A Framework for Testing Software Product

Amjad Farooq[1], M. Junaid Arshad[1] and Muhammad Abuzar Fahiem[2]

[1, 2]Computer Science and Engineering Department, UET, Lahore
[2]Department of Computer Science, Lahore College for Women University, Lahore, Pakistan
amjadfarooq@uet.edu.pk

Abstract: There is a growing need of frameworks for automatic testing of software product because manual testing of huge software product is very time-consuming and costly. Furthermore, the manually testing of complex software becomes more difficult and a challenging activity. However this can be easily achieved through automatic testing strategies. In this paper we propose a framework for testing software automatically. Now errors and bug finding become simpler and easier. It takes less time to test the whole application rather than testing application modules separately. The proposed framework provides programmatic access to most user interface elements. The main propose of our framework is to make testing phase easier and cost efficient. We validate our framework through a case study. By analyzing the results of testing the correctness and completeness of framework is proved. [Journal of American Science. 2010;6(12):164-173]. (ISSN: 1545-1003).

Keywords: Software testing; test automation; test framework

1. Introduction

The manually testing of complex software is a very difficult and a challenging activity. However this can be easily achieved through automatic testing strategies (Yingxiang, 2008). Nowadays, the user interface (UI) testing automation has become an integral part of software development in all big corporations (Jovic and Hauswirth, 2010). UI Automation means testing any User Interface application in an automated fashion. This method of testing is far more effective than just doing manual testing as it is best to catch the last minute bugs and basically improves overall product quality (Mathew and Spraetz, 2009). The automation runs faster and the benefit is that it doesn't require human input like manual testing. By this large application testing become simpler, easier and less time consuming [9]. As for the evolution of the testing field, there have been some innovations over the years. What's striking about the innovations, however, is how few people know about them, and even fewer people are using them. In terms of evolution, here in Pakistan, we're in the Dark Ages of testing. Many test teams work in isolation, knowing little of the existing literature on the subject, and providing little input to improve UI testing. Due to the importance of testing, most probably in few years it will evolve into a proper engineering discipline.

The rest of paper is structured as follows: The background and the related work are given in Section 2. The functionalities, and design description of proposed framework is given in Section 3 whereas; the different test scenarios used to validate the proposed framework are also listed in the same Section. Finally, the paper ends with conclusion in Section 4.

2. Background and Related Work

There is a growing need of frameworks for automatic testing of software (Riungu et al,2010; Berner et al, 200). Both, the functional and non-functional requirements of the software need to be tested but their manual testing consumes a lot of time and budget resources (Jovic and Hauswirth, 2010; Sarkar et al., 2009; Mesbah and van Deursen, 2009). Furthermore, it rises multiple times for a complex software testing. Furthermore, to measure the quality of software the execution-based testing has its own importance (Viswanathan and Peters, 2010). In (Wang and Damata, 2009), GUI testing toolset is described. This toolset supplements the basic testing tasks required in the common GUI testing process. It generates test cases and has a reporting mechanism. Although it is a good toolset but the target software is not a web-based. The Force.com (Mathew and Spraetz, 2009) framework includes some testing utilities to create test cases and those test cases are applied for the automation of individual modules of software. AUTOWEB (Chai et al., 2009) is an interesting tool, it test the online- assignments submitted by students. It generates demo of failed test cases automatically. It aims to provide help to students in order to solve their assignments.

3. Proposed Work

The basic architecture of our framework is given in Figure 1. There is a UI Automation Tester class which contains all the test cases. User firstly has to start the target UI application. The user task then is to select the test case to which he/she wants to run.

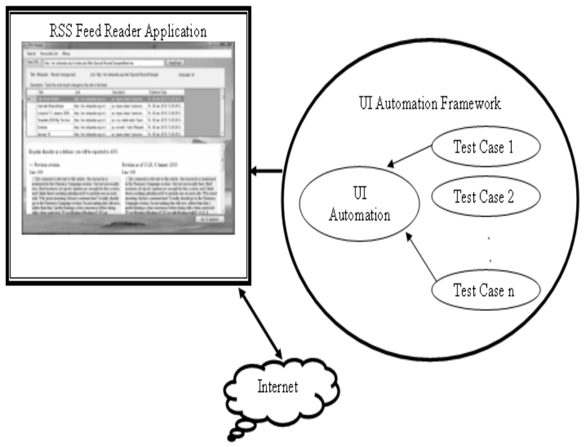


Figure 1: System Architecture of our Proposed Framework

Internet connection should be established to test RSS Feed Reader. RSS Feed Reader needs internet to read latest feed which are continuously upgraded on the sites, regarding the URL specified. The layered architecture of our framework is as given below:

Test Cases
Operational Logics
UI Wrappers – Physical Layer

Figure 2: Layered Architecture of our Proposed Framework

Test cases layer contain the test cases for the application organized by the area names and test names. Each test case represents a user scenario. Some scenarios are higher in priority than others. A test case is collection of a bunch of logical calls that live in logical layer. The test cases use / consume the logical layer in it or interacts with the Logical Layer. Some examples of test case: Reading an RSS Feed and making sure that correct feed is displayed; create a new word document and making sure that a plain document is created without errors.

Logical layer is collection of methods that represents user actions. These are logical methods that do actions needed to complete a test case. For example in order to complete the above mentioned test cases logical methods will be needed to make sure that application is in running state; to read RSS Feed that takes in a custom RSS Feed location. A method in logical layer is merely a collection of calls to physical layer that does physical action e.g. to create a document the physical action that test code will do by clicking the File Menu and then New Document Menu.

Physical layer is also called direct UI Wrappers. Physical layer consists of the wrappers over all the UI Controls that we have in the application. It gives us an interface that we can use to call actions on the UI Controls in the application e.g. Click Read Feed Button, set value in a Location Text box. The mechanism is limited to Testing UI. Applications only developed in C# or VB.Net. The software is application specific; test cases are specific to the application that you want to test. The overall workings of proposed framework have shown in Figure 2 and 3.

For sake of simplicity, the following we have considered the following assumptions and constraints.

- Software is developed for testing a UI application which is RSS feed reader.
- Test cases are specific to the application.
- Test cases are not generic or couldn't run on other UI applications for testing.
- We have developed our software in C# and it is capable of testing UI application.
- We have not purchased any type of software and hardware equipment for our project.

The class diagram of proposed framework is given in Figure 4. We have TestExecution Client App that runs our tests; you can pick a test and run it that class is represented by ExecuteScriptClient. it contains the UI for our execution engine and methods like start app, execute script etc. When a test is selected and executes script is called then we move to Script class or TestCases class. Script class has all our test cases. Script class interacts with Logical Class and Logical Class contains RSS ReaderFunction. RssReaderFuction Class is the one that contains our logical actions. Rss Reader has an association relationship with physicalObjects class.Physical Objects contains wrappers for all the UI controls. Then Physical Objects calls into ScriptFunctons class that has all the methods that help us do actions on these UI controls e.g. click, set text Toggle, Set Value etc.

A selective list of graphical user interfaces used for different requirements are given in the Figures 5-9 followed by their respective input, processing and output. We have used different test scenarios to check the correctness of proposed framework. Some of the scenarios are listed in the Tables 1-8 respectively.

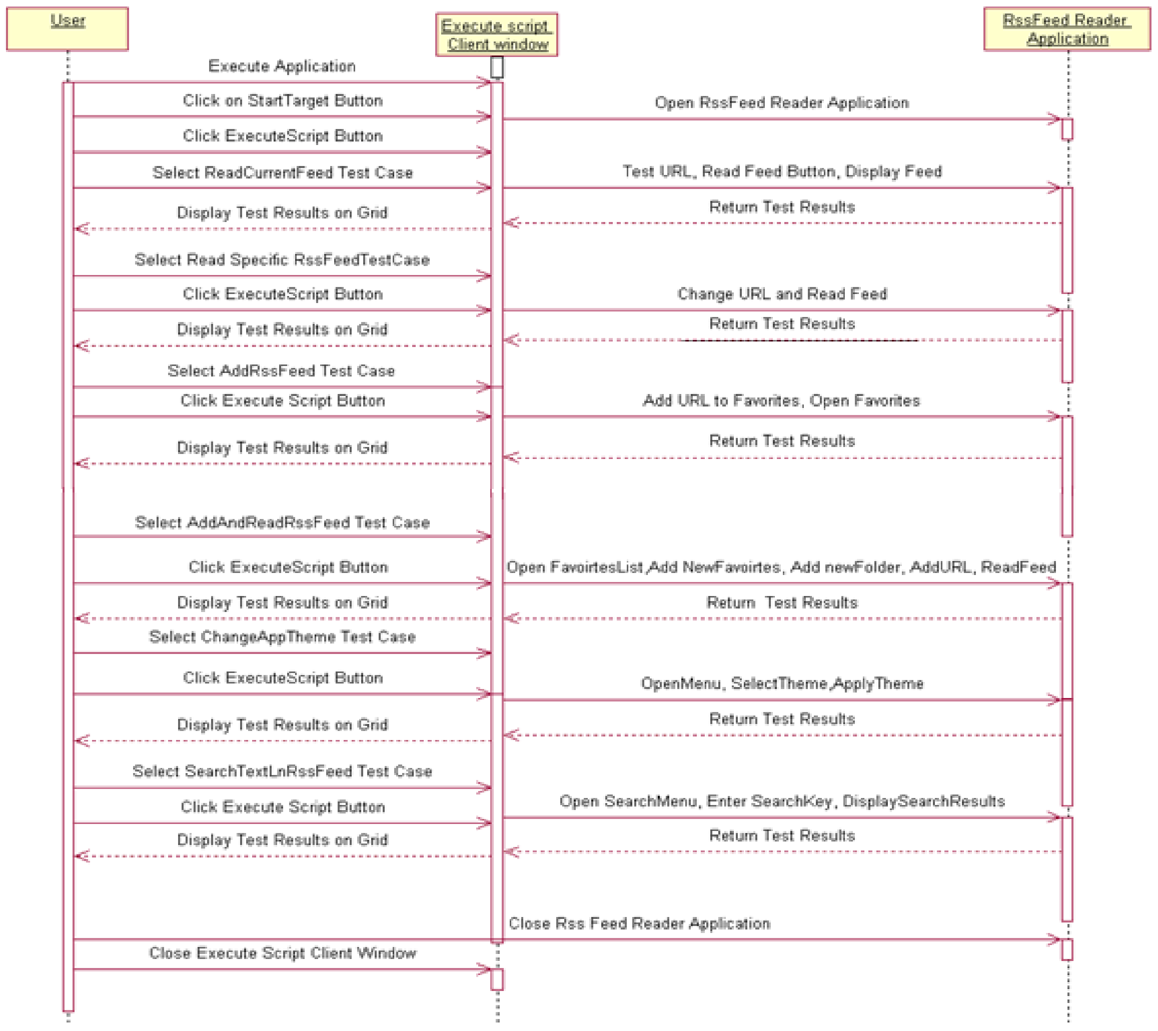

User
Execute script Client window
RssFeed Reader Application
Execute Application
Click on StartTarget Button
Open RssFeed Reader Application
Click ExecuteScript Button
Select ReadCurrentFeed Test Case
Test URL, Read Feed Button, Display Feed
Display Test Results on Grid
Return Test Results
Select Read Specific RssFeedTestCase
Click ExecuteScript Button
Change URL and Read Feed
Display Test Results on Grid
Return Test Results
Select AddRssFeed Test Case
Click Execute Script Button
Add URL to Favorites, Open Favorites
Display Test Results on Grid
Return Test Results
Select AddAndReadRssFeed Test Case
Click ExecuteScript Button
Open FavoirtesList,Add NewFavoirtes, Add newFolder, AddURL, ReadFeed
Display Test Results on Grid
Return Test Results
Select ChangeAppTheme Test Case
Click ExecuteScript Button
OpenMenu, SelectTheme,ApplyTheme
Display Test Results on Grid
Return Test Results
Select SearchTextLnRssFeed Test Case
Click Execute Script Button
Open SearchMenu, Enter SearchKey, DisplaySearchResults
Display Test Results on Grid
Return Test Results
Close Rss Feed Reader Application
Close Execute Script Client Window

Figure 2: System sequence diagram of proposed framework

Figure 5:Graphical User Interface for adding a Favourite

Input	• Automatically click on Favorites List. • Click on Favorites List. • Write URL in name text box. • Specify folder name in folder text box or select already created folder from drop down list. • Click on Add button. • To cancel this window click on cancel button.
Output	URL added to the favorites list in the specified folder.
Processing	Create folder if specified. Add URL to the favorites list. And close the window.

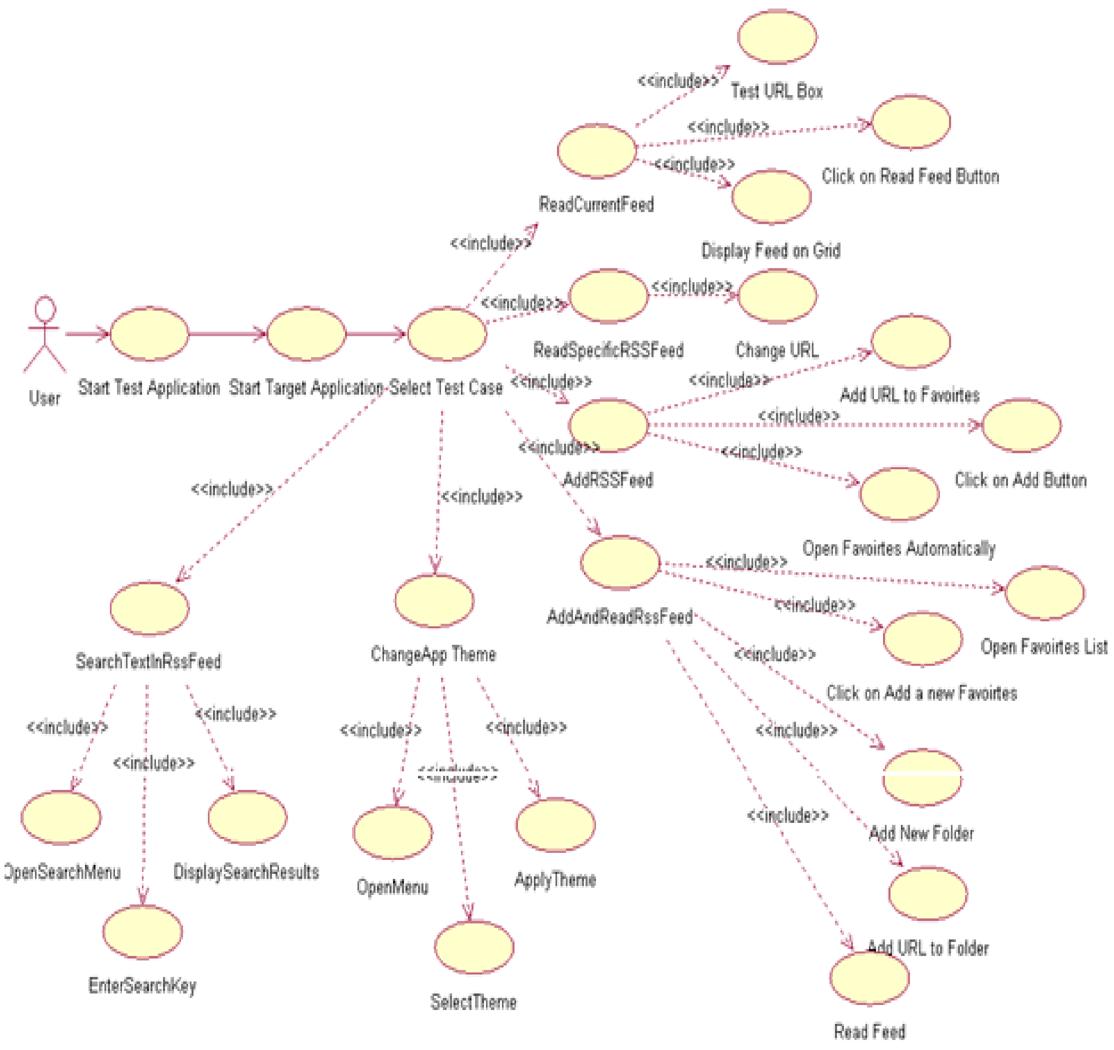


Figure 3: Operations of proposed framework

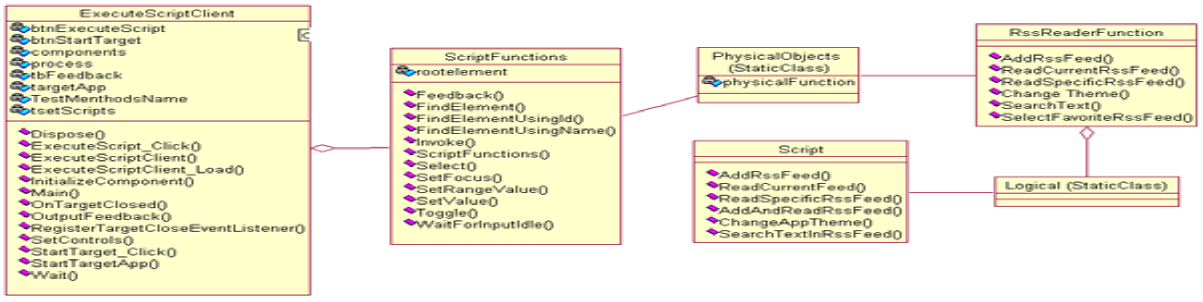
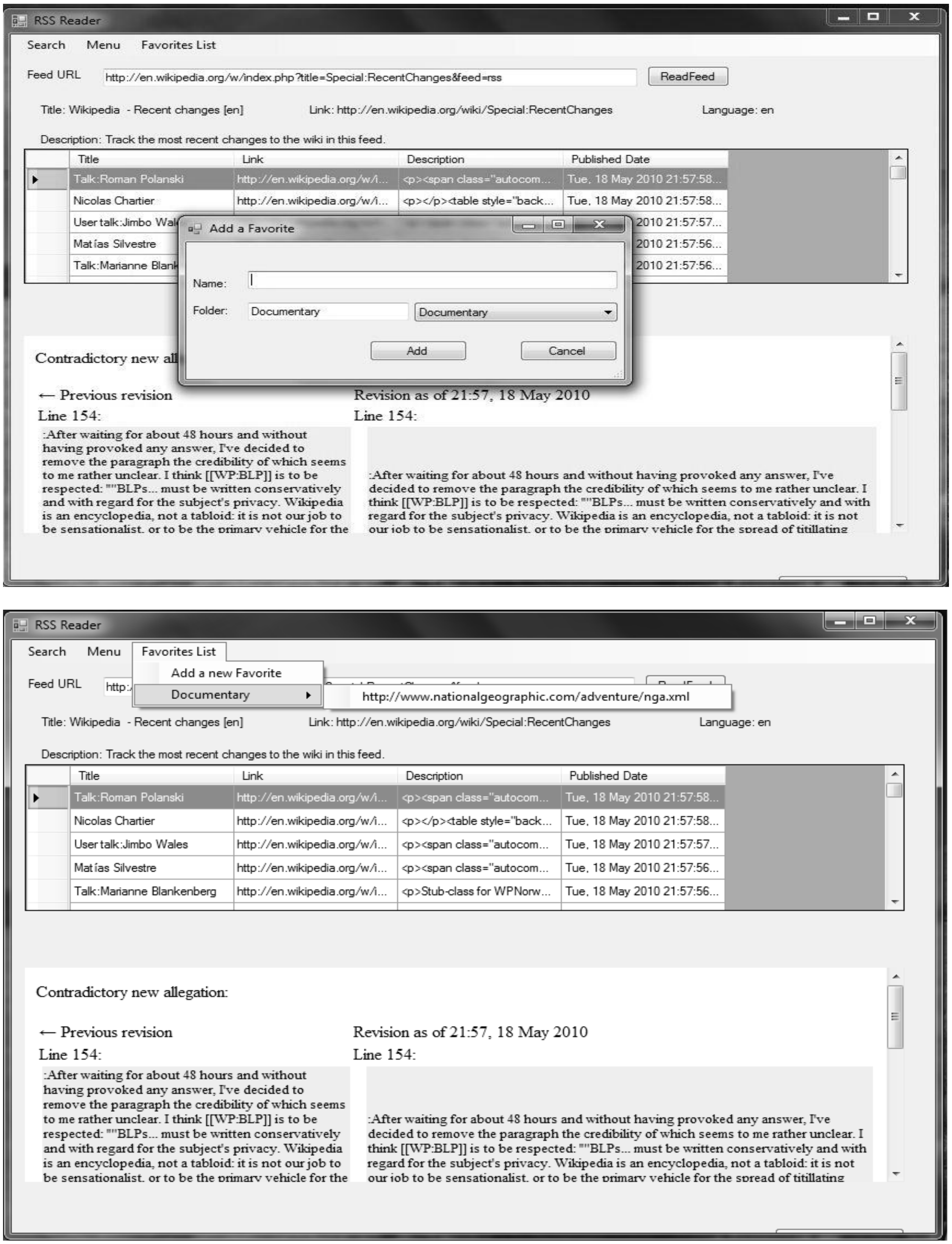


Figure 4: Class Diagram of proposed framework



Feed URL http://en.wikipedia.org/w/index.php?title=Special:RecentChanges&feed=rss

Title: Wikipedia - Recent changes [en] Link: http://en.wikipedia.org/wiki/Special:RecentChanges Language: en

Description: Track the most recent changes to the wiki in this feed.

Title	Link	Description	Published Date
Talk:Roman Polanski	http://en.wikipedia.org/w/i...	<p><span class="autocom...	Tue, 18 May 2010 21:57:58...
Nicolas Chartier	http://en.wikipedia.org/w/i...	<p></p><table style="back...	Tue, 18 May 2010 21:57:58...
User talk:Jimbo Wales	http://en.wikipedia.org/w/i...	<p><span class="autocom...	Tue, 18 May 2010 21:57:57...
Matías Silvestre	http://en.wikipedia.org/w/i...	<p><span class="autocom...	Tue, 18 May 2010 21:57:56...
Talk:Marianne Blankenberg	http://en.wikipedia.org/w/i...	<p>Stub-class for WPNorw...	Tue, 18 May 2010 21:57:56...

Name:

Folder: Documentary Documentary

Favorites List: Add a new Favorite; Documentary ▸ http://www.nationalgeographic.com/adventure/nga.xml

Contradictory new allegation:

← Previous revision

Line 154:

:After waiting for about 48 hours and without having provoked any answer, I've decided to remove the paragraph the credibility of which seems to me rather unclear. I think [[WP:BLP]] is to be respected: ""BLPs... must be written conservatively and with regard for the subject's privacy. Wikipedia is an encyclopedia, not a tabloid: it is not our job to be sensationalist, or to be the primary vehicle for the

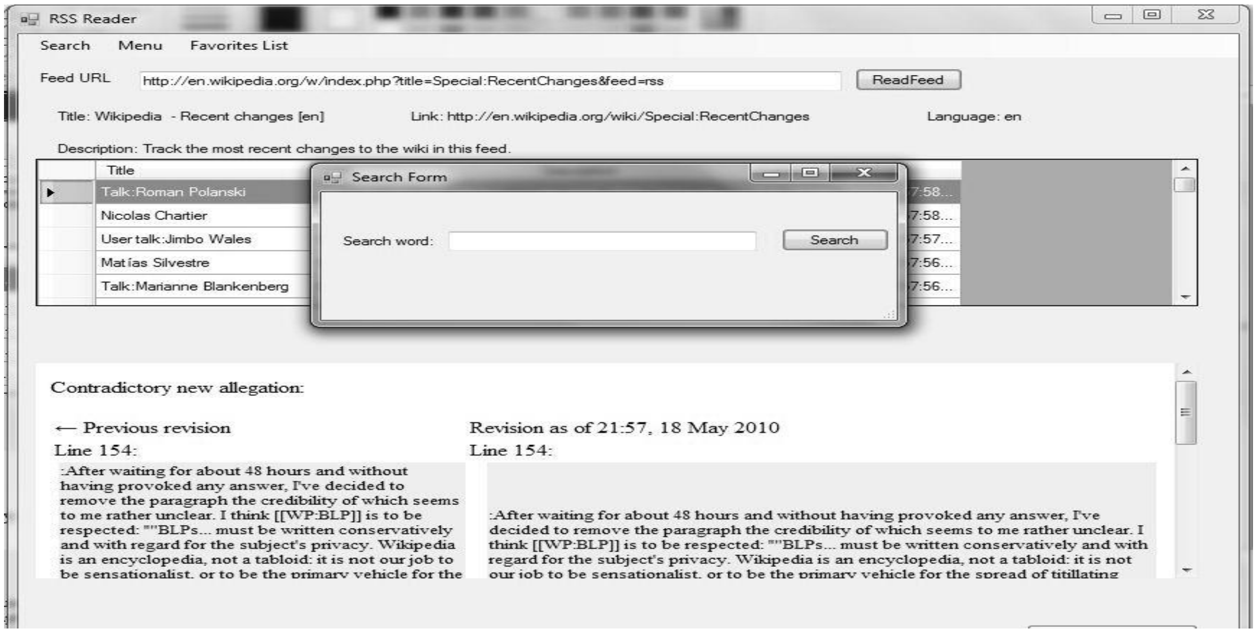
Revision as of 21:57, 18 May 2010

Line 154:

:After waiting for about 48 hours and without having provoked any answer, I've decided to remove the paragraph the credibility of which seems to me rather unclear. I think [[WP:BLP]] is to be respected: ""BLPs... must be written conservatively and with regard for the subject's privacy. Wikipedia is an encyclopedia, not a tabloid: it is not our job to be sensationalist, or to be the primary vehicle for the spread of titillating

Figure 6:Graphical User Interface for Favourite list

Input	Automatically click on Favorites List from menu bar and then go to the Documentary folder from the list.
Output	Display selected URL in the text box.
Processing	Select the URL from the Documentation list.



Feed URL http://en.wikipedia.org/w/index.php?title=Special:RecentChanges&feed=rss

Title: Wikipedia - Recent changes [en] Link: http://en.wikipedia.org/wiki/Special:RecentChanges Language: en

Description: Track the most recent changes to the wiki in this feed.

Title	
Talk:Roman Polanski	7:58...
Nicolas Chartier	7:58...
User talk:Jimbo Wales	7:57...
Matías Silvestre	7:56...
Talk:Marianne Blankenberg	7:56...

Contradictory new allegation:

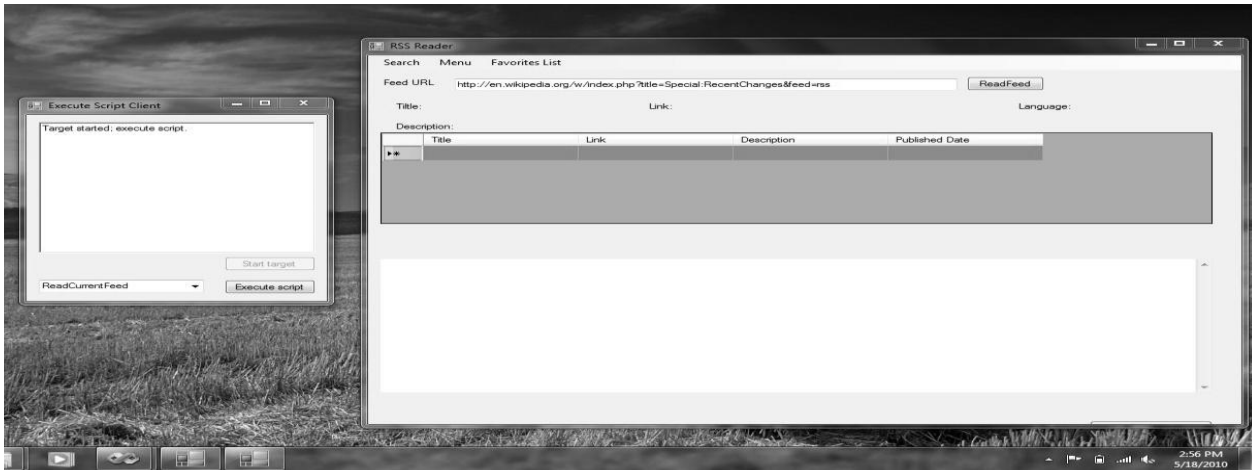
← Previous revision Revision as of 21:57, 18 May 2010
Line 154: Line 154:

:After waiting for about 48 hours and without having provoked any answer, I've decided to remove the paragraph the credibility of which seems to me rather unclear. I think [[WP:BLP]] is to be respected: ""BLPs... must be written conservatively and with regard for the subject's privacy. Wikipedia is an encyclopedia, not a tabloid: it is not our job to be sensationalist, or to be the primary vehicle for the

:After waiting for about 48 hours and without having provoked any answer, I've decided to remove the paragraph the credibility of which seems to me rather unclear. I think [[WP:BLP]] is to be respected: ""BLPs... must be written conservatively and with regard for the subject's privacy. Wikipedia is an encyclopedia, not a tabloid: it is not our job to be sensationalist, or to be the primary vehicle for the spread of titillating

Figure 7:Graphical User Interface for searching

Input	• Automatically click on the search from menu list. • Input what you want to search. • Click on Search button.
Output	Show search results.
Processing	Read input from search word text box and display find the results.



Target started; execute script.

Feed URL http://en.wikipedia.org/w/index.php?title=Special:RecentChanges&feed=rss

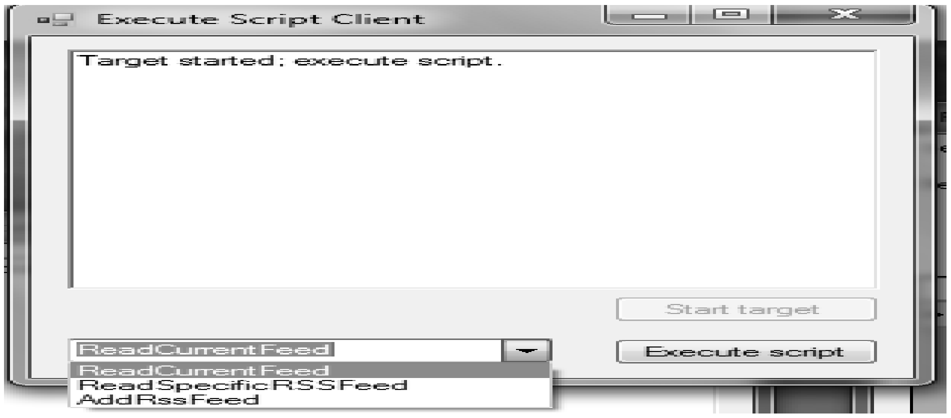
Title: Link: Language:

Description:

Title	Link	Description	Published Date

Figure 8:Graphical User Interface for starting the target application

Input	Click on Start Target button.
Output	Display RSS Feed Reader window.
Processing	Open the target application.



Target started; execute script.

Figure 9:Graphical User Interface for executing script

Input	Select any test case which u wants to run and click on Execute script button.
Output	Display test results on the Data Grid Box.
Processing	Run selected test case to test the target application.

Table 1: Start Test Application

UC-01: Start Test Application		
Actors: User		
Feature: Start application		
Use Case Id:	UC-01	
Pre Condition:	Application should be debugged for starting the application.	
Scenarios		
Step #	**Action**	**Software Reaction**
1.	Start target application.	Software will open target application.
2.	Select test cases from drop down menu.	Run the selected test case and display results in grid box.
Alternative Scenarios:		
Target application should be started before selecting test cases.		

Table 2: Start Target Application

UC-02: Start Target Application		
Actors: User		
Feature: User will start the target application.		
Use Case Id:	UC-02	
Pre Condition:	User should be click on the start target application button.	
Scenarios		
Step #	**Action**	**Software Reaction**
1.	User presses the start target application button.	Software will open the RSS Feed Reader application. Execute Script Client Window will displays a message in grid box “Target started; execute script”.
Alternative Scenarios:		
User should have to click the start target application button in each and every condition.		

Table 3: Select Test Case

UC-03: Select Test Case		
Actors: User		
Feature: User will select test cases from the drop down menu.		
Use Case Id:	UC-03	
Pre Condition:	User should select the test case in order to test the application.	
Scenarios		
Step #	**Action**	**Software Reaction**
1.	User selects the desired test case from the drop down list. Available test cases are: • Read Current Feed • Read specific RSS Feed • Add RSS Feed	
Alternative Scenarios:		
User may directly close the application without running test cases.		

Table 4: Read Current Feed

UC-04: Read Current Feed		
Actors: User, Execute Script Client		
Feature: User will select the "Read Current Feed" test cases from the drop down menu.		
Use Case Id:	UC-04	
Pre Condition:	User should press the Execute Script button in order to run the test case.	
Scenarios		
Step #	**Action**	**Software Reaction**
1.	User will press the Execute script button to run the test case.	Software will automatically test the application and performs the following functionalities. • Test URL Box • Click on Read Feed Button • Display Feed on Grid
Alternative Scenarios:		
User may select any other test case rather than selecting this test case.		

Table 5: Read Specific RSS Feed

UC-05: Read Specific RSS Feed		
Actors: User, Execute Script Client		
Feature: User will select the "Read Specific RSS Feed" test cases from the drop down menu.		
Use Case Id:	UC-05	
Pre Condition:	User should press the Execute Script button in order to run the test case.	
Scenarios		
Step #	**Action**	**Software Reaction**
1.	User will press the Execute script button to run the test case.	Software will automatically test the application and performs the following functionalities. • Change URL
Alternative Scenarios:		
User may select any other test case rather than selecting this test case.		

Table 6: Add RSS Feed

UC-06: Add RSS Feed		
Actors: User, Execute Script Client		
Feature: User will select the "Add RSS Feed" test cases from the drop down menu.		
Use Case Id:	UC-06	
Pre Condition:	User should press the Execute Script button in order to run the test case.	
Scenarios		
Step #	**Action**	**Software Reaction**

1.	User will press the Execute script button to run the test case.	Software will automatically test the application and performs the following functionalities. • Open Favorites Automatically • Click on Add Button • Add URL to Favorites
Alternative Scenarios:		
User may select any other test case rather than selecting this test case.		

Table 7: Execute Client Script

Test Case ID: T-01	**Engineer:** Faiza Aziz,Iram Waheed ,Farah Amjad
Application Name: Testing via UI Automation	**Use Case Id:** UC-Start Test Applcation-01
Purpose: To start the application.	
Scenario: To test target application.	
Environment: Visual Studio .NET 2008, IE 8.0	
Pre-Request: User should debug the program in order to test the application.	
Strategy: 1. User will run the Program. 2. Execute Client Script window will open. 3. Press starts target application. 4. Select required test case from drop down list. 5. Click on execute script button.	
Expected Results: 1. Execute Client Script window will open. 2. RSS Feed Reader window will open. 3. Automated testing regarding the specific test case will start.	
Observations: The testing will start properly and less time is consumed on testing. All results will be displayed in the Data Grid Box.	
Results: No error found. Client Script Window opened properly. All Test cases execute properly and will check all controls of target application RSS Feed Reader. For Example, it will show errors if URL text box is empty.	

Table 8: RSS Feed Reader

Test Case ID: T-02	**Engineer:** Faiza Aziz,Iram Waheed ,Farah Amjad
Application Name: Testing via UI Automation	**Use Case Id:** UC-Start Target Applcation-02
Purpose: To start the target application for testing.	
Scenario: To run all test cases on RSS Feed Reader.x	
Environment: Visual Studio .NET 2008, IE 8.0	
Pre-Request: User should press the Start Target Application button in order to test the application.	
Strategy: 1. Select ReadCurrentFeed from drop down list and click on Execute Script button. 2. Select ReadSpecificRS Feed from drop down list and click on Execute Script button. 3. Select AddRSSFeed from drop down list and click on Execute Script button. 4. Select AddAndReadRSSFeed from drop down list and click on Execute Script button. 5. Select SearchTextlnRSSFeed from drop down list and click on Execute Script button. 6. Select ChangeApp Theme from drop down list and click on Execute Script button.	
Expected Results: 1. RSS Feed Reader window will open. 2. When Read Current Feed test case selected following actions performed automatically ○ Test URL Box ○ Click on Read Feed Button ○ Display Feed on Grid 3. When Read Specific RSSS Feed test case selected following actions performed automatically ○ Change URL 4.When Add RSS Feed test case selected following actions performed automatically ○ Open Favorites Automatically	

○ Click on Add Button ○ Add URL to Favorites
Observations: Different URL is changed to check that whether RSS Feed Reader is properly getting latest feeds from the URL. Different URL is also added to the favorites list to check that it is properly maintaining favorites List.
Results: No error found. All Test cases execute properly and will check all controls of target application RSS Feed Reader. For Example, it will show errors if URL text box is empty etc.

5. Conclusion and Future Work

The proposed framework is purely independent and self contained. And there is no other related application or any third party component involved for the development of the framework. We have developed the framework in C# and enabled to test various C# UI Applications. The proposed framework is capable of testing UI application automatically. In our framework user have to start the application, after the UI Automation starts user task now is to start the target application(RSS Feed Reader), which is the application to which user wants to test. After the target application is selected, now user has to select any test case which he/she wants. After the desire test case is selected automated testing will start by testing all controls of the application. It will tell about all bugs/errors in the application automatically. As manual testing was more time consuming, so our application not only reduces time complexity but also make it efficient and convenient for users.

Acknowledgements

Authors are thankful to the University of Engineering and Technology, Lahore, and HEC – Pakistan for financial support for doing this work.

References

1. Appasami S. User Interface Accessibility and Test Automation for Silverlight Applications. International Journal of Computational Intelligence Research 2009, 5(2), Print ISSN: 0973-1873. Online ISSN: 0974-1259.
2. Berner S, Weber, R., and Keller, R. K. 2005. Observations and lessons learned from automated testing. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005). ICSE '05. ACM, New York, NY, 571-579. DOI= http://doi.acm.org/10.1145/1062455.1062556
3. Chai T, Wang Z and Wang J. Automated Universal Testing and tutoring system for WEB application. Computer Science and Information Technology, International Conference on, pp. 188-192, 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009.
4. Jovic M and Hauswirth M. Performance Testing of GUI Applications. icstw, pp.247-251, 2010, Third International Conference on Software Testing, Verification, and Validation Workshops.
5. Mathew, R. and Spraetz, R. 2009. Test Automation on a SaaS Platform. In Proceedings of the 2009 international Conference on Software Testing Verification and Validation (April 01 - 04, 2009). ICST. IEEE Computer Society, Washington, DC, 317-325. DOI= http://dx.doi.org/10.1109/ICST.2009.46
6. Mesbah A and van Deursen, A. 2009. Invariant-based automatic testing of AJAX user interfaces. In Proceedings of the 31st international Conference on Software Engineering (May 16 - 24, 2009). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 210-220. DOI= http://dx.doi.org/10.1109/ICSE.2009.5070522
7. Riungu L, Taipale O, and Smolander K. Software Testing as an Online Service: Observations from Practice. icstw, pp.418-423, 2010 Third International Conference on Software Testing, Verification, and Validation Workshops, 2010.
8. Sarkar C, Soderston C, Klementiev D, and Bell E. Remote Automated User Testing: First Steps toward a General-Purpose Tool. A chapter in book: Software Engineering Research, Management and Applications 2010.
9. Viswanathan S, and Peters J. C. Automating UI guidelines verification by leveraging pattern based UI and model based development. In Proceedings of the 28th of the international Conference Extended Abstracts on Human Factors in Computing Systems (Atlanta, Georgia, USA, April 10 - 15, 2010). CHI EA '10. ACM, New York, NY, 4733-4742. DOI= http://doi.acm.org/10.1145/1753846.1754222.
10. Wang M and Damata, L. 2009. TAO Project: An Intuitive Application UI Test Toolset. In Proceedings of the 2009 Sixth international Conference on information Technology: New Generations (April 27 - 29, 2009). ITNG. IEEE Computer Society, Washington, DC, 796-800. DOI= http://dx.doi.org/10.1109/ITNG.2009.86
11. Yingxiang Z. Ingrid - An automated testing system for telephony software - A case study. UVic Subject Index::Sciences and Engineering::Applied Sciences::Computer science, 2008.

7/1/2010