Code Clones Detection in Websites using Hybrid Approach

Rubala Sivakumar
Dept. of Computer Science
Sri Manakula Vinayagar Engineering Collage
Puducherry, India

Kodhai.E
Dept. of Information Tech.,
Sri Manakula Vinayagar Engineering Collage
Puducherry, India

ABSTRACT
Code Clone is a pathological form of software reuse because of its effects on the maintenance of large software systems. Probably the existing web applications use a mixture of Designing page and scripting language code as the front-end to business services. Analogously to traditional applications, redundant code is introduced by copy-and-paste practices called 'Code Clones'. This paper proposes the detection of all types of clones by identifying cloned functions within scripting code of web applications using Hybrid approach with the combination of textual and metric analysis. Various metrics had been formulated and their values were utilized during the detection process. Compared to the other approaches, this method is considered to be the least complex and is to provide a most accurate and efficient way of Clone Detection. The results obtained had been compared with an existing tool for the open source of web applications.

Keywords
Clone Detection, Code duplication, Functional Clones, Hybrid approach, refactoring, and web applications.

1. INTRODUCTION
Replication of code occurs recurrently during the growth and evolution of large software systems. Clones are habitually created by copy-and paste programming and expansion of the presented code and also often used in the outward appearance of reuse consists in copying, and ultimately altering, a block of existing code that implements a portion of requisite functionality. Duplicated chunks are called *clones* and be active of copying, together with trivial alterations, is said *cloning*. As soon as complete functions are copied more willingly than segments, duplicated functions are called *function clones*.

According to the different similarities, clone can be categorized into two: One type of similarity considers textual similarity and other second considers the semantic level, which the clone code must have the same behaviors, means the functional similarity (1).

1.1 Textual Similarity
Based on the textual similarity we discriminate the subsequent types of clones:

1.1.1 Type I
Indistinguishable code segments i.e. an accurate replica without alterations.

1.1.2 Type II
Formational wise / syntactically indistinguishable segments excluding dissimilarity in white space, identifiers, literals, types, layout and comments.

1.1.3 Type III
Imitative segments with extra alterations. Statements can be changed, added or removed in accumulation to dissimilarities in identifiers, literals, types, layout and comments.

1.2 Functional Similarity
If the functionalities of the two code segments are indistinguishable or alike and referred as Type IV clones.

1.2.1 Type IV
Code segments that used to perform the comparable computation but implemented throughout different syntactic variants.

The consequences of the code clone identification are more often than not given as clone pairs or clone clusters along with their location or occurrence.
Clone Pair (CP): pair of code segments / fragments which are indistinguishable or comparable to each other.
Clone Cluster (CC): the amalgamation of all clone pairs which have code segments in general.

Researchers have all-embracing deliberated that clones detection for being applied to procedural programs (2, 3, 4, 6, 7, and 8) as well as object-oriented programs (9, 10, 6, 11, and 12). Blemishing software clones for the objective of deterrence or elimination can assist to defy at "software aging" (13), where even miniature alterations turn out to be very complex to apply.

In recent times, clone detection has been wished-for for static web documents (14), written in HTML. On the other hand, modern web applications are a blend of HTML and scripting language code, where scripts can sprint as occurrence handlers on the client-side or achieve HTTP processing on the server-side. A foremost approach to server-side handing out of HTTP requests is using *server pages*, i.e., documents containing HTML make notes on with server-side construed scripts. Many scripting languages are supported and envoy examples are Java Server Pages (JSP), Microsoft's Active Server Pages (ASP), and PHP.

Web applications progressed from web sites by adding up commerce functionality (15). A web application employs web technologies as the front-end to industry services for trouble-free of deployment and minimal client design. The observable fact of replicate code is even shoddier for this class of applications for the reason that many people have flawed

thought that software engineering principles and techniques do not apply to web development (16).

Auspiciously, a lot of techniques for the revealing of code clones have been proposed. They demonstrate that lightweight text-based techniques can find clones with high correctness and confidence, but detected clones time and again do not correspond to apt syntactic units (17, 18). Parser based syntactic (AST-based) techniques, find syntactically significant clones but be inclined to be more heavyweight, requiring a full parser and sub-tree evaluation method.

An Incremental detection technique perceives clones in less time in each revision separately (19). Furthermore, it is useful only for the certification of preceding versions of software with detected clones. The intricacy of all the methods is high and this can be reduced with the computed metrics values.

In this paper, a narrative code clone detection process using hybrid approach with the combination of text based and a metrics-based technique has been proposed. It has also been implemented as a tool using DOT NET. The tool efficiently and accurately detects type-1, type-2, type-3 and type-4 clones found in web applications at function level with the open source QuickAuction210 (20), SnitzTM Forum (21) and Web Wiz Forum (22).

This paper contains four major sections. Section II describes the implementation of the proposed method. Section III gives the results obtained using the proposed method. Finally, Section IV concludes the paper.

2. IMPLEMENTATION OF THE PROPOSED METHOD

The proposed method is implemented as a tool in DOT NET. The system architecture of the tool (see Figure 1). The tool developed initially integrates the given input source code and identifies the various functions present. Having identified the potential functions, Templates are converted for that potential functions and also various metrics formulated are computed for each functions and the metrics stored in the database.

With the help of the metric values and converted templates the possible potential clone clusters are extracted and are further put forth for the textual comparison to detect all types of clones.

2.1 Source files Integration

In this phase, Website (web pages or web documents) will be selected as input project. Selecting the project involves the concatenation of all the files of the same project into a single large file. To be exact, all web pages in website are linked together to form a single large text file for an effective parsing.

2.2 Normalization and Potential code identification

This phase includes source code normalization and identification of potential code. In normalization, the integrated text file is parsed for the removal of, pre-processor statements, white spaces and comments.

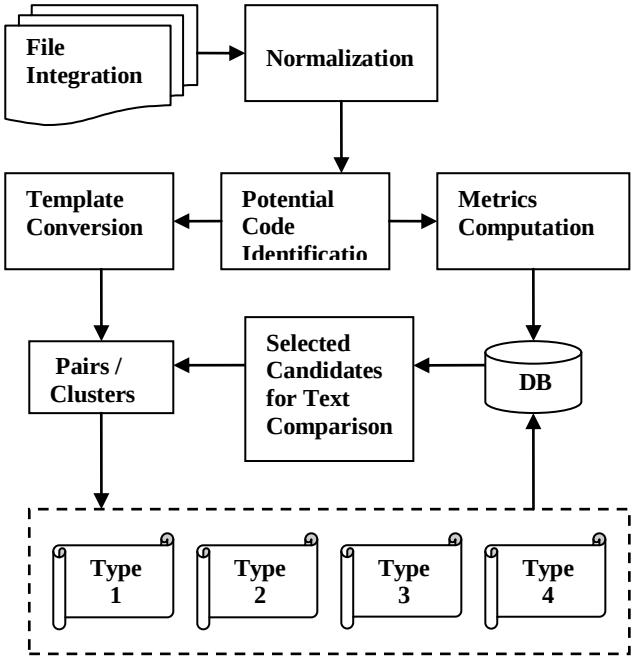

Fig 1: System Architecture of the developed tool

Source code is then re-structured to a standard format which is significant for instituting similarity of the cloned portions. These steps give up a significant gain in the recall. From that normalized code potential functions are extracted and saved for further reference to compute metrics and convert templates.

2.3 Template Conversion

Template conversion is said to be the conversion of the inputted source code into a pre-defined set of statements or translation into a standard intermediary form. For instance, renaming of data types, variables, function names, etc (see Figure 2).

```
Dim sFirst                          Dim sFirst
Dim sSecond                           Dim sSecond
If sFirst = sSecond Then      ➡     Selection
OutMatch = pvMtchOut                  sFirst = sSecond
Else                                  Then
OutMatch = pvBadOut                   OutMatch = pvMtchOut
End If                              Selection
                                      OutMatch = pvBadOut
                                    End
```

Fig 2: Sample inputted method before and after Template Conversion

2.4 Metrics Computation

A set of 7 existing function level metrics are used for the detection of all types of clone functions in web application are as follows:

- No. of Lines of Code in each function.
- No. of effective lines of code in each function.
- No. of conditional statements in each function.
- No. of looping statements in each function.
- No. of Comment lines of code in each function.
- No. of Return Statements in each function
- No. of Variable Declaration in each function

The metrics are computed for each of the methods identified and the values are stored in a database. The various metric values for the code fragment. The descriptive statistics of the metric values obtained for the various methods (see Table 1).

Table 1. Metrics values for the method given in Fig 2.

Function name/ S.No.	Metrics	Values
Function OutMatch(pvFirst, pvSecond, pvMatchOut, pvBadOut)		
1	No. of Lines of Code	7
2	No. of Effective Lines	7
3	No. of Conditional Statements	1
4	No. of Looping Statements	0
5	No. of Variable Declaration	4
6	No. of Return Statements	0
7	No. of Comments	0

2.5 Textual Analysis

This phase includes the selecting candidates for textual comparison and finding Clone Pairs and Clusters. Having computed the metric values, the method pairs with equal or similar set of metrics values are identified by comparison of the records in the database. The short-listed set of candidates is then textually compared with the template file to be confirmed as clone pairs/clusters.

2.6 Type Clone Detection

This phase includes detection of type 1, type 2, type 3 and type 4 clones with the identification of clone pairs / clusters appropriately. If pairs/clusters seem exact then it is named as type 1 clones. Structurally/syntactically identical fragments are said to be Type 2 clones.

Perhaps, there are some modifications in the fragments but there is some similarities means it should be declared as type-3 clones. Then code fragments are completely different but produce similar output, and then it is declared as type-4 clone. Type clone detection gives a clear image of how the methods were cloned and helps to provide an easier review process.

3. RESULT ANALYSIS

Containing implementation of the above process, the tool was trialed with a popular medium sized open source web application, Quick Auction [20], is a basic auction application that can be integrated into other web sites to add simple auctions features. Additionally, web applications, Web Wiz Forums [22] and Snitz Forums 2000 [21], are both web-based bulletin board engines were also examined to assess the effectiveness and efficiency of the process.

All three applications use MS ASP technology to implement web server pages. They range in size from tens to hundreds of web pages, and are currently being maintained and evolved were shown with characterizations of the web applications including size-related statistics (see Table 2).

Table 2. Characterizations of Web Applications Experimented with Proposed Method

Web Applications	Version	Scripting Language	No. of Functions	No. of Files
Quick Auction	2.0.0	VBScript	39	50
SnitzTM Forums 2000	3.4.07	VBScript, JavaScript	79	104
Web Wiz Forums	7.01	VBScript, JavaScript	126	233

Having worked out the number of type-1, type-2, type-3 and type-4 cloned functions for different web applications (see Table 3); the result of proposed method has been compared with the existing tool, eMetrics, developed at University of Bari (see Table 4). The eMetrics tool analyzes web applications that adopt Microsoft's ASP technology which measures the size of a web application to different granularity levels (including script functions), and then selects homonym programmer-defined functions (written in JavaScript or VBScript) as potential cloned functions. Homonymy is defined by the equality of identifiers that are used as function names in the declaration of script functions.

*****Clu – Clusters *****Clo – Clones

Web Applications	Type I		Type II		Type III		Type IV	
	Clu	Clo	Clu	Clo	Clu	Clo	Clu	Clo
Quick Auction	4	14	3	6	4	8	2	7
Snitz TM Forums 2000	8	19	5	11	9	19	4	9
Web Wiz Forums	9	21	9	18	18	43	14	44

Table 3. Types of Clones for Different Web Application

In case of Quick Auction, eMetrics tool reported 12 functional clones as exact match (type-1), only 2 near miss functional clones (type-2), 10 similar functional clones (type-3) and 0 clones found as distinct (type-4). Whereas for Snitz Forums 2000, 20 functional clones as exact match (type-1), only 4 near miss functional clones (type-2), 14 similar functional clones (type-3) and 0 clones found as distinct (type-4). Similarly for Web Wiz Forums 25 functional clones as exact match (type-1), only 9 near miss functional clones (type-2), 14 similar functional clones (type-3) and 0 clones found as distinct (type-4). The eMetrics tool helps only for semi-automatic approach which requires visual inspection of source code that is Instead of using measures to automatically

classify Clones it uses size measures to provide clues and set priorities for the visual inspection.

It is concluded that the proposed method has accomplished the desired output with full-automation. (See Table 5) determined percentage of precision and recall with the respective Chart (see Figure 3).

Table 4. Types Comparison of Proposed Method with an Existing Tool

Types of Clones	Quick Auction		SnitzTM Forums 2000		Web Wiz Forums	
	Proposed Method	eMetrics Tool	Proposed Method	eMetrics Tool	Proposed Method	eMetrics Tool
Type I Clones	14	12	19	20	21	25
Type II Clones	6	2	11	4	18	9
Type III Clones	8	10	19	14	43	14
Type IV Clones	7	0	9	0	44	0

Table 5. Percentage of Precision and Recall

	Quick Auction		Snitz TM Forums 2000		Web Wiz Forums	
	Propo Method	eMetric Tool	Propo Method	eMetric Tool	Propo Method	eMetric Tool
Precision %	95 %	90 %	80 %	61 %	90 %	78 %
Recall %	92 %	90 %	77%	67 %	96%	96 %

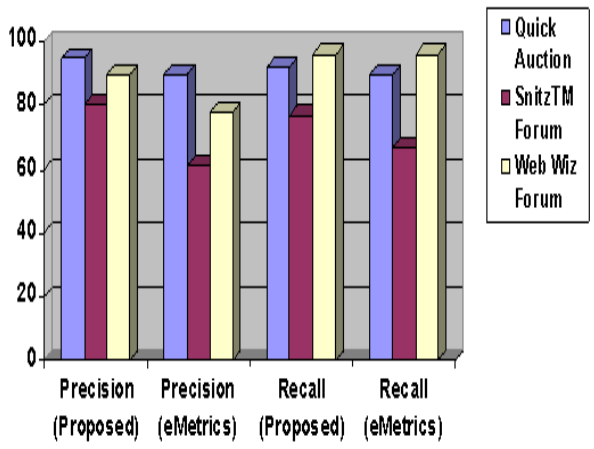

Fig 3: Result of precision and recall

4. CONCLUSIONS

The proposed work uses a light weight technique to detect cloned functions within scripting code of web applications with the computation of metrics based technique with textual analysis technique. The potential cloned script functions are detected from integrated files by a proposed tool, with the usage of metrics the existing exponential rate of comparison of the various functions has been avoided thus improving the precision and reducing the total comparison overhead. Since the string matching/textual comparison is performed over the short-listed candidates, a higher amount of recall could be obtained with the effective detection of type 1, type 2, type 3 and type 4 Clones.

Finding function clones in scripted web applications for the purpose of eliminating duplicated code can be seen as a first step to introduce refactoring and also to investigate how to improve poorly designed web applications. Having implemented the above process, the proposed tool was experimented with popular medium sized open source web applications, such as Quick Auction, Web Wiz Forums and Snitz Forums 2000 and also the result of proposed method has been compared with the existing tool, eMetrics. The result manipulations explored as percentage of precision and recall with the respective Chart. Thus it concludes that the proposed method accomplished the desired output with full-automation.

The future works may be the automatic selection of potential clones could also be extended to other web enabling technologies, such as PHP or JSP (Java Server Pages).

5. REFERENCES

[1] Kodhai.E, Kanmani.S, Kamatchi.A, Radhika.R, Detection of Type-1 and Type-2 Clone Using Textual Analysis and Metrics in ITC,2010.

[2] Antoniol, G., Villano, U., Merlo, E. and Di Penta, M. Analyzing cloning evolution in the Linuxkernel. Information and Software Technology 2002, 44(13). 755–765.

[3] Antoniol, G., Casazza, G., Di Penta, M. and Merlo, E. Modeling Clones Evolution Through Time Series. in International Conference on Software Maintenance, (Florence, Italy, 2001). 273-280.

[4] Balazinska, M., Merlo, E., Dagenais, M., Lague, B and Kontogiannis, K. Measuring Clone Based Reengineering Opportunities. in Sixth IEEE International Symposium on Software Metrics, (Boca Raton, USA, 1999). 292-303.

[5] Boldyreff, C. and Kewish, R. Reverse Engineering to Achieve Maintainable WWW Sites. in EightWorking Conference on Reverse Engineering (WCRE'01), (Stuttgart, Germany, 2001). 249-257.

[6] Fioravanti, F., Migliarase, G. and Nesi, P. Reengineering Analysis of Object-Oriented Systems via Duplication Analysis. in International Conference on Software Engineering, (Florence, Italy, 2001); 577-590.

[7] Baker, B.S. On Finding Duplication and Near-Duplication in Large Software Systems. in Second Working Conference on Reverse Engineering, (Toronto, Canada, 1995). 86-95.

[8] Balazinska, M., Merlo, E., Dagenais, M., Lague, B. and Kontogiannis, K. Partial Redesign of Java Software Systems Based on Clone Analysis. in Sixth Working Conference on Reverse Engineering, (Atlanta, USA, 1999). 326-336.

[9] Carpenter, B (Ed.). Architectural Principles of the Internet. RFC 1958, June 1996.

[10] Johnson, J.H. Substring Matching for Clone Detection and Change Tracking. in Proceedings International Conference on Software Maintenance, (Victoria, Canada, 1994). 120-126.

[11] Johnson, J.H. Identifying Redundancy in Source Code using Fingerprints. in CAS Conference.(Toronto, Canada, 1993). 171–183.

[12] Berners-Lee, T. Principles of Design. 1998, last change: January2002.

[13] http://www.w3.org/DesignIssues/Principles.html

[14] Baxter, I.D., Yahin, A., Moura, L., Santa Anna, M. and Bier, L. Clone Detection Using Abstract Syntax Trees. in International Conference on Software Maintenance, (Washington DC, USA, 1998). 368-377.

[15] Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue. Refactoring support based on code clone analysis. In Proc. of the Product Focused Software Process Improvement Int'l Conference,pages 220{233, 2004.

[16] Kamiya, T., Kusumoto, S. and Inoue, K. CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code. IEEE Transactions On Software Engineering 2002, 28(7). 654–670.

[17] S.Bellon, R. Koschke, G. Antoniol, J. Krinke and E. Merlo, Comparison and Evolution of Clone Detection Tools IEEE TSE, 33(9):577-591, 2007

[18] F.V. Rysselberghe and S.Demeyer, Evaluating Clone Detection Techniques. In ELISA, 12pp., 2003.

[19] Rainer Koschke, Raimar Falke, Pierre FrenZel, Clone Detection Usimg Abstract Syntax Suffix Trees- Working Conference on Reverse Engineering 2006.

[20] Xcent,QuickAuction.Available:http://www.xcent.com/products/QuickAuction.htm [Access éd January 2004].

[21] Snitz Forums 2000, Product Specifications and Downloads. Available:http://forum.snitz.com/spec.asp [Accessed January 2004].

[22] Web Wiz Guide, Web Wiz Forums Downloads. Available: http://www.webwizguide.info/web_wiz_forums/forum_download.asp [Accessed January 2004].