ISSN: 2278-4632
Vol-10 Issue-12 No.03 December 2020

Utilization of the Particle Swarm Optimization Technique for the Detection of Code-Smells

M. BALACHANDRA
Assistant Professor
Balachandra.1202@gmail.com

N. UPENDRA BABU
Assistant professor
Upendra.alts@gmail.com

V. ARUN KUMAR REDDY
Assistant Professor
Arunkumar.alts@gmail.com

Abstract:

The software development process in software engineering is concerned with the actions performed as intended. Analysis of need, designing, coding, testing, and maintenance are among the tasks. Typically, code smells are found during the coding process. Typically, code smells are discovered during the coding stage. Code smell is a surface sign that often points to a more serious issue inside the system. Code-smell is a sign of a program's incompleteness, which might make it challenging to carry out software system development and maintenance. If odours are ignored, they might result in software problems and mistakes. There are many other odours, including comments, lengthy methods, excessive literals, lengthy parameters, duplicated routines, complicated conditional logic, primitive obsessions, lazy classes, and others. This study primarily focuses on lazy classes that appear during the coding process. Particle Swarm Optimization, one of the optimizers, is used to detect these odours. The optimizer intelligently lowers the odours' search entanglement. By doing this, the software's performance improves and the system's efficacy is attained..

I. INTRODUCTION

II.

It is common for mistakes, defects, and failures to happen while programming computers. There is a chance for mistakes, defects, and even failures to occur when developing a program's source code. Human errors are included among these; programmers may make grammatical or logical errors, either consciously or inadvertently. When mistakes are ignored, they open the door for software flaws, which lead to improper programme operation. Therefore, a software fault is the cause of every software failure. So one should focus on finding the faults in order to prevent software breakdowns.

Code smells aren't really faults or errors, and they don't stop the programme from working. The scent is a clue that something is amiss somewhere in the programme; if the programmer ignores this, mistakes, difficulties, and sometimes more significant problems like software failure result. As a result, early on in testing, code smells should be detected. According to Martin Fowler [1], "code-smell is an obvious sign that often points to a deeper issue in the system." They just serve as tips or warnings of bad code. The phrase "code-smells" has been developed as a means of assisting software engineers in identifying defective programmes and determining when such scripts need refactoring.

Refactoring

Refactoring helps to clean up code problems, make it simpler, uncover errors, cut down on debugging time, and avoid "design delay." Software developers may clean up, make their code more intelligible, and

improve its design without altering its operation by using a process called refactoring[2]. Refactoring is used to enhance software development by enhancing its extensibility, maintainability, and low cost of change. The work that has been done in the past for smelling detection is discussed in Section II. Particle Swarm Optimization is a suggested approach that is discussed in Part III. The next section discusses experimental findings related to the study, Section IV discusses conclusions, and Section V discusses upcoming work. The last section discusses references that are cited.

III. RELATED WORK

Various tactics and methodologies are utilised in this section to identify odours that emerge during the design phase of the software development life cycle. These are listed below.

First method:

The first method describes detection tactics. The "quantifiable statement of a rule by which design fragments that are adhering to that rule may be discovered in the source code," according to Chidamber and Kemerer's[3] definition of detection strategy. As a result, it is a general approach to employing metrics to locate the odours. Metrics essentially refer to the evaluation of software quality. Binary trees and function sets have both been used to identify design-smells using metrics (UNION & INTERSECTION).

Second Approach: The statistical classifiers are discussed in the second approach. Statistical classifiers "can estimate the class membership probabilities, such as the likelihood that a given tuple belongs to a certain class," according to F. Khomh et al. [4] This classifier is also used to identify the BLOB design smell, sometimes known as the GOD class. determining the likelihood that a stakeholder will take into account an anti-pattern for a class. Manual detection has been done based on likelihood. Only BLOB, also known as GOD CLASS, has been found using this method. As its name suggests, the GOD CLASS is a class with an excessive amount of obligations. Here, a directed acyclic network with a probability distribution is known as a Bayesian Belief Network (BBN).

Classifer (C) = "notsmell,"

Here, conditional probabilities are used to teach set operators like OR and AND.

Third Approach: The third approach discusses the search-based strategy. Here, parallel evolutionary techniques are used to identify code odours. To get fitness values for the solutions, two techniques, genetic algorithms[6] and genetic programming[5], are used here in simultaneously. The fitness value of algorithm (a1) in genetic algorithms relies on the fitness value of the (a2). The issue is not that you can smell n different types of codes. The degree and danger of code-smells must be determined. That claims that codes are discovered one by one. One should utilise the accuracy and recall features to acquire the best results. Where recall offers the properly detected scents in the supplied bunch of code-odors that are manually recognised and accuracy provides the number of correctly detected smells in the given batch of detected smells. Each code smell's severity may be determined using the formula severity=(risk + rules detected)/2.

When using a search-based technique, each and every line of source code is searched in order to find any smells[7]. By doing this, the search window will be extended, and there is also the potential to conduct the search even in the absence of odour. Increases in time slots will have an impact on the system's effectiveness. In order to solve this issue, the Particle Swarm Optimization approach is used to identify the few instances of lazy classes. Whereas the complexity of the search process is also reduced by applying particle swarm optimization techniques.

IV. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization is used for detecting the code-smells, actually it is a stochastic optimization technique based on social behaviour of group of birds.

ISSN: 2278-4632
Vol-10 Issue-12 No.03 December 2020

Working of Algorithm

Consider group of birds are searching for a piece of food in an search area. No birdknows where the food is hidden, based on the smell of food birds will start searching in a particular search area. At first, one bird starts searching at a particular search area and if the piece of food is not found at the first iteration, the remaining birds won't go where the first bird had searched. By this, time consumption for searching is benefited. Now in second iteration,if the bird found the food then it starts sounding because to alert the other birds and all other birds will stop searching and go to the bird which found the food. In this process if the food is distributed among other birds it is called as neighbour solution or Global solution, if the food is not shared then it is called as Local solution. Everybird is taken as a particle. How intelligently the birds are searching for the food in each iteration is known here and following is the Pseudo code for PSO[8]

For each Initialization of a smell

Calculate the fitness value

If the current fitness value is greater than previous value

Then

Set current value as new Pbest.

End

Here fitness value is calculated based on one of the metrics called Precision. If the precision range is high, then the method using is correct. Precision for detecting the smells is as follows:

$$Precision = \frac{number\ of\ smells\ that\ have\ been\ acknowledged\ and\ resolved\ by\ the\ user}{total\ detected\ smells}$$

Procedure:

Following is the procedure or method for detecting code-smells by using optimized technique called particle swarm optimization. This technique is used as a detector for identifying code-smells in the program.

Programmer writes source code. The source code which is written by programmer is not tested at coding phase and which may contains defects. Now, the proposed code-smell detector i,e. Particle Swarm Optimization is used and it starts checking the smells called as lazy class. Detector work is to identify the smells and if there is need for correcting the smells, then refactoring will comes into picture. Few are the refactoring tools, namely Insrefactoring[9], refactoring browser, JRefactory, DECOR[10], crawler tools. By identifying and correcting the smells one can get the quality and efficient source code without smells.

Programmer:

Here, programmer or developer writes the source code of the project. This, process will performed at coding phase. While writing the source code anomalies will occurs usually those are called as smells. In this step smells are not identified, just writing the code is done.

Code-smell detector:

The smells like long methods, long parameter list, lazy classes, duplicated codes, large classes are identified by tester. Tester uses one of the most optimized techniques for detecting the smells and the

ISSN: 2278-4632
Vol-10 Issue-12 No.03 December 2020

optimizer is PSO. It is one of the stochastic algorithm. The major benefit of this technique is, it reduces the search complexity of smells. Compare to other methods, PSO is faster and cheaper too.

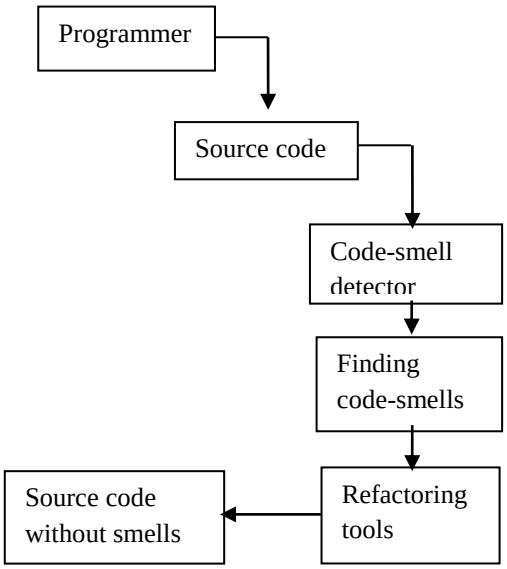


Fig 1. Procedure for detecting code-smells

Finding code-smells:

Identifying huge number of smells is not the matter, based on severity and risk, smells should be identified and detected. To get accurate results precision metric is used. In general precision gives "code-smells that are correctly detected among the set of all detected code-smells."

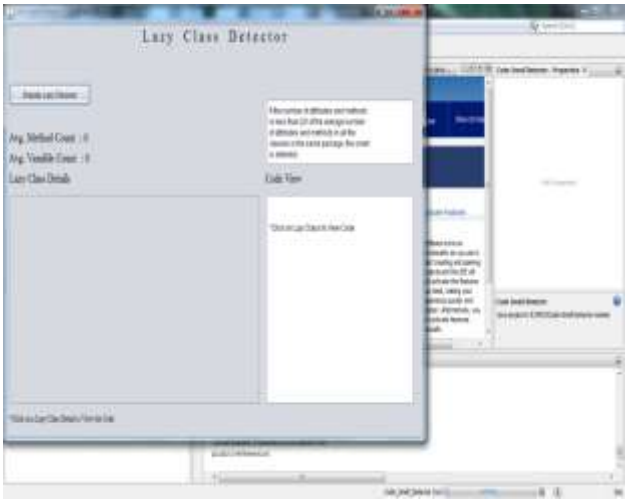
Refactoring tools:

After identifying and detecting the smells, if one want to correct the smells, then refactoring tools will comes into picture. Refactoring is the one which cleans the disorders in the code. Some of the refactoring tools are Insrefactoring[9], refactoring browser, Jrefactory, crawler tools.

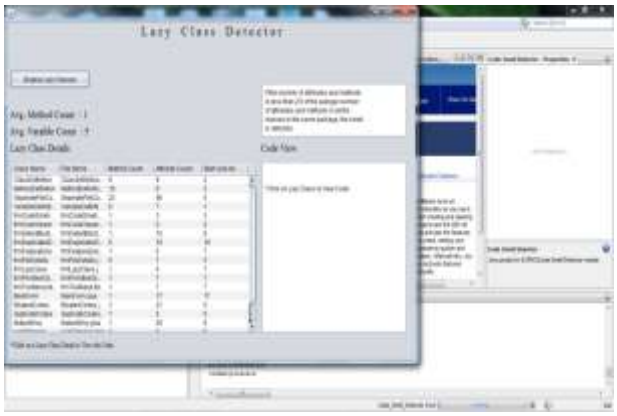
V. RESULTS

If the number of attributes and methods are less than two third(2/3) of the average number of attributes and methods in all the classes in the package, then the smell is considered as lazy class.

By taking java files, practically lazy classes have been detected and are shown below.



ISSN: 2278-4632
Vol-10 Issue-12 No.03 December 2020



The graphshown represents accurate results between existing and proposed system. Here, accuracy is calculated based on number of smells that have been detected with respect to number of lines of code. In existing system, by using genetic algorithms, smells are detected and are shown in blue color in graph, where the algorithm is not showing the optimization of code and in proposed system PSO, which is a stochastic optimization does not contain any genetic operators and searching of smells is performed intelligently. There by, accuracy and efficient of the system is achievedand in graph the proposed results are shown in brown color.

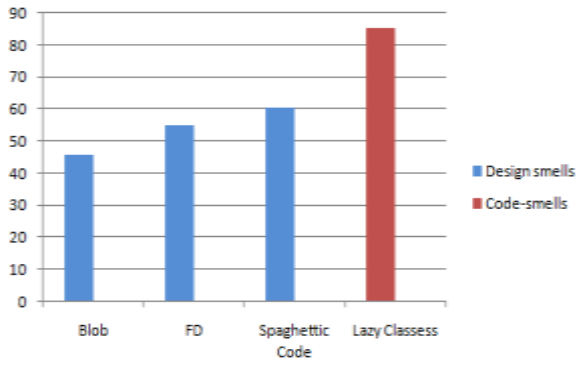


Fig. 2. accuracy of detected code-smells

VI. CONCLUSIONS AND FUTURE WORK

Detecting and identifying the smells by using PSO technique provides essential results, it reduces the complexity in finding the smells, system performances will increase and gets for cheaper cost. Due to these benefits, many researchers and in various applications this algorithm has been used. In simple, smells should be find and detected at the starting stages only there should not be neglected. At first, PSO searches for the smells that are affecting the code more, by using the precision metric.Here, the work showedthe analysis on design-smells and code-smells that have been detected. In proposed work to get the precise results PSO technique has been used along with the precision metric. This work proposes and develops a framework that combines the processes of identifying and applying refactoring for correcting those detected smells. Thus, the framework allows to find code-smells like lazy classes in the coding phase.

Therefore, it helps improve the accuracy of the source code and hence, efficiency of the overall system is achieved. In proposed work, detection performed on the smells that occurred in coding phase and future work includesthe smells that raises in testing phase can be detected and they are named as test-smells.

ISSN: 2278-4632
Vol-10 Issue-12 No.03 December 2020

REFERENCES

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, Refactoring: Improving the Design of Existing Code. Reading, MA,USA:Addison Wesley, 1999.
[2] W. J. Brown, R. C. Malveau, W. H. Brown, and T. J. Mowbray, Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis. Hoboken, NJ, USA: Wiley, 1998.
[3] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object-oriented design," IEEE Trans. Softw. Eng., vol. 20, no. 6,pp. 293–318, Jun. 1994
[4] F. Khomh, S. Vaucher, Y. G. Gu_eh_eneuc, and H. A. Sahraoui, "A bayesian approach for the detection of code and design smells,"in Proc. Int. Conf. Quality Softw., 2009, 305–314.
[5] M. Tomassini and L. Vanneschi, "Guest editorial: Special issue on parallel and distributed evolutionary algorithms, part two,"Genetic Program. Evolvable Mach., vol. 11, no. 2, pp. 129–130, 2010.
[6] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA,USA: Addison Wesley, 1989.
[7] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," ACM Comput. Surv., vol. 45, no. 1, 61 pages.
[8] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," IEEE Trans. Evol. Comput., vol. 16, no. 2,pp. 210–224, Apr. 2012.
[9] P. Vignesh and P. ramya, "Detection and Removal of Bad Smells instantly using a InsRefactor" , international Journal of Computer Science & Engineering Technology(IJCSET)
[10] N. Moha and Y. G. Gu_eh_eneuc, "Decor: A tool for the detection of design defects," in Proc. Int. Conf. Autom. Softw. Eng., 2007, pp. 527–528.