Journal of Computer Science 7 (9): 1343-1352, 2011
ISSN 1549-3636

Algorithms for the Detection of Defects in GUI Applications

[1]B. Uma Maheswari and [2]S. Valli
[1]Department of MCA, St. Joseph's College of Engineering,
Old Mamallapuram Road, Chennai, 600119, Tamilnadu, India
[2]Department of CSE, College of Engineering, Guindy, Anna University Chennai,
Sardar Patel Road, Guindy, Chennai, 600025, Tamilnadu, India

Abstract: Problem statement: Developing software using graphical interfaces has grown rapidly in recent years. Products developed by, using GUI have to be tested for the reliability and quality of the software. The objective of software testing is to reveal all the errors in the desired software with minimum cost. GUI application comprises of several controls for user interaction and each control has several properties assigned to it. Even though default attributes and actions are available for the controls, some performance problems exist, which are undetected. **Approach:** The VBCT tool is developed to extract the incorrect statements in the code and various algorithms are written for detecting the logical errors in the GUI Applications. **Results:** This study addresses the errors in the visual applications and alerts the developer to modify the source code. This study performs structural and behavioral testing for GUI based applications. **Conclusion:** Identifying and removing the undetected logical errors in GUI based applications has been developed. Both white box and black box testing is attempted. The written test scripts, work in a robust manner to detect the faults.

Key words: Software testing, visual applications, tokenizer, logical errors, software faults, GUI applications, Visual Basic Control Testing (VBCT)

INTRODUCTION

Visual Basic (VB) is an event driven programming language and it is an Integrated Development Environment provided by Microsoft. The VB development environment has a default form, tool box, project explorer window and property window. They are used to develop GUI applications and to build many complex applications in a rapid manner. Now-a-days, the software interacts with the user with the help of GUI. So the GUI software is tested to ensure its correctness, reliability and quality.

Software faults and failure data were analyzed by (Hamill and Goseva-Popstojanova, 2009). They have found that requirement faults, coding faults and data problems are the common types of software faults. There are various testing tools (HP.HpWinRunner) such as capture/replay tools. If the size of the test scripts grows, it becomes unreadable. In GUI Testing Automation (GUITA) technical testers are required for creating scripts. The initial work of building frameworks and creating libraries is time consuming. Building functional logic in the scripts for complex applications is tedious.

Errors can be classified as syntax errors, runtime errors and logical errors. Syntax errors are introduced by the programmers and the compiler will produce the alerts if the written code is syntactically incorrect in the editor window. They can be fixed in the coding environment itself. Run time errors are those which occur on the execution of the code. They can be fixed by modifying the original code in the application. Logical errors are those which produce unexpected results.

Some logical errors are not shown by the Visual Basic compiler. The objective of this study is to improve the quality of the source code, which will reveal the defects which are not detected by the compiler. Even though default attributes and actions are available for the controls, some performance problems are not detected by the compiler. The aim of this study is to identify these problems and to rectify them. The proposed Visual Basic Control Testing (VBCT) detects these kinds of errors in the VB program and produces alert to the user, on how to perform property assignment to the Visual Basic controls.

The rest of the study is organized as follows. The related works of software testing in various domains, Visual Application Testing which describes the implemented testing frame work, the working

Corresponding Author: B. Uma Maheswari, Department of MCA, St.Joseph's College of Engineering,
Old Mamallapuram Road, Chennai, 600119, Tamilnadu, India

methodology of the VBCT tool and the various algorithms used in detecting the logical errors in the GUI Applications, the experimental results and the conclusion with the directions for future work.

MATERIALS AND METHODS

Related work: There are various literatures in software testing to discover the bugs in the software. The software may have faults, some of which lead to failures. Understanding the relationship between faults and failures improves the quality of the software. The classification of faults and failure data is done by (Hamill and Goseva-Popstojanova, 2009). A failure is the departure of the system's behavior from its required behavior. A fault is an accidental condition, which leads to failure. Faults are problems noticed by the developer and failures are problems noticed by the user.

GUI testing has been attempted by (McMaster and Memon, 2008). They have analyzed the sequence of the active calls of an executing program, which was applied to the Space application (Antenna Steering System) developed by the Euro Agency and TerpOffice Suite. (Pai and Dugan, 2007) used a Bayesian network model to relate object oriented software metrics to software fault content and proneness. They have addressed the internal product metrics and external quality metrics.

There are many approaches and capture and play back tools for preparing test scripts, which do not show the logical error of the program and if the lines of the code of an application are more, the test scripts are unreadable and unstructured. Data flow testing has been attempted by (Harrold *et al*., 1997). They have used the program dependence graph to detect faults either at the structural level (modifying the structure of the program) or at the statement level (modifying the statement of the program).

The GUI testing framework GUITAR (Xie and Memon, 2006) is a tool which automates the creation and execution of test cases. The outputs are compared using the oracle procedure. The relationship between the test-suite and fault-related factors in GUI testing is analyzed by (Strecker and Memon, 2008). They found that the statement coverage and GUI-event coverage are statically related to detect certain kinds of faults.

Sampath *et al*. (2007) used concept analysis for testing web applications. User Session is the duration between the arrival of a new IP address and the user exit from the website. Client URL requests and name-value pairs are collected for the concept analysis. Extremely used 40 websites were tested for 15 minutes. 75% of these sites exhibited application failures, blank pages, incomplete and incorrect pages. Five types of faults namely, data store, logic faults, form faults, appearance faults and link faults were seeded in the original applications. Oracle comparators were used to compare the original and the fault seeded applications.

Chen *et al*. (2008) developed an object based approach for testing the GUI applications. Component abstractions modeled the structure of the GUI. Test scripts were written using the GUI Testing Modeling Language (GTML). Object Oriented testing was performed by (Sarala and Valli, 2006). The authors have developed flex rules to detect the missing new operator in the context of dynamic memory allocation, unhandled exceptions and missing operator in the context of run time polymorphism and also created rules for detecting the defects in console based applications.

Jones and Harrold (2003) developed test suite reduction techniques. A test suite contains modified conditions/decisions. Each decision statement is evaluated for true and false conditions. The test suite is reduced by eliminating uncovered modified condition / decision coverage pairs. The weakest test cases are discarded. Test suites are prioritized by selecting the test case that has the highest entity coverage.

Algorithms for regression test suite reduction were analyzed by (Li *et al*., 2007) for Greedy, Additional Greedy, 2-Optimal, Hill Climbing and Genetic Algorithms. All the algorithms were used for code coverage and inefficient for fault detections. Automatic failure identification was done by (Travison and Staneff, 2008) by using test instrumentation and pattern matching.

Visual Application Testing: There are various GUI controls available in Visual Basic. All the controls have their own properties and actions. Actions are performed by the event handlers. Even though default attributes and actions are available for the controls, some performance problems exists, which are not detected by the Visual Basic compiler. The VBCT tool reveals such problems to the user. The framework of the tool, the working methodology and the various algorithms developed for revealing the logical errors of the Visual Basic controls, specifically, textbox control are as follows.

Definition: A Visual Basic form has various controls, with a fixed set of properties. A Form can be modeled in terms of controls $C = (c_1, c_2 \ldots c_n)$ and their Properties $P = (p_1, p_2, \ldots\ldots, p_n)$. P's are assigned to C's for designing the forms during the product development.

Controls	=	{ Picture Box, Text box, Command Button, Option Button, List box, Vertical , Scroll bar, Drive list box, File List box, Data, Line, Pointer , Label, Frame, Checkbox, Combo box, Horizontal scroll, Timer, Directory list box, Shape, Image, OLE } (a)
Properties of the textbox	=	{Appearance, Alignment, Back color, Border style, Drag mode, Enabled,Font, Fore color, Height, Index, Left, Locked, Max length, Mouse Icon, Mouse pointer, Multi line, Password char,Right to left, Tab index, Tab stop,Text, Top,visible, What's thishelp ID, Width } (b)

Fig. 1: (a) Visual basic controls and (b) properties of the controls

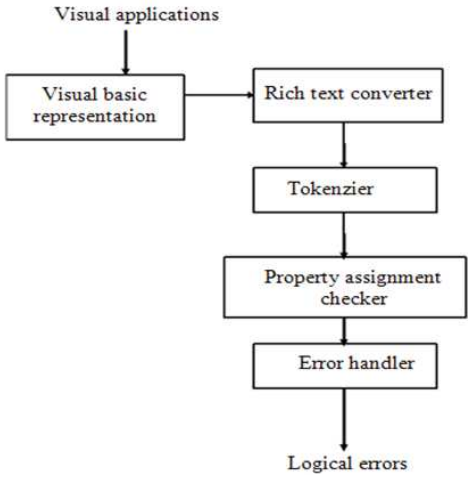

Fig. 2: An overview of the VB Control Testing Framework

Table 1: Assignment of Text property to the textbox control using the code editor

HELLO.Text	=	" VisualBasic''	//Valid assignment
WORLD.Text	=	34223	//Valid assignment
JAVA.Text	=	VisualBasic	//Invalid assignment

Table 2: Assignment of text property to the textbox control using the property window

HELLO.Text	=	Text1
WORLD.Text	=	Text2
JAVA.Text	=	Visual basic

As in Fig. 1 the necessary controls are arranged in the form during design. The properties are assigned at a designed time using the application source code. The discussion of VBCT tool and the working principle are as follows.

Visual Basic Control Testing (VBCT) Framework: The VBCT tool given in Fig. 2 has several components, such as the rich text converter, tokenzier, property assignment checker and error handler.

Visual Basic representation is a collection of objects such as forms, controls, properties and the values of those properties. The properties of the product are converted into richtext by the richtext converter. The tokenzier analyzes the coding and splits the entire file into tokens using space as the delimiter. Some of these tokens are given in Fig. 6.1 and 6.2. The property assignment checker scans the tokens from the tokenzier and extracts the assigned values. These assigned values are used by the error handler for detecting the logical errors in coding and produce alerts to the user to indicate the existence of logical errors.

Algorithms for validating textbox control: The working of the algorithm to detect the violations in the code for property assignments, such as text, alignment, backcolor, enabled, fontbold, fontitalic, fontstrikethru, fontunderline, locked, height, left, top, maxlength, passwordchar, visible, width and tabstop, are discussed.

Text Property: Input box, text box, list box, combo box and richtext box controls are used as input controls for processing. List and combo boxes are restricted input controls, using which the input can be entered either by using the code editor or through the property window. Text box and richtext box are unrestricted input controls. Any content can be given for processing. The text property is validated by examining the assigned value using the property window assignment, or with the help of the source code.

The text assigned to the textbox JAVA is invalid in Table 1, because the string should be assigned within double quotations. But, if the text is assigned without double quotes using the Visual Basic code editor, no error alert is shown .On execution, the string Visual basic is not assigned to the Textbox named JAVA, even though the property window assignment is VisualBasic as in Table 2.

Algorithm for detecting logical errors of Text property of the textbox control: This module identifies the Text property of all the Text Boxes in the form. It checks if the value assigned to it begins with a double quote in the case of a string, or it is a digit in the case of a numeral:

Read the file to be tested

/* tokens [ ] is an array of strings which holds the list of tokens in the file.*/
tokens [ ] ← split the entire file into separate words, using space as a delimiter
n ← count (tokens[ ])
for i = 1 to n do
begin If ((tokens [i] = " *.Text ") and
(tokens [i +1] = " = ")) then
testtext ← tokens [i + 2]
If ((left(testtext = " ")) or (left(testtext = "0 - 9")))
then
write ("Valid Text property is assigned ")
else
write ("Invalid Text property is assigned.
It should start with double quotes or with any numerals")
end

Alignment property: The alignment property of the text box assumes 0 for left justification, 1 for center and 2 for right justification. The assignment of the alignment property using the code, suppresses the value assigned using the property window.

As given in Table 3 on execution, there is no compilation errors and the alignment for entering into the Textboxes PQR and XYZ is left, even if 2 (right) is assigned in the property window. This is a logical error.

Algorithm for detecting logical errors of Alignment property of textbox control

/* This module identifies Alignment property of all Text Boxes in the form. It checks whether the value assigned to it is 0 / 1 / 2 */
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file */
tokens [ ] ← split the entire file into separate words, using space as a delimiter
n ←count (tokens[ ])
for i = 1 to n do
begin If ((tokens[i] = " *.Alignment") and
(tokens[i+1]=" = ")) then
testalignment ← tokens [i+2]
If ((left (testalignment = 0)) or (left(testalignment = 1))
or (left(testalignment = 2))) then
write("Valid Alignment property is assigned")
else
write("Invalid Alignment is assigned. It should
be 0 or 1 or 2")
end

Backcolor Property: The background color assignment should start with &H8000000 and should end with any numerals. &H80000001, &H80000002, &H80000003 and &H80000004 specify blue color variations, &H80000005, &H80000006 and &H80000009 represent white color, &H80000007 and &H80000008 represent black color and &H80000000 represent ash color.

In Table 4, the assignments to the textboxes PQR, XYZ are arithmetic expressions. Hence background color is black and it is a logical error.

Algorithm for detecting logical errors of the BackColor property of the textbox control

/* This module identifies the BackColor property of all the Text Boxes in the form. It checks if the value assigned starts with &H8000000 and whether the last character is between 0 - 9 */
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file */
tokens [ ] ← split the entire file into separate words, using space as a delimiter
n ←count (tokens[ ])
for i = 1 to n do
begin If ((tokens[i] = " *.BackColor") and
(tokens [i+1]= "=)) then
testbackcolor←tokens [i +2]
If ((left (testbackcolor = "&H8000000") and
right (testbackcolor = "0-9")) then
write("Valid Textbox.BackColor property is assigned")
else
write ("Invalid Textbox.BackColor is
assigned. It should start with &H800000 and end
with any numerals")
end

Table 3: Assignment of the alignment property to the textbox control using the code editor

Hello.Alignment	=	0	//Valid assignment
World.Alignment	=	1	//Valid assignment
Java.Alignment	=	2	//Valid assignment
PQR.Alignment	=	HAI	//Invalid assignment
XYZ.Alignment	=	A	//Invalid assignment

Table 4: Assignment of the backcolor property to the textbox control using the code editor

Hello.Back color	=	&H80000003	//Valid assignment
PQR.Back color	=	(A * B)	// Invalid assignment
XYZ.Back color	=	A + B	// Invalid assignment

Table 5: Assignment of the Enabled property to the textbox control using the code editor

HELLO.Enabled	=	1	// Valid assignment
WORLD.Enabled	=	0	// Valid assignment
JAVA.Enabled	=	True	// Valid assignment
PQR.Enabled	=	(a+b)	// Invalid assignment
XYZ.Enabled	=	hei	// Invalid assignment

Table 6: Assignment of the font and locked properties to the Textbox control using the code editor

XYZ.FontBold	=	hello	// Invalid assignment
XYZ.FontItalic	=	hello	// Invalid assignment
XYZ.FontStrikethru	=	heiil	// Invalid assignment
PQR.FontUnderline	=	a + b - c * d	// Invalid assignment
XYZ.FontUnderline	=	construction	// Invalid assignment
JAVA.Locked	=	k	// Invalid assignment
PQR.Locked	=	a	// Invalid assignment
XYZ.Locked	=	construction	// Invalid assignment

Table 7: Assignment of the height, width, left and top properties of the textbox control using the code editor

HELLO.Height	=	1000	// Valid assignment
JAVA.Height	=	False	// Valid assignment
PQR.Height	=	a + b - c * d	// Invalid assignment
WORLD.Width	=	GFDGDFG	// Invalid assignment
WORLD.Left	=	"4000"	// Valid assignment
JAVA.Left	=	k	// Invalid assignment
XYZ.Left	=	construction	// Invalid assignment
WORLD.Top	=	A	// Invalid assignment
PQR.Top	=	FKSDFS	// Invalid assignment

Enabled, font and locked properties: The string can be entered in the textbox, when the enabled property is true or 1 and if the enabled property is false or 0, the value cannot be entered in the textbox.

Even if true is assigned in the property window, it is impossible to enter the data in the textboxes PQR and XYZ, since programmatically enabled property is set to an arithmetic expression and string respectively. On compilation it is free of defects. But on execution, the data cannot be entered in to the textboxes PQR, XYZ. This is a logical error.

Algorithm for detecting logical errors of the Enabled property of the textbox control

/* This module identifies the Enabled property of all the Text Boxes in the form. It checks whether the value assigned to it is 0 / 1 / true / false */.
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file. */
tokens [ ] ← split the entire file into separate words, using space as a delimiter
n ← count (tokens[ ])
for i = 1 to n do
begin If ((tokens[i] = " *.Enabled") and
(tokens [i +1] = " = ")) then
testenabled ← tokens [i+ 2]
If ((left (testenabled = 0)) or (left (testenabled = 1))
or (left(testenabled ="True")) or
(left(testenabled = "False"))) then
write ("Valid Enabled property")
else
write ("Invalid Enabled property is assigned. It
should be 0 or 1 or True or False")
end

Table 6 lists the invalid assignment of FontBold, FontItalic, FontStrikethru, FontUnderline and Locked properties in the Visual Basic code editor.If the property is set as given in Table 5, the text in the corresponding textbox is not bold, nor italic, nor stroked and not underlined even if the correct property is assigned in the property window. The data can be entered in the textboxes JAVA, PQR, XYZ of Table 6, even though the Locked assignment is True in the property window and programmatic assignment is some character or string.The algorithm for the Enabled property can be used to detect logical errors by the replacement of *.FontBold, *.FontItalic, *.FontStrikethru ,*.FontUnderline and *.Locked in the place of *.Enabled property.

Height, Width, Left and Top Properties: The Design of the textbox height,width,space from the left of the form and from the top of the form are obtained by the height,width,left and top properties.The valid and invalid assignments of these properties are given in Table 7.

Even if the correct values are assigned in the property window, incorrect height, incorrect negligible width, no space from the left and no space from the top of the window results, when the property is assigned as given in Table 7 in the code editor. This is a logical error.

Algorithm for detecting logical errors in the height property of the textbox control

/* This module identifies the Height property of all the Text Boxes in the form. It checks whether the value assigned to it is numerals or the numerals are within double quotes */.
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file. */
tokens [ ] ← split the entire file into separate words, using space as a delimiter
n ← count (tokens[ ])
for i = 1 to n do
begin If ((tokens[i] = " *. Height") and
(tokens [i +1] = "= ")) then
testheight←tokens [i+2]
If ((left(testheight = " ")) or (left(testheight = "0-9")))
then
write ("Valid Height assignment")
else
write ("Invalid Height is assigned. It should start with any number or numerals within double quotes!")
end

The same algorithm can be used to detect errors for width, left and top properties with the replacement of *.Width, *.Left, *.Top in place of *.Height.

Table 8: Assignment of the maxlength property of the textbox control using the code editor

Hello.Max length	=	60	// Valid assignment
World.Max length	=	3	// Valid assignment
Java.Max length	=	0	// Invalid assignment
PQR.Max length	=	"3443"	// Invalid assignment
XYZ.Max length	=	heiil	// Invalid assignment

Table 9: Assignment of the passwordchar property to the textbox control using the code editor

Hello.Password char	=	"2"	// Valid assignment
World.Password char	=	4	// Valid assignment
Java.Password char	=	"g"	// Valid assignment
PQR.Password char	=	A + B + C * B	// Invalid assignment
XYZ.Password char	=	h	// Invalid assignment

Table 10: Assignment of visible and tab stop properties to the textbox control using the code editor

Java.Visible	=	jksadas	//Invalid assignment
PQR.Visible	=	fhkgm	//Invalid assignment
Java.Tab stop	=	aaa	// Invalid assignment

Maxlength property: The maximum number of characters the textbox can accommodate is 65535. Table 8 shows the invalid assignment to the maxlength property in spite of valid length in the property window. The Maxlength assignments in Table 8 do not show the compilation errors. The number of characters to be entered in case of invalid assignments depends upon the size of the text and the width of the textbox.

Algorithm for detecting logical errors in the MaxLength property of the textbox control

/* This module identifies the MaxLength property of all the Text Boxes in the form. It checks whether the value assigned to it is any numerals 1-9 and it should be < 65535 */.
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file. */
tokens [ ] ← split the entire file into separate words, using space as a delimiter,
n ← count (tokens[ ])
for i = 1 to n do
begin If ((tokens[i] = " *. MaxLength ") and
(tokens [i +1]="= ")) then
testmaxlength ← tokens[i+2]
If ((left (testmaxlength = 1-9) or
(left(testmaxlength < 65535)) then
write
("Valid MaxLength property is assigned")
else
write("Invalid MaxLength is assigned. It
should start with 1-9 and the value must be < 65535")
end

PasswordChar Property: The decryption character is a single alphabet within double quotes, or a numeral. In Table 9, the PQR textbox is assigned an arithmetic expression and the XYZ textbox is assigned a single character without double quotes.

As in Table 9, if invalid expressions are assigned to password character, the password character is taken as zero. The characters without double quotes are not considered and the typed text is not decrypted. This is a logical error.

Algorithm for detecting logical errors in the PasswordChar Property of the Textbox Control

/* This module identifies the PasswordChar property of all the Text Boxes in the form. It checks whether the value assigned to it is any numerals or begins with double quotes in case of alphabets. */.
Read the file to be tested
/* tokens [ ] is an array of strings which holds the list of tokens in the file. */
tokens [ ] ← split the entire file into separate words,
using space as a delimiter
n←count (tokens[ ])
for i = 1 to n do
begin
If((tokens[i]="*.PasswordChar")and
(tokens[i+1]="=")) then
testpasswordchar ←tokens[i+2]
If ((left (testpasswordchar = 0 to 9) or
left(testpasswordchar = " ")) then
write ("Valid PasswordChar is assigned")
else
write ("Invalid PasswordChar is assigned. It should
be 0-9 or any character input which starts
with double quotes")
end

Visible and TabStop Properties: The textbox control should be visible and receives the focus only if the visible property is true in the property window. If the source code is used for assignment it works with respect to the source code. The assignment in the property window is ignored.

The incorrect assignments and the resulting logical errors in Table 10 are handled by the algorithm which follows. Even if the visible, tabstop properties are set to true in the property window, due to the assignment in the program using Table 10, logical errors such as "text box is not seen", "tab control is not focused" arise in the form.

Algorithm for detecting logical errors in the Visible property of the textbox control:

/* This module identifies the Visible property of all the Text Boxes in the form. It checks whether the value assigned to it is true / false / 0-9 / any numerals within double quotes */.

Read the file to be tested

/* tokens [ ] is an array of strings which holds the list of tokens in the file. */

tokens [ ] ← split the entire file into separate words, using space as a delimiter

n ← count (tokens[ ])

for i = 1 to n do begin

If ((tokens[i] = " *. Visible ") and (tokens[i+1] ="= ")) then

testvisible ←tokens [i+2]

If ((left (testvisible = 0 to 9)) or (left(testvisible = " ") or (left(testvisible = "True")) or left (testvisible = "False"))) then

write ("Valid Visible property is assigned")

else

write ("Invalid Visible property assignment .It should be True or False or 0 to 9 or any numeral which starts with double quotes!")

end

The algorithm detects errors for the tabstop property with the modification of *.TabStop in the place of *.Visible. Thus, all the algorithms efficiently analyze the source code, scan the appropriate properties of the textbox control and alert the developer by giving suggestions on how to rectify the code.

RESULTS

In the above Fig. 4.1-4.3 the HELLO textbox is assigned "Text1", the WORLD text box is assigned "Text2" and the JAVA textbox is assigned "Visual Basic". After the compilation of the above code, the VB compiler does not show any syntax error. It displays the form as in Fig. 5. The HELLO and WORLD textboxes are assigned correctly as "VisualBasic" and "34223" according to the coding in Table 11. The JAVA text box is empty and the developer can enter any string since in Table 11, the assignment to the JAVA text box is set as Visual Basic without double quotations. The VBCT identifies these kinds of incorrect assignments in the source code as in the Fig. 6.1-6.2

Table 11: Assignment of the Text property to the textbox control using the code editor

Hello.Text	=	"Visual basic"	//Valid assignment
World.Text	=	34223	// Valid assignment
JAVA.Text	=	Visual basic	//Invalid assignment

Java text box property	
Java text box	
Alphabetic	Categorized
(Name)	Java
Alignment	0-Left justify
Appearance	1-3D
Tag	
Text	Visual basic
Tool tip text	
Top	5640
Visible	True
Whats this help ID	0
Width	5895

Fig. 4.1: Assignment of the text property for Java textbox in the property window

Hello textbox property	
Hello text box	
Alphabetic	Categorized
(Name)	Hello
Alignment	0-Left justify
Appearance	1-3D
Tag	
Text	Text1
Tool tip text	
Top	5640
Visible	True
Whats this help ID	0
Width	5895

Fig. 4.2: Assignment of the text property for Hello textbox in the property window

World Text box property	
World text box	
Alphabetic	Categorized
(Name)	World
Alignment	0-Left justify
Appearance	1-3D
Tag	
Text	Text2
Tool tip text	
Top	5640
Visible	True
Whats thishelp ID	0
Width	5895

Fig. 4.3: Assignment of the text property for World textbox in the property window

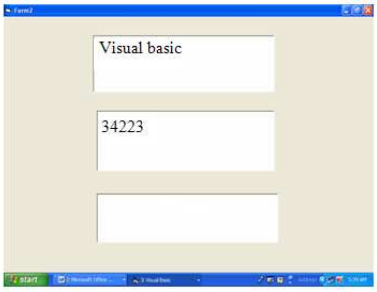

Fig. 5: Textboxes as a result of Fig. 4.1-4.3 and Table 11

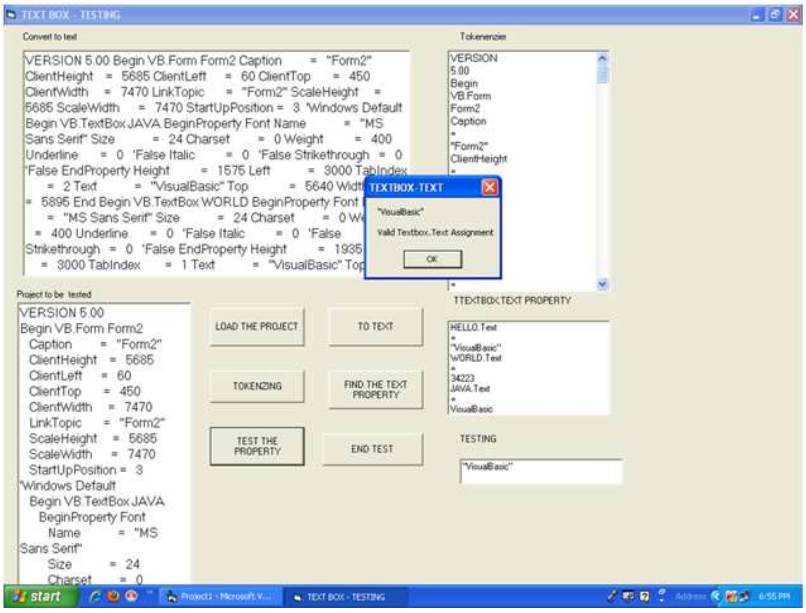

Fig. 6.1: The Visual Basic Control Testing (VBCT) tool result

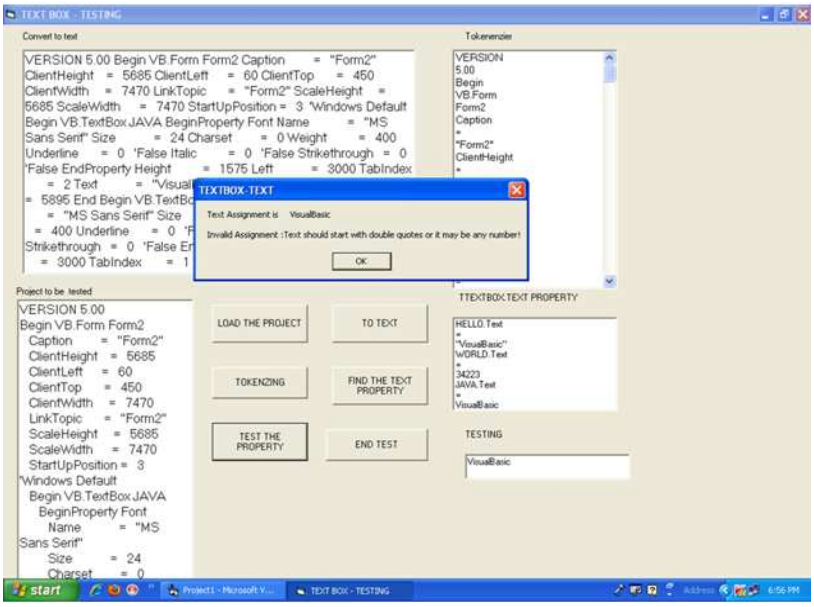

Fig. 6.2: The Visual Basic Control Testing (VBCT) tool result

DISCUSSION

The following Fig. 7.1-7.3 explain the logical errors detected by the VBCT and the VB compiler. The VBCT finds the logical errors in the properties such as text, alignment, backcolor, enabled, font, locked, height, left, top, maxlength, passwordchar, visible, width and tabstop which are not detected by the compiler. Logical errors partially detected by the compiler and the VBCT tool, for the properties such as appearance, borderstyle, dragmode, forecolor, index, mouseicon, mousepointer and mulitline are also mentioned in the following graphs.

Thus the VBCT tool finds the properties of the text box with the help of the tokenzier, detects the invalid assignments in the source code and alerts the developer.

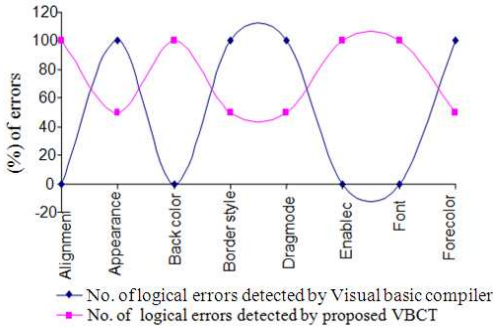

Fig. 7.1: VBCT Logical error detection

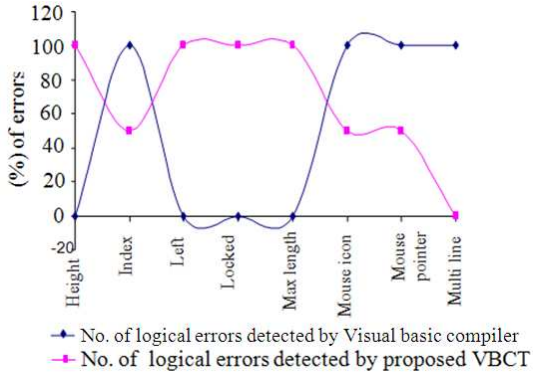

Fig. 7.2: VBCT Logical error detection

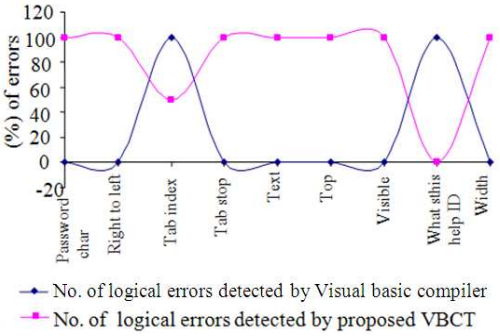

Fig. 7.3: VBCT Logical error detection

CONCLUSION

Since every user wishes to work with fault free software, revealing the logical errors is vital. Some of the performance problems are not identified by the compilers of the appropriate software languages. This study addresses the design and coding errors which are not identified by the Visual Basic compiler. The VBCT tool uncovers the problems and alerts the programmer, on how to modify the source code and control the properties of the textbox. There are various controls available in the visual environment. These kinds of issues may be present in those controls also. Analyzing and solving those issues in the rest of the Visual Basic controls and finding the logical errors which are not found by the compiler are taken up for future research study. This approach can be extended to other visual environments such as VC++, VB.net and Visual Studio, which may also have such problems. Identifying and removing the undetected logical errors in GUI based applications has been attempted.

REFERENCES

Chen, W.K., Z.W. Shen and C.M. Chang, 2008. GUI test script organization with component abstraction. Proceedings of the 2nd International Conference on Secure System Integration and Reliability Improvement, Jul. 14-17, IEEE Xplore Press, Yokohama, pp: 128-134. DOI: 10.1109/SSIRI.2008.16

Hamill, M. and K. Goseva-Popstojanova, 2009. Common trends in software fault and failure data. IEEE Trans. Software Eng., 35: 484-496. DOI: 10.1109/TSE.2009.3

Harrold, M.J., A.J. Offutt and K. Tewary, 1997. An approach to fault modeling and fault seeding using the program dependence graph. J. Syst. Software., 36: 273-295. DOI: 10.1016/S0164-1212(96)00175-6

Jones, J.A. and M.J. Harrold, 2003. Test-suite reduction and prioritization for modified condition/decision coverage. IEEE Trans. Software Eng., 29: 195-209. DOI: 10.1109/TSE.2003.1183927

Li, Z., M. Harman and R.M. Hierons, 2007. Search algorithms for regression test case prioritization. IEEE Trans. Software Eng., 33: 225-237. DOI: 10.1109/TSE.2007.38

McMaster, S. and A.M. Memon, 2008. Call-stack coverage for GUI test suite reduction. IEEE Trans. Software Eng., 34: 99-105. DOI: 10.1109/TSE.2007.70756

Pai, G.J. and J.B. Dugan, 2007. Empirical analysis of software fault content and fault proneness using Bayesian methods. IEEE Trans. Software Eng., 33: 675-686. DOI: 10.1109/TSE.2007.70722

Sampath, S., S. Spernkle, E. Gibson, L. Pollock and A.S. Greenwald, 2007. Applying concept analysis to user-session-based testing of web applications. IEEE Trans. Software Eng., 33: 643-657. DOI: 10.1109/TSE.2007.70723

Sarala, S. and S. Valli, 2006. Algorithms for defect detection in object oriented programs. Inform. Technol. J., 5: 876-883. http://www.doaj.org/doaj?func=abstract&id=600667&openurl=1&uiLanguage=en

Strecker, J. and A.M. Memon, 2008. Relationships between test suites, faults and fault detection in GUI testing. Proceedings of 1st International Conference on Software Testing Verification and Validation. Apr. 9-11, IEEE Xplore Press, Lillehammer, pp: 12-21. DOI: 10.1109/ICST.2008.26

Travison, D. and G. Staneff, 2008. Test instrumentation and pattern matching for automatic failure identification. Proceedings of 1st International Conference on Software Testing Verification and Validation,, Apr. 9-11, IEEE Xplore Press, Lillehammer, pp: 377-386. DOI: 10.1109/ICST.2008.69

Xie, Q. and A.M. Memon, 2006. Studying the characteristics of a "Good" GUI test suite. Proceedings of 17th Symposium on Software Reliability Engineering, Nov. 7-10, IEEE Xplore Press, Raleigh, NC, pp: 159-168. DOI: 10.1109/ISSRE.2006.45