ISSN: 2278 – 1323
*International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*
*Volume 2, Issue 4, April 2013*

# Quantitative Code Review based on Weighted Checklist

**Amol Sharma, Ajay Rana**

***Abstract-*** **Code Review is a useful technique for detecting defects in the code. It can considerably enhance quality and reduce effort during testing. An important part of the code review process is the use of a checklist to examine the program for common defects. A checklist is a cross reference list stated in the form of questions, concentrates on providing hints to enable reviewers detecting the defects. In this paper a technique for code review is proposed that makes use of a weighted checklist, a checklist in which weights are assigned to different checklist items. Using this technique, a flavor of quantification can be given to code review by computing a code review metric termed Code Review Score (CRS) on the basis of compliance of code under review for a checklist item and the weight assigned to the respective checklist item. It is proven that information expressed in quantitative terms is always more effective to be put into use for analysis, comparison and verification purposes so is the objective of proposing this technique to obtain the code review results in quantities that can be used to manage and track the detection of defects more effectively.**

***Index Terms-*** **Quantitative Code Review, Defect, Weighted Checklist, Metric, CRS.**

## I. INTRODUCTION

Code review is quite effective non computer based testing (“human testing” [1]) technique in finding errors. So every software project is recommended to use this technique. Code reviews are usually held after the code has been successfully completed and other form of static tools have been applied but before any testing has been performed [2]. Being non computer based in nature one might be skeptical about the usefulness of code reviews but it is well proven that they substantially contribute to productivity and reliability because of the fact that earlier errors are found, the lower the cost of correcting the errors.

Checklist is an indispensable aid that assists in performing code reviews. In manual review, a reviewer detects visible defects using project-specific checklist. Here checklist

- Summarizes previous experience in defect detection.
- Stated in question forms concentrating on hints, enabling the detection of defects.
- Managed by men [3].

Despite the fact that doing reviews is the most important step you can take to improve your software engineering performance [4], but this technique still face challenges in industry due to several reasons. One of the main reasons for this is the lack of quantification in the code review process, this technique is still more or less subjective and that does not go with the modern software engineering practices. By incorporating quantitative measures into code review process we can derive enormous benefits from this technique. This paper focuses on this quantitative approach for performing code reviews using weighted checklists.

This paper is structured as follows: section II describes what a weighted checklist is meant for and how it is prepared for the quantitative code review process. Section III describes the quantitative code review process itself, making use of weighted checklists. Section IV discusses the implementation technique of the process presented in section III. Section V lists the benefits and challenges of using this technique and, finally section VI concludes the paper.

## II. WEIGHTED CHECKLIST

Code Review method mainly has three stages: *organization* (selecting the participants, planning, etc.), *fault detection* (performing the actual review) and *completion* (reporting the faults and checking the work done) [5]. During the detection stage most descriptions [6], [7], [8] distinguish between an individual phase and a group phase. During the individual fault detection phase, Fagan [6] mentions the use of checklists to help inspectors focus their efforts on the most worthwhile parts of the document. Single-inspector phases as mentioned by Knight [9] make use of a rigorous checklist, with the reviewer deciding whether the document does or does not comply with each item.

So checklist is slightly extended in the code review technique described in this paper. Here every checklist item has an assigned weight based on the relative impact of its compliance in the code under review. Each item needs to be rated on a scale of 1 to 5 where 1 signifies *incidental* impact where as 2 signifies *moderate*, 3 signifies *average*, 4 signifies *significant* and 5 signifies *essential*. This scale is derived from the one that is used to calculate the Complexity Adjustment Factor in Function Point Analysis [10].

Here a sample checklist is presented having 10 checklist items with corresponding weights:

*Table I*: Sample Weighted Checklist

| **S.No.** | **Checklist Item** | **Weight** |
|---|---|---|
| 1 | Does a referenced variable have a value that is unset or uninitialized? [1] | 4 |
| 2 | Do all variables and different program elements have meaningful names? | 1 |
| 3 | For all array references, is each subscript value within the defined bounds of the corresponding dimension? [1] | 3 |
| 4 | Are there any mixed-mode computations? [1] | 5 |
| 5 | Is it possible for the divisor in a division operation to be zero? [1] | 5 |
| 6 | Are the comparison operators correct? [1] | 3 |
| 7 | Are there any off-by-one errors, such as one too many or too few iterations? [1] | 3 |
| 8 | Does the number of parameters received by this module equal the number of arguments sent by each of the calling modules? [1] | 4 |
| 9 | Do the attributes (e.g., data type and size) of each parameter match the attributes of each corresponding argument? [1] | 3 |
| 10 | Is there any unreachable code? | 2 |

Same checklist is used in the description of quantitative code review process in the next section.

## III. QUANTITATIVE CODE REVIEW PROCESS

Once the weighted checklist is prepared and we have our code segment to be reviewed with us, Quantitative Code review process can be initiated as per the following algorithm:

***Algorithm***

```
QCR (Code Segment, Checklist: with weights assigned to all checklist items)

TotalWeight: =0
ScoreGain: =0
CodeReviewScore: =0

For each statement in the code segment do
   For each applicable checklist item do
     If statement is compliant with checklist
     item then
        ScoreGain: = ScoreGain + Weight of
        the corresponding checklist item
     End If
     TotalWeight: = TotalWeight + Weight of
     the corresponding checklist item
   End For
End For
CodeReviewScore: = (ScoreGain /
TotalWeight) *100
Output CodeReviewScore
```

In this algorithm we have used three variables namely TotalWeight, ScoreGain and CodeReviewScore. The purpose of each of the following is explained below:

***TotalWeight:*** This variable containes the total weight of all those checklist items against which code is checked in the entire process.

***ScoreGain:*** This variable hold the total weight of only those checklist items for which code segment under review is compliant with.

***CodeReviewScore:*** This variable contains the final outcome of the process i.e. the percentage score gained by the code segment under review.

In the next section, the implementation technique for the above specified algorithm is given so as to support the understanding of how to apply this algorithm when performing the actual code review on a code piece.

## IV. IMPLEMENTATION TECHNIQUE

To improve the understanding of the entire process, one sample java code is taken, on which quantitative code review algorithm will be applied to obtain the code review score of the sample code
The code is given below:

```
1 class Student
{
2         String nm;
3         int marks[]=new int[3];
4         int mm=300;

5         Student(String n,int m1,int m2,int
m3,int m)
          {
6                  nm=n;
7                  marks[0]=m1;
8                  marks[1]=m2;
9                  marks[2]=m3;
10                 mm=m;
          }
11        String get()
          {
12                 return(nm);
          }
13        double getPercentage()
          {
```

```
14
        return(((marks[0]+marks[1]+marks[
2])/mm)*100);
        }
}
15 class StudentMain
{
16      public static void main(String args[])
        {
17              Student s = new
Student("Amol",30,40,20,300);
18              System.out.println("Name:
"+s.get());
19      System.out.println("Percentage:
"+s.getPercentage());
        }
}
```

For the sake of identifying each statement individually, unique numbers have been assigned to each program statement.

After applying quantitative code review algorithm, following results are obtained:

*Table II:* Code Review Results obtained after applying Quantitative Code Review Algorithm

| S.No. | Statement No | Applicable Checklist Item No | Weight | Is Compliant? | Score Gain |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | YES | 1 |
| 2 | 2 | 2 | 1 | NO | 0 |
| 3 | 3 | 2 | 1 | YES | 1 |
| 4 | 4 | 2 | 1 | NO | 0 |
| 5 | 5 | 8 | 4 | YES | 4 |
| 6 |  | 2 | 1 | NO | 0 |
| 7 | 6 | - | - | - | - |
| 8 | 7 | 3 | 3 | YES | 3 |
| 9 | 8 | 3 | 3 | YES | 3 |
| 10 | 9 | 3 | 3 | YES | 3 |
| 11 | 10 | - | - | - | - |
| 12 | 11 | 2 | 1 | NO | 0 |
| 13 |  | 8 | 4 | YES | 4 |
| 14 | 12 | 1 | 4 | YES | 4 |
| 15 | 13 | 2 | 1 | YES | 1 |
| 16 |  | 8 | 4 | YES | 4 |
| 17 | 14 | 1 | 4 | YES | 4 |
| 18 |  | 3 | 3 | YES | 3 |
| 19 |  | 4 | 5 | NO | 0 |
| 20 |  | 5 | 5 | NO | 0 |
| 21 | 15 | 2 | 1 | YES | 1 |
| 22 | 16 | 2 | 1 | YES | 1 |
| 23 |  | 8 | 4 | YES | 4 |
| 24 |  | 9 | 3 | YES | 3 |
| 25 | 17 | 2 | 1 | NO | 0 |
| 26 |  | 9 | 3 | YES | 3 |
| 27 | 18 | 9 | 3 | YES | 3 |
| 28 | 19 | 9 | 3 | YES | 3 |
| **TOTAL** |  |  | **68** |  | **53** |

In order to understand that how these entries in the table have been made, let us take an example of program statement number 14 in the sample code. For this statement, the checklist items 1, 3, 4, and 5 are applicable as this statement involves:

- *Variables*, that should be checked for initialization.
- *Array references*, for which subscript value should be checked to be within the defined bounds of the corresponding dimension.
- *Computation*, that should be checked for being mixed-mode.
- *Division Operation* that should be checked for the possibility of the divisor in a division operation to be zero.

The weights of the checklist items 1, 3, 4, and 5 are 4, 3, 5, and 5 respectively (see Table I).

Out of these four checklist items, the program statement 14 of the code under review is compliant with 1 and 3 as referenced variables are initialized and array references are with in bounds but it is not compliant with 4 and 5 as it involves mixed mode computation and there is a possibility for divisor to be zero if corresponding actual parameter value for mm is assigned to be 0.

So the score gain for checklist item 1, 3, 4, and 5 are 4, 3, 0, and 0 respectively and the total score gain of the statement is 7 out of 17.

Like wise the other code statements are reviewed and remaining entries of the table are filled.

Once we have the table with all the entries done, Code Review Score can be computed by putting values in the following formula:

CRS = (Score Gain / Total Weight) * 100

***CRS = (53 / 68) *    100***
***= 77.94 %***

So by applying this technique, finally we have arrived at a numerical value in terms of code review score that can be used to arrive at a conclusion whether this score is satisfactory or do we have to change the code for the checklist items for which code is found to be non compliant and then re review it.

## V. BENEFITS AND CHALLENGES

It is said that software reviews “purify” software engineering work products [11]. So code review process described above may also offer ample benefits if carried out with proper attention and care. Some of the *benefits* are listed below:

1) As the effect of compliance/ non compliance of the code to all the checklist items is not same in terms of the impact and importance of the checklist items, so the reflection of the same will be reflected in the results obtained by applying this technique.
2) Quantitative results are always better candidate for the evaluation, comparison and analysis rather than the subjective results.
3) As the review process requires follow up for the correction of detected defects, so once the corrections are done and the code is re reviewed then, by comparing the code review scores of the original and corrected code, one can verify that corrections are properly done or not.
4) As the code can’t be re reviewed again and again, so code review score can be used as the termination criteria in a way that if the code has obtained a specified threshold code review score then there is no need to go for another review.
5) Code review score can be used as the effective metric to know about the efficiency of the programmers.

6) Code review score can be used as the evaluation criteria to get the effectiveness of the review process itself.

Apart from the benefits that this technique offers, there are some *challenges* in the implementation of this technique:

1) As the process of this technique requires rigorous and careful understanding of various terms, one might find it difficult to understand, but if understood properly this process is for sure easy to apply and offers significant benefits.
2) Code reviews are always criticized for the high costs so as this process can be, but the cost of detecting and correcting errors later will surely be higher than the cost of carrying out this process and detecting and correcting the errors at the time of review process.
3) As for now this technique is proposed to be carried out manually, so this can be subjected to human errors, but with the help of automation we can overcome this challenge.
4) This technique might take a long to be carried out, but for sure the time taken will be shorter then the time it will take to detect and correct an error at some later point in time during software development or even after that.

Attention must be paid to the above mentioned points so as to enable the decision of moving on with the quantitative code review technique.

## VI. CONCLUSION

In this paper, we have described a code review technique that is quantitative and makes use of weighted checklists.

As described above, this technique allows to first associating an impact factor in terms of weight with the points on which the code needs to be reviewed so that one can make sure that code is compliant with all those checkpoints which are critical, without which code cannot move further in the software development and even for those, which might be less critical in nature but desired to be there in the code under review. But this level of criticality will be reflected in the code review process through the weights being assigned to checklist items. After that, other feature of this technique is its quantitative base, each and every aspect of this technique goes along the quantities being expressed in numeric numbers which are always praised to be more conversant with comparison, analysis and evaluation sort of activities.

So we wish to conclude that by using this technique, early detection and correction of defects can be ensured and that in turn, will result in enhanced quality and customer satisfaction.

## ACKNOWLEDGMENT

No task is a single man’s effort. Co-operation and co-ordination of various people at different levels go into a successful venture. There is a sense of gratitude, which we wish to express for many people for their helpful and needy services they rendered in accomplishing this work but it is impossible to thank every one of them individually but we hereby humbly pay our sincere thanks to all faculty members and lab staff and M.Tech. (CSE) students of 2010 – 2013 batch of Amity School of Engineering and Technology, Noida who have not only given us the valuable inputs but also were always ready with all their resource to help us out in the time of need.

We are thankful to all our reviewers for their valuable suggestions and comments as well.

## REFERENCES

[1] Glenford J. Myers, “The Art of Software Tesing,” Wiley *India*, p. 121, 2009.

[2] Pankaj Jalote, “An Integrated Approach to Software Engineering”, *Narosa Publishing House*, p.384, 2004.

[3] Jun-Suk Oh, Ho-Jin Choi, “A Reflective Practice of Automated and Manual Code Review for a Studio Project”, Fourth *Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pp. 37-42, 2005.

[4] W.H. Humphrey, “A Discipline for Software Engineering”, *Addison-Wesley*, 1995.

[5] Fevzi Belli, Radu Crisan, “Empirical Performance Analysis of Computer-Supported Code Reviews”, *IEEE*, pp. 245-255, 1997.

[6] M.E. Fagan, “Design and code inspection to reduce errors in program development”, *IBM Systems Journal, 15(3)*:*182-211*, 1976.

[7] D. Freedman and G.M, Weinberger. “Handbook of Walkthroughs, Inspections and Technical Reviews”, *Little, Brown*, 1990.

[8] W.S. Humphery, “Managing the Software Process”, *Addison-Wesley, Mass*, 1989.

[9] J.C. Knight and E.A. Myers, “An improved inspection technique”, *Communications of the ACM, 11(11):S1-61*, November 1993.

[10] K.K. Aggarwal, Yogesh Singh, “Software Engineering”, *New Age International Publishers*, p. 146, 2011.

[11] Roger S. Pressman, “Software Engineering, A Practitioner’s Approach”, *McGraw Hill*, p. 416, 2012.

**Dr. Ajay Rana** is Founder Director/ Professor/ Mentor of more than 19 different departments and innovative programs at Amity University, Noida. He is having rich experience of Industry and Academics of more than 13 years. He obtained Ph.D. and Master of Technology in Computer Science and Engineering. He has published more than 108 research papers in reputed journals and proceedings of international and national conferences. He has co-authored 04 books and co-edited 15 conference proceedings.

**Mr. Amol Sharma** is pursuing Master of Technology in Computer Science and Engineering at Amity School of Engineering and Technology of Amity University, Noida. His areas of interest are Educational ERP systems, Software Engineering and Software Testing. He is currently doing his dissertation work at Amity University, Noida.