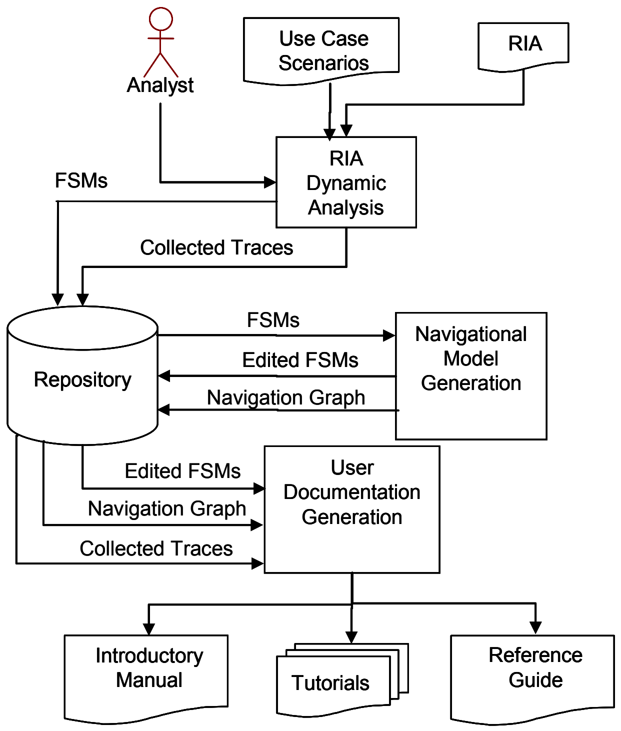


Figure 2. The documentation generation process

A. *Web application Dynamic Analysis*

The purpose of this activity is to record user sessions devoted to exercising the single use cases offered by the Web application. The software engineer who is in charge of generating the end user documentation will carry out this activity having the attention of exercising all the use case scenarios of the application that will have to be included in the user manual. This activity must be performed to reach two aims: a) obtaining user session traces that will be transformed into walkthroughs showing how given user tasks can be accomplished, and b) abstracting an FSM describing the behavior of the User Interface for each user session.

To record the execution traces and to obtain the corresponding FSMs, the RIA comprehension process proposed in [4] can be used, which assumes that an FSM can be obtained incrementally through the iterative steps of User Interaction, Extraction, Abstraction and Concept Assignment. This process is supported by the CReRIA reverse engineering tool and is illustrated in Figure 3.

The process starts with the *User Interaction* step where the user interacts with the RIA and fires an event on its current user interface: this interaction is performed in the controlled navigation environment offered by the CReRIA tool that observes and registers all the interactions and the needed information about them.

In the successive *Extraction* step, information about current interface, fired user event and user interface obtained after the event processing must be retrieved and persistently stored. Hereafter, we call each user interface captured during the dynamic analysis an 'Interface Instance', and each transition recorded during the interaction a 'Transition Instance'. Interface Instances and Transition Instances are captured and stored in the CReRIA tool repository.

The *Clustering* step is performed using some heuristic clustering criteria that evaluate the degree of similarity of the current user interface instance with the previously produced ones, as well as the similarity among occurred transitions. These clustering heuristics are described in detail in [2] and [4]. Using these criteria, clusters of similar interfaces, as well as clusters of similar transitions can be obtained. From here on in, we call each cluster of similar interface instances an 'Interface Class', and each cluster of similar transition instances a 'Transition Class'.

The *Concept Assignment* is actually a comprehension [10] and validation step where the software engineer has to validate the clustering proposed by the heuristic criteria and accepts or refuses it. If an interface (transition) clustering is refused, he has to propose the correct clustering of the interface instance (transition). In this way, the expert incrementally reconstructs a FSM modeling the behavior of the RIA GUI, since he either associates the current interface with a new interfaces class (and a new FSM state), or with an already existing interface class (and FSM state). Analogously, he associates the current transition either with a new class of transitions, or with an already existing one.

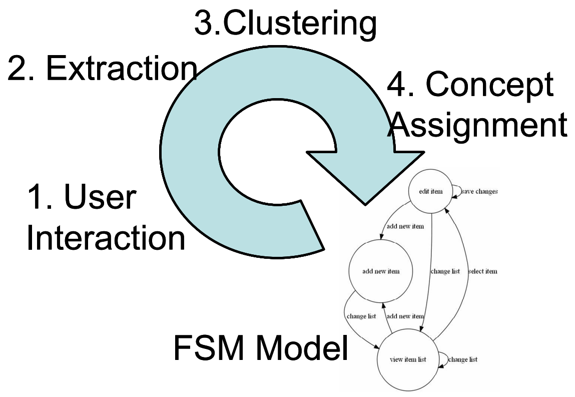


Figure 3. The iterative comprehension process of a RIA

Figure 4 shows the conceptual model of the information that is captured during the dynamic analysis step and that is used to build the Navigation Graph.

B. *Generation of the Navigation Graph*

During this step, the single FSMs obtained by dynamic analysis are merged and transformed into the Navigation Graph of the Web application that will be included in the Introductory Manual of the Web application. This step will be performed using the CReRIA tool: indeed, the software engineer will have to select the set of traces that he wants to include in the final user documentation and the tool will produce the overall Navigation Graph. In this graph, each node will represent a user interface class and will be labelled with the textual string provided to it in the Concept Assignment step. Analogously, each edge will be associated with a transition between states and will be labelled with the corresponding transition description.

The user will be able to edit this graph and associate each node or edge with additional textual annotations, or to correct the labels, if he considers them incorrect due to wrong past interpretations of Interfaces and Transitions.

At the end of this step, the Navigation Graph data will be stored in the tool repository.

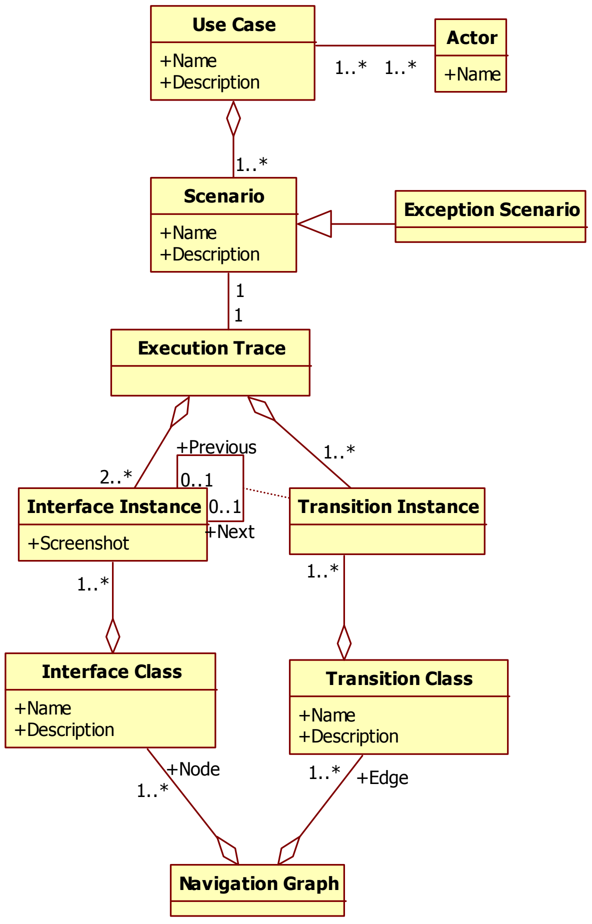


Figure 4. Information Model

C. *End User Documentation Generation*

This step will be devoted to the automatic generation of the end user documentation of the application and will be carried out by the CReRIA tool using the data stored in its repository. The documentation will include the Introductory Manual, the Tutorial Guide and a Reference Guide and will be provided both in textual and in hypertextual format.

The hypertextual documentation can be seen as an interactive site map and can be accessed both online (by publishing it on a Web server and linking it to the Web application) and offline, as a downloaded hypertext on a client machine. The hypertextual format provides links and shortcuts between the different pages in which the same items are cited. As an example, Interfaces and Transitions in the Navigation Graph are clickable and the link reaches the page containing the detailed description of the Interface or of the Transition.

V. THE CRERIA TOOL

The CReRIA tool (version 5) provides an integrated environment for dynamic analysis of Rich Internet Applications implemented with Ajax-based technologies whose main functionalities include:

- *incorporating* a Web browser (implemented with JavaXPCOM technology) for navigating the Rich Internet Application;
- *extracting* and *storing* in a Mysql database the relevant information about traced user sessions, such as user interfaces, events and transitions that occurred during the navigation;
- *capturing and storing* the screen shots of the navigated interfaces;
- *proposing* clustering of interfaces and transitions according to heuristic clustering criteria;
- *supporting* the Concept Assignment task on the basis of information collected or abstracted in the previous steps of the process;
- *supporting* the interactive navigation and editing of the Navigation Graph information, such as the collected scenario executions, screen shots and details of collected Interfaces and Transitions;
- *generating* user documentation in textual (rtf) or hypertextual (html, css, and javascript) format.

The original basic version of the CReRIA tool was born to exclusively support the FSM abstraction from RIAs and provided the former five functionalities of the above list. A more recent prototype version of the tool has been tailored for software re-documentation processes and implements the latter two functionalities too. This version is now being tested and, as soon as a more stable version will be available, it will be published online at http://wpage.unina.it/ptramont/Download/.

Using this tool, several versions of the user documentation can be generated by selecting different subsets of execution traces. As an example, it is possible to generate the user documentation only for the subset of use cases scenarios related to a given actor, or to generate documentation for a new release of the application by selecting only recently updated use cases and scenarios.

The basic version of CReRIA is available for download at http://wpage.unina.it/ptramont/Download/CReRia_4.0.zip

VI. An Example

In this section an example of using the proposed document generation approach will be described with the purpose to show the feasibility of the process and to present some details about the RIA documentation that it produced.

The involved application is an Ajax-based open source Web application called Tudu [28], available at http://www.julien-dubois.com/tudu-lists and offering functionalities for the management of lists of tasks (the so-called 'todos') such as adding, deleting, searching for todos, organizing lists of todos, and so on. Tudu provides an exemplar Rich Internet Application that has been frequently used for experimenting reverse engineering and testing techniques of RIAs [3, 4, 22, 23, 25]. Indeed, Tudu is a simple (but not trivial) RIA composed of about 10 KLOC, whose server side is implemented with Java/JSP technology, while its client side includes typical 'rich' pages that modify themselves at run-time on the basis of the user interaction with the pages.

As a consequence, for comprehending which are the user functionalities offered by Tudu and how they can be replayed, static analysis of its server pages does not suffice, while dynamic analysis must be carried out. Moreover, the user documentation retrievable on the Tudu website is very poor. Only a brief list of the 8 main use cases and 4 screen shots are reported in the Web page http://www.julien-dubois.com/tudu-lists/user-documentation.

Instead, on the basis of our past knowledge about Tudu, we know that it offers a wider set of use cases, including 23 use cases and 120 scenarios offered to three different actors, i.e. a Generic User "GU" (not yet logged in) (involved in 2 use cases), the Logged User "LU" (with 17 use cases) and the Administrator "A" (with 21 use cases, 17 of which are shared with the Logged User). The overall list of use cases, involved actors and # of related scenarios is reported in Table 1.

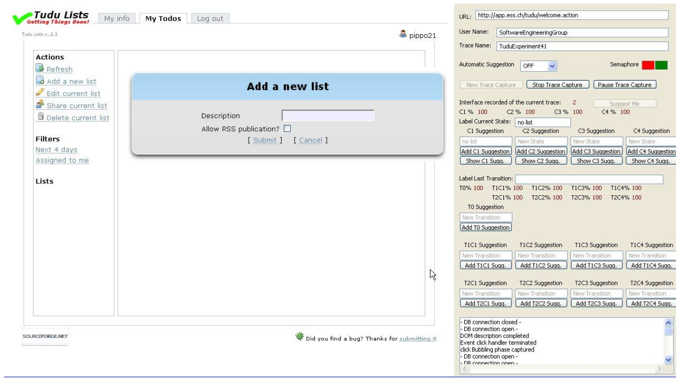
TABLE I. USE CASES OF TUDU LISTS

Use Case	Actor	Scenarios #
User Login	GU	5
Register a New User	GU	12
Quick Add of a Todo	LU & A	2
Advanced Add of a Todo	LU & A	11
Manage Completed Todos	LU & A	3
Filter Listed Todos	LU & A	6
Edit a Todo	LU & A	11
Delete a Todo	LU & A	2
Backup	LU & A	1
Restore	LU & A	4
Order Listed Todos	LU & A	3
Refresh	LU & A	1
Add New Todo List	LU & A	7
Open a Todo List	LU & A	2
Edit a Todo List	LU & A	5
Share a Todo List	LU & A	6
Delete a Todo List	LU & A	5
User Logout	LU & A	2
Show User Info	LU & A	16
User Monitoring	A	6
Configuration	A	4
Manage Users	A	5
Dump Database	A	1

In order to re-document the user functions provided by Tudu, we followed the process proposed in the paper.

In the first step of the process, one author performed dynamic analysis of Tudu and collected 119 Execution Traces exactly, corresponding to the known use case scenarios of the application. These execution traces included 425 Interface Instances and 306 Transition Instances. During the dynamic analysis, the concept assignment activity was performed and these instances were grouped into 42 Interface classes and 138 Transition classes.

Figure 5 reports the interface shown by the CReRIA tool during the execution of the clustering and concept assignment steps, where the clustering suggestions provided by the tool can be accepted or refused by the software engineer. In this case, the clustering suggestions were accepted 420 out of 425 times for the interface instances, and 278 out of 306 times for the transition instances. The overall dynamic analysis activity was accomplished in about ten hours, mainly devoted to the manual and the semi-automatic tasks of the user interaction and concept assignment. At the end of this step, 120 FSM models associated with the execution traces were obtained and stored in the tool repository.



URL: http://app.ess.ch/tudu/welcome.action
User Name: SoftwareEngineeringGroup
Trace Name: TuduExperiment41
Automatic Suggestion OFF
Interface recorded of the current trace: 2
C1 % 100 C2 % 100 C3 % 100 C4 % 100
Label Current State: no list
Label Last Transition:
T0% 100 T1C1% 100 T1C2% 100 T1C3% 100 T1C4% 100
T2C1% 100 T2C2% 100 T2C3% 100 T2C4% 100

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

Figure 5. An Example of CReRIA GUI during the execution of the comprehension process

In the second process step, the overall Navigation Graph of Tudu was generated by merging the available FSMs. This task was performed using the features of CReRIA of selecting the set of execution traces, navigating them possibly refining the concepts assigned with the related FSMs states and transitions, and finally adding extra-information to be included in the final documentation. Figure 6 shows some snapshots of the CReRIA tool during the execution of the tasks of trace selection, concept refinement, and extra-information editing.

As the lower part of Figure 6 shows, to support the concept refinement task, the CReRIA tool allows a user to select a FSM node and to get a view about the original associated screens of the RIA, as well as to get further data captured during dynamic analysis (see the right panel in the Figure).

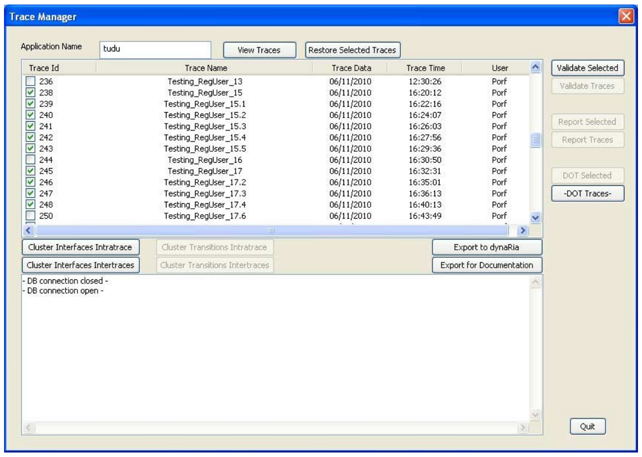
The Navigation Graph of Tudu obtained at the end of this second step is reported in Figure 7 (an extended and more clear version of this figure is downloadable from http://wpage.unina.it/ptramont/images/NavigationGraph.jpg).

This graph provides an overview about the complete application from the end user point of view. More useful maps will be shown in the following, including only the set of interfaces and transitions involved in a single use case or scenario.

In the third step of the process, the final user documentation was obtained. In particular, the overall Navigation Graph of Tudu was generated both in textual format and in hypertextual one. In the hypertextual format, the Navigation Graph was implemented by a HTML clickable map that allows the reader to click on graph nodes and edges and to jump to related pages describing additional details about each Interface and Transition.

As to the tutorial guide, it had to include separate descriptions of how each use case scenario can be executed. Each scenario description comprised:

- An explicative text describing the use case scenario;
- the list of actors involved in that scenario;
- the part of the Navigation Graph including only Interface nodes and edges involved in the scenario;
- the sequence of interface screen shots and the description of the user events needed in order to replicate the scenario execution.

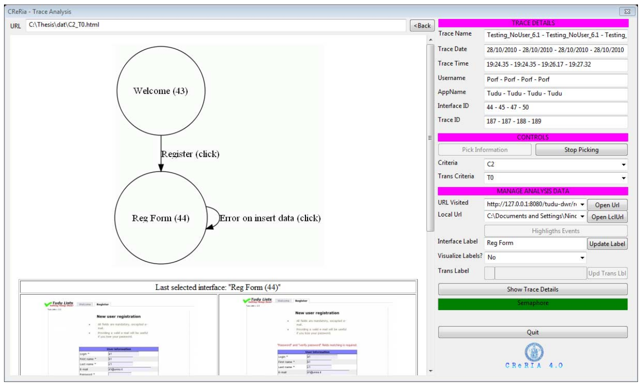


Trace Manager

Application Name: tudu

	Trace Id	Trace Name	Trace Data	Trace Time	User
☐	236	Testing_RegUser_13	06/11/2010	12:30:26	Porf
✓	238	Testing_RegUser_15	06/11/2010	16:20:12	Porf
✓	239	Testing_RegUser_15.1	06/11/2010	16:22:16	Porf
✓	240	Testing_RegUser_15.2	06/11/2010	16:24:07	Porf
✓	241	Testing_RegUser_15.3	06/11/2010	16:26:03	Porf
✓	242	Testing_RegUser_15.4	06/11/2010	16:27:56	Porf
✓	243	Testing_RegUser_15.5	06/11/2010	16:29:36	Porf
☐	244	Testing_RegUser_16	06/11/2010	16:30:50	Porf
✓	245	Testing_RegUser_17	06/11/2010	16:32:31	Porf
✓	246	Testing_RegUser_17.2	06/11/2010	16:35:01	Porf
✓	247	Testing_RegUser_17.3	06/11/2010	16:36:13	Porf
✓	248	Testing_RegUser_17.4	06/11/2010	16:40:13	Porf
☐	250	Testing_RegUser_17.6	06/11/2010	16:43:49	Porf

```
- DB connection closed -
- DB connection open -
```



CReRia - Trace Analysis

URL: C:\Thesis\dat\C2_T0.html

TRACE DETAILS

Trace Name	Testing_NoUser_6.1 - Testing_NoUser_6.1 - Testing_
Trace Date	28/10/2010 - 28/10/2010 - 28/10/2010 - 28/10/2010
Trace Time	19:24.35 - 19:24.35 - 19:26.17 - 19:27.32
Username	Porf - Porf - Porf - Porf
AppName	Tudu - Tudu - Tudu - Tudu
Interface ID	44 - 45 - 47 - 50
Trace ID	187 - 187 - 188 - 189

CONTROLS

Criteria	C2
Trans Criteria	T0

MANAGE ANALYSIS DATA

URL Visited	http://127.0.0.1:8080/tudu-dwr/r
Local Url	C:\Documents and Settings\Ninc
Interface Label	Reg Form
Visualize Labels?	No
Trans Label	

Last selected interface: "Reg Form (44)"

Figure 6. Two examples of CReRIA GUI showing Execution Traces selection (on the upper side) and concept refinement (on the lower side)

In the following, we show the part of tutorial guide documenting just a use case related to the insertion of a new todo. In particular, we considered the use case labelled "Advanced Add" that allows a todo to be inserted by specifying several parameters of it (such as todo description, priority, due date, assigned user, notes). The latter use case has 11 different scenarios, 7 of which correspond to correct insertions of a todo (with different valid combinations of input data) and 4 of which correspond to exceptional scenarios (due to incorrect input data).

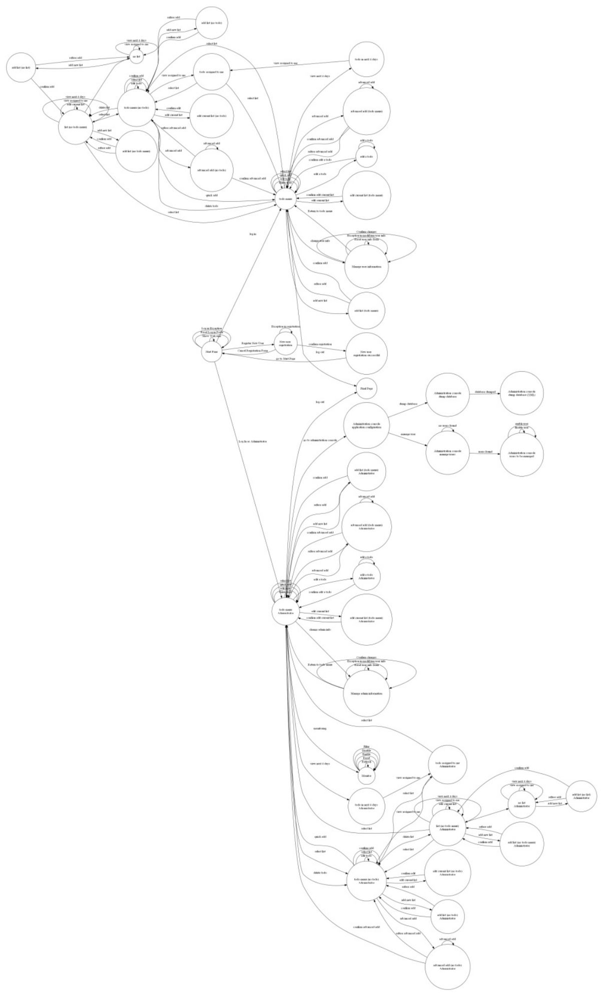


Figure 7. The overall Navigation Graph of Tudu

Two different views of the Navigation Graph for the "Advanced Add" use case are reported in Figure 8. The view on the left is a typical graph visualization, while in the right view graph nodes have been substituted by an interface instance of the Interface Class. Both the views can be included in the textual and in the hypertextual versions of the documentation (as a HTML map). The last one is a more intuitive view that can be used to have an immediate exemplification of the real appearance of the Interfaces shown by the application.

Moreover, the tutorial guide includes a walkthrough description of user tasks needed for accomplishing each "Advanced Add" scenario. Figure 9 shows the walkthrough generated for the first scenario of "Advanced Add". This scenario explains the procedure for inserting a todo in a todo list by specifying its parameters, and reports a synthetic description of the use case scenario, the sequence of screen shots of the obtained interface instances (labelled with the names given to the corresponding interface classes) and the description of the events causing the transitions between the interface instances (with the values set in the input fields in the collected instance of the scenario). Further information such as the preconditions with which the scenario has been obtained and a semantic description of the input data can be added manually by the software engineer who is in charge of producing the end user documentation.

In conclusion, the produced documentation was more complete and detailed with respect to the one obtainable by using the most part of user documentation generation tools described in the literature and commercially available, such as SmartTutor [31], DocWizards [8], JTutor [21], EpiDocX [14], that essentially produce only walkthroughs of user tasks.

Of course, our approach was more expensive as to the time needed for accomplishing the dynamic analysis of the RIA. However, this extra-effort was valuable for obtaining more abstractions and details about the application usage, that could be transformed automatically into user documentation by our tool.

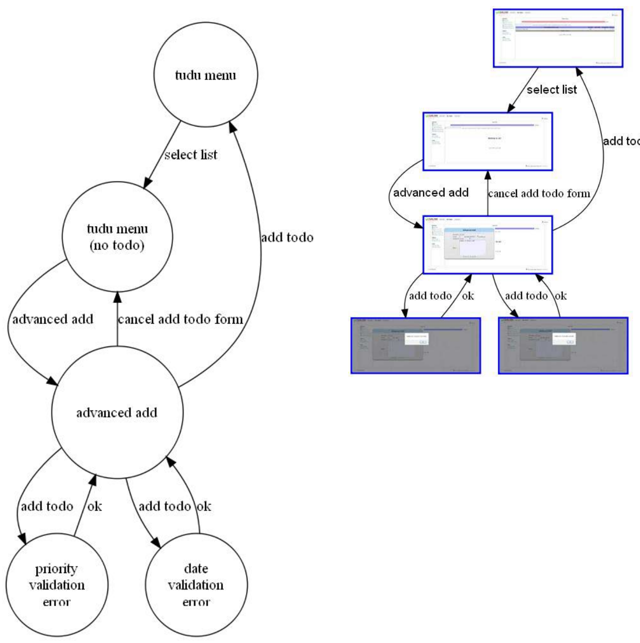


Figure 8. The Navigation Graph (in its classical view on the left, with thumbnails on the right) for the Advanced Add use case

VII. CONCLUSIONS AND FUTURE WORKS

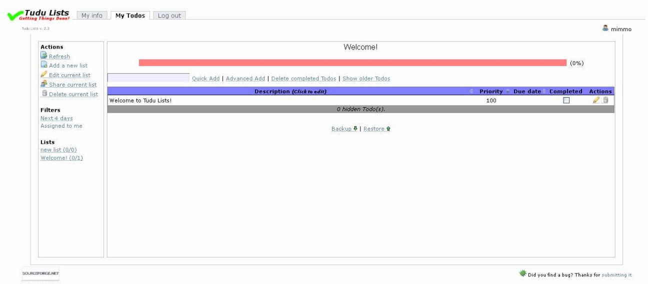
According to the survey presented by [15], software documentation generation processes should rely on technologies that improve automation of the documentation process, as well as facilitating documentation maintenance.

In this paper we have proposed a technique and a tool for semi-automatic generation of end user documentation about Web 2.0 applications. The technique is innovative since it exploits reverse engineering processes and tools for generating the documentation, differently from most existing solutions supporting user documentation production. With respect to other competing tools, ours is able to generate a more flexible, complete and accurate documentation.

Tutorial: ***Advanced Add***

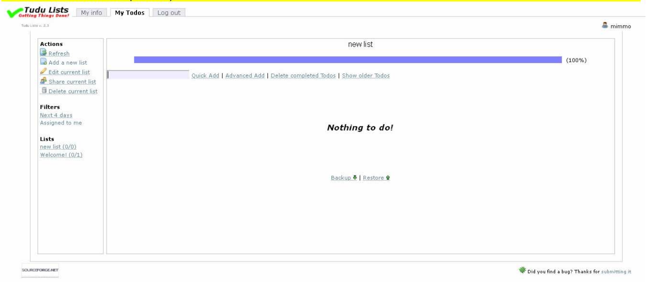
Actor: Logged User, Administrator
Use case: Advanced Add
Scenario Description: Insertion of a todo in a todo list by specifying a description, a positive level of priority, a future due date, the user to which the todo is assigned and an annotation.

1 - Interface: tudu menu



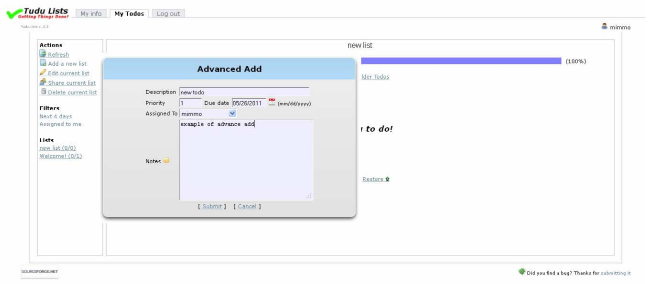
Transition: select list
Event: click on link "st (0/0)"

2 - Interface: *tudu menu (no tudu)*



Transition: Advanced Add
Event: click on link "Advanced Add"

3 - Interface: Advanced Add

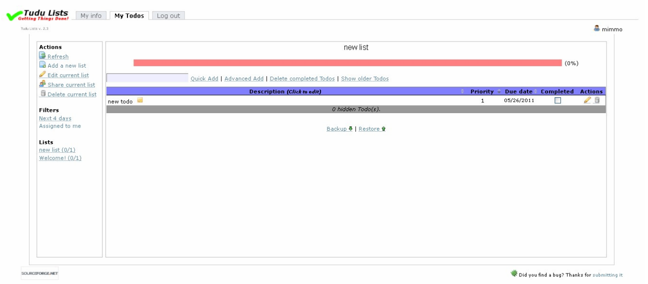


Description: new todo
Priority: 1 Due date: 05/26/2011
Assigned To: mimmo
example of advance add

Transition: add todo
Input:
insert '*new todo*' in *description Text field*
insert '*1*' in *priority Text field*

insert '*05/26/2011*' in *dueDate Text field*
select '*mimmo*' in *assignedUser Select field*
insert '*example of advanced add*' in *notes Text area*
Event: click on link "Submit"

4 - Interface: tudu menu



new todo 1 05/26/2011

Figure 9. The walkthrough description generated for a scenario of the Advanced Add use case

The paper presented the features of the tool that we designed to support the proposed process and showed an example of using it for re-documenting an existing application implemented using Ajax technology. The resulting documentation provides both overview and more detailed descriptions of the user functions offered by the application, and was obtained effectively thanks to the tool support. With respect to the requirements of the IEEE 1063 Standard, the accuracy of the documentation is guaranteed by the fact that the documentation reports real screenshots and scenarios of the RIA. Moreover, the completeness of obtained documentation will depend on the degree of coverage of all the application scenarios in the executed dynamic analysis activity.

Our approach is scalable since the effort needed for the documentation production just depends on the complexity of the software interactions. We expect a linear grown of the needed effort with the number of scenarios and with their length. Experiments will be carried out in future works in order to confirm this hypothesis.

Our approach is potentially applicable to any RIA, provided that the software engineer has all the needed execution rights for all functionalities and, if needed, he is able to set needed preconditions.

In future work, we will extend the features of our documentation generation tool in order to allow the generation of other types of contents suggested by the IEEE Standard on User Documentation, such as documentation about error management. Effort will be devoted, too, to the proposal of techniques for the evaluation of the consistence of the produced documentation.

Of course, in order to demonstrate the validity of the proposed approach, experiments are needed for showing both the effectiveness of the documentation process and of the obtained documentation. In particular, experiments will be carried out to assess the usefulness of the produced documentation in comprehension, maintenance and reengineering tasks. In future work we will carry out experiments aiming at comparing effectiveness and scalability of our approach against other ones, and at obtaining a systematic evaluation of the actual contribution given to software usability by the produced documentation.

REFERENCES

[1] D. Amalfitano, A. R. Fasolino and P. Tramontana, “Reverse Engineering Finite State Machines from Rich Internet Applications”, Proceedings of the Working Conference on Reverse Engineering (WCRE 2008), 2008, IEEE CS Press, pp. 69-73

[2] D. Amalfitano, A. R. Fasolino and P. Tramontana, “Experimenting a Reverse Engineering Technique for Modelling the Behavior of Rich Internet Applications”, Proceedings of International Conference on Software Maintenance (ICSM 2009), 2009, IEEE CS Press, pp.571-574

[3] D. Amalfitano, A. R. Fasolino and P. Tramontana, “Rich Internet Application Testing Using Execution Trace Data,” Second International Workshop on TESTing Techniques & Experimentation Benchmarks for Event-Driven Software (TESTBEDS 2010), IEEE CS Press, pp. 274-283

[4] D. Amalfitano, A. R. Fasolino and P. Tramontana, “An Iterative Approach for the Reverse Engineering of Rich Internet Application User Interfaces”, Proceedings of the Fifth International Conference on Internet and Web Applications and Services, ICIW 2010, IEEE CS Press, pp. 401-410

[5] D. Amalfitano, A. R. Fasolino, A. Polcaro and P. Tramontana, “DynaRIA: a Tool for Ajax Web Application Comprehension”, Proceedings of the 18th IEEE International Conference on Program Comprehension, ICPC 2010, IEEE CS Press, pp. 46-47

[6] D. Amalfitano, A. R. Fasolino, P. Tramontana, “Techniques and Tools for Rich Internet Applications Testing”, Proceedings of the 12nd Symposium on Web Systems Evolution, WSE 2010, IEEE CS Press, pp. 63-72

[7] G. Antoniol, M. Di Penta, M. Zazzara, “Understanding Web Applications through Dynamic Analysis”. Proceedings of the International Workshop on Program Comprehension, IWPC 2004, IEEE CS Press, pp. 120-131

[8] L. Bergman, V. Castelli, T. Lau and D. Oblinger, “DocWizards: a system for authoring follow-me documentation wizards”. In *Proceedings of the 18th annual ACM* symposium on User interface software and technology (UIST '05). ACM, New York, NY, USA, 191-200.

[9] M. L. Bernardi, G. A. Di Lucca and D. Distante., “The RE-UWA approach to recover user centered conceptual models from Web applications”. International Journal on Software Tools for Technology Transfer, STTT 11(6): 485-501 (2009)

[10] T.J. Biggerstaff, B.G. Mitbander and D. Webster, “Program understanding and the concept assignment problem”, Communications of the ACM, 1994, vol. 37 (5), pp. 72- 83

[11] E.J. Chikofsky and J.H. Cross II, “Reverse engineering and design recovery: a taxonomy”. IEEE Software, vol.7, no.1, pp.13-17, Jan 1990

[12] G. A. Di Lucca, A. R. Fasolino, P. Tramontana, “Reverse engineering Web applications: the WARE approach”. Journal of Software Maintenance 16(1-2): 71-101 (2004)

[13] Doxygen, Generate documentation from source code. available at: http://www.stack.nl/~dimitri/doxygen/ Last accessed May 13th, 2011.

[14] Epiance Software, epiDOCX, http://www.epiplex500.com/ (also available at http://download.cnet.com/windows/epiance-software/3260-20_4-6311213.html). Last accessed May 13th, 2011.

[15] A. Forward and T. Lethbridge, “The relevance of software documentation, tools and technologies: a survey”. In Proceedings of the 2002 ACM symposium on Document engineering (DocEng '02). ACM, New York, NY, USA, 26-33

[16] J.Garrett, AJAX, “A new approach to Web applications”. Adaptive Path, 2005.

[17] Institute of Electrical and Electronics Engineers, “Glossary of Software Engineering Terminology”, IEEE, New York, 1990. IEEE Standard 610.12-1990.

[18] IEEE Standard for Software User Documentation, IEEE Std 1063-2001, 2001

[19] ISO, Software Product Evaluation - Quality Characteristics and Guidelines for Their Use (ISO/IEC IS 9126). Geneva, Switzerland: International Organization for Standardization 1991.

[20] Javadoc Tool, available at http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html Last accessed May 13th, 2011.

[21] C. Kojouharov, A. Solodovnik, and G. Naumovich, “JTutor: an Eclipse plug-in suite for creation and replay of code-based tutorials”. In Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange (eclipse '04). ACM, New York, NY, USA, 27-31.

[22] A. Marchetto, P. Tonella and F. Ricca, “State-Based Testing of Ajax Web Applications”. Proceedings of 2008 Internationa Conference on Software Testing, Verification and Validation, IEEE CS Press, 2008, pp. 121-130.

[23] A. Marchetto and P. Tonella, “Search-Based Testing of Ajax Web Applications”. Proceedings of 1st International Symposium on Search Based Software Engineering, IEEE CS Press, 2009, pp. 3 - 12

[24] A. Mesbah, E. Bozdag and A. Van Deursen, “Crawling AJAX by Inferring User Interface State Changes”. Proceedings of Eighth

International Conference on Web Engineering (ICWE 2008), 2008, IEEE CS Press, pp. 122-134

[25] A. Mesbah and A. van Deursen, “Invariant-based automatic testing of AJAX user interfaces”. Proceedings of International Conference on Software Engineering (ICSE 2009), 2009, IEEE CS Press, pp. 210-220

[26] F. Ricca and P. Tonella,
“Understanding and Restructuring Web Sites with ReWeb”.
IEEE Multimedia magazine, special issue on Web Engineering,
pp. 40-51, April-June 2001, Vol 8, N. 2.

[27] I. Sommerville, Software Engineering: Nineth edition, 2011, Addison-Wesley

[28] Tudu lists, available at: http://www.julien-dubois.com/tudu-lists Last accessed May 13th, 2011.

[29] W. Wang, Yu Lei, S. Sampath, R. Kacker, R. Kuhn and J. Lawrence, “A Combinatorial Approach to Building Navigation Graphs for Dynamic Web Applications”, *Proceedings of the 2009 International Conference on Software Maintenance* (ICSM '09). IEEE Computer Society Press, pp. 211- 220

[30] Wikipedia, Comparison of Screencasting Tools, available at: http://en.wikipedia.org/wiki/Comparison_of_screencasting_software Last accessed May 13th, 2011.

[31] Y. Zhang, G. Huang, N. Zhang, and H. Mei, “SmartTutor: Creating IDE-based interactive tutorials via editable replay”. In *Proceedings of the 31st International Conference on Software Engineering* (ICSE '09). IEEE Computer Society Press, pp. 559-562.