An Automated Testing Tool Using UI Structure

Nutharat Harnvorawong, Taratip Suwannasart, Member, IAENG

Abstract-Testers usually run a new version of software against existing test cases to validate that changes do not cause any unexpected results in legacy functionalities when the software is modified or enhanced. A solution that can reduce cost is automated testing. However regression testing and automated testing are resources consuming and high cost. In this paper we propose a framework called Neo Automation Framework (NAF) which allows testers to create and develop automated test cases easily and efficiently. The framework can generate UI structure of a given form inside application. The UI structure is a list of usable UI controls in hierarchical data structure in a class format of programming language. Automated test cases can be automatically generated from the UI structure. The framework also provides tools for simply modifying test case and analyzing usage of UI controls in the test cases. The analyzer tool can identify UI controls which are not used and so testers can be aware of area lacking of test in their test cases.

Index Terms—Automated Testing, GUI, Automation, Software Testing, Neo Automation Framework

I. INTRODUCTION

Toftware testing is one of essential parts in software D development life cycle to ensure software works correctly as expected [1]. Software testers need to understand software under test in order to design test cases properly and therefore defects in the software can be detected or discovered. When a new feature is developed, new test cases are created to validate functionalities and ensure if quality of the software is satisfied. Not only the new feature is tested but also existing features should be tested to ensure what was previously working still works correctly [2]. This is regression testing. Obviously number of test cases can be increasing rapidly and effort to execute test cases is numerous whereas resources are still being the same. It is a hard work of testers to validate new developed features and also validate all existing features every time a new version of software is released.

Nowadays automated testing is widely used in software testing [3] especially for regression testing as it is fast, reliable, and repeatable. It can reduce cost of testing significantly. However, there are some concerns when creating scripts to use in automated testing such as overhead for testers to learn an automation framework, overhead of maintaining automated test cases when software is changed, how much automated test cases are enough?, where is area lacking of test?, and etc. In this paper we propose an automation test framework which can help testers to

Manuscript received December 9, 2013; revised December 23, 2013.

N. H. Author is a student at the faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (e-mail: Nutharat.H@student.chula.ac.th).

T. S. Author is with the faculty of Engineering, Chulalongkorn University, Bangkok, Thailand (e-mail: Taratip.S@chula.ac.th).

automate test cases easily and help identify parts in software which lack of testing in automated test cases.

II. RELATED WORKS

There are many techniques used in automated testing. A technique which is easy for creating automated test cases called capture & playback [4] which was introduced in many testing tools in market such as Microsoft Visual Studio [5], and etc. An advantage of this approach is that automated test cases can be created fast but a disadvantage is that UI controls are only recorded if it is used in the automated test cases. If testers want to add more test cases for other UI controls, testers cannot reuse an existing list of UI controls and need to rerecord actions for new test cases specially.

Another technique that is frequently used is model based [6]. André M. P. Grilo and his team [7] have worked in GUI model for software testing. They have proposed to extract structural information of GUI and store it in XML format. From their work, it shows if UI controls in application are recorded in hierarchical data structure, i.e. XML, it could be easy to use and can be processed for other purposes easily.

Another interesting technique used when executing automated test has been proposed by Alex Ruiz and his team [8]. They have proposed an approach how to write automated test cases by using library with concept of "fluent interfaces" to simplify automated test cases in coding style. Their work shows that automated test cases written in code like programming language is readable and is also powerful as testers can use power of programming language to control flow of a test scenario.

III. NEO AUTOMATION FRAMEWORK

Neo Automation Framework (NAF) is an automation test framework which provides tools to facilitate testers' works. It has been designed to support automated testing with Windows application. We chose Windows application developed by using .NET framework as a target of application under test because it is a well-known programming language and is widely used. NAF has a library of API for each supported UI control [9] which wraps UIA [10] commands for easy to use. The framework diagram is shown in **Fig 1**.



Fig 1. Neo Automation Framework

A. Software

A Graphical User Interface (GUI) application consists of UI controls that can interact with users, for example button, radio button, and text box. Layout of these UI controls are designed and set since design phase by programmers. In case of many UI controls, they probably are grouped in a UI container control such as group box, tab, and etc., for ease of use.

A Windows application developed by using .NET framework and running on Windows operating system can be used in automated testing of this paper.

B. UI Structure Generator

UI Structure Generator is a tool that analyzes application under test and extracts information of UI controls such as automation ID, type of control, position of control, and etc., and generates a list of UI controls as a hierarchy and saves it into a format of class. Testers can use the generated UI structure in their test project and can access the UI control by accessing properties of UI structure class.

This tool has an ability to find a form which is expected to be a root in UI structure class. Testers just drag and drop cursor onto target form of application under test, then the tool will automatically extract UI control information of selected form and its children UI controls.



Fig 2. UI structure Generator tool with an example of UI structure

UI Controls that are usable in automated test will be displayed in a hierarchical tree as shown in **Fig 2**. Testers can select each UI control and see where the UI control is and also be able to change its name to be more understandable or descriptive. The original name of each UI control comes from automation ID of the control, therefore sometimes it is not a good name or is a code defined by programmer. If the UI control is not supposed in automated test such as label control, testers can mark it as hidden control so it will not be generated in a UI structure class.

One of useful data retrieved from UI control is automation ID. It is unique under the same parent control since it is used as an identifier. In some cases, UI controls do not have automation ID. NAF has a process to handle such cases. For example tab item control which is an individual tab in tab list control, by default it does not have automation ID, NAF uses name of tab item control instead when finding a target UI control, e.g. running automated test cases to access the tab item.

Once testers have completed reviewing the UI controls, they can simply generate a UI structure and the tool will show a generated UI structure in the preview area and testers are able to save it into a file, as shown in **Fig 3**.



Fig 3. UI structure Generator tool with an example of generated UI structure class

C. UI Structure

UI structure contains list of UI controls in hierarchical data structure recorded in format of C# language. Each form in application under test is extracted and recorded in a separate class.

A form will be generated as a class and its children controls will be generated as properties with appropriated type of control. The class has a variable storing automation ID value of the form. Also property members have automation ID values in a get method. An example of UI structure class is shown in **Fig 4**.



Fig 4. An example of UI structure class

D. Test Case Generator

Once testers have the UI structure class they should add it into the test project and compile it into an assembly file such as DLL.

Test Case Generator is a tool that analyzes the assembly which contains the UI structure class and automatically generates automated test cases based on the given UI structure class. All UI controls in the given UI structure class will be used at least once in a sequence of Top-Down and then Left to Right based on position of UI control. This is to ensure all UI controls have been accessed in order to ensure UI controls of application have been tested. However, the generated test cases by accessing UI controls in such sequence may not be valid test cases. Testers can modify test cases manually or use the editor tool which will be described below. **Fig 5** shows our Test Case Generator tool and an example of generated test case in the tool.

📧 Test Case Tool		x
Generate Test Case	e Edit Test Case	
Assembly: C:\Us	sers\nutharath\Desktop\Test-Dlls\UITest.dll	
	Load Assembly	
	Euga vesembly	
UI structure class:	UIStructure.FormUIStructureGenerator4	
Generate UI Tes	st Class	_
Namespace:	UIStructure	
Class name:	FomUIStructureGenerator4_UITestClass	
Method name:	UITestMethod	
UI test class:		_
// This code wa // Changes to th //	s generated by a tool. is file may cause incorrect behavior and will be lost if the code is regenerated. st; awing: incovs.Automation; visual Studio. Test Tools. Unit Testing: auton Framework. Common; tructure { Jass] class FormUIStructureGenerator4_UITestClass { [Test Method] public void UITestMethod() { // ####### Jack Automation Framework: Settions ########	H
		-
	•	
Neo Automation	Framework Settings Generate Save As	
	Log File Qiose	•

Fig 5. Test Case Generator tool and an example of generated test case

Test case generation consists of 3 parts.

1. Configuring NAF settings

This part contains commands to configure setting values of NAF such as Show highlight of target UI control, Use caching of UI control, and Duration to delay for searching UI control or after action performed.

2. Initializing variable of target form

This part creates and instantiates a variable of target form so it can be called by later test steps.

3. Test steps

Actions to be performed on each UI control are included in the test steps. If the UI control has multiple actions, the commonly used action will be selected. For example, checkbox control has Toggle, GetState actions; the default action is Toggle.

After a test case is generated, it can be saved into a new file in C# format.

1	
2	// This code was generated by a tool.
3	// Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
4	[//
2	
6	eusing system.urawing;
2	using Microsoft.Visualstudio.lestloois.Unitlesting;
8	using NeoAutomationFramework.Common;
9	
10	Enamespace UIStructure {
11	
2	[iestLiass]
	public class screeni_ullestclass {
4	FT
2	[lestrethod]
0	Dublic Vola Ullestmethod() {
	// manage the Automatical Second Contribute Manage
.0	// ##### Neo Automation Framework Settings #####
19	// [[_this] Negotichicki
10	// [Setting] wante-showinghinght
12	NeoAutomationFrameworkCenter.ShowFigniignt = true;
2	// [Satting] Name-UnitUnitIntiana Time-utTuWilliseands
10	// [Setting] waine-waitontitivaliable_lineoutinitiseconds
5	Neodatonation rameworktenter .wartontiinvariabre_imeoutiniiiisetonus = 10000,
6	// [Satting] Name-TargetElement HighlightColor
27	NeckstantionEconomous/Content TagetElement WithlightColor = (alor Economy/c65536);
8	Representation remember remember in general in general in general in the second of the second s
0	// ##### Toitialization Commands #####
10	
	<pre>// [Initialization] ObjectName=screen1, UIStructureClass=UIStructure.Screen1</pre>
2	UIStructure.Calculator/UStructure_screen1 = new UIStructure.Calculator/UStructure():
33	
14	// ##### Test Steps #####
15	
36	// [TestStep] Description=, Name=NumOperand1. Type=NeoAutomationFramework.Controls.NeoSpinner, Bas
37	<pre>screen1.NumOperand1.SetValue("0"):</pre>
8	
9	// [TestStep] Description=, Name=NumOperand2, Type=NeoAutomationFramework.Controls.NeoSpinner, Bas
10	<pre>screen1.NumOperand2.SetValue("0");</pre>
11	
12	// [TestStep] Description=, Name=TxtExpression, Type=NeoAutomationFramework.Controls.NeoEdit, Base
13	<pre>screen1.TxtExpression.SetValue("");</pre>
14	
15	// [TestStep] Description=, Name=TxtAnswer, Type=NeoAutomationFramework.Controls.NeoEdit, BaseNeoC
6	<pre>screen1.TxtAnswer.SetValue("");</pre>
17	
18	}
19	}
60	
51	

Fig 6. An example of a generated test case

Fig 6 shows an example of a generated test case.

E. Test Case Editor

Test Case Editor is a tool that provides basic operations to modify test cases so testers can update test cases according to test design. Testers can add a new test step, remove a test step or change order of a test step as they want to follow a real user scenario. **Fig 7** shows the Test Case Editor tool and an example of a test case being edited.

Test Case	e Tool							
enerate Te	est Case	Edit Test Case						
UI Structu	ire							
Assembly:	C:\Use	rs\nutharath\Des	ktop\Test-Dils\UITes	t.dl				Load
Class:	UIStruc	cture FormUIStruc	tureGenerator4				•	
III Test								
Class	D:\Proje	cts\Nott-NeoAuto	mationFramework\UI	Test\UITe	et ce			Load
Method:	UlTestM	ethod111222						
Test Case								
Method na	ame: Ul	TestMethod1112	22					
Neo Arto	mation Fra	amework Settinos	Edit					
Name				Value				
ShowHig	hlight			true				0
Caching8	Enabled			true				
TrySearc	hDescen	dants		false				
ThrowEx	ception#P	PerformingActionIs	NotSuccessful	true				
Test step	a:					_		
	Select	Description	Neo Control		Operation		Command	Firs
1		Nott step 1	<cannot co<="" edit="" neo="" td=""><td>ntrol of</td><td><cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.LabelTargetInstruction.GetText();</td><td></td></cannot></td></cannot>	ntrol of	<cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.LabelTargetInstruction.GetText();</td><td></td></cannot>	ex	formUIStructureGenerator.LabelTargetInstruction.GetText();	
2	m		«Cannot edit Neo Co	ntrol of	<cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.LinkLabelLogFile.Invoke():</td><td></td></cannot>	ex	formUIStructureGenerator.LinkLabelLogFile.Invoke():	
3		Nott step 52	<cannot co<="" edit="" neo="" td=""><td>ntrol of</td><td><cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.ButtonClose.Click();</td><td>Dow</td></cannot></td></cannot>	ntrol of	<cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.ButtonClose.Click();</td><td>Dow</td></cannot>	ex	formUIStructureGenerator.ButtonClose.Click();	Dow
4		Nott step 7	<cannot co<="" edit="" neo="" td=""><td>ntrol of</td><td>Cannot edit operation of</td><td>ex</td><td>formUIStructureGenerator.ButtonClose.Click();</td><td>Las</td></cannot>	ntrol of	Cannot edit operation of	ex	formUIStructureGenerator.ButtonClose.Click();	Las
5		222333	<cannot co<="" edit="" neo="" td=""><td>ntrol of</td><td><cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.TabItem_2.ComboBox2.Collapse();</td><td></td></cannot></td></cannot>	ntrol of	<cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.TabItem_2.ComboBox2.Collapse();</td><td></td></cannot>	ex	formUIStructureGenerator.TabItem_2.ComboBox2.Collapse();	
6	m	Nott-Copy1	<cannot co<="" edit="" neo="" td=""><td>ntrol of</td><td><cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.Tabitem_2.CheckBox2.Toggle(true):</td><td></td></cannot></td></cannot>	ntrol of	<cannot edit="" of<="" operation="" td=""><td>ex</td><td>formUIStructureGenerator.Tabitem_2.CheckBox2.Toggle(true):</td><td></td></cannot>	ex	formUIStructureGenerator.Tabitem_2.CheckBox2.Toggle(true):	
7	1		CheckBoxShowHigh	light 🔻	Toggletrue)	-	formUIStructureGenerator.CheckBoxShowHighlight.Toggle(true);	
▶ 8			ButtonClose	-	Click()	-	formUIStructureGenerator.ButtonClose.Click();	
Add	ł 🛛	Copy & Paste	Delete)				
							Librio All	Save
							20100.1	2010

Fig 7. Test Case Editor tool

F. UI Usage Analyzer

UI Usage Analyzer is a tool that analyzes usage of UI controls in automated test cases. The tool requires 2 inputs in order to analyze UI usage: assembly of UI structure class and assembly of test cases.

A technique used to find usage of UI control in test cases is code refactoring. All UI controls in a specified UI structure will be listed out and then iterates a process of finding usage for each UI control one by one. UI control can be called by a test method directly or other shared method. For example, testers may create a common method for the login screen so each test method calls to the shared method instead of having steps perform in the login screen. Therefore, the result of UI control usage will be from traversing through all call chains in the application.

The result shows number of UI controls in the UI structure and number of UI controls which are used and not used in a table. Moreover, it can point out which UI controls are not used so testers can consider adding more tests to increase coverage in their testing. **Fig 8** shows the UI Usage Analyzer tool and the UI usage result from UI structures and test cases.

😥 UI Usage Analyzer								×
UI Structure		U	Test					
Select File Select Folder 📃 Incl	ude subfolder		Select File	Select Folder	📄 Inc	lude subfolder		
UI structure classes: 2/2 selected	Clear	Te	st methods:	3/4 selected			ç	lear
Select UI Structure Class /	Assembly	s	elect Test M	ethod	Test Clas	5	Assembly	
UIStructure.FrmMain C	:\Users\nutharath\De		UITest	/lethod	UIStructur	e.FmMain_UITestClass	C:\Users\nutharath\[Deskt
UIStructure CalculatorUIStructure C	:\Users\nutharath\De		UITest	/ethod2	UIStructur	e.FmMain_UITestClass	C:\Users\nutharath\[Deskt
			☑ StartLa	SimulatorServer	UIStructur	e.FmMain_UITestClass	C:\Users\nutharath\[Deskt
			☑ UITest	lethod	UIStructur	e.Screen1_UITestClass	C:\Users\nutharath\[Deskt
<	F	٠			III			Þ
UI Usages UI structure classes:				Usages: 92 N	eo controls	. 62 usages in 2 methods	in 1 classes in 1 asse	lear mblie
UI Structure Class Assembly Contr Court	rol Used (Percent)	Not Used	Not Used (Percent)	Neo Control (Ci BtnClose	slice) 🔺	Method (Caller)	Class (Caller)	
UIStructure.FmMain C:\Users\ 92			66.30%	BtnDriver_Apply		StartLabSimulatorServer	UIStructure.FmMa	
UIStructure.Calculator C:\Users\ 5	4 80.00%	1	20.00%	BtnDriver_Apply		UITestMethod2	UIStructure.FmMa	
				BtnMode_Start		UITestMethod2	UIStructure.FmMa	
				BtnMode_Start		StartLabSimulatorServer	UIStructure.FmMa	
				BtnMode_Stop				
				BtnOpenLogFile				
				BtnParser_Parse	•			
				Btn Send Messag	le_Quic			-
				Btn Send Messag	le_Quic			-
				Btn Send Messag	e_Send			-
				<			•	
						Analys	te <u>R</u> eport	
						ما	g File Qose	

Fig 8. UI Usage Analyzer tool

The result can also be shown in a printable format as a usage report as shown in **Fig 9** and **Fig 10**.

eneral Informatio	n						
	/11						
ssembly: TestProject1, Ve	ersion=1.0.0.0), Culture=ne	sutral, Public	KeyToken=n	ull		
ath: C:IU sersinuthan	ath\Desktop\T	estP roject1\	TestProject1	Vbin\Debug\T	TestProject1.	511	
amespace: UIStructure							
lass: Fm Main							
UI Structure Class	Neo Control	Count	lised	Used (%)	NotUs	ed.	Not Used (%)
m Main		92	31	33.	70	61	66.3
	,	eo Control	Usage				
	- V						
			>				
		Used Court		sed Count			
	-	Used Court		sed Count			
	-	Used Court	R Not U	aed Count			
		Used Court	R Not U	sed Count			
	-	Used Court	Not U	sed Count			
covered Neo Cont	rols	Used Court	n Not U	sed Count			_
covered Neo Cont	rols	Used Court	r Not U	sed Count			_
Covered Neo Cont	rols	Used Coun	Type	sed Count	Count		_
Covered Neo Contr GrpMode	rols	Used Coun	Type	red Count	Count	2	_
Covered Neo Contr GrpHoe Contr GrpHoe Tabiten	rols	Used Coun	Type	sed Count	Count	2 2	_
Covered Neo Cont GrpHode Tabltem_1	rols	Used Coun	Type n n	ted Count	Count	2 2 2 2	_
Neo Contro GrgMode Tabitem Tabitem_1 Tabitem_2	rols	Used Count Used Count Neo Group Neo Tablter Neo Tablter	Type n n	sed Count	Count	2 2 2 2	_
Covered Neo Contr Optil de Tabitem_1 Tabitem_1 Tabitem_3	rols	Used Coun Neo Group Neo Tabiter Neo Tabiter Neo Tabiter	Type n n n	sed Count	Count	2 2 2 2 2 2	_
Covered Neo Contr Grplode Tablem Tablem_1 Tablem_2 Tablem_3 Num Kode_Server_Port	rols	Used Count Neo Group Neo Tablier Neo Tablier Neo Tablier Neo Tablier Neo Tablier	Type n n n r	ed Count	Count	2 2 2 2 2 2 2 2 2	
Covered Neo Cont Heo Contr Tablen_1 Tablen_2 Tablen_2 Tablen_3 Num Mode_Server_Port Robit de_Server	rols t	Used Coun Neo Group Neo Tablter Neo Tablter Neo Tablter Neo Tablter Neo Spinne Neo RadioB	Type n n r	sed Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
International States	rols	Used Coun Neo Group Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Spinne Neo RadioB	Type n n r r	ted Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	_
Rec Contr Orpitole Tablem Tablem_1 Tablem_1 Tablem_2 Tablem_2 Tables_Sec_ Fable Gener Entitiod_Stat	rols of	Used Coun Used Coun Neo Group Neo Tabler Neo Tabler Neo Tabler Neo Spinne Neo Spinne Neo Spinne Neo Spinne Neo Spinne Neo Spinne	Type n n n r	ed Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	_
Covered Neo Cont Gradea Tablea Tablea Tablea, J Tablea, J Ta	rols not	Used Coun Used Coun Neo Group Neo Tablier Neo Tablier Neo Tablier Neo RadioB Neo Button Neo Group Neo Group	Type n n r tutton	eed Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
Record Orpitode Tablem, 1 Tablem, 2 Tablem, 2 Tablem, 2 Tablem, 3 Tablem, 3 Tablem, 3 Tablem, 3 Tables, 3	rols ol	Used Count Neo Group Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Group Neo Group Neo Group	Type n n n r r Box	sed Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	_
Covered Neo Cont Gradioa Tatima 1 Tatima 2 Tatima 3 Humilion Server J P Robio Server J Robio Server J	rois ol	Vsed Count Vsed Count Neo Tablier Neo Tablier Neo Tablier Neo Tablier Neo RadioB Neo Button Neo Group Neo Group Neo Group Neo Group	Type n n r Box	red Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
Records Orplote Tablem, 1 Tablem, 2 Tablem, 2 Tablem, 2 Tablem, 3 Tablem, 3 Tablem, 3 Tablem, 3 Tablem, 4 Tables, 3 Tables, 3	rois ol t maracterForint Top	Vised Court Neo Group Neo Tabiter Neo Tabiter Neo Tabiter Neo Tabiter Neo Spinne Neo Spinne Neo Group Neo Group Neo Combo Neo CheckE	Type n n n r eBox Box	ted Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	_
Records Alex Control Gradient Tablem, 2 Tablem, 2 Tablem, 2 Tablem, 2 Sector, 2 Sector, 2 Table, 2 Tab	TOIS of t Top Color_Detsuit	Used Count Neo Group Neo Tablier Neo Tablier Neo Tablier Neo Spinne Neo Spinne Neo Group Neo Group Neo Group Neo Combo Neo Combo	Type n n n r futton	ted Count	Count	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	

Fig 9. An example of UI usage report

Ladading, Remotifical Calor, Def Henhysefink 2 Ladading, Ernot-Stor (Chang Weinhysefink 2 Caborner, Japp Weinhom 2 Cabo	Lnk:Settings_ErrorColor_Default	NeoHyperlink	4
Liskeinag, EmrColor_Change Meehyenink 2 Liskeinag, EmrColor_Change Meehyenink 2 CbdOniver, Udag Meehyenink 2 CbdOniver, Udag Meehyenink 2 Challoniac (interpreted transport (interpreted transport Meehyenink 2 Liskeinag Constructions Meehyenink 2 Challoniac (Southamatag) Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Meehyenink 2 Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Challonia (Lasimatag) Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyenink 2 Meehyeni	LnkSettings_RemoteHostColor_Def ault	NeoHyperfink	2
Intolmer, Apply HeeButton 2 Ecolorier, II. doal HeeConsbox 2 OpOHrw. JI. doal HeeOrop 2 Charlon Ling, Meyneted Heasgen HeeConsbox 2 2 Charlon Ling, Constrol Meyneted Heedon 2 3 Charlon Ling, Constrol Markey Heedon 2 3 Charlon Ling, Constrol Markey Heedon 2 3 Charlon Constrol Meedon 2 3 Charlon Constrol Meedon 2 3 Charlon Constron	LnkSettings_ErrorColor_Change	NeoHyperlink	2
Challonding Necl Canababa 2 Chalonding Necl Canababa 2 <	BtnDriver_Apply	NeoButton	2
IngOnerup Modi 2 OrgOnerup Modi 2 Chall obsitue / Ingoreal Measure Meet Centrolloo X 2 Chall obsitue / Ingoreal Measure Meet Centrolloo X 2 Chall obsitue / Ingoreal Meet Centrolloo X 2 Chal	CboDriver_Model	NeoComboBox	2
Childonic (Independent Message Head Net Chead Sox 2 Latitotic (Independent Message Head Net Consolito (S) 2 Latitotic (Londino): Sovie Message Head Net Chead Sox 2 Latitotic (Londino): Sovie Message Head Net Chead Sox 2 Latitotic (Londino): Sovie Message Head Net Chead Sox 2 Challonic (Londino): Sovie Message Head Net Chead Sox 2 Challonic (Londino): Sovie Message Net Red Solution 2 Challonic (Londino): Sovie Message Net Red Solution 2 Challonic (Londino): Sovie Message Net Red Solution 2 Rockac(Linker Message Net Red S	GrpDriver_Model	NeoGroup	2
Chall donk (Second Second Seco	ChkMonitor_InterpretedMessage	NeoCheckBax	2
Lakinda Lakinda Lakinda Karana	C boM onitor_ShowM essageH eader	NeoComboBox	2
Child Gold C, Show Time stemp Hero (Head Box 2) Child Gold C, Show Star C The Rec (Head Box 2) Child Gold C, Child C, Show Star C Child C, Sh	LnkM on itor_InsertC omment	NeoHyperlink	2
Lakiforbig (Castaf onter Meenhysefink 2) Charlondic, Shook Sharri Mee Coleada ox 2) Robif onter Meenhalt ox 2) Charlondic, Shook Sharri Meenhalt ox 2) Charlondic, Shook Sharri Meenhalt ox 2) Charlondic, Shook Sharri Meenhalt ox 2) RobAcu, Shook Sharri Meenhalt ox	ChkMonitor_ShowTimestamp	NeoCheckBox	2
Childonic Cheva Antoneo Antone	LnkMonitor_Clearl/Ionitor	NeoH yperfink	2
hted oder, J. Anster H. ec Gooment 2 Challonger, Working H. ec Cheallow 2 Challonger, Scotting H. ec Cheallow 2 Reak-Ac_Show Address H. ec Cheallow 2 Reak-Ac_Show Address H. ec Cheallow 2 Reak-Ac_Show Address H. ec Cheallow 2 Covered Neo Controls To Covered Neo Controls The Control Head Cover 1 Reak-Ac_Show Address H. Reak-Address H. Reak-Address Reak-Ac_Show Address H. Reak-Address Reak-Ac_Show Address Reak-Ac_Show Address Reak-Ac_Show Address Reak-Add	ChkMonitor_ShowStackTrace	NeoCheckBox	2
Childonic Colling Verdina Mechanica (Childonic Colling Verdina (Childonic Childonic Colling Verdina (Childonic Childonic Child	RbdM onitor_Monitor	NeoDocument	2
Childonic Scolliubum dalain Hechadox 2 Rechadostant Anna Scollando 2 Rechadostant Anna Scol	ChkMonitor_WordWrap	NeoCheckBox	2
ResAuc].Showlarge NeeRadobutin 2 NeeRadobutin 2 NeeRadobutin 2	ChkMonitor_ScrollAutomatically	NeoCheckBox	2
Redukcj_ShowGebedestitem NeoritadeButon 2 Covered Neo Controls	RdoAsci_ShowRange	NeoRadioButton	2
Number Type Standbort Neelhysefielk Standbort Neelhysefielk Standbort Neelhysefielk Standbort Neelhysefielk Standbort Neelhysefielk Standbort Neelhysefielk Tationedion Status_Demicticell Neelhysefielk Tationedion Status_Demicticell Neelhysefielk Status Status Status Neelhysefielk Status Status Status Neelhysefielk Status Neelhysefielk Status Neelhysefielk Status Neelhysefielk Status Neelhysefielk Status Neelhysefielk <td></td> <td>Nee Dedie Rolling</td> <td>2</td>		Nee Dedie Rolling	2
BitolgesLight Pf E Heelbidton TacConsection Status_ConnectionSt Heelbidton TacConsection Status_ConnectionSt Heelbidton TacConsection Status_ConnectionSt Heelbidton Status Anot Status_ConnectionSt Status Anot Status_ConnectionSt Status Anot Status Status NeoT Status<	Covered Neo Controls	Туре	2
tinc/Sam Th/Consection Statiun_Connection Th/Consection Statiun_Connection Th/Consection Statiun_Connection Th/Consection Statiun_Connection Th/Consection Statiun_Connection Th/Consection Statiun_Connection Th/Consection Th/Co	Covered Neo Control	Type	2
Tacionadon Salani, Connection3 Hadomedon Salani, Connection3 Marcianedon Salani, Connection3 Marcianedon Tatatem, 4 NetTablem Salatem, 4 NetTablem Salatem, 4 NetTablem Salatem, 4 NetTablem Salatem, 4 NetTablem Salatem, 4 NetTablem NetSalatem, 1 NetSalatem, 1	Rooked_showeedteditems Covered Neo Controls Neo Control Link About Roologia as Fis	Type NeoHyperink NeoByten	
Tacionados Salaria_ConnectionSi Neo G di Tastem, 1 Neo Tablem Tastem, 2 Neo Tablem Tastem, 5 Neo Tablem Neo Tablem	Noo-sol_snowselecteditems Covered Neo Controls Neo Control LnkAbout BtnOpenLogFie BtnClose	Type NeoHyperlink NeoButton	
Tanten, 4. Hen Tablem Tanten, 5. Hen Tablem Orgifsen/Usesage HenoProp Bindide, Stop HenoProp Bindide, Stop HenoProp Statiod, Cleart (P Addres HenoShi Maldow, Cleart (P Addres HenoShi Maldow, Cleart (P Addres HenoShi Maldow, Cleart (P Addres HenoShi Maldow, Cleart (P Addres HenoShi HenoProp Statistica, Lines Maldow, Cleart Maldow, Cleart Maldow, Labimulator, Clear, HenoPyselink Mal LineSetting, Labimulator, Clear, HenoPyselink Mal	Rec Control Internet Neo Control Internet Intern	Type NeoHyperlink NeoButton NeoDournent	
Taktem_5	Non-Aci_snovo-recteotems Reo Control Reo Control LnA-About BrinOpenLogF /e BrinCose TaConnedio Status_ConnectionSt TaConnectionStatus_ConnectionSt	Type Type NeoHypefink NeoButton NeoButton NeoDoument NeoDoument	
Orgebourd/essage MecoOrusp Districtor_Size MecoMuton Restrate_Class MecoTastacourus Station_Class MecoTastacourus Martine Lastacourus Martine Lastaria Martine Lastaria Martine Lastaria Martine Lastaria Martine Lastaria Martine Lastaria	Non-Aci_Snovace ecteditions Reo Control Int-Acout Bit-Opencion Status, Remotert out Tatic Connection Status, ConnectionSt atus Tatic Connection Status, ConnectionSt atus Tatic Connection Status, ConnectionSt atus Tatic ConnectionStatus, ConnectionSt atus Tatic ConnectionStatus, ConnectionSt atus Tatic ConnectionStatus, ConnectionSt atus ConnectionStatus, ConnectionSt atus ConnectionStatus, ConnectionSt atus ConnectionStatus, ConnectionSt atus ConnectionStatus, ConnectionSt atus ConnectionStatus, ConnectionSt atus ConnectionStatus Connect	Type Type NeoHyperink NeoButton NeoButton NeoButton NeoButton NeoButton NeoEdit NeoEdit NeoEdit NeoTablem	4
Bitmid of, Brogo Heel Button Heel Button Tatil do C, Line II, PA detes Heel Button Tatil do C, Line II, PA detes Heel Bonner Orgettings, Server NeoDrop Gradetings, LabS mulator: Color_Chat NeoDrop LabSenigation: Color_Chat NeoDropenix	Covered Neo Controls Neo Control International Networks Enclose Enclose TraConceton Status, ConnectionSt TraConceton Status, ConnectionSt Tatatem, 4 Tatatem, 5	Neckspectron Necks	-
Radi dag, Client NeoSadobutton Tadoda, Client J. Padra NeoSado Num Node, Client, Pad NeoSadomer Orpädattag, Sarver NeoSadopu Ladistratus, Universitationa Color, Chan NeoSystemik Ladistratus, Ladistratutor: Color, Det NeoSystemik anti.	Non-Kot-scover exteriors Neo Control Neo Control Un About Un About Bitro per Log File Bitro Control Tuto Connection Status, Remoteriord Tuto Connection Status, ConnectionSt atai ataitem4 Taitem5 Orginer diressop	Type Type NeoHypefink NeoButton NeoDuton NeoDuton NeoCourment NeoEdf NeoTablee NeoTabl	4
Tatidad_Ciner_UP-Adorea NeoEdft Tatidad_Ciner_UP-Adorea OrgEstmag_Server NeoDrop OrgEstmag_Server NeoDrop Generations_CineAn NeoDropedina Generations_CineAn NeoDropedina LasSemigateColor_Cine NeoDropedina LasSemigateColor_Cine NeoDropedina	Non-Karjanovce extentions No Control Na Control Biogranicy Fe Biogranicy Fe Biogranicy Fe Tracioned Status, Connection() Antennet Tratience, S Orgenet Biographics Statience, S Statience, S Stat	Type Type Nechopatink Nechotan Nechotan Nechotan Nechotan Nechotan NecTablem	4
Num Mode, Cient, Port Num Schonner Größerlang, Linko mation-Calver, Meed/Sporting, Linko Settinga, Linko mation-Calver, Chan Need/sportink Linko Settinga, Lakösmulator-Color, Der Need/sportink Auth	Konsko-skowerskowers Keo Controls Keo Control Konochers	Type NeoHype fink NeoButton NeoDourment NeoCourment NeoColument NeoColume NeoColume NeoColume NeoColume NeoRofue NeoRofu	4
Orgisteting Server NeoDrop Unstending Line Marcio Cr. Lan Kerkingketin ge. Lindfening LakSmulator Calor, Def Veolfgeefink Lindfening LakSmulator Calor_Ch. Neofigeefink	Non-Ka ² -Shower extendem s No Control Na Control Michael Mich	Type Topic State Type Neckyton	
Lui Sentogu, Into mationColor_Chan Heotype filok ge Lui Sentogu, LabSim ulatorColor_Def Heotype filok aut Lui Sentogu, LabSim ulatorColor_Ch. Heotype filok ang	Konska, skower external Keo Controls Keo Control Konono Kononoo Konooo Kononoo Kononoo Kononoo Kononooo Konoooo	Type NeoHype fink NeoButton NeoButton NeoBotton NeoBotton NeoBotton NeoBotton NeoFablem N	
LnkSettings_LabSimulatorColor_Def NeoHyperlink auft nnSettings_LabSimulatorColor_Ch NeoHyperlink ange	Non-Kof-shower entertainers No Control Na Control Michael Mich	Type Nextype fink	
LnkSettings_LabSimulatorColor_Ch NeoHyperlink ange	Neo Control Second Seco	Type Newlype fink Newburton Newburton Newburton Newburton New Tablem New Tablem	
	Kookad-Johoverendenen Kookad-Johoverendenen Kookado	Type Type Neotypefink Neobuton Neobuton Neobuton Neoboton Neofoton	

Fig 10. An example of unused UI controls in UI usage report

G. Test Manager

Automated test cases we have created in previous steps are in C# language. It can be run by using test command of .NET framework, i.e. mstest.exe, or can be easily run within Microsoft Visual Studio IDE which already provides functionalities of running test and test result report.

IV. APPLICATIONS

We have tested NAF by creating a few automated test cases for example applications. The process of creating UI structure is done within few minutes and testers can finish writing automated test cases according to test design shortly. However, after testers have finished creating the automated test cases, we found some UI controls that are not used in the test cases. Obviously that UI Usage Analyzer tool could help identify what UI controls should be added into automated test cases to increase test coverage as much as possible.

V. CONCLUSION

We have presented an automation testing framework – Neo Automation Framework (NAF). NAF consists of three main parts: 1. generating UI structure of specified form in an application under test, 2. generating and modifying automated test cases which is kind of unit test in C# language, and 3. analyzing usage of UI control in automated test cases. Our approach is to keep information of UI controls in hierarchical data structure which can be used in other parts of framework easily. Testers can access UI controls directly via a class of UI structure which is stored in a class. Therefore, there is no extra work for preprocessing of use and it can also be used to verify the automated test cases to find unused UI controls.

Since we have a model of application under test in hierarchical data structure so for our future work, it is possible to validate the UI structure against the software. It could be a quick test to ensure that there is no change in UI. If the result of checking UI structure with the software is failed, then there might be new UI controls added into the software or existing UI controls do not exists anymore. In latter case, if it is not an intended change, then it is likely a bug in the software detected by UI structure.

REFERENCES

- James A. Whittaker, "What Is Software Testing? And Why Is It So Hard?", IEEE, 2000
- [2] Wei Jin et al., "Automated Behavioral Regression Testing", in 3rd Int. Conf. on Software Testing, 2010, IEEE, pp. 137-146
 [3] ZHU Xiaochun et al., "A Test Automation Solution on GUI
- [3] ZHU Xiaochun et al., "A Test Automation Solution on GUI Functional Test", pp. 1413-1418
- [4] Pekka Aho; Nadja Menz; Tomi Räty; Ina Schieferdecker, "Automated Java GUI Modeling for Model-Based Testing Purposes", IEEE, 2011
- [5] Microsoft, "Visual Studio Test Professional", Available from: http://www.microsoft.com/visualstudio/eng#products/visual-studiotest-professional-2012+product-edition-testpro
- [6] Automated GUI Test Coverage Analysis using GA
- [7] Reverse Engineering of GUI Models for Testing
- [8] GUI Testing Made Easy
- Microsoft, "Control Types and Their Supported Control Patterns", Available from: http://msdn.microsoft.com/enus/library/windows/desktop/ee671193(v=vs.85).aspx
- [10] Microsoft, "Microsoft UI Automation", Available from: http://msdn.microsoft.com/en-us/library/ms747327.aspx