International Journal on
"Technical and Physical Problems of Engineering"
(IJTPE)
Published by International Organization of IOTPE

ISSN 2077-3528
IJTPE Journal
www.iotpe.com
ijtpe@iotpe.com

June 2023 | Issue 55 | Volume 15 | Number 2 | Pages 168-174

WAVELETS TRANSFORM FOR SOFTWARE BUG LOCALIZATION

S. Zouairi M.K. Abdi

Department of Computer Science, University of Oran1 Ahmed BenBella, Oran, Algeria
ntssaim@aol.com, abdimk@yahoo.fr

Abstract- Nowadays, the cost of maintaining a large and evolving IT project is constantly increasing and consumes a lot of time and effort of the engineering team. Typically, during the software maintenance step, a bug report is used to investigate the location of a failure. After receiving a bug report, it's appropriate to have an automatic way to point out the files that the developer needs to change to fix the problem. In our paper, we use a bug localization approach, through the Information Retrieval (IR) field, to represent textual data by signals, which will be used later by the wavelet transform, a technique widely used in signal processing, and whose use is still young in the IR field. The results of the experiments conducted on the AspectJ 1.6.1 project affirm the adequacy of the proposed approach. The analyses likewise show that the proposed approach beats the Vector Space Model (VSM).

Keywords: Error Finding, Data Mining, Bug Issue, Wavelets Transform, Program Maintaining, Software Debugging.

1. INTRODUCTION

The term "software crisis", which refers to the general situation that characterized software development failures, appeared in the 1960s and 1970s. There are three main causes of this phenomenon: - The complexity of the software - Excessive user and customer expectations; and - Changes to be made to software products. The paradox in our time, and since the 60's, is the difficulty to produce quality software at a good price and in a reasonable time. The causes of the software crisis are numerous: First, there are reasons related to the essence, to the very nature of software, what we call the essential properties of software, particularly its virtuality. Then there are reasons related to the problems surrounding the production of the software, what we call the accidental properties of the software, for example the needs of the users which are constantly changing, the demand for sophisticated functionalities, the short delivery times, the inexperience of the developers, etc. Famous software bugs (like the explosion of the Mariner space probe in 1962, the Intel Pentium V bug in 1994) show that it is necessary to pay particular attention to software development and its interaction with the environment.

A bug can be defined as the appearance of an anomaly in the software product that prevents it from performing its functions or behave abnormally [1]. Upon the occurrence of a malfunction of an operated program, a report is sent to the technical team with the indications of the error, which is then recorded in the bug database. Verifying the validity of the bug is preceded by several steps, such as its duplicate and validate state, and semantic content. The developer is then credited with the error and examines the data in the report to find the source files that need to be changed to correct the problem. Typically, a description of the bug, a summary of the faulty problem, software version, stack traces, etc. are attached to the components of the bug report. Table 2 provides an illustration.

It is the responsibility of the person receiving the bug to locate and correct the root cause that triggered the failure in the software project. Depending on the size of the software project and the number of source files involved, the manual localization process can take 30-40% of the total time required to fix the problem [2]. As a result, fixing bugs takes longer and the project is more expensive to maintain.

Hence fixing problem task comprises implicit subtasks like "bug understand", "bug validated", "identifying", and "solving " the majority of the developer's time is spent on the bug localization process [3, 4]. From this point of view, bugs localization requires resorting to automated tools or approaches to deal to this problem. The two main categories of current localization methods are spectral-based and information retrieval 'IR'-based [5]. The IR-based strategy evaluates project sources and the vocabulary (i.e., questions) used in bug reports to identify bugs, while spectrum-based approaches may rely on the results of ray runs [5].

The basic principle of these IR methods is that reports are first considered as queries and the source files considered are document collections. After sorting the documents by predicted relevance, the IR approach generates an ordered list of potential error source files [5]. A lot of the recommended methods for IR-based fault localization automatically look for relevant files in connection with issue reports [5-9]. A similarity function serves as the foundation for this IR approaches, which give as result a score to the document according a query and a document passed as arguments. This score represents who much the document is relevant to the query [5].

When representing documents and queries, some strategies use vectors, where each word in the document

set has its own dimension in the vector space [5]. The vector's components are the linked term's frequency of occurrence in the document. To have a score on likelihood of relevance, we apply a similarity function on weights between query and document vectors [10]. To convert a document into vector, we consider only terms appearance number in documents.

It is necessary for many applications to format data into vector, this process is used in multi-dimensional biomedical imaging, video and image processing, to process audio and speech problems [5]. This preprocessing step is necessary to several mathematical methods to achieve our goal. One of these methods are wavelets transform (WT) represents data or other functions by vectors for specific mathematical applications (i.e., signals).

In the literature, the simplest case is the Haar wavelet. In discrete form, it is called Haar transform [11]. Haar transforms were mainly used for pattern recognition and image processing due to their low computational requirements [12], [13]. It should be noted that this method is not frequently employed for IR bug localization.

Wavelet analysis has been applied in numerous publications, such as in integral text information search, software code clone detection [14], document information retrieval [15], and picture processing [22]. data reduction techniques for nonstationary data curves with possibly enormous and complex shapes [16].

One major benefit of using wavelet transform is the computational complexity that is only O(N) multiplications [17]. We can use fewer resources by reducing the amount of project data processed in each transformation by two factors. This justifies the use of wavelet techniques in 'IR'-based fault localization, which is rare in this field.

Our study tries to fulfil this research gap, in order to answer the subsequent main research question: How effective are wavelets transform for bugs localization?

This study evaluates the value of the wavelet transform in locating faults. To do this, we develop a tool called BugLocWT that uses Haar transforms to describe text data as signals and rank relevant files. The remaining sections of the work are structured as follows: section 2 lists related works, section 3 discusses wavelets, the suggested approach, and the methods used to carry out the study. Section 4 presents the findings and subsequent debates, and section 5 brings this study to a close.

2. RELATED WORKS

Generally, automated bug localization approaches fall into two parts: the dynamic approach and the static approach. In dynamic approach, program semantics and information of its execution are used. There are two methods: model-based fault location and "spectrum-based" fault location. This type of strategy is used by Saha et al., and his tool BLUiR (Bug Localization Using Information Retrieval), examines source code structure information such as comments, class names, methods, and variables to improve translation accuracy [9].

The static approaches depend only on the bug reports information and on the program code, and can be arranged in two classes: IR-based and program analysis. Bug localization with program analysis-based approach need predefined bug template. Hovemeyer, et al. [18] proposed a model named FinBugs using this approach

Techniques like Naive Bayes, VSM, rVSM, TF-IDF, LSA and LDA [19], can be contained in the IR-based type or Machine Learning approaches. These techniques implement Learning-To-Rank IR issue for the bug localization question.

Rao, et al. [6] adopt for bug localization the VSM methods. The cosine similarity between phrases in the documents is the key concept behind this technique. BLIA (Bug Localization with Integrated Analysis) is a program created by Youm, et al. [20] that uses the text and stack traces in bug reports and source code change histories. The tool BugLocator was introduced by Zhou, et al. [21], by doing an automatic search for pertinent files under an initial bug report. file ranking, this tool explores the revised Vector Space Model (rVSM).

The present work is in line with the previous work and proposes the tool called BugLocWT, it borrows from signal processing field that map textual data to signals. To achieve this, we use a mathematical theory of the signal called the wavelet transform, which is currently largely unused in the IR range.

3. METHODS

In the following section, we will not dwell on the theory of wavelets, but give an overview of this theory. For a deeper insight, the reader is directed to the literature [22], [11], [23].

3.1. The Haar Wavelets Transform

The wavelet transform is a tool that cuts up data or functions or operators into different frequency components, and then studies each component with a resolution matched to its scale.

Historically, the first orthonormal wavelet basis is the Haar basis, constructed long before the term "wavelet". It was developed by the mathematician Alfred Haar, and have been in use since 1910 [24], for additional information, see wavelets transform [23], [25], [22]. The Haar transform is a mathematical operation that is related to Haar wavelets in discrete form. All other wavelets transforms are based on the Haar transform.

The key words for the Haar transform, defined in section 3.3, are presented in this part, along with examples of how the Haar transform is commonly applied for bug localization.

In this work, we projected vectors (functions of time with values occurring at discrete points in time) as discrete signals. A discrete signal can usually be represented as S= (s_1, s_2…, s_N), where N is a positive even integer equal to the length of S. S's values s_1, s_2…, s_N are logically real numbers. These values are typically measurements of an analogue signal g taken over time. In short, the values of S are:

$$s_1 = w(t_1)\ ,\ s_2 = w(t_2)\ ,...,\ s_N = w(t_N) \quad (1)$$

The Haar transform proceeds to the decomposition of a discrete signal S by generating two signals of half its length. The first resulting signal is formed by the averages, the second is formed by the differences, resulting from the decomposition.

For the signal S, we calculate the first trend (average or mean) sub signal $a^1 = (a_1, a_2, ..., a_{N/2})$ by taking a running average as follows:

Its initial value a_1, is calculated by averaging the first two values of *S* which is $(s_1+s_2)/2$, and then multiplying it by $\sqrt{2}$, the result is $a_1 = (s_1 + s_2)/\sqrt{2}$. In this manner of iteration, all values of a^1 are calculated by averaging successive pairs of values of S multiplied by $\sqrt{2}$.

A general equation for the values of a^1 is

$$a_m = \frac{(s_{2m-1} + s_{2m})}{\sqrt{2}} \quad (2)$$

for *m*=1, 2, …, *N*/2

For example, suppose *S* is defined by *S*= (8, 8, 4, 0, 5, 3, 7, 1); Then its first average sub signal is $a^1 = (8\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 4\sqrt{2})$.

The first fluctuation is the second sub signal (difference). It is possible to calculate the signal S's first fluctuation, designated by $d^1 = (d_1, d_2, ..., d_{N/2})$, by taking a running difference in the manner shown below. The first term d_1 is calculated by taking the difference between the first half pair of $S = (s_1 - s_2)/2$ values and multiplying it by $\sqrt{2}$. The subsequent value is $d_2 = \frac{(s_3 - s_4)}{\sqrt{2}}$, and so on. Thus, all d^1 values are calculated using the Equation (3):

$$d_m = \frac{(s_{2m-1} - s_{2m})}{\sqrt{2}} \quad (3)$$

for *m*=1, 2, …, *N*/2

For example, for the signal *S*= (8, 8, 4, 0, 5, 3, 7, 1) considered above, its first fluctuation d^1 is $(2\sqrt{2}, 2\sqrt{2}, \sqrt{2}, 3\sqrt{2})$.

3.2. Haar Transform Multiresolution

Several decompositions of the Haar transform are carried out (stages, or levels). The mapping H_1, which is the first decomposition, is described by:

$$S \xrightarrow{H_1} \left(a^1 \middle| d^1\right) \quad (4)$$

It follows that the Haar transform 1-level for the signal *S*= (8, 8, 4, 0, 5, 3, 7, 1) is as showed above

$$(8,\ 8,\ 4,\ 0,\ 5,\ 3,\ 7,\ 1) \xrightarrow{H_1} \xrightarrow{H_1} (8\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 4\sqrt{2} \,|\, 2\sqrt{2}, 2\sqrt{2}, \sqrt{2}, 3\sqrt{2})$$

The next 2-level for the signal *S*, which is $S' = (10, 8 \,|\, 0, -2)$, are obtained by repeatedly using the Equations (2) and (3). Repeat this method once more until the entire decomposition is obtained, as shown in Table 1.

Table 1. decomposition process results of the Haar wavelet for signal *S*

Level	Averages	differences
1	$(8\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 4\sqrt{2})$	$(2\sqrt{2}, 2\sqrt{2}, \sqrt{2}, 3\sqrt{2})$
2	(10, 8)	(0, -2)
3	$(9\sqrt{2})$	$(\sqrt{2})$

We can carry on with this process up to level P, where there is one average and one difference coefficient, if the length *S* is $N=2^P$ (in the example shown P=3). Lastly, *S*'s wavelet transform is $\left(9\sqrt{2}, \sqrt{2}, 0, -2, 2\sqrt{2}, 2\sqrt{2}, \sqrt{2}, 2\sqrt{2}\right)$, this process is called multiresolution analysis.

3.3. Proposed Methodology

In order to use the AspectJ Eclipse (www.eclipse.org/aspectj/) project for empirical analysis, we need to prepare data for both source code files and bug reports. This pretreatment has five phases. File assembly, corpus building, indexing, query formulating, retrieve and rank step.

Data preprocessing is required for both the source code files and the bug reports in order to use the AspectJ Eclipse (https://www.eclipse.org/aspectj/) project for the empirical evaluation. This preprocessing entails five steps: files assembly, corpus building, indexing, query formulation, and retrieval and ranking. In general, the structure of BugLocWT for bug localization is shown in Figure 1. An illustration of the IR bug localization strategy is provided in this section through an example. A real bug report (ID: 28974) for AspectJ version 1.6.1 can be seen in Table 2.

3.3.1. Module Description

3.3.1.1. Files Assembling

In order to speed up the bug-finding process, we removed all comments from files and group them in one repository.

3.3.1.2. Corpus Creation

Word tokenization, a useful function from Natural Language Processing, is performed for creating corpora; for further information, see [26], Each source code file undergoes a lexical analysis to provide a vector of lexical tokens. Some tokens are eliminated since they are used by all programs, including separators, operators, and keywords (such as int, double, char, etc.). The removal of English "stop words" (such as "a," "the," and similar words), punctuation, whitespace, tabs, and carriage return, this lowers the volume of data that is converted.

Table 2. A bug report for bug id 28974
(https://bugs.eclipse.org/bugs/show_bug.cgi?id=28974)

BugID	28974
resume	Compiler error when introducing a ""final" " field
Status	RESOLVED FIXED
Reported	2003-01-03 10:28 EST by Adrian Colyer
Product	AspectJ
Component	Compiler
Version	1.6.1

Description	The aspect below fails to compile with 1.1b2, producing the compilation error: ---------- $ ajc com/ibm/amc/*.java com/ibm/amc/ejb/*.java d:/eclipse/runtime-workspace-ajsamples/MockEJBs/com/ibm/amc/DemoBeanEJB.java:1: Cannot assign a value to the final field com.ibm.amc.DemoBean.ajc$interField$com_ibm_amc$verbose !! no source information available!! 1 error ---------- package com.ibm.amc; import com.ibm.amc.ejb.SessionBean; /** * @author colyer ---------- Making the inter-type declaration non-final solves the problem...
Fixed	2003-01-14 14:30 EST
Fixed files	org.aspectj/modules/weaver/src/org/aspectj/weaver/AjcMemberMaker.java

3.3.1.3. Indexing

All of the files in the corpus are indexed once it has been produced. To do this, each file is given a length (term's number), and the entire collection of AspectJ source code files is then indexed using a top-down method.

3.3.1.4. Query Construction

In order to create the query vector, which will be used to look for pertinent files. The vector query is created by word tokenizing the bug's name and text that describe it in the same way as was done while building the corpus.

3.3.1.5. Retrieval and Ranking

When scores for each file in the corpus have been determined, we use the Haar transform of the query and each file to retrieve and rank the relevant buggy files. The bag of words paradigm is applied to any collection of documents [27]. The sequence of the terms in a text is disregarded in this stage, but the frequency with which each term appears is counted. Application of the Haar transform on a file resulted in a vector with components corresponding to each keyword in the query. The process of calculating the score is modeled around [27]'s chapter 6, section 6.3.3. The score of a file is determined by dividing the total number of vector components created by the Wavelets transforms by the vector's length.

3.4. How to Localize Bugs with Wavelets

Here, we explain the approach used for locating the relevant items files that match a specific query.

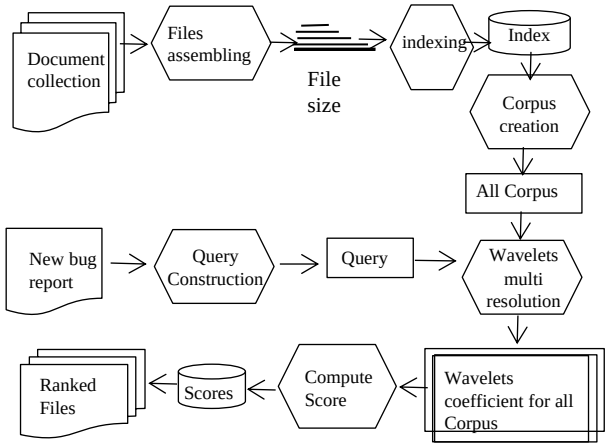

Figure 1. Structure of BugLocWT

3.4.1. Pseudo Algorithm

The actual study, BugLocWT was developed using Python 3.6. that includes tools for programming computers to comprehend human language and respond appropriately.

Algorithm 1. Pseudo Algorithm

BeginAlgorithm
1- Token vector creation
$k=1$
Do while $k \le m$
calculate $vect\_File_k[w_1, w_2, ..., w_i, ..., w_n]$
$k=k+1$
EndDo
w_i keyword $\in File_k$
2 - Creating pertinent vector for query vector tokens for each file
For j=1, m, compute vect_ReqFilej [Tfwq1, Tfwq2, .., Tfwqi,.., Tfwqn]
Where: $Tf_{wq_i} = \begin{cases} Termfequency & \text{if } w_i \in vect\_query \\ 0 & \text{otherwise} \end{cases}$
$wq_i \in File_k$
3- Applying wavelets transforms
$p=1$
Do wile $p \le m$
calcultate wavedec (vect_ReqFile$_p$)
$p = p+1$
EndDo
according to equations (2) and (3)
4- Scoring all m documents
$t=1$
Do while $t \le m$
$Score\_File_j = \sum \left| Haar\_Transform\left(vect\_\mathrm{Re}qFile_j\right) Components \right|$
$t=t+1$
EndDo
EndAlgorithm

Some of these packages include tokenization, stemming, lemmatization, punctuation, character count, and word count. PyWavelets using the Python programming language is available for free and is Open Source.

3.5. Approach Validation with Metrics

The following metrics were used, in order to judge the effectiveness of the proposed bug tracking method:

- Top *N* Rank, represents the number of bugs whose associated files are ranked at the top *N* (*N*= 1, 5, 10, 20) of returned results. Given a bug report, if the first N results of the query contain at least one file to which the bug should be fixed, we consider that the bug is localized. The higher the value of the metric, the better the bug localization performance.
- MRR (Mean Reverse Rank), is a measure that allows the evaluation of a process generating a list of possible answers to a query. For a set of query *Q*, it is given by Equation (5):

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \quad (5)$$

The higher the MRR value, the better the bug localization performance.

• MAP (Mean Average Precision), which provides a single digit measure of information retrieval quality. The average precision of a query (AvgP) is the average of the precision values obtained for the query, which is calculated as follows:

$$avgP = \frac{1}{GTP}\sum_{i=1}^{M} P@i \times rel_i \quad (6)$$

where,

GTP: Ground True positive
P@i: precision at rank *i*
rel_i: rank of the *i*th relevant document
M: number of returned documents
Then *MAP* for a set of query *Q* is:

$$MAP = \frac{\sum_{i=1}^{|Q|} avgP_i}{|Q|} \quad (7)$$

Table 3. Top ranking pertinent files related to bug report ID 28974 with BugLocWT tool

Rank	File name	Score
1	D:/Data1_DW/file_copied\AjcMemberMaker.java	0.7092239
2	D:/Data1_DW/file_copied\FieldGet.java	0.58876
3	D:/Data1_DW/file_copied\BcelField.java	0.4244899
4	D:/Data1_DW/file_copied\ConditionalFlowInfo.java	0.3417179
5	D:/Data1_DW/file_copied\DOMField.java	0.3270305
6	D:/Data1_DW/file_copied\SelectionRequestor.java	0.299778
7	D:/Data1_DW/file_copied\FieldGetCall.java	0.29438
8	D:/Data1_DW/file_copied\JobManager.java	0.2861517
9	D:/Data1_DW/file_copied\ProblemReporter.java	0.2768924
10	D:/Data1_DW/file_copied\CodeSnippetScope.java	0.2675704
11	D:/Data1_DW/file_copied\CompletionEngine.java	0.2663975
12	D:/Data1_DW/file_copied\QualifiedNameReference.java	0.2385462
13	D:/Data1_DW/file_copied\LazyClassGen.java	0.2277787
14	D:/Data1_DW/file_copied\AjcCompilerAdapter.java	0.2262577
15	D:/Data1_DW/file_copied\SetContainerOperation.java	0.2218946
16	D:/Data1_DW/file_copied\ConstructorLocator.java	0.2180203
17	D:/Data1_DW/file_copied\Ajc.java	0.2147939
18	D:/Data1_DW/file_copied\AjTypeImpl.java	0.2094218
19	D:/Data1_DW/file_copied\FieldSignatureImpl.java	0.2061156
20	D:/Data1_DW/file_copied\UserLibraryClasspathContainerInitializer.java	0.2061156
21	D:/Data1_DW/file_copied\ExactAnnotationFieldTypePattern.java	0.1968944
22	D:/Data1_DW/file_copied\TypeDeclaration.java	0.1884213
23	D:/Data1_DW/file_copied\AddJarFileToIndex.java	0.1838055
24	D:/Data1_DW/file_copied\SetVariablesOperation.java	0.1664209
25	D:/Data1_DW/file_copied\InstructionShort.java	0.1648528
26	D:/Data1_DW/file_copied\EclipseResolvedMember.java	0.1635153
27	D:/Data1_DW/file_copied\InterTypeMemberFinder.java	0.1584026
28	D:/Data1_DW/file_copied\JavadocRunner.java	0.1583571
29	D:/Data1_DW/file_copied\BasicSearchEngine.java	0.1581298
30	D:/Data1_DW/file_copied\JRockitAgent.java	0.1545867

DCG (Discounted Cumulative Gain) measures the utility or gain of a document based on its position in the result list. The gain is accumulated from the top of the result list downwards, with the gain of each result reduced to the lower ranks. DCG focuses on highly relevant documents that appear early in the results list using the logarithmic scale for reduction. DCG is the measure of document classification quality. It is primarily used in information retrieval problems such as measuring the effectiveness of the search engine algorithm in ranking the articles it displays based on their relevance in terms of the search keyword. It is given with Equation (8):

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \quad (8)$$

4. RESULTS AND DISCUSSIONS

4.1. Evaluation

The 1844 files of the AspectJ 1.6.1 project were used to evaluate the performance of BugLocWT. Table 4 displays the proportion of Top N using both the Haar Wavelets Transform (WT) and the VSM.

Table 4. Applying WT and VSM to rank pertinent files on AspectJ project

	Top 5%	Top 10%	Top 15%	Top 20%	Top 30%
Haar transform	13.33	36.66	63.33	76.66	90
VSM	0	3.33	13.33	20	53.33

According to Table 4, the top 5, top 10, top 15, and top 20 files with BugLocWT are significantly higher than those with VSM technique. It results those wavelets provided a better position for pertinent; this allows to help the team project in locating the troubling files and then eliminating the bug that has occurred.

According to Table 2, the "AjcMemberMaker.java" file is the subject of bug report ID 28974. The latter is listed as the most important file in position one Table 3.

Additionally, as shown in Figure 2, each bug complaint addressed by our tool BugLocWT has a location of the relevant file that is fewer than 30. This guarantees the

software maintenance project will be time and cost efficient.

➢ RQ: How well does the wavelets transform work for pinpointing bugs? Table 4 shows relevant file rank score with their bugs using Wavelet and VSM. According to the experimental findings, the proposed WT approach works better than the VSM method. For instance, in our AspectJ experiment, we discovered that the relevant source files for only (13.33%) of the top 5 results, (36.66%) of the top 10 results, and (63.33%) of the top 15 results were returned utilizing the proposed WT technique. While those values are (0%), (3.33%), and (13.33%) accordingly with the VSM technique.

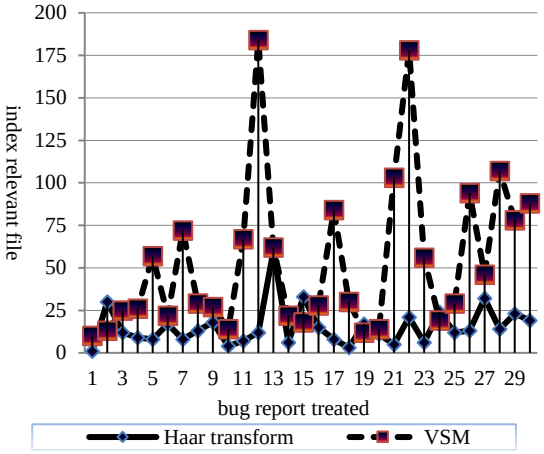

Figure 2. relevant file rank score with their bugs using Wavelet and VSM

Since our tool use IR base bug localization techniques, it is suitable to validate our approach with some metrics. In Table 5, we can notice that the MRR values following the wavelet localization approach is better than the one following the Vector Space models.

Table 5. The effectiveness of bug localization using VSM and WT

Method	MRR	MAP	DCG/Top50
WT	0.09192124	0.04974732	64
VSM	0.05064554	0.04592643	44

Moreover, we can see this difference by a visualization of Figure 3, which shows in the histogram diagram a clear difference in the scales of the values MRR and MAP of the localization via wavelets and the vector model.

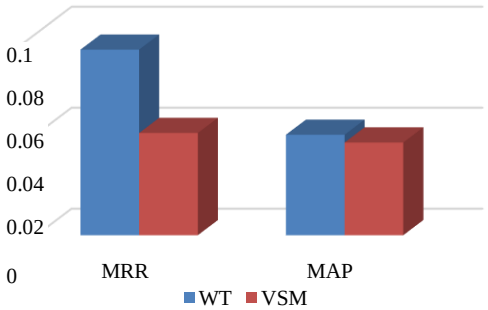

Figure 3. MRR and MAP values WT and VSM

When talking about important relevant files retrieved, it is normally suitable to have these document in top list, these is what Discounted Cumulative Gain (DCG) metric can provide. As shown in Table 5, DCG/Top50 value is 64 according to the Wavelets bug localization approach, which is significantly better than DCG value according to VSM approach. Figure 4 provides a screenshot of these evaluation, as show how bug localization with wavelet is useful compared with VSM technic.

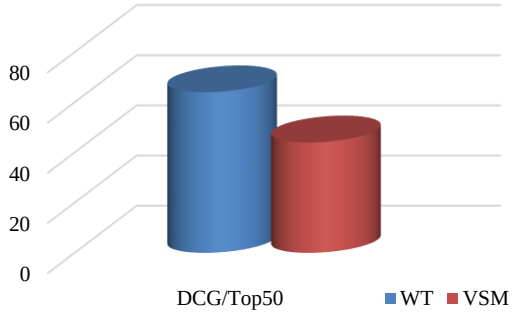

Figure 4. DCG values WT and VSM

5. CONCLUSION

Throughout the use of software, users are confronted with numerous bugs. When the engineering team receives a new bug report, the person who was assigned the report has to identify the files that need to be modified to fix the bug. However, this task will consume a lot of time to find the files to be modified, especially if it is done manually and for a large project.

This work proposes a bug localization tool named BugLocWT, to classify the relevant files from the initial bug report. The contribution is based on information extraction, using Wavelet Transforms. Experimental results on AspectJ project show that the BugLocWT tool allows a significant localization compared to the one based on vector space models (VSM). Future research will examine the utility of additional Wavelet Transforms in IR to enhance this theory's effectiveness in the areas of data mining and bug localization.

REFERENCES

[1] E.J. Braude, M.E. Bernstein, "Software Engineering: Modern Approaches", Waveland Press, p. 802, 2016.

[2] S.S. Murtaza, A. Hamou Lhadj, N.H. Madhavji, M. Gittens, "An Empirical Study on the Use of Mutant Traces for Diagnosis of Faults in Deployed Systems", Journal of Systems and Software, Vol. 90, pp. 29-44, 2014.

[3] K.H. Chang, V. Bertacco, I.L. Markov, "Simulation-Based Bug Trace Minimization with BMC-Based Refinement", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Issue 1, Vol. 26, pp. 152-165, 2006.

[4] S. Kottam, V. Paul, "Soft Set Based Approach for Mining Frequent Item Sets", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 45, Vol. 12, No. 4, pp. 50-56, December 2020.

[5] Z. Shi, J. Keung, K.E. Bennin, X. Zhang, "Comparing Learning to Rank Techniques in Hybrid Bug

Localization", Applied Soft Computing, Vol. 62, pp. 636-648, 2018.
[6] S. Rao, A. Kak, "Retrieval from Software Libraries for Bug Localization: A Comparative Study of Generic and Composite Text Models", The 8th Working Conference on Mining Software Repositories (ACM), pp. 43-52, 2011.
[7] T.D.B. Le, R.J. Oentaryo, D. Lo, "Information Retrieval and Spectrum-Based Bug Localization: Better Together", The 10th Joint Meeting on Foundations of Software Engineering, pp. 579-590, 2015.
[8] S. Wang, D. Lo, "Version History, Similar Report, and Structure: Putting them Together for Improved Bug Localization", The 22nd International Conference on Program Comprehension, pp. 53-63, 2014.
[9] R.K. Saha, M. Lease, S. Khurshid, D.E. Perry, "Improving Bug Localization Using Structured Information Retrieval", The 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 345-355, 2013.
[10] H. Taher, M. Abdulameer, B. Mahdi, "Information Retrieval Scheme Via Similarity Technique", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 51, Vol. 14, No. 2, pp. 375-379, June 2022.
[11] J.S. Walker, "A Primer on Wavelets and their Scientific Applications", Chapman and hall/CRC, 2008.
[12] F.B.J. Stankovic, S. Radomir, "The Haar Wavelet Transform: Its Status and Achievements", Computers and Electrical Engineering, Vol. 29, pp. 25-44, 2003.
[13] N. Shyamala, S. Geetha, "Improved Integer Wavelet Transform (IIWT) Based Medical Image Compression Method", International Journal on Technical and Physical Problems of Engineering (IJTPE), Issue 51, Vol. 14, No. 2, pp. 339-346, June 2022.
[14] S. Karus, K. Kilgi, "Code Clone Detection Using Wavelets", The IEEE 9th International Workshop on Software Clones (IWSC), pp. 8-14, 2015.
[15] M.Y. Dahab, M. Kamel, S. Alnofaie, "An Empirical Study of Documents Information Retrieval Using DWT", Intelligent Natural Language Processing: Trends and Applications, Springer, pp. 251-264, 2018.
[16] M.K. Jeong, J.C. Lu, X. Huo, B. Vidakovic, D.L. Chen, "Wavelet-Based Data Reduction Techniques for Process Fault Detection", Technometrics, Vol. 48. pp. 26-40, 2006.
[17] O. Maimon, L. Rokach, "Data Mining and Knowledge Discovery Handbook", Springer-Verlag, p. 1306, 2005.
[18] D. Hovemeyer, W. Pugh, "Finding Bugs is Easy", ACM SIGPLAN notices, Issue 12, Vol. 39, pp. 92-106, 2004.
[19] R. Khoury, C. Drummond, "Advances in Artificial Intelligence" The 29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Springer, Vol. 9673, p. 362, Victoria, BC, Canada, May-June 2016.
[20] K.C. Youm, J. Ahn, J. Kim, E. Lee, "Bug Localization Based on Code Change Histories and Bug Reports", Asia-Pacific Software Engineering Conference (APSEC), pp. 190-197, 2015.
[21] J. Zhou, H. Zhang, D. Lo, "Where Should the Bugs be Fixed? More Accurate Information Retrieval-Based Bug Localization Based on Bug Reports", The 34th International Conference on Software Engineering (ICSE). pp. 14-24, 2012.
[22] A. Graps, "An Introduction to Wavelets", IEEE Comput. Sci. Eng., Issue 2, Vol. 2, pp. 50-61, 1995.
[23] D.F. Walnut, "An Introduction to Wavelet Analysis", Springer Science and Business Media, pp. 115-140, 2013.
[24] R.S. Stankovic, B.J. Falkowski, "The Haar Wavelet Transform: Its Status and Achievements", Computers and Electrical Engineering, Issue 1, Vol. 29, pp. 25-44, 2003.
[25] Z. Abba, P. Rain, "A Study on Applications of Wavelets to Data Mining", International Journal of Applied Engineering Research, Issue 12, Vol. 13, pp. 10886-10896, 2018.
[26] C.D. Manning, H. Schutze, "Foundations of Statistical Natural Language Processing", MIT Press, p. 704, 1999.
[27] C. Manning, P. Raghavan, H. Schutze, "Introduction to Information Retrieval", Cambridge University Press, p. 506, 2008.

BIOGRAPHIES

Name: **Saim**
Surname*:* **Zouairi**
Birthdate: 09.02.1973
Birth Place: Oran, Algeria
Bachelor: Computer Engineer, Department of Computer Science, University of Sciences and Technology of Oran, Oran, Algeria, 1996
Master*:* Risk and Materials Sciences, University of Oran, Oran, Algeria, 2009
Doctorate: Student, Computer Science, Department of Computer Science, University of Oran1 Ahmed BenBella, Oran, Algeria, Since 2015
Research Interest: Software Testing and Debugging, Natural Language Processing, Reverse Engineering

Name: **Mustapha Kamel**
Surname: **Abdi**
Birthdate: 04.03.1966
Birth Place: Oran, Algeria
Bachelor: Computer Science, Computer Engineering, Computer Science Department, Oran1 University, Oran, Algeria, 1990
Master: Computer Science/ Master of Computer Science, Computer Science Department, Oran1 University, Oran, Algeria, 1995
Doctorate: Computer Science, Computer Science Department, Oran1 University, Oran, Algeria, 2007
The Last Scientific Position: Prof., Computer Science Department, Faculty of Exact and Applied Sciences, Oran1 University, Oran, Algeria, Since 2020
Scientific Publications: 25 Papers, 3 Projects, 6 Thesis
Scientific Memberships: Member of Program Committees of Several National and International Conferences