International Journal of Computer Applications (0975 – 8887)
Volume 107 – No 12, December 2014

Software Refactoring Technique for Code Clone Detection of Static and Dynamic Website

K. Kanagalakshmi. Ph.D.
Associate Professor
Department of Computer Application
Vidyasagar College of Arts and Science
Udumalpet – 642 126.

R.Suguna
Research Scholar
Department of Computer Application
Vidyasagar College of Arts and Science
Udumalpet – 642 126

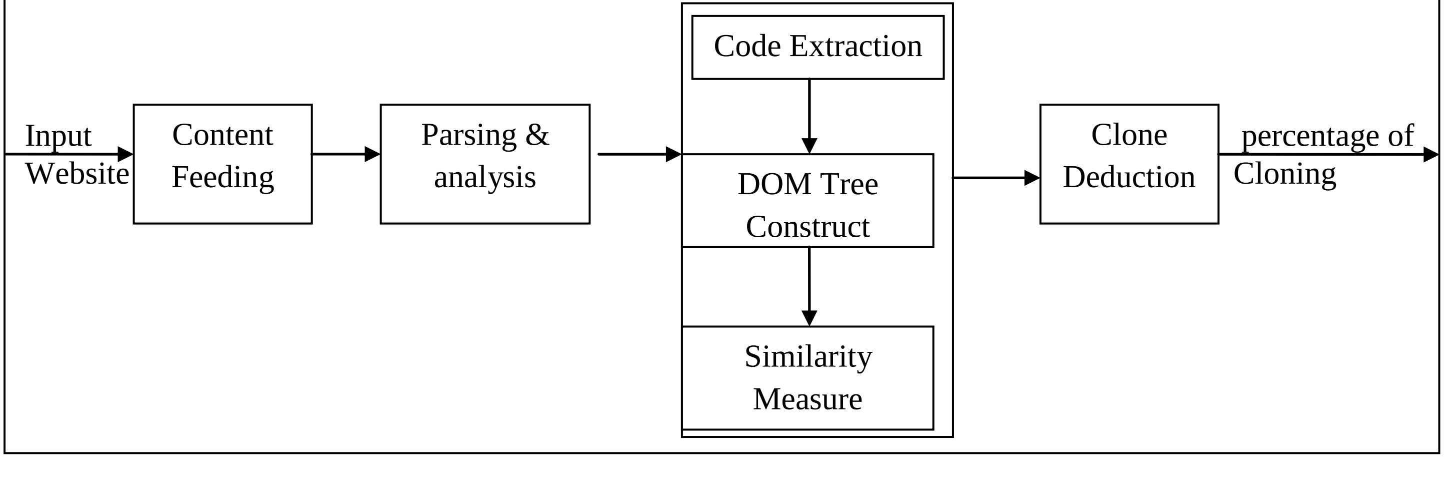
ABSTRACT
Now-a-days cloning of codes or programs of the developer or authorized person leads a positive approach. But the code cloning is done by unauthorized person leads a negative approach. In the recent years, many clone detection tools have been proposed. It produces an over whelming volume of simple clones of data or structure [3]. Code clone detection the content similarity between the programs or webpages. An attempt is made to desgn a method called "SD Code Clone Detection" for both static and dynamic webpages. It is based on levenshtein's approach. This method comprises some steps like, parsing & analysis, tree construction, code similarity measure and clone detection. Experiments are carried out with open source websites and webpages created by some volunteers. Experimental results are recorded and are showing the better detection rate.

Keywords
Refactoring, clone detection, code clone, static and dynamic pages, DOM tree construct, Levenshtein distance algorithm.

1. INTRODUCTION
Refactoring is a process of transforming the program without affecting the behavior and semantics and to improve the quality [24]. In other term code refactoring is the process of restructuring the existing computer code by changing the factors without affecting its external behavior [24]. The refactoring process also involves in the removal of duplication and simplification of unclear code[34]. The refactoring process offers many advantages such as improved code readability and reduced complexity to improve source code maintainability, creation of expressive internal structure [24]. The maintainability and extensibility are the two major benefits of refactoring. But the other side of code refactoring is called code clone. It is about the similarity of codes. Code clone can be defined as a similar program or code structure of considerable size and significant similarity [1]. Section 2 provides the literature review. In section 3 the proposed methods is discussed. Experimental results are recorded in section 4 and section 5 concludes the work.

2. LITERATURE REVIEW
The literature survey shows that cloning is an active area of research [1]. Many refactoring and clone detection tools and approaches have been proposed. A literature survey has been made to have a knowledge on code clone detection and its techniques[3]. Daniel. B [5] proposed a techniques and described some examples of refactoring such as renaming program element to be better convey its meaning, replacing field references splitting large classes etc., many other code refactor techniques have been proposed for code or software systems [2,4,6,7,8,9,10,15, 16,17,11,12,20,13,14,19,28,22,24,25,26,27,28,29,30,31,32, 33].

3. PROPOSED METHOD: SD CODE CLONE DETECTION TECHNIQUE
A approach to clone mining for Web applications has been proposed together with a prototype implementation for dynamic web pages. The proposed methods analyze the page structure, implemented by specific sequences of HTML tags, and the content displayed for both dynamic and static pages. Moreover, for a pair of dynamic web pages we also consider the similarity degree of their source is considered. The similarity degree can be adapted and tuned in a simple way for different web applications in one- to- many. The proposed method called "SD Code Clone detection technique (SDCC) aims the detection of clones on both static and dynamic web pages. The proposed model consists of 4 phases namely content feeding, parsing and analysis refactoring (code extraction, DOM tree and similarity calculation), clone deduction as shown in fig.



Step 5: Perform refactoring and Similarity measure using Levenshtein's approach.

Step 6: Detect the Clone from the code

The algorithm of the proposed methodology based on levenshtein distance measure is given below

3.2 Description

3.2.1 Input and content extraction

In the initial step, WebPages are read. The given input WebPages is transferred to the next phase to extract the contents. Web page are extracted one by one and the content (or) pieces of webpage code and extracted sequentially. Further these contents are forwarded to parse analysis [2].

3.2.2 Parsing and analysis

During this phase, the HTML parsing module accesses the HTML as tokens. It gives one token at a time, much as a file handler which gives one line at a time from a file. The HTML is tokenized from the input file as a string. The tokenize decodes the entities in attributes [35].

3.2.3 Tree construction

The tokenizes passes the output to construct tree. The data instances of the same type have the same path from the root in the DOM tree of the input page according to the page generation model. This method focuses on all levels of nodes. It starts from the root node <HTML>. It uses multiple string arguments approach to the first level child node [35].

3.2.4 Similarity Measure

The next level of the method is to computes the similarity measures using the levenshtein distance approach. It is based on matrix. A matrix is reserved to hold the distance between all prefix of the first string and all prefix of the second Afterwards computation is done on values of the matrix in a dynamic program. Fashion and them the distantness but the two full strings can be measure [36].

3.2.5 Clone Detection

The last step of the stage of the method is to detect the clone values from the outcome of the previous step. Clone detects values (%) and clone index values are identified. The experimental results are discussed in the next section. Path from the root in the DOM tree of the input page according to the page generation model. This method focuses on all levels of nodes. It starts from the root node <HTML>. It uses multiple string arguments approach to the first level child node [4].

4. EXPERIMENTAL RESULTS & DISCUSSION

The proposed refactoring techniques for clone detection have been implemented in C# and experimental results are observed. The following sources shown in table 1 and table 2 are used for the experiments.

Table 1 : The HTML files analyzed in the experimental

File ID	File Name	KB
1	\Index.html	8.07
2	\Special list \main frame.html	0.411
3	\Special list \Special list.html	1.75
4	\Special list text.html	2.30
5	\Special list \title.html	0.363
6	\Novita \Brugaletta.html	6.57
7	\Novita \CalendariotarNA.html	10.6
8	\Novita \ text.html	3.30
9	\Title.html	0.409
10	\Forum \main frame.html	0.506
11	\Forum \taxt.html	0.237
12	\Forum \title.html	0.4
13	\Common frame left.html	4.78
14	\Common \bottom frame.html	3.21
15	\Main frame.html	0.494
16	\irctc.html	0.46
17	\just dial.html	0.58
18	\Chisiamo \text.html	3.24
19	\Chisiamo \title.html	0.407
20	\Cerca.html	1.87
21	\Cerca \main frame.html	0.501
22	\Cerca \text.html	27.3
23	\Cerca \title.html	0.4
24	\Honda.html	0.48
25	\Swift.html	0.24
26	\TNEB.html	0.20
27	\Redbus.html	0.44
28	\NDTV.html	0.90
29	\Default.html	0.96
30	\Sample.html	0.79
31	\Naukri.html	0.125
32	\VAT.html	0.52
33	\Live cricket.html	0.269
34	\naukri.html	0.125

Table 2 : Real time HTML files created by the volunteers

File ID	File Name	KB
1	\A1.html	0.5
2	\A2.html	0.2
3	\B1.html	0.7
4	\B2.html	0.4
5	\C1.html	0.2
6	\C2.html	0.3
7	\C2.html	0.1
8	\C3.html	0.20

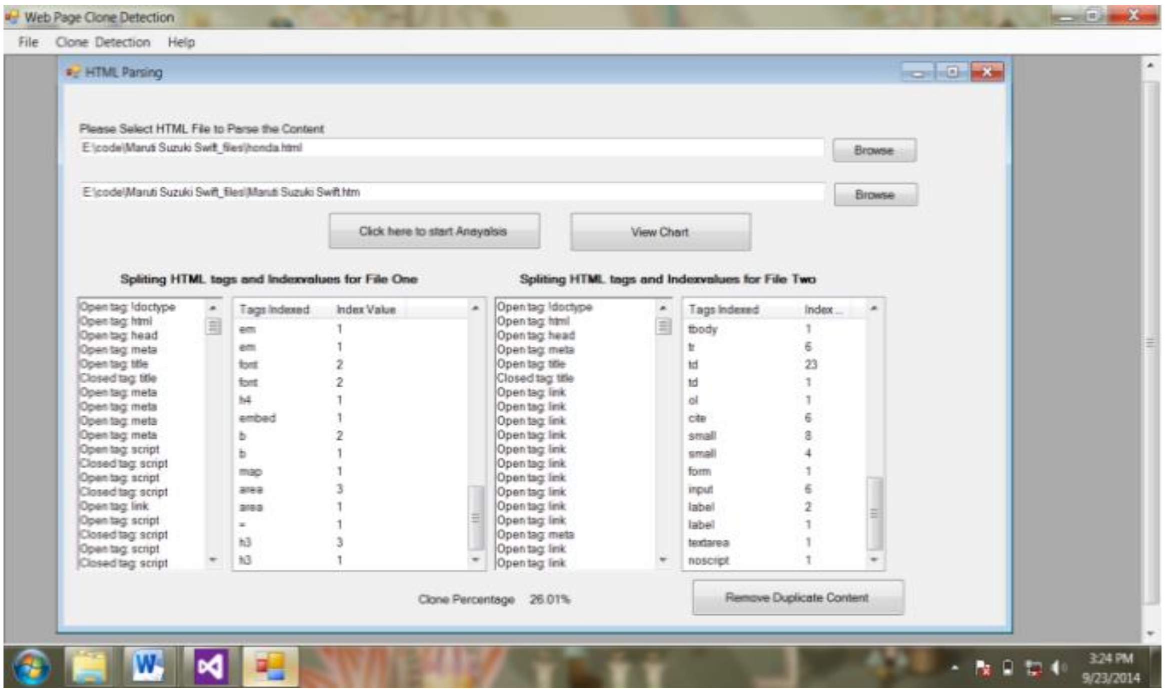
9	\D1.html	0.12
10	\D2.html	0.10
11	\E1.html	0.7
12	\E2.html	0.4

The results of two files from the above mentioned table are shown below in table 3. For instance the results of two files namely Honda.html and SuzukiSwift.html are listed in table1. It lists the tag index and clone detection value for both files which are taken from open sources as mention in table 1.

Table 3 : Tag index and clone detection value for two files (Honda, Swift)

F1 : Honda.html		F2 : Swift.html	
Tags index	**Clone detection value**	**Tags index**	**Clone detection value**
doc type	1	doc type	1
Html	1	html	1
Head	1	head	1
Meta	5	mea	2
Title	1	File	1
Script	55	Link	12
Script	55	Link	1
Link	8	Body	1
Style	3	Div	12
Body	1	Ul	6
Form	1	Li	48
Div	153	A	59
Input	44	A	59
Input	43	Sript	14
Div	153	Script	10
Input	44	Ins	10
Input	43	Ins	5
A	288	Fname	5
Img	153	H1	1
Select	1	H2	2
Option	1	P	32
Strong	46	P	31
Span	74	Img	2
Table	30	B	11
Tbody	30	B	10
Tr	78	H3	9
Td	166	Strong	2
Br	89	Br	42
Ul	38	Br	30
Li	204	Table	1
Li	73	Tbody	1
Form	1	Tr	6
Div	153	Td	23
Input	44	Td	1
Input	43	Ui	1
A	288	File	6
Img	153	Small	8
Select	1	Small	4
Option	1	Form	1
Strong	46	Input	6
Table	30	Lable	2
Tbody	30	Lable	1
Tr	78	Text area	1
Td	166	No script	1
H2	1	-	-
Em	1	-	-
Em	1	-	-
Font	2	-	-
Font	2	-	-
H4	1	-	-
Embed	1	-	-
B	2	-	-
B	1	-	-
Map	1	-	-
Area	3	-	-
H3	3	-	-

Fig. 2 Visualizes clone detection value of the above mentioned files. From the result, it is observed and calculated the clone detection value. This result shows the html tags and index value of first file (Honda.html) and second file (Swift.html). About 26.1% of code clones are identified from the two files.



Please Select HTML File to Parse the Content

E:\code\Maruti Suzuki Swift_files\honda.html

E:\code\Maruti Suzuki Swift_files\Maruti Suzuki Swift.htm

Spliting HTML tags and Indexvalues for File One

```
Open tag: !doctype
Open tag: html
Open tag: head
Open tag: meta
Open tag: title
Closed tag: title
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: script
Closed tag: script
Open tag: script
Closed tag: script
Open tag: link
Open tag: script
Closed tag: script
Open tag: script
Closed tag: script
```

Tags Indexed	Index Value
em	1
em	1
font	2
font	2
h4	1
embed	1
b	2
b	1
map	1
area	3
area	1
=	1
h3	3
h3	1

Spliting HTML tags and Indexvalues for File Two

```
Open tag: !doctype
Open tag: html
Open tag: head
Open tag: meta
Open tag: title
Closed tag: title
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: link
Open tag: meta
Open tag: link
Open tag: link
```

Tags Indexed	Index ...
tbody	1
tr	6
td	23
td	1
ol	1
cite	6
small	8
small	4
form	1
input	6
label	2
label	1
textarea	1
noscript	1

Clone Percentage 26.01%

3:24 PM
9/23/2014

Figure 2: Comparison of clone detection value of two files (F1 : Honda, F2 : Swift).

In Fig. 3 The upper portion of the screen shows the individual clone detection of F1 and F2. Lower portion of the screen shows the comparison of clone detection value of two files.

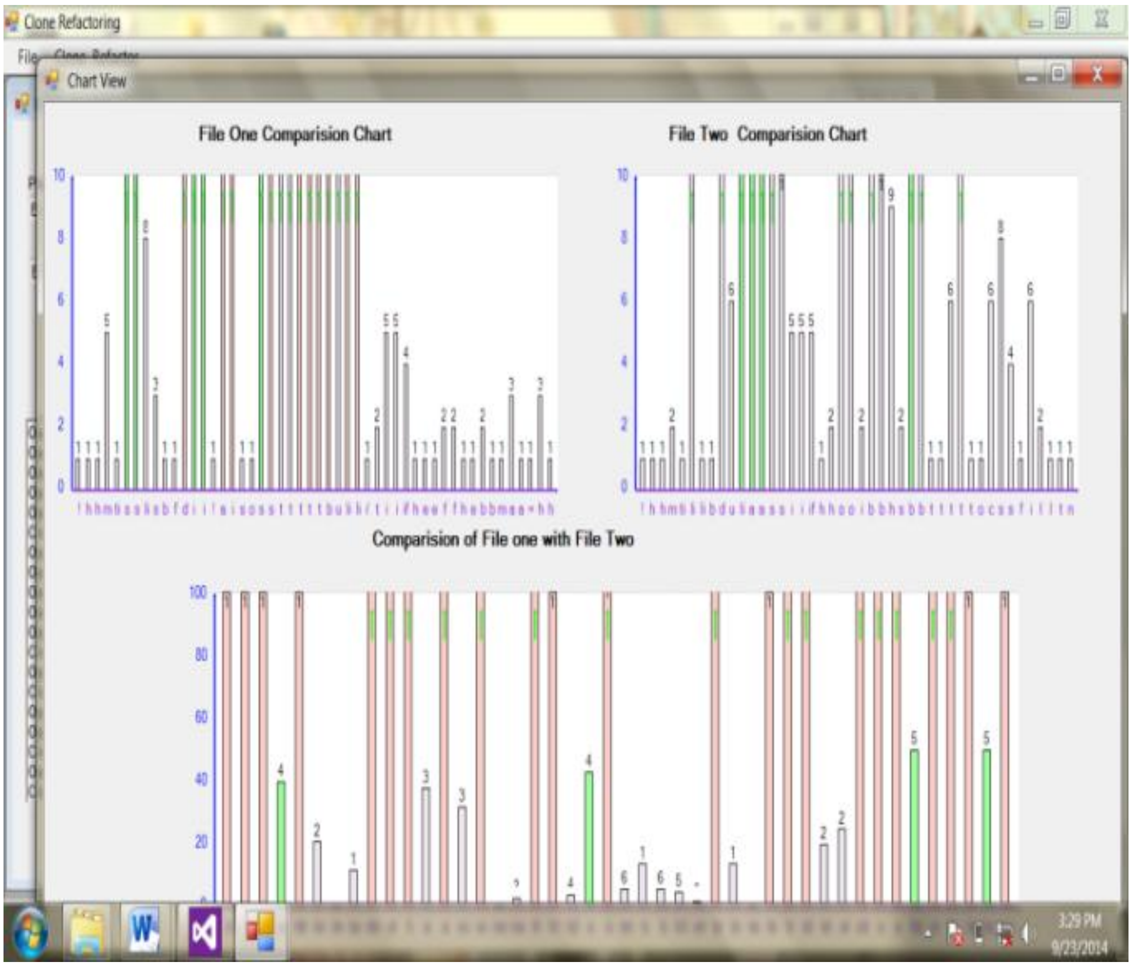
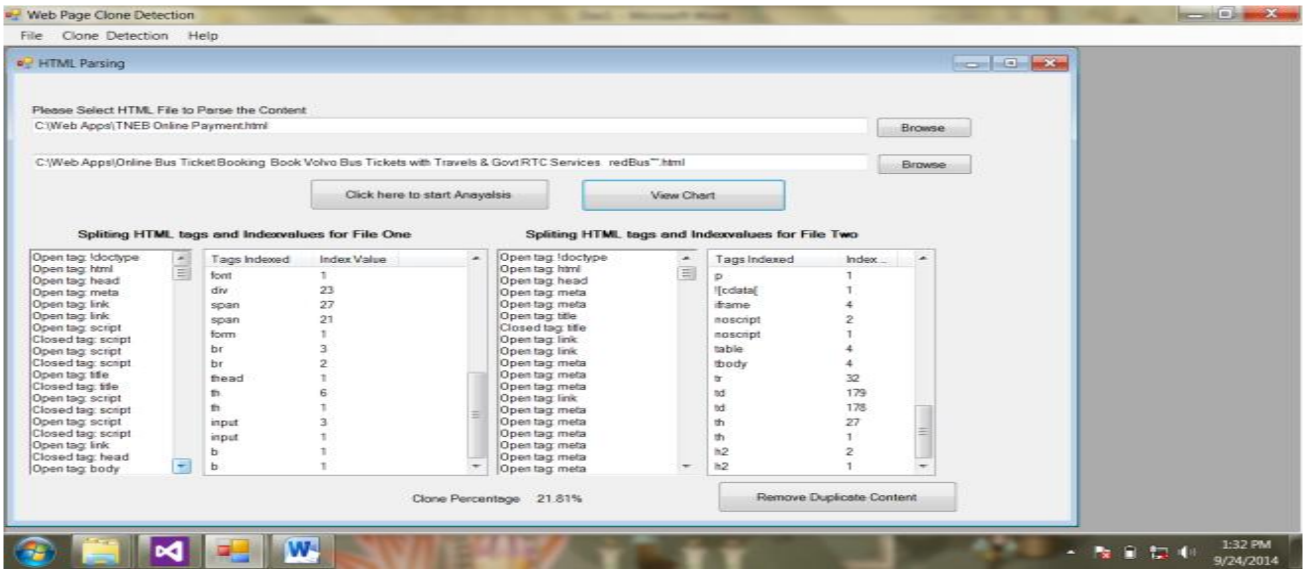


Figure 3: Comparison chart of clone detection on individual values (html tags / and index values).

Table 4: Tag index and clone detection value for two files (TNEB, Redbus)

F3: TNEB.html		**F4 : Redbus.html**	
Tags index	**Clone deduction value**	**Tags index**	**Clone value**
doc type	1	doc type	1
html	1	Html	1
head	1	Head	1
Meta	1	Meta	2
Link	2	Title	1
Link	2	Link	4
Script	4	Link	3
Title	1	Script	21
Body	1	Script	18
Table	1	Body	1
Tbody	4	Div	100
Tr	18	Header	1
Td	53	Ul	6
Title	1	Li	73
Body	1	A	87
Table	4	A	85
Tbody	4	Span	77
Tr	18	Span	21
Td	53	Img	6
Img	9	Br	1
P	2	H3	1
A	44	Section	2
Font	1	H1	1
Div	23	Label	15
Span	27	Input	13
Span	21	Input	11
Form	1	Button	9
Br	3	Aside	1
Br	2	Footer	1
Thead	1	H6	1
Th	6	Sup	1
Th	1	P	1
Input	3	Fname	4
Input	1	Noscript	2
B	1	Noscript	1
-	-	Table	4
-	-	Tbody	4
-	-	Tr	32
-	-	Td	179
-	-	Td	178
-	-	Th	27
-	-	Th	1
-	-	H2	2

Fig. 4 Visualizes clone detection value of the above mentioned files. From the result , the clone detection value is calculated. This result shows the html tags and index value of F3 and F4. About 21.81% of code clones are identified in between two files (TNEB.html, Redbus.html).



Please Select HTML File to Parse the Content
C:\Web Apps\TNEB Online Payment.html
C:\Web Apps\Online Bus Ticket Booking Book Volvo Bus Tickets with Travels & Govt RTC Services redBus™.html

Spliting HTML tags and Indexvalues for File One

```
Open tag: !doctype
Open tag: html
Open tag: head
Open tag: meta
Open tag: link
Open tag: link
Open tag: script
Closed tag: script
Open tag: script
Closed tag: script
Open tag: title
Closed tag: title
Open tag: script
Closed tag: script
Open tag: script
Closed tag: script
Open tag: link
Closed tag: head
Open tag: body
```

Tags Indexed	Index Value
font	1
div	23
span	27
span	21
form	1
br	3
br	2
thead	1
th	6
th	1
input	3
input	1
b	1
b	1

Spliting HTML tags and Indexvalues for File Two

```
Open tag: !doctype
Open tag: html
Open tag: head
Open tag: meta
Open tag: meta
Open tag: title
Closed tag: title
Open tag: link
Open tag: link
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: link
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: meta
Open tag: meta
```

Tags Indexed	Index ...
p	1
![cdata[	1
iframe	4
noscript	2
noscript	1
table	4
tbody	4
tr	32
td	179
td	178
th	27
th	1
h2	2
h2	1

Clone Percentage 21.81%

1:32 PM
9/24/2014

Figure 4 : Comparison of clone detection value of two files (F3 : TNEB, F4 : Redbus).

In fig. 5 the upper portion of the chart shows the individual clone detection of F3 and F4. The lower portion of the represents the comparison of clone detection value of two files. This chart shows the individual clone detection value of F3 and F4. The below chart shows the comparison of clone detection value of two files.

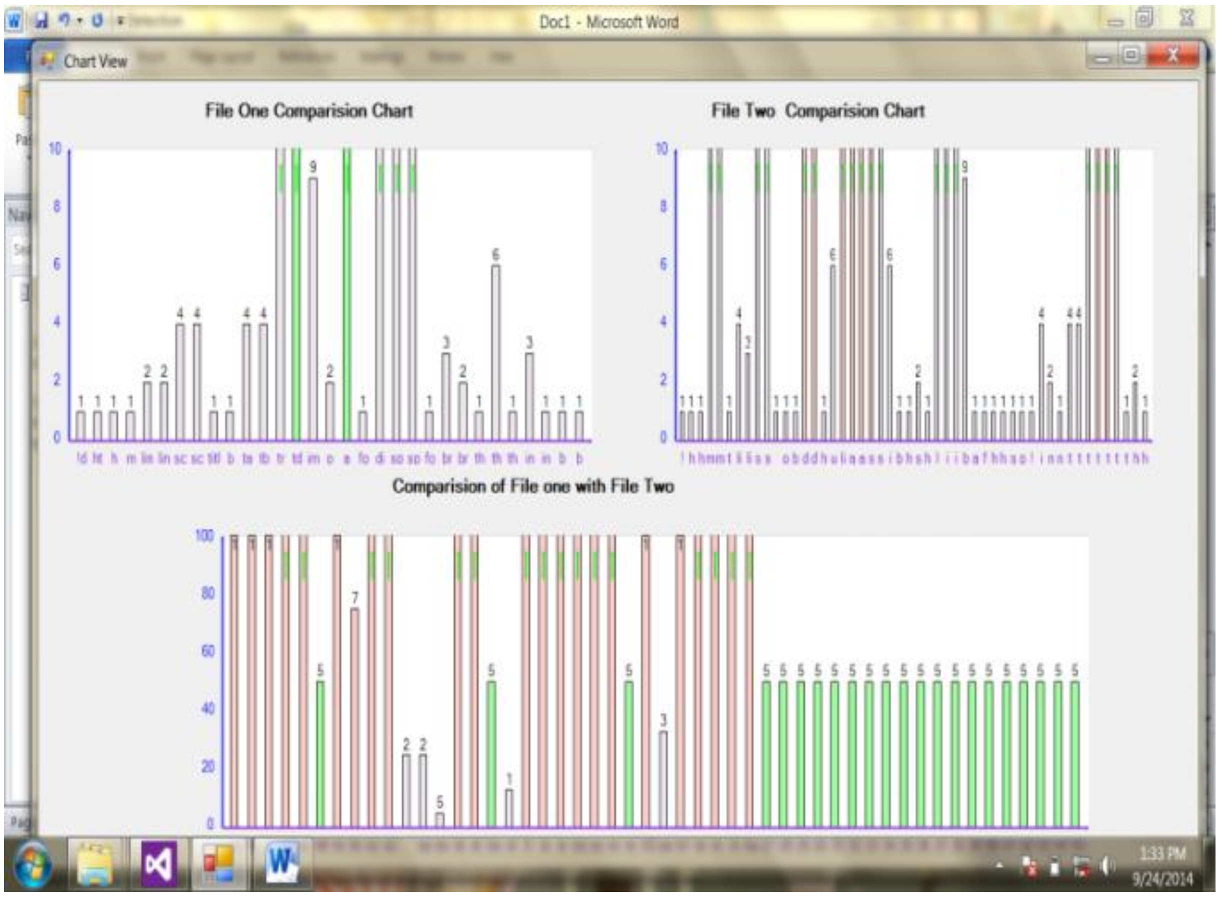
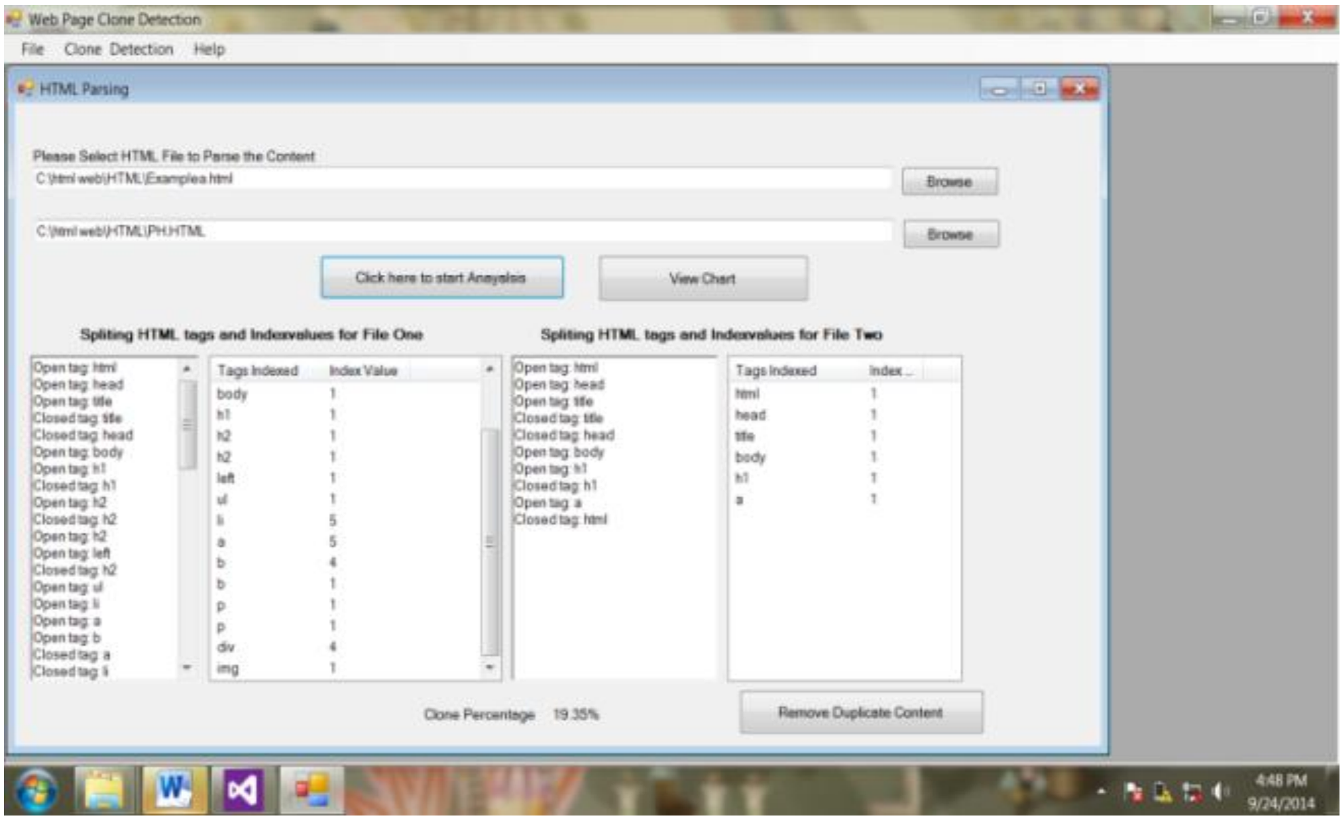


Figure 5: Comparison chart of clone detection on individual values (html tags / and index values).

Table 5: Tag index and clone detection value for two files (A1.html , A2.html) created by voluntaries.

F5:A1.html		F6 : A2.html	
Tags index	**Clone deduction value**	**Tags index**	**Clone value**
Html	1	Html	1
Head	1	Head	1
Title	1	Title	1
Body	1	Body	1
H1	1	H1	1
H2	1	H2	1
H2	1	H2	1
Left	1	Left	1
Ul	5	A	1
Li	4	B	5
A	1	B	5
B	1	B	4
B	1	-	-
P	4	-	-
P	1	-	-
Div	1	-	-

Fig. 6 Visualizes clone detection value of the above mentioned files that the result shows the html tags and index value of F5 and F6. About 19.35% of code clones are identified between two files (A1.html, A2.html).



Please Select HTML File to Parse the Content

C:\html web\HTML\Examplea.html

C:\html web\HTML\PH.HTML

Spliting HTML tags and Indexvalues for File One

```
Open tag: html
Open tag: head
Open tag: title
Closed tag: title
Closed tag: head
Open tag: body
Open tag: h1
Closed tag: h1
Open tag: h2
Closed tag: h2
Open tag: h2
Open tag: left
Closed tag: h2
Open tag: ul
Open tag: li
Open tag: a
Open tag: b
Closed tag: a
Closed tag: li
```

Tags Indexed	Index Value
body	1
h1	1
h2	1
h2	1
left	1
ul	1
li	5
a	5
b	4
b	1
p	1
p	1
div	4
img	1

Spliting HTML tags and Indexvalues for File Two

```
Open tag: html
Open tag: head
Open tag: title
Closed tag: title
Closed tag: head
Open tag: body
Open tag: h1
Closed tag: h1
Open tag: a
Closed tag: html
```

Tags Indexed	Index ...
html	1
head	1
title	1
body	1
h1	1
a	1

Clone Percentage 19.35%

Figure 6: Comparison of clone detection value of two files (F5: A1, F6 : A1).

In fig. 7, the upper portion of the chart shows the individual clone detection of F5 and F6. The lower portion of the chart compares the clone detection value of two files.

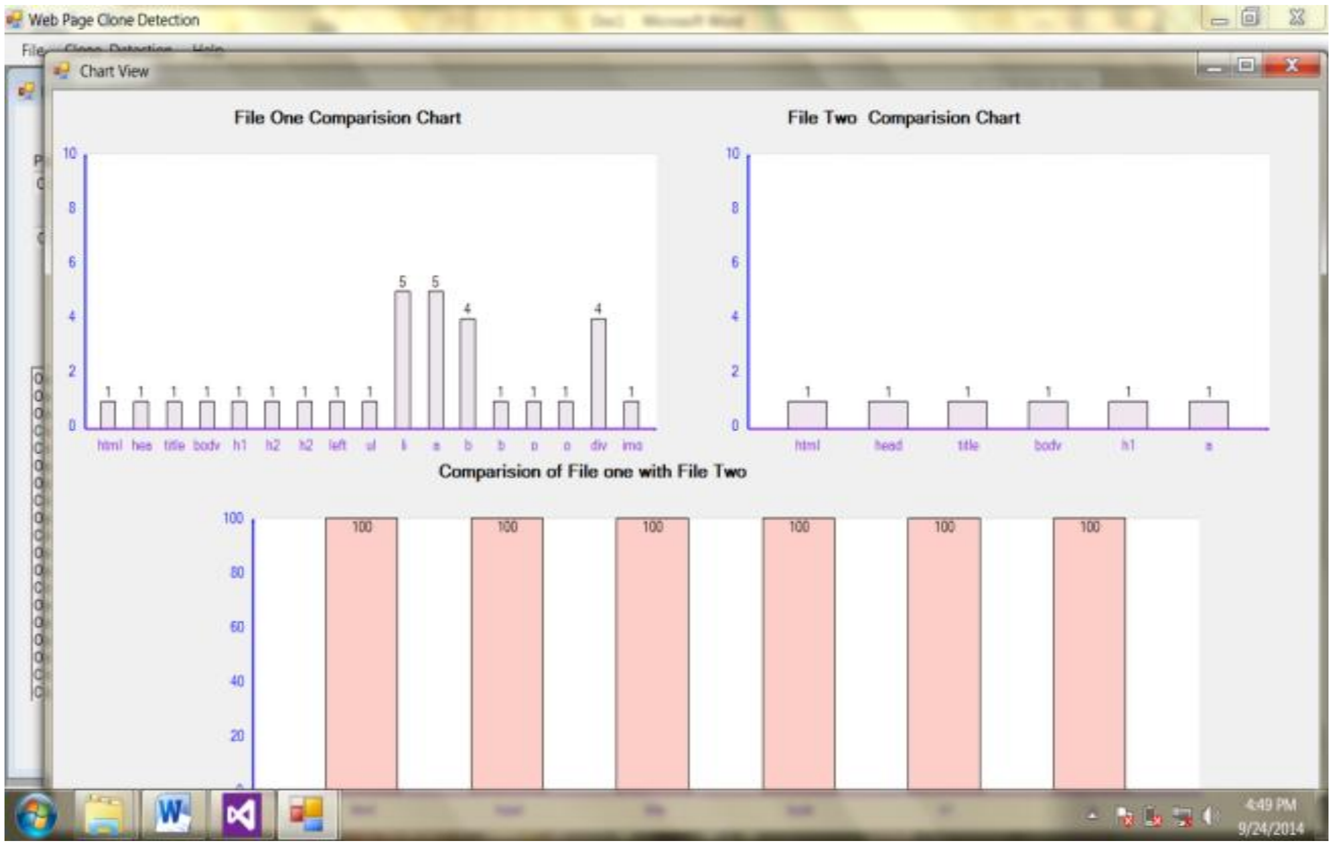


Figure 7: Comparison chart of clone detection on individual values (html tags / and index values).

4.1 Performance Measure

The performance of the proposed methods is based on clone percentage and also time taken to detect the clone. Table 6 lists the two measures for some files.

Table 6: Performance measure on clone percentage

Name of the webpage(s)		Clone percent (%)	
		Static web pages	**Dynamic web pages**
Default.html	Net carry .html	100	40
Swift.html	Net carry .html	33	27
Search.html	Compile.asp	24	46
Code.html	Sample.asp	29	70

Chart describes fig. 8 clone detection percentage of static and dynamic web pages.

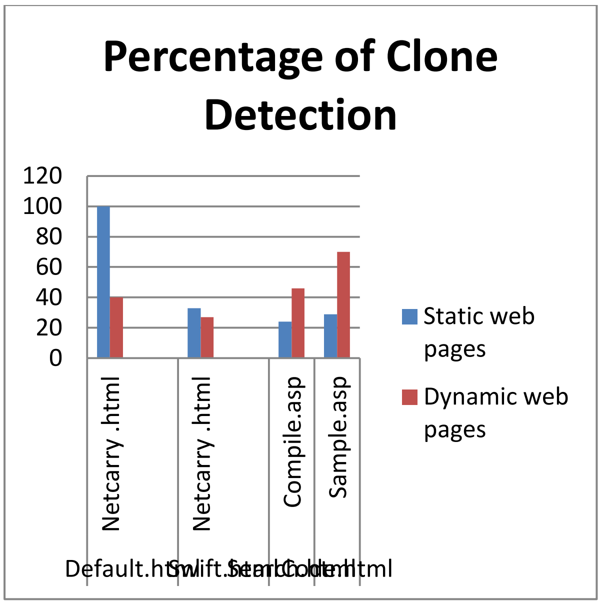


Figure 8: This chart measure comparison of clone percentage.

Table.7: Performance measure to time taken of clone detection

Name of the webpage(s)		Clone detection time (in seconds)
Default.html	Net carry .html	0.55
Swift.html	Net carry .html	0.12
Search.html	Compile.asp	0.06
Code.html	Sample.asp	0.05

Table 7 and Fig. 9 indicate the time measure of clone detection in open source web pages.

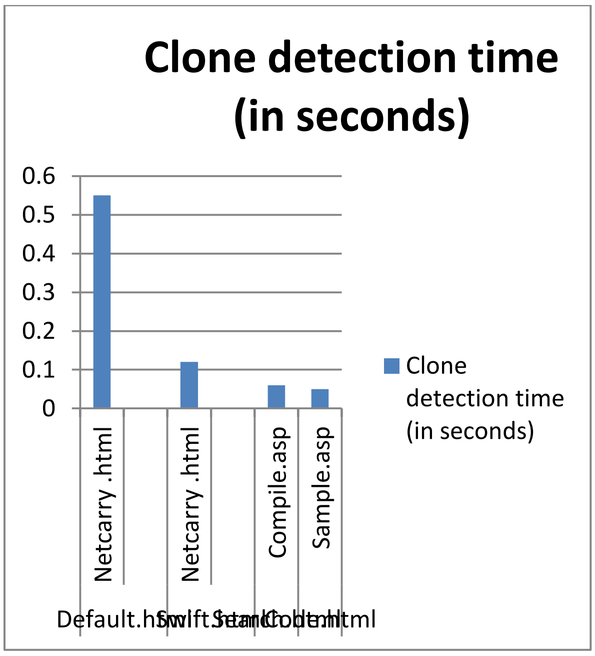


Figure 9: This chart measures clone detection time (In seconds).

Table 8: Performance measure on clone percentage

Name of the webpage(s)		Clone percent (%)	
		Static web pages	**Dynamic web pages**
B1.html	A1.html	49	80
B2.html	A2.html	67	55
VAT.html	Compile.asp	58	93
C1.html	Sample.asp	49	35

Table 8, 9 and Fig. 10, 11 give information about the clone detection percentage of static and dynamic web page.

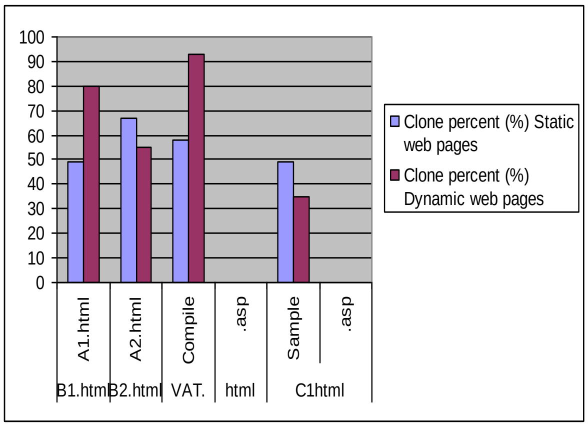


Figure 10: This chart measure comparison of clone percentage.

Table 9 : Performance measure to time taken of clone detection

Name of the webpage(s)		Clone detection time (in seconds)
B1.html	A1.html	0.58
B2.html	A2.html	0.10
VAT.html	Compile.asp	0.05
C1.html	Sample.asp	0.03

Fig. 11 gives information about the clone detection percentage of static and dynamic web page.

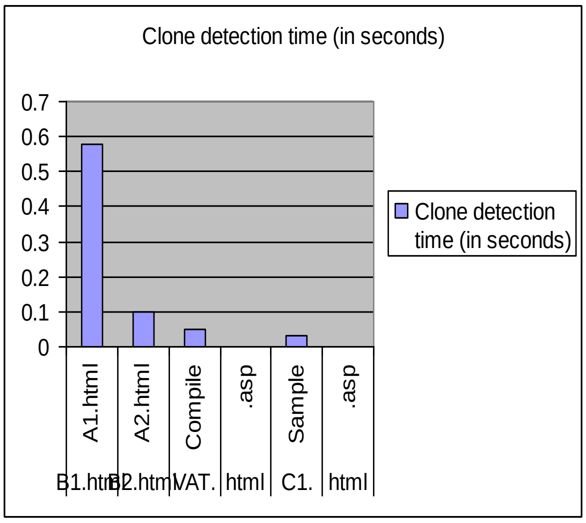


Figure 11: This chart measures clone detection time (In seconds).

5. CONCLUSION

Code clone detection is an art of detecting the content similarity between the programs or WebPages. An attempt is made to design a method called "SD Code Clone Detection" for both static and dynamic WebPages. It is based on levenshtein's approach. This method comprises some steps like, parsing & analysis, tree construction, code similarity measure and clone detection. Experiments are carried out with open source websites and WebPages created by some volunteers. Experimental results are recorded and are showing the better detection rate. Future research on Web data extraction focuses on comparing the contents appearing on the page as well as the code to measure the standard and originality of the web page. However, they are redesigned or applied in a different sequence and scenario to solve key issues in page-level data extraction and comparison to the code of web site and its contents to find the fake and the real. The System can also be enhanced work to detect the script injection and projected towards the detection of malwares attached to web pages that harms the user's machine and acts as a spy ware and sends the information of the end user to the attacker. These systems are still in research to prevent the attackers. It is planned to exploit the results of the clone mining method to support web application reengineering activities.

6. REFERENCES

[1] Aversano, L., Canfora, G., De Lucia, A., and Gallucci, P., 2001. Web Site Reuse: Cloning and Adapting. Proc. Of 3rd International Workshop on Web Site Evolution, Florence, Italy, IEEE CS Press, pp. 107-111.

[2] Chang C.-H. and S.-C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," Proc. Int'l Conf. World Wide Web (WWW-10), pp. 223-231, 2001.

[3] Cloning http://msdn.microsoft.com/en-us/library/hh205279.aspx

[4] De Lucia, A., Scanniello, G., and Tortora, G., 2004."Identifying Clones in Dynamic Web Sites Using Similarity Thresholds," Proc. Intl. Conf. on Enterprise Information Systems (ICEIS'04), pp.391-396.

[5] Daniel B, D. Dig, K. Garcia, and D. Marinov, "Automated Testing of Refactoring Engines," Proc. Sixth Joint Meeting European Software Eng. Conf. and ACM SIGSOFT Symp. The Foundations of Software Eng., pp. 185-194, 2007.

[6] Dig D, and R. Johnson, "The Role of Refactorings in API Evolution," Proc. 21st IEEE Int'l Conf. Software Maintenance,pp. 389-398, 2005.

[7] Eclipse.org, "Eclipse Project,"at http://www.eclipse.org, 2011.

[8] EmbarcaderoTechnologies,"JBuilder," http://www.codegear.com/br/products/jbuilder, 2011.

[9] "JDT Core Component," Eclipse.org, http://www.eclipse.org/ jdt/core/, 2011.

[10] Fowler M., Refactoring: Improving the Design of Existing Code. Addison-Wesley Longman Publishing Co., 1999.

[11] Goodenough J.B. and S.L. Gerhart, "Toward a Theory of Test Data Selection," SIGPLAN Notes, vol. 10, pp. 493-510, Apr. 1975.

[12] Gligoric M, T. Gvero, V. Jagannath, S. Khurshid, V. Kuncak, and D. Marinov, "Test Generation through Programming in UDITA," Proc. 32nd Int'l Conf. Software Eng., vol. 1, pp. 225-234, 2010.

[13] Hoffman D.M, D. Ly-Gagnon, P. Strooper, and H.-Y. Wang, "Grammar-Based Test Generation with YouGen," Software: Prac- tice and Experience, vol. 41, pp. 427-447, Apr. 2011.

[14] Jackson D., I. Schechter, and H. Shlyahter, "Alcoa: The Alloy Constraint Analyzer," Proc. 22nd Int'l Conf. Software Eng., pp. 730-733, 2000.

[15] Jin W, A. Orso, and T. Xie, "Automated Behavioral Regression Testing," Proc. 23rd Int'l Conf. Software Testing, Verification and Validation, pp. 137-146, 2010.

[16] Kushmerick, D. Weld, and R. Doorenbos, "Wrapper Induction for Information Extraction," Proc. 15th Int'l Joint Conf. Artificial Intelligence (IJCAI), pp. 729-735, 1997.

[17] Muslea I., S. Minton, and C. Knoblock, "A Hierarchical Approach to Wrapper Induction," Proc. Third Int'l Conf. Autonomous Agents (AA '99), 1999.

[18] Mens T. and T. Tourwe ́, “A Survey of Software Refactoring,” IEEE Trans. Software Eng., vol. 30, no. 2, pp. 126-139, Feb. 2004.

[19] Demeyer. S, Mens T, and D. Janssens, “Formalising Behaviour Preserving Program Transformations,” Proc. First Int’l Conf. Graph Transformation, pp. 286-301, 2002.

[20] Marinov D and S. Khurshid, “TestEra: A Novel Framework for Automated Testing of Java Programs,” Proc. IEEE 16th Int’l Conf. Automated Software Eng., pp. 22-34, 2001.

[21] Opdyke W.F, “Refactoring Object-Oriented Frameworks,” PhD dissertation, Univ. of Illinois at Urbana-Champaign, 1992.

[22] Overbey J.L and R.E. Johnson, “Differential Precondition Check-ing: A Lightweight, Reusable Analysis for Refactoring Tools,” Proc. 26th IEEE/ACM Int’l Conf. Automated Software Eng., pp. 303-312,

[23] Refactoring http://www.informit.com/articles/article.aspx

[24] Simon K and G. Lausen, “ViPER: Augmenting Automatic Information Extraction with Visual Perceptions,” Proc. Int’l Conf. Information and Knowledge Management (CIKM), 2005.

[25] Sun Microsystems, “NetBeans IDE,” http://www.netbeans.org/,2011.

[26] Scha ̈fer M, M. Verbaere, T. Ekman, and O. Moor, “Stepping Stones over the Refactoring Rubicon,” Proc. 23rd European Conf. Object-Oriented Programming, pp. 369-393, 2009.

[27] Moor O. de, Scha ̈fer M and “Specifying and Implementing Refactorings,” Proc. 25th ACM Int’l Conf. Object-Oriented Programming, Systems, Languages, and Applications, pp. 286-301, 2010.

[28] Ekman T. Scha ̈fer M, and O. de Moor, “Challenge Proposal: Verification of Refactorings,” Proc. Third Workshop Programming Languages Meets Program Verification, pp. 67-72, 2008.

[29] Gheyi. R Soares G, D. Serey, and T. Massoni, “Making Program Refactoring Safer,” IEEE Software, vol. 27, no. 4, pp. 52-57, July/ Aug. 2010.

[30] Mongiovi M., Soares G, and R. Gheyi, “Identifying Overly Strong Conditions in Refactoring Implementations,” Proc. Conf. Software Maintenance, pp. 173-182, Sept. 2011.

[31] Silva L, A. Sampaio, and Z. Liu, “Laws of Object-Orientation with Reference Semantics,” Proc. Sixth IEEE Int’l Conf. Software Eng. And Formal Methods, pp. 217-226, 2008.

[32] Moor O. de, Scha ̈fer M, T. Ekman, and “Sound and Extensible Renaming for Java,” Proc. 23rd ACM SIGPLAN Conf. Object Oriented Programming, Systems, Languages, and Applications, pp. 277-294, 2008.

[33] Tokuda L and D. Batory, “Evolving Object-Oriented Designs with Refactoring,” Automated Software Eng., vol. 8, pp. 89-120, Jan.2001.

[34] Sourcemaking http://sourcemaking.com/refactoring/introduction-to-refactoring

[35] DOM Tree Algorithm :http://dbs.snu.ac.kr/papers/xsym09.pdf.

[36] Levenshtein Edit Distance Algorithm: http://www.levenshtein.net/ levenshtein measure http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance.