Solving Crosscutting Concerns Identification Problem in Object-Oriented Software with Genetic Algorithm

Richa Singh
Department Of Computer Science
Banaras Hindu University, Varanasi

Manjari Gupta
Department Of Computer Science
Banaras Hindu University, Varanasi

Abstract: Identifying crosscutting concerns in existing developed software systems is a well known problem. This problem is known as aspect mining and is useful in reverse engineering as well as refactoring the concerns into Aspect Oriented Programming (AOP). Refactoring the existing system using AOP suffices the goal of making the existing system easier to maintain and evolve. Discovering crosscutting concerns from existing object oriented software by facilitating semi automated system is the main issue. To address this issue, proposed technique provides a solution using genetic algorithm (GA) based on the concept of traditional method invocations. In this work, it is found that proposed experiments based on genetic algorithm resulted in keeping the number of identified crosscutting concerns seeds high. The proposed technique also find out that the proposed solution worked well on standard benchmark case studies previously used in aspect mining.

Key Word -Aspect Mining, Crosscutting Concern Identification, Genetic Algorithm, Object-Oriented Software, Aspect oriented programming (AOP).

I. INTRODUCTION

Crosscutting Concern Identification (CCI) is about discovering many different program entities like classes and methods representing simple or tangled concerns in the source code. Some of these concerns include logging, transaction management, and exception handling. Code for debugging, logging or locking is hard to keep hidden using object oriented programming alone. As a result, concerns are scattered across many classes, methods and packages [26]. At the same time the particular classes and methods do not only deal with the primary concerns they address, but also may need to take account some other concerns [23]. Thus there is a need to identify such concerns and implement them in AOP. This does not only improves the understandability of the concern in particular and of the software in general, but also provides a first step in migration towards applying aspect-oriented software development techniques. Thus, CCI is often identified as the most important activity for maintenance and evolution of object oriented software systems. Though, there are many methodologies for CCI, they all have similar limitations, like poor precision of results (i.e., the percentage of relevant aspect candidates reported by a given technique is relatively low); subjectivity to analyze the results because of ambiguity in results produced (eligibility of a method for a crosscutting concern depends on software engineers' experience); difficulty on comparing the results of different techniques; and difficulty on composing of results from multiples aspect mining techniques [4]. Following are the characteristics of CCI problem:-

1. CCI problem is found in legacy systems. Many techniques ([23],[5],[18]&[21]) were applied to solve CCI problem in open source codes like JHot Draw framework, pet store and junit having more than 10,000 lines of code.

2. There is no known efficient and complete solution because the existing solution approaches for CCI problem (pattern matching, formal concept analysis, natural language processing on source code, software metrics and heuristics, clone detection and fan in analysis) use specific assumptions for deciding a concern to be crosscutting.

II. CROSSCUTTING CONCERN IDENTIFICATION PROBLEM

*Corresponding Author

Understanding system at source code level requires understanding of each concern that it addresses, which in turns requires a way to identify these concerns [23]. Some concerns are easily identified because they are explicitly represented by program entities like classes and methods whereas some concerns are spread over many program entities or are mixed up with other concerns in the system and thus these concerns are called crosscutting concerns. Several examples of crosscutting concerns have been identified such as logging, transaction management and exception handling etc. Therefore theses concerns are not completely captured by using abstraction mechanism provided by traditional programming approaches.

In OOP the system is decomposed into modular units like functions and classes, in which some functionality are decomposed into more than one modular unit. These kinds of functionalities are called crosscutting concerns and finding these functionalities in the system is called CCI problem. It is very difficult to overcome with this problem, because to identify concerns it is needed to go through all code portions (like packages classes and methods) containing it. This problem is because of code scattering. OOP code is also affected by code tangling as implementation of crosscutting concerns is part of the implementation of some rest of the code in the system.

Crosscutting concerns identification may be used in many contexts. Many legacy software systems are need to be maintained and thus sometimes need to be re-implemented into aspects oriented programming language as required. Thus identifying crosscutting concern would ease the process of maintenance of legacy software.

Many researchers are working in the field of crosscutting concerns identification and its refactoring. Tonella et.al., Marin et.al. and Serban et.al. are prominent researchers in this area ([23],[5],[18]&[21]). Genetic algorithms have been widely used in search problems such as TSP (Travelling sales man problem) and scheduling jobs [27]. CCI can be treated as a search problem where a small number of methods, that belong to some crosscutting concerns, are searched from large number of methods in any source code. Thus, search space is very large. This motivates the application of genetic algorithms in this approach. To automate the process of CCI genetic algorithm is presented and evaluated in this paper. Contributions of this paper can be summarized as follows:

1. Aspect mining technique is proposed by using genetic algorithm based on the concept of method-method interaction and method and calling class relation. This technique does not give the best results but this is the initial effort and can be improved further by taking into consideration other characteristics of crosscutting concerns to judge whether these characteristics are important to define crosscutting behavior of concerns or not.

2. A list of all crosscutting concern seeds are presented that the proposed technique identified in JHotDraw framework, Junit and pet store. The obtained results of JHotDraw are compared with the results of other crosscutting concerns mining tools. Junit and pet store results of this paper cannot be compared with other techniques' results because existing techniques do not provide the results on Junit and pet store. But these results are valuable for those aspect mining researchers who will apply their technique on Junit and pet store and can compare their results with our results.

The rest of the paper is structured as follows: Related work is described in Section 3. Section 4 describes quality metrics that are used in fitness function evaluation in our proposed algorithm. Chromosome representation and genetic algorithm operators are described in section 5. Proposed genetic algorithm for CCI and case studies are described in section 6 and section 7 respectively. Conclusions are drawn in section 8.

III. RELATED WORK

Several approaches have been proposed for aspect mining till now. Leandro & Gustavo [17] proposed a strategy to identify crosscutting concerns at requirement level, that is at early stages in the software development process, by using the Language Extended Lexicon. First approaches in aspect mining were the query based search techniques. The developer had to search so called seed and associated tool showed all the places where the seed was found [5]. Many tools have been developed for this as Aspect mining [13], Aspect browser [30], AMTEX [1], Feature Exploration and analysis tool (FEAT) [20]. In the next enhancement developer focused on semi-automated techniques to identify crosscutting concerns in aspect mining. The state-of-the-art in aspect mining is represented by the collection of techniques described below:

Marin et.al. [21] have proposed an aspect mining technique using fan in value. The idea behind this approach was the methods that have fan in value greater than a given threshold are identified as crosscutting concerns.

Tourwe et.al. [28] have proposed an aspect mining technique based on identifier analysis. They used formal concept analysis on the identifiers to group entities with the same identifiers and obtained group of concepts that satisfy some constraint. Number of elements in the group larger than a given threshold are considered as aspect candidates.

Tonella & Ceccato [23] have proposed aspect mining technique using dynamic analysis. An instrumented version of mined software system is run and execution traces for each use case are obtained. Formal concept analysis technique is used for execution traces.

Shepherd et.al. (D.Shephered et.al., 2005) have proposed clone detection technique for Aspect mining. They search for code duplication in existing system using program dependency graph.

He & Bai [18] have proposed aspect mining technique based on dynamic analysis. They also applied execution traces for each use case, but they used clustering and association rules to identify crosscutting concerns.

There are just a few existing aspect mining techniques that use genetic algorithm and distance metric to identify crosscutting concerns.

Serban & Cojocar [10] have proposed a GAAM (graph algorithm for Aspect mining) algorithm for crosscutting concerns identification.

Renata Rand & Frank J. Mitropoulos [24] have proposed model-based clustering on heuristic methods such as hierarchical or partitional clustering for crosscutting concerns identification.

Moldavan & Serban [9] proposed a new genetic clustering based approach in Aspect mining. The measure of within cluster variance (*SSE*) is used as a fitness function in *GAM*, meaning that if K is the partition generated by an individual i, the fitness function of i is: fitness(i) = Max−SSE(k), where Max is the maximum value of the square sum error.

IV. QUALITY METRICS USED IN OUR APPROACH

This section represents the metrics used in the aspect of crosscutting seeds. As described above, the proposed technique is based on genetic algorithm. GA fitness function is based on two measures MMI and MCI described below. These measures correspond to crosscutting concerns symptoms such as code scattering and tangling and depend on calling classes and calling methods in a system. Let us first define the set of methods and set of classes in the system say $X = \{m_1, m_2, \ldots m_n\}$ and $Y = \{c_1, c_2, \ldots c_l\}$. A method m is characterized by a set of method-method interaction and a set of method-class interaction defined below [8].

Defnition1: Set of Method-Method interaction (MMI): A method-method relationship: for a method m, that is called from other methods belonging to X, a count is defined as the number of methods that call m. This is denoted by MMI (m) = {N| m is invoked by N number of methods ϵ X}.

Defnition2: Set of Method-Class interaction (MCI): A method-class relationship: a method m is related all those application classes (AC_i) such that they call it. Thus another count is defined for m denoted by MCI (m) = {L| m invoked by L number of classes ϵ Y}.

After calculating the above counts for all methods in the system, the Euclidian distance is calculated between any two methods in the system by using the following equation 1:

$$D_E=\sqrt{\sum_{k=1}^{2}\left(m_{i_k}-m_{j_k}\right)^2} \quad \ldots\ldots \ (1)$$

Where, m_{i_k} and m_{j_k} are 2-dimensional vector for ith and jth methods (having components MMI and MCI) as described above. Minimum Euclidian distance shows that either both methods belong to crosscutting concerns or both do not.

V. CHROMOSOME REPRESENTATION AND GENETIC OPERATORS

This section represents the chromosome structure and genetic operators used in the proposed genetic algorithm for identification of crosscutting concerns. We assume that the reader is familiar with the basic terminology of genetic algorithm (population, operators and fitness function).

Defnition1: Binary Representation: Let us consider a set of all methods $\{m_1, m_2, \ldots m_n\}$ in the system and then the chromosome is n-length string of 0 and 1. The ith gene of the chromosome corresponds to the ith method of the system. The allele of the corresponding ith gene in the chromosomes is set to '1' or '0' randomly. The allele '1' corresponding to the ith gene shows that m_ith method belongs to the crosscutting concerns and the allele '0' corresponding to the ith gene shows that m_ith method does not belongs to the crosscutting concerns.

Defnition2: Population Initialization: Let *P* be the size of the population. Each chromosome in the population represents a possible solution i.e. those set of methods that belong to any crosscutting concern. The initial population is generated randomly.

Defnition3: Fitness Function Evaluation: Taking each chromosome from the population P, its fitness is calculated using equation 2 that is based on MMI and MCI of only those methods that corresponds to allel '1' in the chromosome. Thus fitness function described as follows:

$$\text{Fitness}(Ch_r) = \sum_{\text{over all allels whose value is } 1} \sqrt{\sum_{k=1}^{2} (m_{ik} - m_{jk})^2}$$

Since, fitness evaluation process is quite time-consuming due to the repeated computation of distances between methods occurring in each chromosome. So it is advantageous to construct a fitness-table that saves the computed MMI, MCI and Euclidian distances between all pairs of methods in advance. During the fitness function evaluation process distances can be obtained simply by looking them up in the table rather than computing again and again.

Defnition4: Genetic Operators: reproduction, crossover, mutation and selection are basic operators of any genetic algorithm. There can be many different possibilities for all these operators. This section describes the particular operators used in the proposed algorithm.

Defnition4.1: Reproduction: In GA the selection of the best individual is based on an evaluation on fitness function. The reproduction operator ensures that, in probability, the better a solution is in the current population, the more replicates it has in the next population. In order to raise the possibility of the chance that best individuals are survived and replicates in the next population the proposed GA used "good individual replacing" step prior to the roulette wheel selection. In the first stage, a new chromosome Ch_{new} is generated and its fitness is calculated with fitness of any randomly selected chromosome in the population. The randomly selected chromosome *Chj* is replaced with the newly chromosome Ch_{new} if $Fitness(Ch_{new}) < Fitness(Ch_j)$. In the second stage a roulette wheel selection is applied [11].

//Good individual replacing step:

1. Evaluate the fitness for each chromosome Ch_i, $i = 1, 2, \ldots, P$.
2. Create Ch_{new} and randomly select a chromosome Ch_j from the current population.
3. Replace Ch_j with Ch_{new}, if $f(Ch_{new}) < f(Ch_j)$.

//Roulette wheel selection step [11]:

1 Evaluate the fitness, f_i , of each individual in the population.

2 Compute the probability (slot size), p_i of selecting each member of the population:

$P_i = f_i \backslash \sum_{j=1}^{n} f_j$ where *n* is the population size.

3 Calculate the cumulative probability, q_i, for each individual: $q_i = \sum_{j=1}^{i} p_j$

4 Generate a uniform random number, $r \in (0, 1]$.

5 If $r < q_1$ then select the first chromosome, x_1, else select the individual x_i such that $q_{i-1} < r \leq q_i$.

6 Repeat steps 4–5 *n* times to create *n* candidates in the mating pool.

Defnition4.2: Crossover: After selection, individuals from the mating pool are recombined (or crossed over) to create new, better, offsprings. The number of '1' in each of a parent pair is counted, say, NC_1 and NC_2, respectively, a random integer N_c is generated from the range $[1, M]$, where $M = \min(NC_1, NC_2)$, and then N_c gene positions having allele 1's in each of the parents are randomly selected for crossover [11]. Then apply crossover on parent pair by exchanging alleles at the selected gene position to find a new pair of their offspring. The steps used in crossover operation are illustrated as follows [11]:

```
// Crossover on multiple positions:
Generate a random (float) number r from the range [0,1].
If r < $p_c$ then
For each pair of chromosomes $Ch_a$ and $Ch_b$
1. Evaluate
$NC_1$ = number of 1's in $Ch_a$.
$NC_2$= number of 1's in $Ch_b$.
M = min {$NC_1$, $NC_2$}
2. Generate a random integer $N_c$ from the range [1, $M$].
3. Randomly select $N_c$ gene positions among the genes with allele '1' from $Ch_a$ and form a set $S_a$ of indices of such selected positions. Randomly select $N_c$ gene positions among the genes with allele '1' from $Ch_b$ and form a set $S_b$ of indices of such selected positions.
4. $S = S_a \cup S_b$
for each index $i$ in $S$
Exchange the alleles of chromosomes $Ch_a$ and $Ch_b$ at gene position $i$.
```

Defnition4.3: Mutation: Mutation is often the secondary operator in GAs, performed with a low probability. One of the most common mutations is the bit-wise mutation. In bitwise mutation, each bit in a binary string is changed (a 0 is converted to 1, and vice versa) with a certain probability, p_m, known as the mutation probability [15].

```
// Mutation under given condition
1. Generate a random (float) number r from range [0,1]
2. For i=1 to P do
4. If r < $p_m$ then      //mutate the allele of Ch.
5. Randomly generate a gene of $Ch_i$
6. Mutate allele of this gene
```

VI. PROPOSED GENETIC ALGORITHM FOR CCI

The starting point to identify crosscutting concerns through genetic algorithm is the creations of population of chromosomes and chromosome's random representation of genes as '0' and '1'. Different steps must be followed and repeated until a particular number of generations are achieved.

Step1: Select the number of population and generate chromosomes in which chromosome's length is equal to total number of methods in the system and randomly represent gene allele as '0' and '1'.
Step2: The best chromosomes generated in the current generation is added to the next generation, where the best chromosome is obtained on the basis of minimum fitness values.
Step3: "Good individuals replacing" and roulette wheel selection method is used to select two parents until a new complete generation is derived.
Step4: Recombination of parents is conducted by crossover described in section3.4.2 so that two offspring can be obtained.
Step5: Chromosomes are allowed to undergo mutation using the methodology described in section3.4.3.
Step6: The offsprings are placed in the next generation.

When the maximum number of generations has been obtained, the individual chromosome that has minimum fitness value obtained in the last generation is selected and the methods in the chromosome represent the possible crosscutting concern seeds in the existing software system.

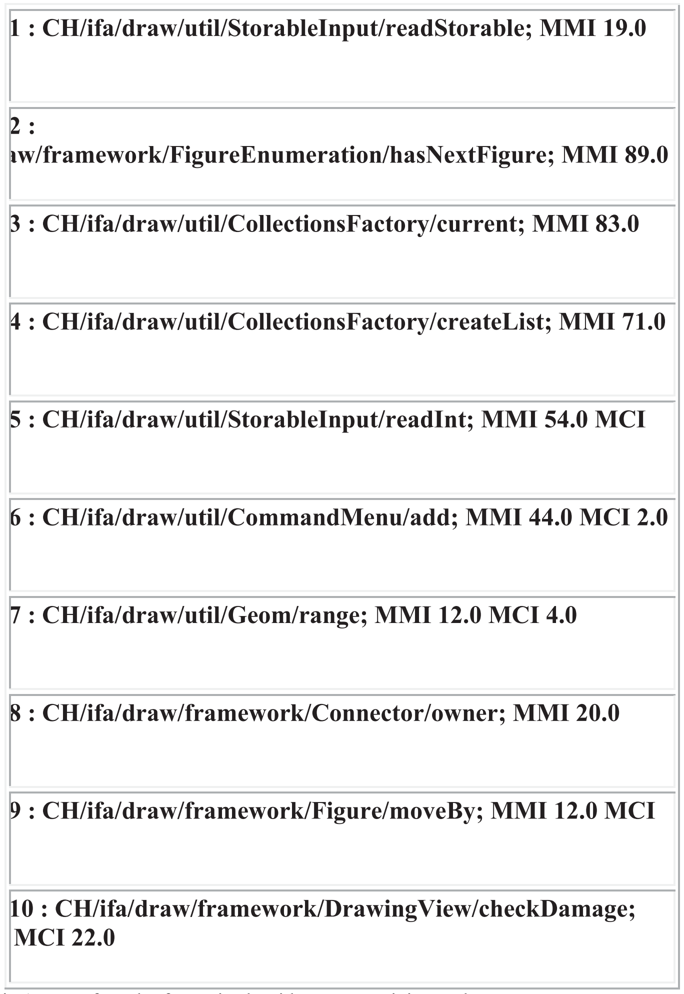
VII. CASE STUDY

In order to validate the approach, the case studies are performed on open source software: Pet Store, JHotDraw and Junit. The goal was twofold: 1) Validate whether the approach is identifying meaningful aspect candidates. 2) Demonstrate that the proposed genetic-algorithm approach brings significant benefits over existing aspect mining approaches. The developed tool can work on different parameter values that are used in genetic algorithm (population size, crossover probability, number of generations). The results in this paper are shown over population size of 100 chromosomes and 500 numbers of generations.

1. JHotDraw

This section presents the results of applying our approach to version 5.4b of JHOT Draw, a java object oriented framework, with approximately 18,000 non commented lines of code and around 2800 methods. In order to compare and validate results on JHotDraw, we compared our results with the crosscutting concerns discovered by [21].

The proposed genetic algorithm tool takes java archive file (JAR) as input, and then MMI and MCI is calculated for each method in JHotDraw. Methods are filtered using the same method described in [21]. Total 48 numbers of seeds are generated by our tool as a result. Out of which 10 seeds are shown in the fig. 1.



1 : CH/ifa/draw/util/StorableInput/readStorable; MMI 19.0
2 : aw/framework/FigureEnumeration/hasNextFigure; MMI 89.0
3 : CH/ifa/draw/util/CollectionsFactory/current; MMI 83.0
4 : CH/ifa/draw/util/CollectionsFactory/createList; MMI 71.0
5 : CH/ifa/draw/util/StorableInput/readInt; MMI 54.0 MCI
6 : CH/ifa/draw/util/CommandMenu/add; MMI 44.0 MCI 2.0
7 : CH/ifa/draw/util/Geom/range; MMI 12.0 MCI 4.0
8 : CH/ifa/draw/framework/Connector/owner; MMI 20.0
9 : CH/ifa/draw/framework/Figure/moveBy; MMI 12.0 MCI
10 : CH/ifa/draw/framework/DrawingView/checkDamage; MCI 22.0

Fig 1: Part of result of genetic algorithm aspect mining tool on JHotDraw

In the list of the figure 1 that represents the possible crosscutting seeds are identified by genetic algorithm. Few of these identified methods belong to some crosscutting concerns. For example, Persistence concern is performed by methods inherited from the Storable interface [21]. The Figure classes implement the methods to write/read themselves to/from a storableOutput/Input object, which is basically a specialized output/input stream. The set of the methods with minimum Euclidian distance and high MMI and MCI values comprises a number of seeds for Observer instances such as the checkDamge() and add/removeFigureChangeListener() methods in Figure classes. Among the other reported candidates the displayBox() method for Figure and Handle is identified. It returns the object's display box and is overloaded for figures for changing a figure's display box [21].

As we discussed above, in the figure 1 from the possible aspect seeds some seeds belong to crosscutting concerns such as; persistence, observer and composite design patterns. The idea to analysis the result is same as used by Marin [21] to identify the crosscutting concerns from the possible seeds. Marin used the concept of fan in value only while we used fan in as well as MCI.

In the second fold, to validate the technique compare obtained results with the other proposed technique “Fan In Analysis” [21]. This technique is able to detect almost many crosscutting concerns that were previously identified by other approaches [21]. The following table represents the comparison of seeds identified by our technique and Marin [21] technique.

Type	alysis]	l e
t behavior		
nforcement		
e		

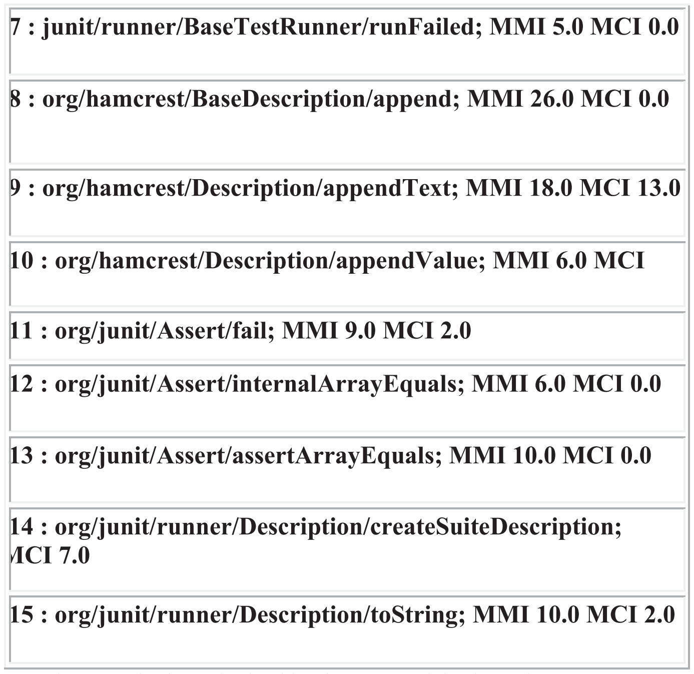
Table 1 summarizes the identified concerns in JHotDraw

In the table 1, column one shows the concerns identified by any one of the technique and second and third column shows whether the technique identified a concern or not by + and – symbols respectively. GA based technique identifies almost every concern as comparison to “Fan In Analysis” based technique [21]. GA based technique also identified one new concern that is factory in JHotDraw that was not identified by Marin’s technique.

2. JUnit:

This section presents the results of our approach applied to version 4.10 of JUnit. JUnit is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java. We have taken junit jar file as input for genetic based tool. Fig. 2 shows the seeds identified in junit as a crosscutting concerns.

1 : junit/framework/Test/run; MMI 5.0 MCI 5.0
2 : junit/framework/Assert/assertEquals; MMI 16.0 MCI 0.0
3 : org/junit/runner/Description/addChild; MMI 5.0 MCI 5.0
4 : junit/framework/JUnit4TestAdapterCache/asTest; MMI .0
5 : org/junit/runner/Description/isTest; MMI 5.0 MCI 2.0
6 : junit/framework/TestSuite/warning; MMI 9.0 MCI 0.0

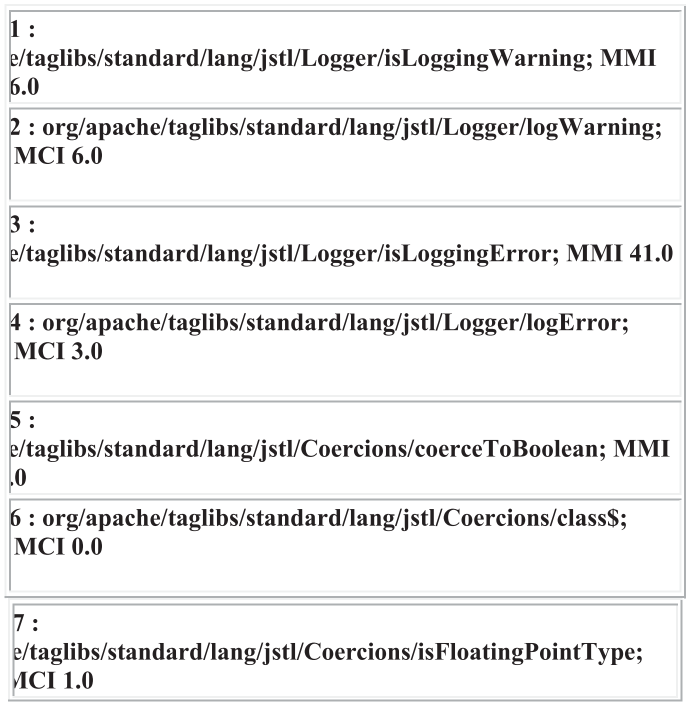


7 : junit/runner/BaseTestRunner/runFailed; MMI 5.0 MCI 0.0
8 : org/hamcrest/BaseDescription/append; MMI 26.0 MCI 0.0
9 : org/hamcrest/Description/appendText; MMI 18.0 MCI 13.0
10 : org/hamcrest/Description/appendValue; MMI 6.0 MCI
11 : org/junit/Assert/fail; MMI 9.0 MCI 2.0
12 : org/junit/Assert/internalArrayEquals; MMI 6.0 MCI 0.0
13 : org/junit/Assert/assertArrayEquals; MMI 10.0 MCI 0.0
14 : org/junit/runner/Description/createSuiteDescription; ICI 7.0
15 : org/junit/runner/Description/toString; MMI 10.0 MCI 2.0

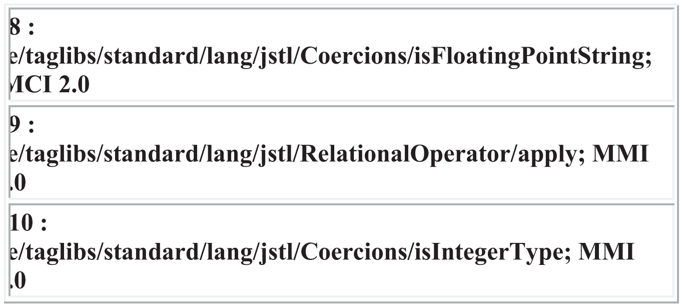
Fig 2: Result of genetic algorithm for Aspect mining in JUnit

3. Pet Store:

A "pet store program" is an example of how to use a particular software framework or set of libraries to build a real program. Its concept is similar to the Hello world program. It has been made popular by Sun Microsystems for illustrating Java EE, and many other vendors followed. Pet store consists of approximately 17,000 non-comment lines of code. We have taken pet store version 1.3.2 standard jar file as input to genetic based tool. Fig. 3 shows the few seeds identified in pet store as a crosscutting concerns. Total numbers of identified seeds in Per Store are 41.



1 : e/taglibs/standard/lang/jstl/Logger/isLoggingWarning; MMI 6.0
2 : org/apache/taglibs/standard/lang/jstl/Logger/logWarning; MCI 6.0
3 : e/taglibs/standard/lang/jstl/Logger/isLoggingError; MMI 41.0
4 : org/apache/taglibs/standard/lang/jstl/Logger/logError; MCI 3.0
5 : e/taglibs/standard/lang/jstl/Coercions/coerceToBoolean; MMI .0
6 : org/apache/taglibs/standard/lang/jstl/Coercions/class$; MCI 0.0
7 : e/taglibs/standard/lang/jstl/Coercions/isFloatingPointType; ICI 1.0



8 : e/taglibs/standard/lang/jstl/Coercions/isFloatingPointString; MCI 2.0
9 : e/taglibs/standard/lang/jstl/RelationalOperator/apply; MMI .0
10 : e/taglibs/standard/lang/jstl/Coercions/isIntegerType; MMI .0

Fig 3: Part of result of genetic algorithm for Aspect mining in pet store

VIII. CONCLUSION

In this paper, the proposed tool used a genetic algorithm to identify methods or concerns that are crosscutting and can be refactored into the AOP. In this work, it captured cross cutting concerns by targeting those methods that were called from a number of classes and sources and calculated the Euclidian distance between methods to evaluate the fitness of chromosomes. This euclidean distance captures the behavior of crosscutting concerns. Three case studies are used to showcase the working as well as the effectiveness of proposed solution. We have also made a comparative study of our technique, i.e. Genetic Algorithm based technique and Fan in Analysis technique [21]. We find out that despite having similarity in results our solution has identified a new unique concern, in JHotDraw, which was missed by existing technique. We applied same assumptions to analyze the results as done by "Fan In analysis" techniques. By keeping the "Fan In Analysis" as a comparison base, we obtained no false negatives in the result. The proposed technique captures appropriate refactoring candidates that would assist developers in realizing the true benefits of Aspect Oriented Programming by helping them to recognize refactoring possibilities in their codes [16]. In future we will focus on exploration of better fitness function by taking into consideration other crosscutting concerns characteristics like cohesion and coupling etc. It would also be interesting to identify the impact of different genetic algorithm operators on the results.

REFERENCES

[1] Charles zhang, Hans-Arno Jacobasan, Multi visualize http://www.eecg.utoronto.ca/ czhang/amtex/.

[2] D. Shepherd and L. Pollock. (2005) "Interfaces, Aspects, and Views" In Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop, Chicago, USA, March 2005.

[3] Esteban S. Abait, Santiago A. Vidal, Claudia A. Morcos (2008) "Dynamic analysis and Association Rules for Aspect Identification" II Latin American workshop on Aspect oriented software development,2008.

[4] Fernanda Madeiral Delfim, Rogerio Eduardo Garcia "Multiple Coordinated Views to Support Aspect Mining Using Program Slicing" SEKE 2013:531-536.

[5] Grigoreta S. Cojocar (2012) "Aspect mining past present and future" Studia Univ. Babes Bolyai Informatica, Volume LVII, Number 4, 2012.

[6] G. S. Moldovan and G. Serban (2006) "Aspect Mining using a Vector-Space Model Based Clustering Approach" In Proceedings of Linking Aspect Technology and Evolution (LATE) Workshop, Bonn, Germany, March 2006, to be published.

[7] G. S. Moldovan and G. Serban (2006) "Quality Measures for Evaluating the Results of Clustering Based Aspect Mining Techniques" In Proceedings of Towards Evaluation of Aspect Mining (TEAM) Workshop, ECOOP, pages 13-16, 2006

[8] G. S. Moldovan and G. Serban (2006) "A new k-means based clustering algorithm in aspect mining", In: 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'06), 2006, pp. 60-64.

[9] G. S. Moldovan and G. Serban (2006) "A new genetic clustering based approach in Aspect mining", Proc. of the 8th WSEAS Int. Conf. on Mathematical Methods and Computational Techniques in Electrical Engineering, Bucharest, October 16-17, 2006.

[10] Gabriela Serban and Grigoreta Sofia Cojocar (2007) "A New Graph-Based Approach in Aspect mining" Proceedings of the international conference on knowledge engineering, Principals and techniques, KEPT2007, Cluj-Naoca (Romania),June 6-8,2007, PP. 252-260.

[11] Hwie-jen Lin, Fu-wen yang and Yang-Ta Kao (2005) "An Efficient GA-based Clustering Technique" Manuscript Received: Mar. 4, 2005 Accepted: Apr. 27, 2005.

[12] Hacoupian Yourik (2013) "Mining aspects through Support vector clustering and Genetic algorithms" Ph.D. Nova Southeastern University Canada, P.no. 3/8/15, 2013.

[13] Jan Hannemann and Gregor Kiczales (2001) "Overcoming the Prevalent Decomposition of Legacy Code" In Advanced Separation of Concerns Workshop,at the International Conference on Software Engineering (ICSE), May 2001.

[14] Kumara Sastry, David Goldberg, Graham Kendall "Genetic Algorithm" Genetic Programming and Evolvable Machines, http://www.kluweronline.com/issn/1389-2576/contents.

[15] Kumara Sastry, David Goldberg, Graham *Kendall* (2004) "GENETIC ALGORITHMS" University of Illinois, USA University of Nottingham, UK ch.4, 2005 *link.springer.com/chapter/10.1007%2F0-387*-28356-0_4.

[16] Kiczels, G.,Lamping, J.,Menhdhekar,A., Maeda,C., Lopes, C., Loingtier, J.M. Irwin, J (1997) " Aspect oriented programming In: proceeding European conference on object oriented programming " 220-242. Volume 1241.Springerverlag (1997).

[17] Leandro Antonelli , Gustavo Rossi, Julio Cesar Sampaio do Prado Leite, Joao Araujo (2013) "Early identification of crosscutting concerns with the Language Extended Lexicon" Received: 28 February 2013 / Accepted: 7 November 2013, Springer-Verlag London 2013, Requirements Eng DOI 10.1007/s00766-013-0193-4

[18] L. He and H. Bai. (2006) " Aspect Mining using Clustering and Association Rule Method" International Journal of Computer Science and Network Security, 6(2A):247–251, February 2006.
[19] M.Ceccato, M.Marin, K-Mens, P.Tonella and T. Tourewe (2004) "Applying and Combining three different aspect mining techniques", TUD-SERG – 002.
[20] Martin P. Robillard and Gail C. Murphy (2002) " Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies" In ICSE '02: Proceedings of the 24th International Conference on Software Engineering, pages 406–416, 2002".
[21] M. Marin, A. van, Deursen, and L. Moonen (2004) "Identifying Aspects Using Fan-in Analysis" In Proceedings of the 11th Working Conference on Reverse Engineering (WCRE2004), pages 132–141. IEEE Computer Society, 2004.
[22] .O. A. M. Morales (2004) "Aspect Mining Using Clone Detection" Master's thesis, Delft University of Technology, the Netherlands, August 2004.
[23] P.Tonella, M. Ceccato (2004) "Aspect Mining through the Formal Concept Analysis of Execution Traces" In: Proceedings of the IEEE Eleventh Working Conference on Reverse Engineering (WCRE 2004). (2004) 112–121.
[24] Renata Rand & Frank J. Mitropoulos (2012) "Aspect Mining Using Model-Based Clustering" 978-1-4673-1375-9/12/$31.00 ©2012 IEEE
[25] Sayyed G.Maisikeli& Frank J.Mitropoulos (2009) "Aspect Mining Using Self-Organizing Maps With Method Level Dynamic Software metrics as input vector"2010 2nd International Conference on Software Technology and Engineering(ICSTE) By March 2009.
[26] Silvia Breu, Thomas Zimmermann, Christian Linding (2006) "Aspect Mining for large systems" OOPSLA06, October 22-26, 2006, Portland Oregon, USA, ACM- 59593-491-X/06/0010.
[27] Steven Beaty &Darrell Whitley (1991) "Motivation and Framework for Using Genetic Algorithms for Microcode Compaction" ACM SIGMICRO Newsletter, Volume 22 Issue 1, January 1991, Pages 20 – 27, doi-10.1145/1096503.1096505
[28] Tourwe,T.Mens (2004) " Mining aspectual views using formal concept analysis" In SCAM'04, Proceeding of source code analysis and manipulation, fourth IEEE international workshop on (SCAM'04), IEEE computer society, Washington,DC,USA,2004,pp.97-106.
[29] Ujjwal Maulik, Sanghamitra Bandyopadhyay (2004) "Genetic Algorithm-based clustering techniques" Presented by Hu Shu-chiung 2004.5.27.
[30] William G. Griswold, Yoshikiyo Kato, and Jimmy J. Yuan (2000) "Aspect Browser: Tool Support for Managing Dispersed Aspects" in first workshop on multi-dimensional separation of concerns in object oriented systems-OOPSLA-99, Technical Report CS1999-0640, UCSD, March 2000.