

Comparative analysis between UFT and Ranorex test automatization frameworks

Contents

Tests behavior without repository.....	3
Step 0:	3
Step 1:	3
Step 2:	3
Step 3:	3
Step 4:	3
Step 5:	4
Step 8:	4
Tests behavior with Repository	4
Step 0:	4
Step 1:	4
Step 2:	4
Step 3:	5
Step 4:	5
Step 5:	5
Step 8:	5
UFT Utilization.....	5
Ranorex Utilization.....	8
Repositories	8
UFT repository.....	9
Ranorex repository	10
Conclusion.....	13

Tests behavior without repository

Step 0:

Action	UFT	Ranorex	Conclusion
Launch DiagBox	OK	OK	No difference between them
Manage/Close Popups	NOK	NOK	Both frameworks have problems identifying pop-ups
Close DiagBox	OK	OK	No difference between them
Manage authentication	NOK	NOK	The authentication is done using a pop-up

Step 1:

Action	UFT	Ranorex	Conclusion
Select existent brand	NOK	OK	Element not clickable using the descriptive programming method
Select inexistent brand	NOK	OK	Error is shown in the report log

Step 2:

Action	UFT	Ranorex	Conclusion
Select vehicle without variant	NOK	OK	Element not clickable using the descriptive programming method
Select vehicle with variant	NOK	OK	Element not clickable using the descriptive programming method
Select inexistent vehicle	NOK	OK	Error is shown in log
Automatically read VIN	NOK	OK	Element not clickable using the descriptive programming method
Manually enter VIN	NOK	OK	Element not clickable using the descriptive programming method

Step 3:

Action	UFT	Ranorex	Conclusion
Menu "Livraison client"	NOK	OK	Element not clickable using the descriptive programming method
Menu "Reparation"	NOK	OK	Element not clickable using the descriptive programming method
Menu "Recherche des pannes"	NOK	OK	Element not clickable using the descriptive programming method
Menu "Entretien"	NOK	OK	Element not clickable using the descriptive programming method

Step 4:

Action	UFT	Ranorex	Conclusion
Manage authentication	NOK	NOK	The pop-up is not identified

+APC on/off pop-ups	NOK	NOK	The pop-up is not identified
Select an ECU	NOK	OK	Element not clickable using the descriptive programming method

Step 5:

Action	UFT	Ranorex	Conclusion
Manage information page	OK	OK	No difference between them

Step 8:

Action	UFT	Ranorex	Conclusion
Access the sub-menus	NOK	OK	Element not clickable using the descriptive programming method
Access the parameter measurements	NOK	OK	Element not clickable using the descriptive programming method
Retrieve data	OK	OK	No difference between them

Tests behavior with Repository

Step 0:

Action	UFT	Ranorex	Ranorex Conclusion
Launch DiagBox	OK	OK	
Manage/Close Popups	OK	OK	
Close DiagBox	OK	OK	
Manage authentication	OK	NOK	The popup is not identified

Step 1:

Action	UFT	Ranorex	Conclusion
Select existent brand	OK	OK	
Select inexistent brand	OK	OK	

Step 2:

Action	UFT	Ranorex	Conclusion
Select vehicle without variant	OK	OK	
Select vehicle with variant	OK	OK	
Select inexistent vehicle	OK	OK	
Automatically read VIN	OK	OK	
Manually enter VIN	OK	OK	

Step 3:

Action	UFT	Ranorex	Conclusion
Menu "Livraison client"	OK	OK	
Menu "Reparation"	OK	OK	
Menu "Recherche des pannes"	OK	OK	
Menu "Entretien"	OK	OK	

Step 4:

Action	UFT	Ranorex	Conclusion
Manage authentication	OK	NOK	The pop-up is not identified
+APC on/off pop-ups	OK	NOK	The pop-up is not identified
Select an ECU	OK	OK	Sometimes, the element is not correctly identified in repository by Ranorex

Step 5:

Action	UFT	Ranorex	Conclusion
Manage information page	OK	OK	Sometimes, the element is not correctly identified in repository by Ranorex

Step 8:

Action	UFT	Ranorex	Conclusion
Access the sub-menus	OK	OK	Sometimes, the element is not correctly identified in repository by Ranorex
Access the parameter measurements	OK	OK	Sometimes, the element is not correctly identified in repository by Ranorex
Retrieve data	OK	OK	Sometimes, the element is not correctly identified in repository by Ranorex

UFT Utilization

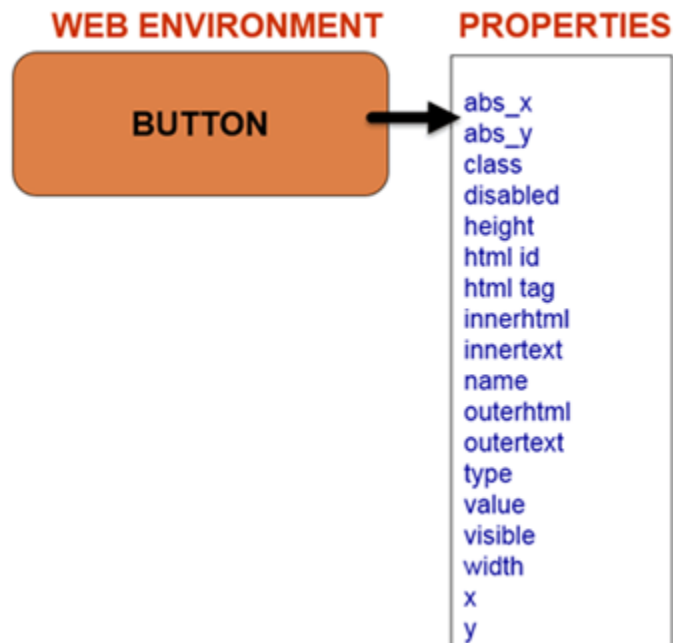
To use UFT for web-tests automatization without repository, the only method is using descriptive programming. Descriptive Programming (also known as Programmatic Description) provides a way to perform operations on objects that are not present in object repository.

In order to do so, UFT de-composes the HTML page, breaking it into multiple elements (or objects) that are stored in the run-time memory (only during the test execution):



The breakdown is done explicitly, using the *ChildObjects()* function.

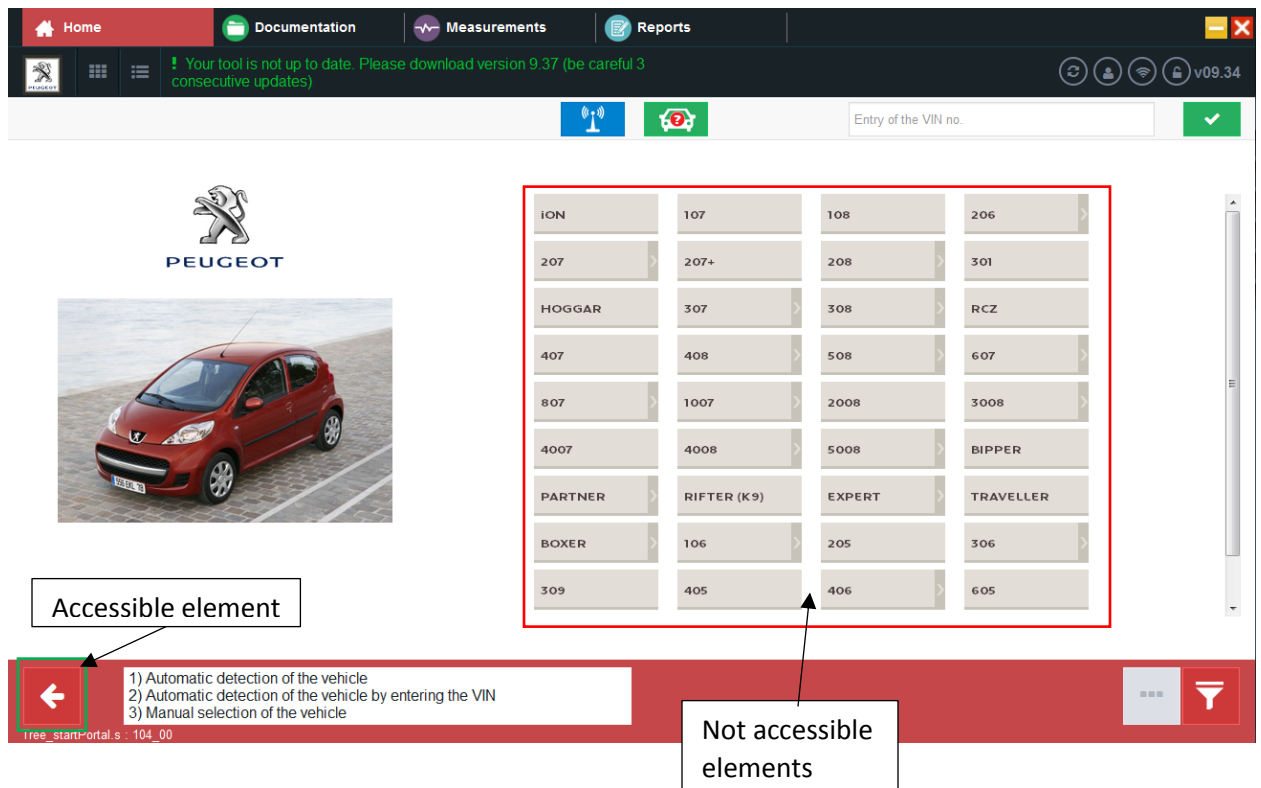
Each element is stored together with his attributes:



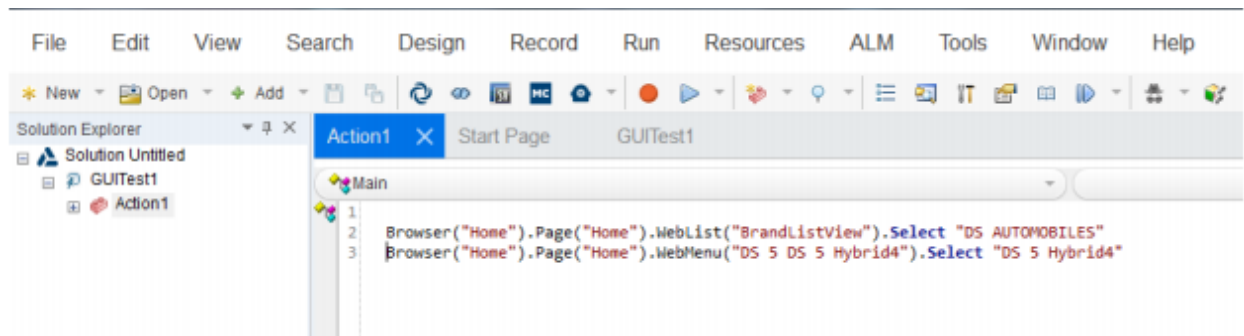
The problem encountered is that, on multiple clickable elements, UFT framework is not able to categorize them into one of the following classes, which are the only classes that supports the *click()* function:

- *WebButton*
- *Link*

Instead, UFT doesn't see this attribute for the majority of GUI elements, like manufacturer or vehicle name, different menus or sub-menus, making it impossible to automatize the actions that are related to these elements. Only a small category are seen as buttons (none as link) and are able to be accessed using this method:



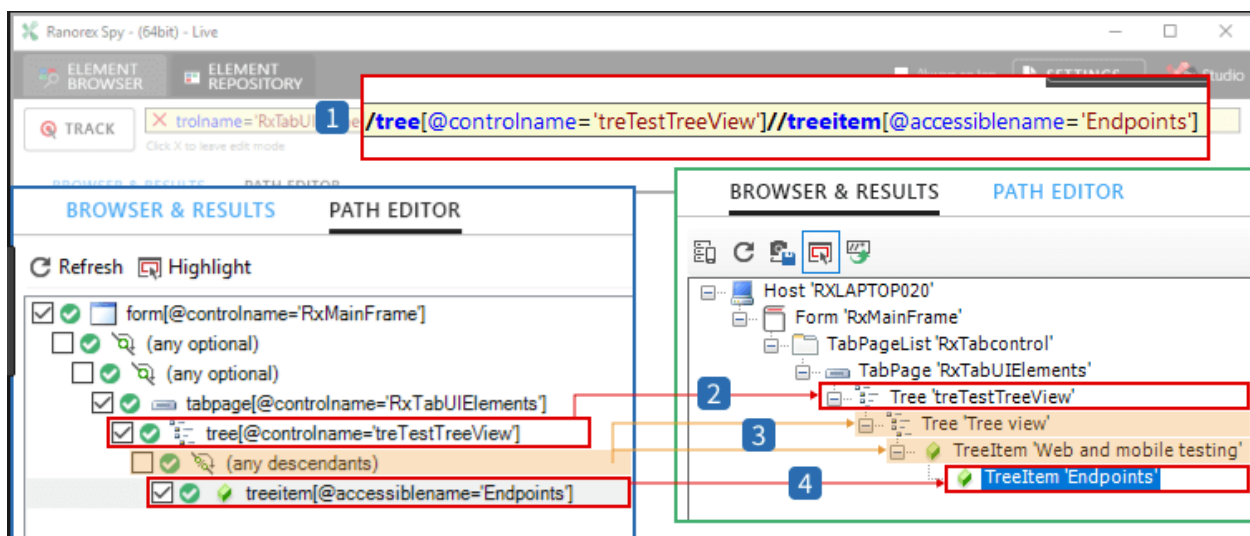
The recorded scripts cannot be used to generalize the user-actions, because the functions that are recorded cannot be used in the descriptive programming method. In the example below, the recorded functions *WebList()* and *WebMenu()* requires that their arguments are to be mapped to repository:



However, this decomposition is ideal for text recognition and the steps that requires to check if a specific element is present or has a specific value can be automatized.

Ranorex Utilization

For dealing with the same type of behavior (objects not present in the repository), Ranorex offers a method called RanoreXPath. Basically, this method also decomposes the webpage, but instead of having multiple objects and accesing them by their attributes, now we have a tree element:



The GUI elements that in UFT were converted into objects, now they are seen as tree nodes or leafs. Based on this internal representation of the webpage, the elements can be accessed using their RanoreXPath (which is basically a normal XPath expression, but adapted to the Ranorex vocabulary).

The breakdown of the page is done implicitly, without any user input or action.

The automatization functions uses this paths to access the elements, modifying and adapting the recorded actions to obtain a generic goal (e.g. choose a vehicle).

Repositories

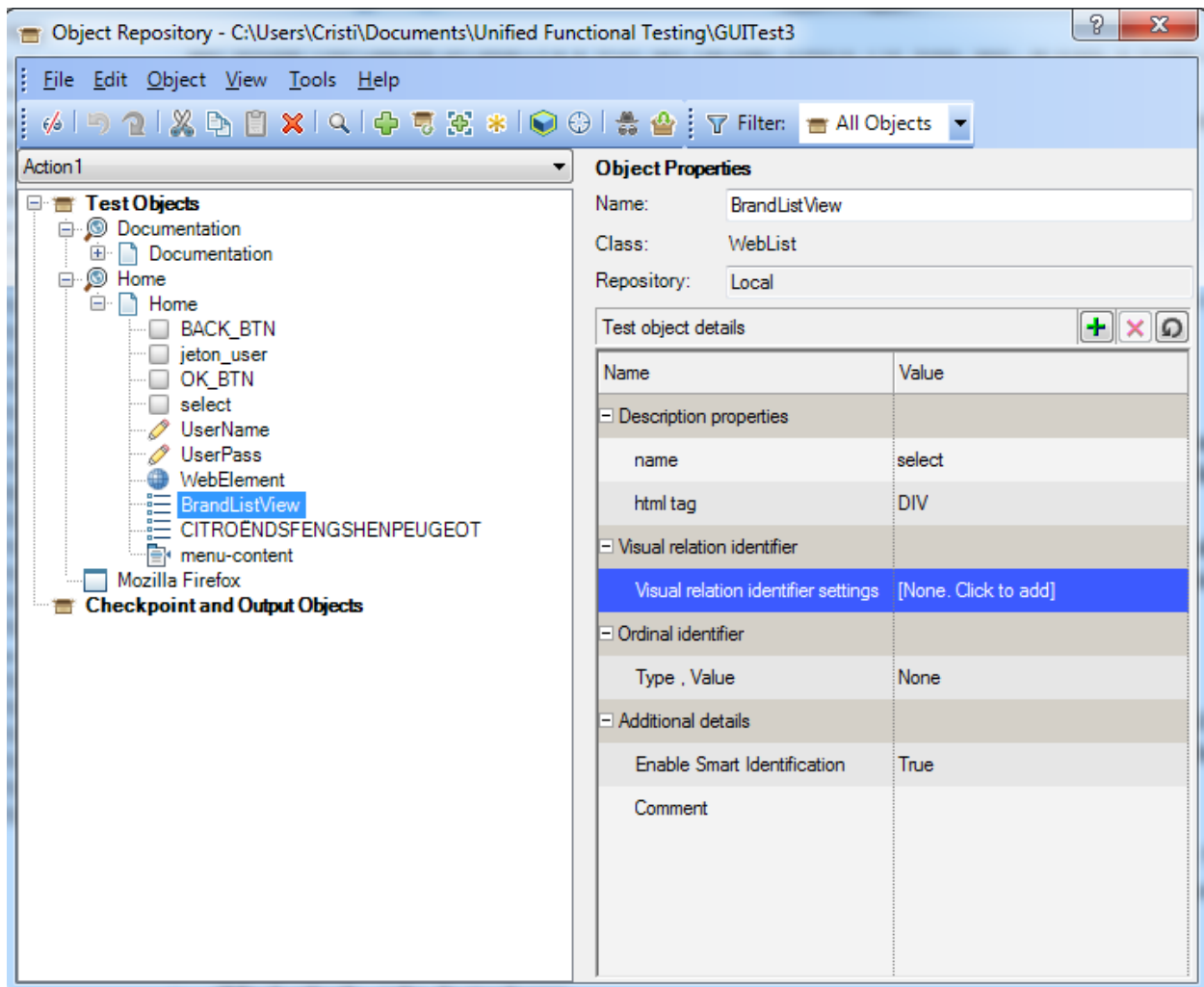
The repository behaves like a database, in storing information and access routes to various elements. In both frameworks, objects that are stored in repositories can be directly accessed by automation scripts.

UFT repository

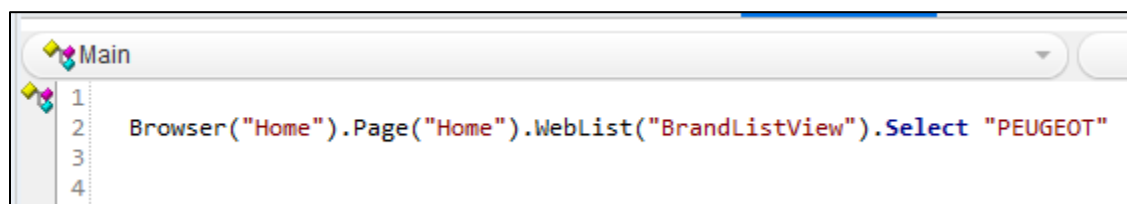
In UFT, there are several ways to add GUI objects to the repository:

<https://www.softwaretestingclass.com/uft-qtp-training-tutorial-8-object-repository-in-uft/>







Once added, there is no need to use the “descriptive programming” method: it can simply call the object from repository:



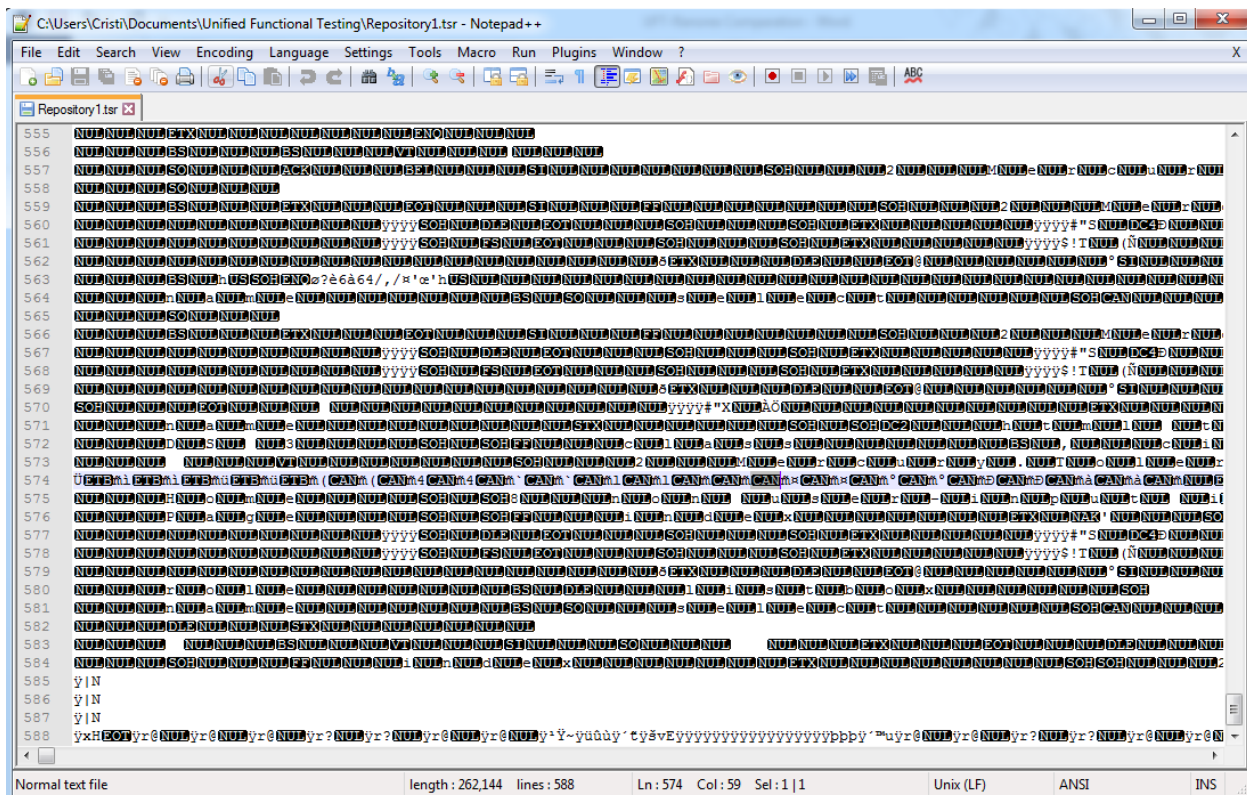
and use it in the script:



The repository is saved in a file with the extension `.tsr`:

Name	Date modified	Type	Size
 GUITest1	12/17/2018 6:18 AM	File folder	
 GUITest2	12/17/2018 6:23 AM	File folder	
 GUITest3	12/18/2018 3:25 PM	File folder	
 Test	12/16/2018 6:18 PM	File folder	
 Test2	12/16/2018 6:14 PM	File folder	
 Repository1.tsr	12/18/2018 2:24 PM	TSR File	256 KB

However, this file can be opened and edited (objects added or removed) only through the UFT visual interface. When trying to open this file using an external editor (Notepad++, in this case), the protected content was not properly displayed:



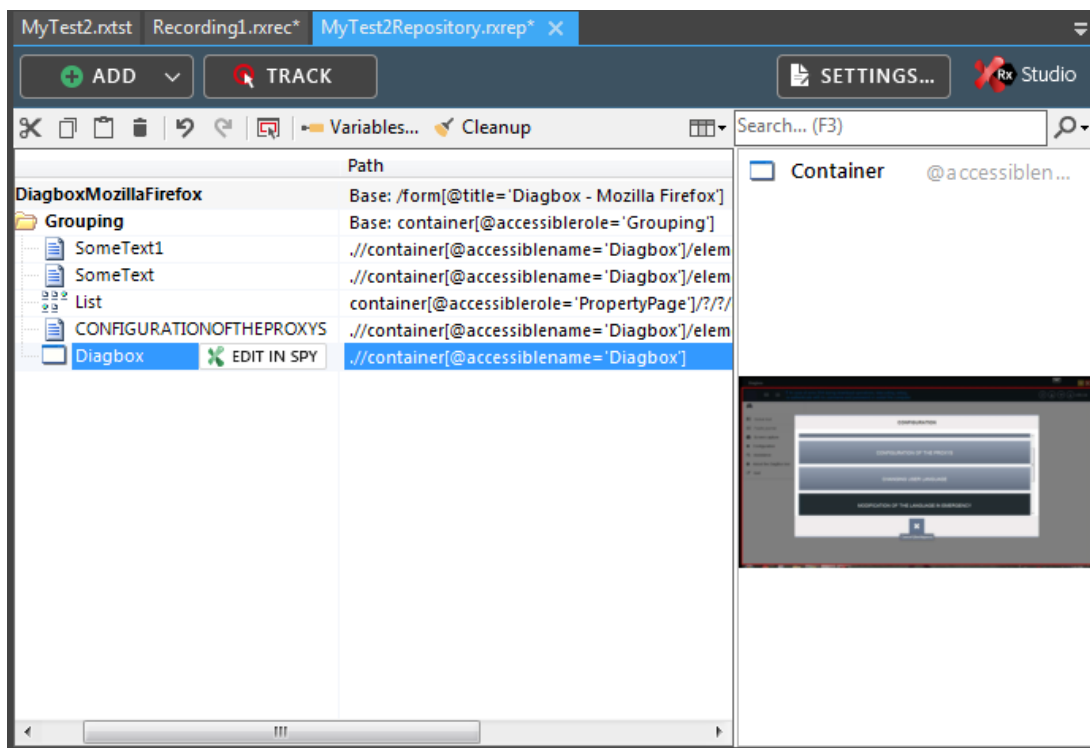
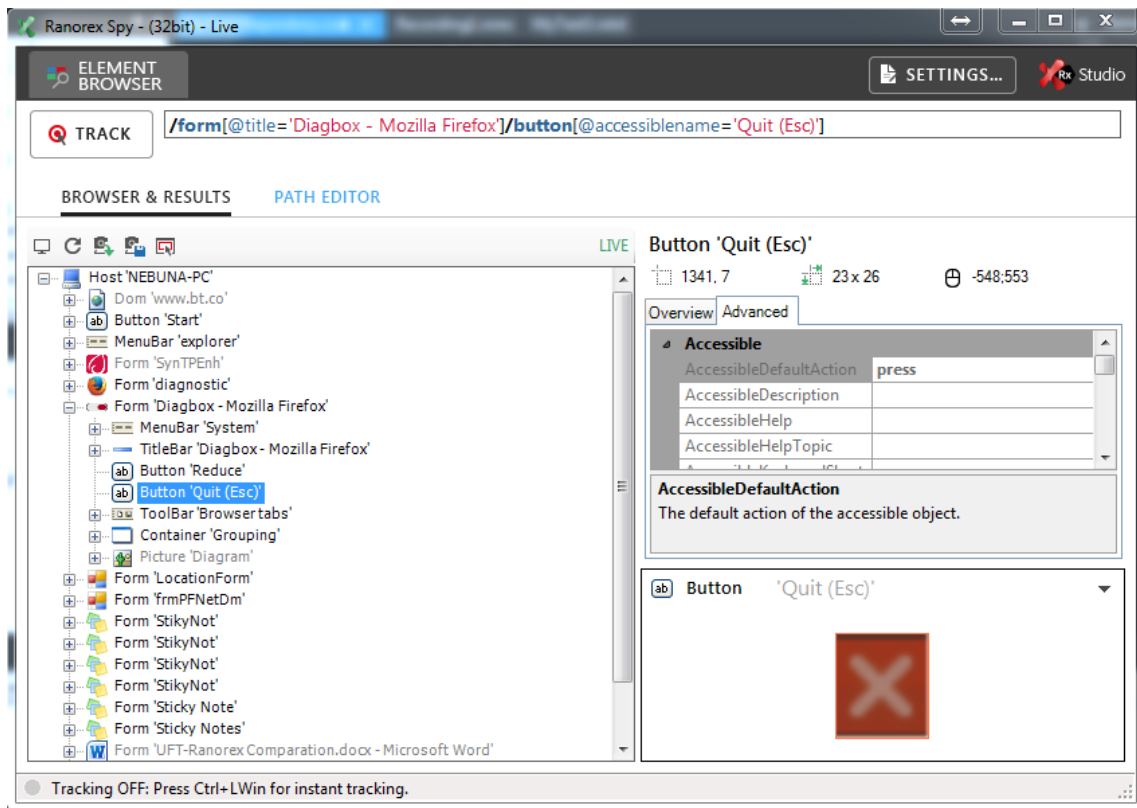
In conclusion, with our current level of knowledge, we are not able to automatically create the repository using an HUR file. The only way to create this repository would be to manually add objects, using one of the methods described in the given link.

Ranorex repository

As UFT, Ranorex has more than one way to create an object repository:

<https://www.ranorex.com/help/latest/ranorex-studio-fundamentals/repository/creation-repository-items/>

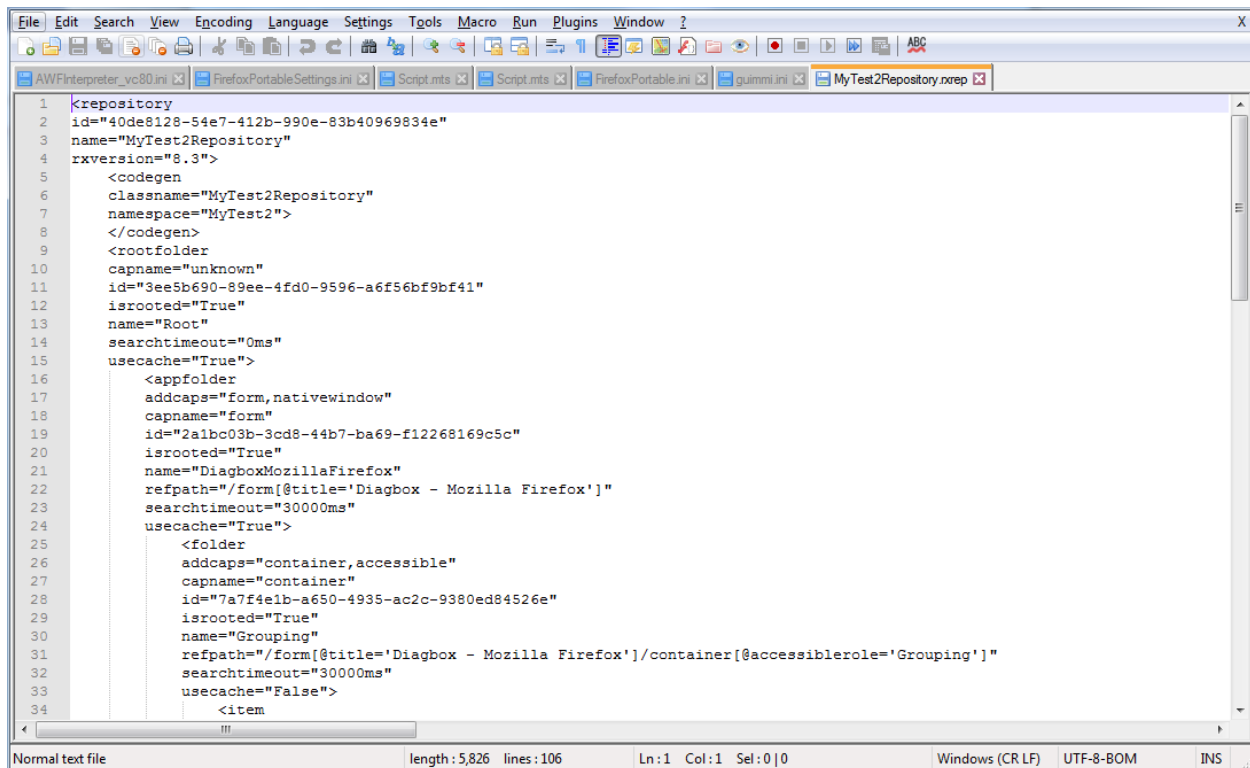
Like UFT, the objects are saved similarly in the repository, with the difference that only few object properties are present, but for each object the RanoreXPath is identified:



The repository is saved in the project, with the extension *.rxrep*:

Name	Date modified	Type	S
bin	04-Dec-18 8:33 AM	File folder	
obj	04-Dec-18 8:33 AM	File folder	
app.config	04-Dec-18 8:33 AM	XML Configuratio...	
AssemblyInfo.cs	04-Dec-18 8:33 AM	Visual C# Source f...	
MyTest2.csproj	04-Dec-18 8:33 AM	Visual C# Project f...	
MyTest2.csproj.pref	04-Dec-18 9:17 AM	PREF File	
MyTest2.csproj.rxuser	04-Dec-18 9:17 AM	RXUSER File	
MyTest2.nxtmg	04-Dec-18 8:33 AM	Ranorex Test Suite	
MyTest2.nxtst	04-Dec-18 8:33 AM	Ranorex Test Suite	
MyTest2Repository.cs	19-Dec-18 7:31 PM	Visual C# Source f...	
MyTest2Repository.ximg	19-Dec-18 7:31 PM	RXIMG File	
MyTest2Repository.rxrep	19-Dec-18 7:34 PM	Ranorex Repository	
Program.cs	04-Dec-18 8:33 AM	Visual C# Source f...	
Recording1.cs	04-Dec-18 8:33 AM	Visual C# Source f...	
Recording1.rxrec	04-Dec-18 8:33 AM	Ranorex Recording	
Recording1.UserCode.cs	04-Dec-18 8:33 AM	Visual C# Source f...	

When opened with an external editor, the file content can be read and interpreted:



```
1 <repository
2 id="40de8128-54e7-412b-990e-83b40969834e"
3 name="MyTest2Repository"
4 rxversion="8.3">
5   <codegen
6     classname="MyTest2Repository"
7     namespace="MyTest2">
8   </codegen>
9   <rootfolder
10     capname="unknown"
11     id="3ee5b690-89ee-4fd0-9596-a6f56bf9bf41"
12     isrooted="True"
13     name="Root"
14     searchtimeout="0ms"
15     usecache="True">
16     <appfolder
17       addcaps="form,nativewindow"
18       capname="form"
19       id="2a1bc03b-3cd8-44b7-ba69-f12268169c5c"
20       isrooted="True"
21       name="DiagboxMozillaFirefox"
22       refpath="/form[@title='Diagbox - Mozilla Firefox']"
23       searchtimeout="30000ms"
24       usecache="True">
25       <folder
26         addcaps="container,accessible"
27         capname="container"
28         id="7a7f4e1b-a650-4935-ac2c-9380ed84526e"
29         isrooted="True"
30         name="Grouping"
31         refpath="/form[@title='Diagbox - Mozilla Firefox']/container[@accessiblerole='Grouping']"
32         searchtimeout="30000ms"
33         usecache="False">
34         <item
```

Normal text file length: 5,826 lines: 106 Ln: 1 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8-BOM INS

Thus, there is a chance to translate the HUR files into Ranorex repository items, and creating a new repository for each DiagBox evolution. However, this aspect must be confirmed after a HUR file format analysis (together with a full dissection of the repository file).

Conclusion

After the comparative analysis between UFT and Ranorex automatization frameworks, these are the result:

1. If an object repository is used (the necessary elements are present in the internal database), UFT is simpler to use (lighter code, bigger versatility) than Ranorex; also, it correctly manages the tasks that require pop-up handling
2. If the object repository is not to be used, Ranorex can handle multiple elements than UFT (all the lists and menus that are used as a link) and is better suited to be used

Taking into account the fact that the DiagBox interface can have evolutions (new elements to be added – new vehicles, new I/O controls, elements to be deleted – old parameters), Ranorex is the framework to be used if the repository is not already populated with DiagBox objects.

However, there are some limitations:

- Ranorex is not able to fill in the user details in the authentication phase (the user and password fields are not distinguishable)

If there is a way to automatically populate the repository with DiagBox objects, UFT behaves better and can handle all the necessary actions.