International Journal of Research in Advent Technology, Vol.7, No.6, June 2019
E-ISSN: 2321-9637
Available online at www.ijrat.org

Evaluation of Software Evolution based Aspect Mining Technique

Dr. Yasmin Shaikh, Dr. Sanjay Tanwani

Abstract— Aspect Mining is the search for candidate aspects in existing software systems and isolating them from the system into separately described aspects. A number of aspect mining techniques (AMT) have been proposed in literature for identifying crosscutting concerns. Software Evolution based Aspect Mining (SEAM) is also an AMT that identifies candidate aspects from version archives of software. The candidate aspects from two open source projects have been identified to assess the applicability of SEAM. One of the major limitation of most of the AMTs that have been proposed in literature is that no validation of their result is provided. In this paper, the evaluation of results produced by SEAM is presented. The evaluation process determines if the candidate aspects recommended by SEAM actually contain crosscutting functionality. The accuracy of recommendations produced by SEAM is determined by comparing the results of SEAM with the results of a benchmarking tool.

Index Terms— Aspect mining, cross-cutting concern, version history mining, software evolution.

I. INTRODUCTION

Aspect mining is defined as a specialized reverse engineering process, which aims at investigating legacy systems (source code) in order to discover which parts of the system can be crosscutting concern i.e. candidate aspect [1]. SEAM is used for mining candidate aspects from version history files [2]. In this approach, while mining aspects from legacy code, the source files that have been changed frequently and set of source code files that have been changed together frequently during the evolution of system are mined. Mined frequent change patterns are then visualized for structural relationship. On the basis of the structural relationship between the files, candidate aspects are recommended for the pattern. Two types of candidate aspects are reported – simple candidate aspects and complex candidate aspects.

In order to assess the applicability of the SEAM in aspect mining and validate the proposed algorithms, SEAM is applied on the version histories of two open source software namely, JHotDraw and Weka written in Java [3]. The simple and complex aspect candidate aspects are identified for both the systems and top ranked candidate aspects are listed. A systematic technique to collect data from version archives is also proposed. A detailed data preprocessing approach is introduced. An algorithm is proposed to map the version archive data in the form of transactions. The results are extremely useful in guiding software maintenance process and enhance maintainability of software. The results produced by SEAM for JHotDraw and Weka shows that the approach can be applied easily to any project with rich development history maintained in the form of CVS or SVN.

In this paper, first the candidate aspects are determined using a benchmarking aspect mining tool FINT [4]. In the next phase, the recommendations made by SEAM are compared with the recommendations made by FINT. Two predictability measures, precision and recall are used to determine the accuracy of results produced by SEAM.

The rest of the paper is organized as follows: Section 2 includes related work. In Section 3, predictability validation is presented. Section 4 includes result analysis and discussion. Section 5 draws conclusions from the presented analysis.

II. RELATED WORK

Breu et al. have developed an approach of identifying aspects from the version history [5]. It states that crosscutting functionality does not exist from the beginning. Instead, it is introduced over time. They analyzed CVS repository and identified those changes that are likely to introduce crosscutting concerns. It is assumed that two method calls that are inserted together in the same transaction are related to each other. This observation is used to mine pairs of functions that form usage patterns from version archives [6]. History-based aspect mining (HAM) identifies and ranks crosscutting concerns by analyzing

Manuscript revised June 8, 2019 and published on July 10, 2019
Dr. Yasmin Shaikh, *Assistant Professor at International Institute of Professional Studies (IIPS), Devi Ahilya Vishwavidyalaya (DAVV), Indore*
Dr. Sanjay Tanwani, *Professor & Head at School of Computer Science & IT, Devi Ahilya University, Indore.*

doi: 10.32622/ijrat.76201929

where developers add code to a program [7].

A concern mining technique named COMMIT (Concern Mining using Manual Information over Time) analyzes the source code history to statistically cluster functions, variables, types, and macros that have been changed intentionally [8]. The links between the clusters represent the seed. The approach is based on clustering references that have been added or removed together.

FINT is an aspect mining tool i.e. a tool for identifying crosscutting concerns from Java code [4]. It is implemented as Eclipse plug-in. FINT implementation includes three source code analysis techniques to identify crosscutting concerns: Fan-in analysis, grouped calls analysis, and redirections finder. The first two techniques look for concerns that are implemented as scattered method calls, such as logging, exception wrapping, authentication/ authorization, and so on. Redirection finder is a technique to identify wrapper classes, such as instances of the decorator pattern.

III. PREDICTABILITY VALIDATION

To assess the applicability of SEAM, candidate aspects of two open source software JhotDraw and Weka are mined using SEAM. The experimental results produced by applying the techniques to both the software are evaluated. The predictability of the recommendations is evaluated by comparing with the known aspects of the system. In the evaluation process, the aspects from the systems under experiment are extracted using a freely available aspect mining tool FINT [4]. The resulting candidate aspects are compared with the candidate aspects recommended by SEAM. Two predictability measures, precision and recall are used to determine the accuracy of results produced by SEAM.

Precision is a common performance measure. In the present context, precision refers to how well the frequent patterns generated from version history uncover the crosscutting concerns. Recall is the ratio of the number of correctly identified crosscutting concerns to the number of all crosscutting concerns existing in the system. Thus, it is a measure to find how well the technique works in determining crosscutting concerns. The correctness of recommended candidate aspects is determined by comparing them with the known aspects of the system.

Formally, *precision (m, sc)* for any candidate aspect *m* and strongly change coupled set *sc* is the fraction of number of correctly identified candidate aspects from *sc* to the number of files that are strongly change coupled. *correct (m, sc)* is the set of correctly identified crosscutting concerns. The *recall (m, sc)* is the fraction of correctly identified crosscutting concerns from *sc* to all the possible crosscutting concerns in *sc_tot*.

$$\boldsymbol{precision}(\boldsymbol{m}, \boldsymbol{sc}) = \frac{|\boldsymbol{correct}(\boldsymbol{m},\boldsymbol{sc})|}{|\boldsymbol{sc}|} \quad (1)$$

$$\boldsymbol{recall}(\boldsymbol{m}, \boldsymbol{sc}) = \frac{|\boldsymbol{correct}(\boldsymbol{m},\boldsymbol{sc})|}{|\ sc\_tot - \boldsymbol{sc}|} \quad (2)$$

Precision is used to evaluate the proposed technique in two ways: First to determine the accuracy of the technique and secondly to determine the limit of accuracy. To determine the accuracy of SEAM, the candidate aspects of each project are computed. Then the structural relationship is visualized among the strongly change coupled files and these aspects are classified into true and false crosscutting concern.

The recall value is computed on the basis of how many crosscutting concerns have been detected from all the existing crosscutting concerns in the system. To compute recall value, all the existing crosscutting concerns in the system are required to be known beforehand. But to have the knowledge of the existing crosscutting concerns of the system is nearly impossible for a real world industry size project. So, a completely correct recall value cannot be determined.

The limit of precision and recall is determined by evaluating how many frequent patterns are never crosscutting concerns and how many files that may contain crosscutting concerns are never included in the results of1 SEAM. We denote these two measures by $\text{precision}_{\text{lim}}$ and $\text{recall}_{\text{lim}}$. Formally, $\text{precision}_{\text{lim}}$ is the fraction of the total number of files contained in a set of strongly change coupled files sc to the number of files that actually contains crosscutting functionality *cc*. The limit of recall can be defined as $\text{recall}_{\text{lim}}$ is the fraction of the total number of crosscutting concerns in *sc* to the number of crosscutting concerns being identified *cc*.
The experimental results are validated using precision as a measure of performance. The results of experiment are compared with known aspects in the systems i.e. the results produced by FINT.

A. Simple Candidate Aspects

A simple candidate aspect is a set of strongly change coupled files with structural relationship between them [2]. To extract simple candidate aspects, the maximal frequent itemsets (MFSs) is considered and the logical coupling among files is determined. The logical coupling is determined by visualizing the coupling relationship between files in the pattern [9].

Fig. 1 shows the precision value of simple candidate aspects generated from different size of frequent patterns. The X-axis shows the size of the pattern and Y-axis shows the average precision percent of the results of simple candidate aspects. Each line of the graph shows the precision of results for one of the

doi: 10.32622/ijrat.76201929

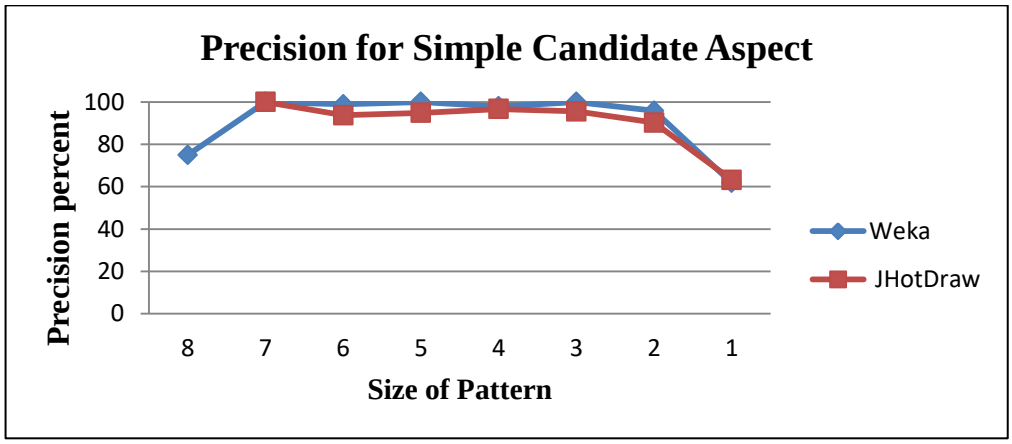

Figure 1: Precision for Simple Candidate Aspects

system under consideration. The precision stays almost flat between the second largest pattern and the pattern of size two. The precision falls greatly at pattern size one as there is no strongly change coupled files at size one.

B. Pruning

While finding simple candidate aspects, the maximal frequent itemsets (MFSs) is considered and the logical coupling among files is determined [2]. Since all the subsets of frequent pattern are also frequent (A priori principle) so the algorithm starts with MFS. After finding the relationship among change prone files in MFS the remaining patterns are filtered. All the patterns that are subsets of MFS are pruned from the candidate set. The pruning step eliminates the redundant patterns from being considered again, thus, improves the efficiency of algorithm. From the remaining frequent patterns, the MFS is considered and the process is repeated.

As pruning is applied while determining simple candidate aspects, a limited number of candidates remain in the subsequent passes. The results of pruning are shown in Fig. 2. The X-axis shows the size of frequent patterns. Y-axis shows the percentage of candidate patterns remained after pruning for each system. The pruning step reduces the number of candidate patterns significantly.

C. Complex Candidate Aspects

Crosscutting functionality cut across several files so combining simple candidate and then obtaining structural relationship among them finds complex candidate aspects [2].

For determining complex candidate aspects the set of frequent patterns (FS) is used. The union of set of files in FS is taken incrementally and candidate sets are generated to determine coupling relationship and crosscutting concerns in them.

Fig. 3 and Fig. 4 show the precision of results for complex candidate aspects in Weka and in JHotDraw respectively. The X-axis shows the size of the pattern and Y-axis shows the average precision value of the results of complex candidate aspects. The average precision of complex candidate aspects is higher for small sized patterns. As the size grows the precision percent falls and it lies around 50%.

IV. RESULTS AND DISCUSSION

The evaluation process reveals that the precision of candidate aspects recommended by SEAM lies in the range of 60% to 100%. The frequent patterns were generated from size one to size eight. There is no coupling in the patterns of size one so the precision at this level is not significant. For patterns of size two to seven, the precision is 100% that shows a high level of accuracy in result.

The precision of complex candidates, in both the systems under experiment, lies between 40% to 80%. Since the number of complex candidate aspect is very less, the interesting patterns out of these patterns can be manually identified.

Pruning is applied on set of frequent patterns after every iteration. It eliminates the patterns that have already been considered for candidate aspect. It significantly improves the efficiency of algorithm. It is evident from the results of pruning that it reduces significant number of patterns from being reconsidered.

Overall, the recommendations made by SEAM have higher precision value so it can be applied to any software having rich version history. Also, since SEAM does not involve investigation of source code, it is scalable to industry-size projects.

doi: 10.32622/ijrat.76201929

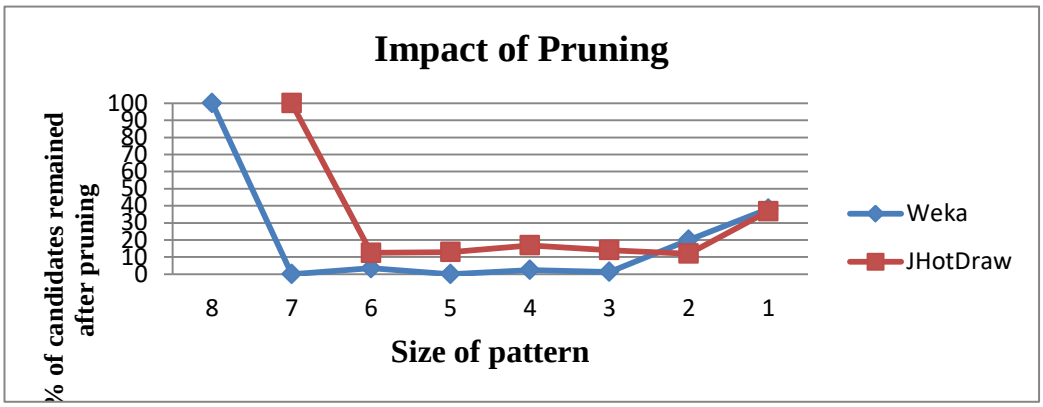

Figure 1: Impact of pruning on patterns from MFS.

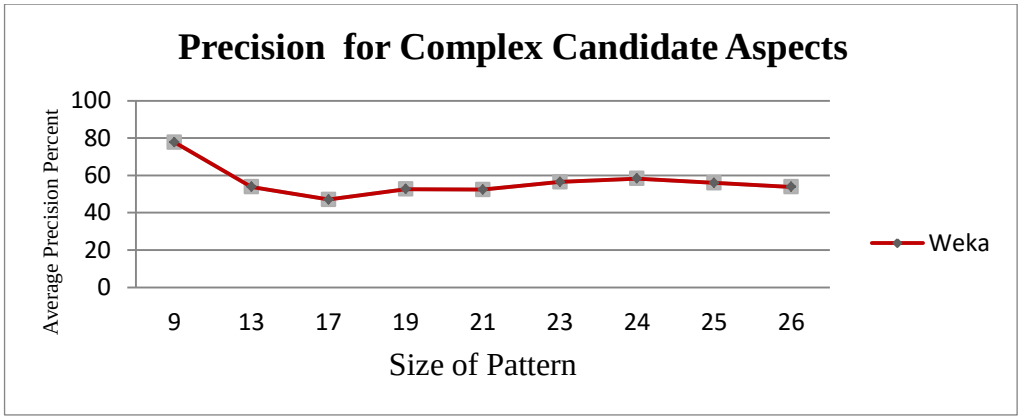

Figure 2: Precision for complex candidate aspects for Weka System.

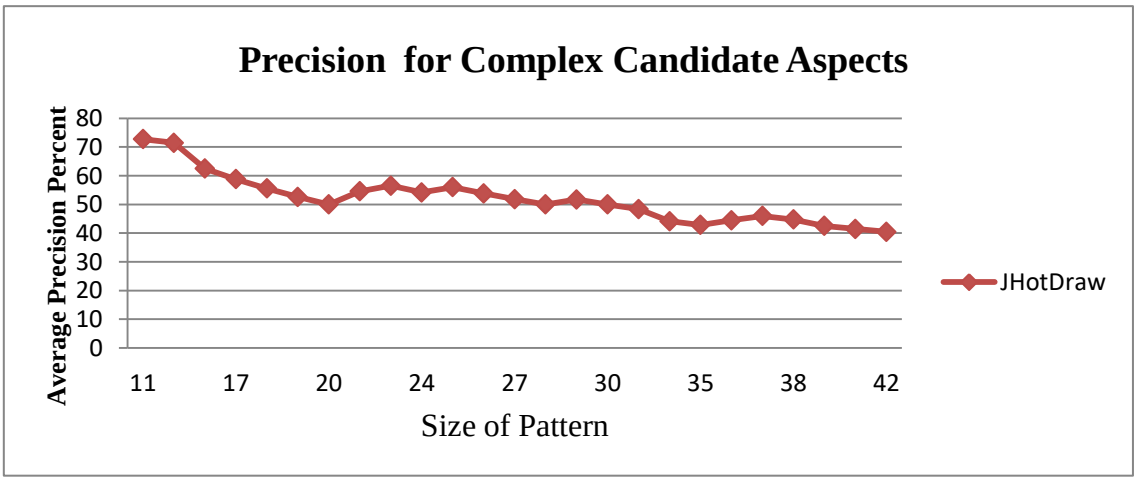

Figure 1: Precision for complex candidate aspects for JHotDraw system.

doi: 10.32622/ijrat.76201929

V. CONCLUSION

In this paper, the results produced by SEAM, when applied to two open source software, are validated. The results of validation process shows that SEAM can identify candidate aspects from legacy systems efficiently and with high precision. Most of the existing aspect mining techniques are platform specific. SEAM is applied on version history extracted from software repository to generate frequent pattern and candidate aspects. Thus, generation of candidate aspects is not platform specific.

The limitation faced while validating the result of SEAM is that very few aspect mining tools are available. Also no such commercial tool is available for validation of result. Therefore, only the results of FINT are used for comparing and validating the results of SEAM.

REFERENCES

[1] Deursen A.V, Marin M, Moonen L. "Aspect Mining and Refactoring." *Proceedings of the 1st International Workshop on Refactoring: Achievements, Challenges, Effects (REFACE), with WCRE*, 2003.

[2] Shaikh Yasmin, Tanwani Sanjay. "Software Evolution-based Aspect Mining: A Novel Approach." *International Journal of Data Mining and Emerging Technologies*, vol. 7, no. 2, pp. 97-106, 2017

[3] Shaikh Yasmin, Tanwani Sanjay.. Assessing Applicability of Software Evolution based Aspect Mining Approach. *International Journals of Management, IT & Engineering*, vol. 8, no. 7, pp. 375-399, 2018.

[4] Marin M, Moonen L, Deursen A.V. "FINT: Tool Support for Aspect Mining." *IEEE 13th Working Conference on Reverse Engineering*, pp. 299-300, 2006.

[5] Breu S, Zimmermann T. "Identifying Crosscutting Concerns from History." *Softwaretechnik-Trends*, vol. 26, no. 2, 2006.

[6] Williams C.C, Hollingsworth J. K. "Recovering System Specific Rules from Software Repositories." *Proc. Intl. Workshop on Mining Software Repositories, pp. 1-5*, 2005

[7] Breu S, Zimmermann T. "Mining Aspects from Version History." *Proc. 21st IEEE/ACM International Conference on Automated Software Engineering*, pp. 221-230, 2006

[8] Adams B, Jiang Z.M, Hassan A.E. "Identifying Crosscutting Concerns using Historical Code Changes." *Proc. 32nd ACM/IEEE International Conference on Software Engineering*, vol. 1, pp. 305-314, 2010

[9] Pinzger M, Gall H, Fischer M. "Towards an Integrated View on Architecture and its Evolution." *Electronic Notes in Theoretical Computer Science*, vol. 127, no. 3, pp. 183-196, 2

AUTHORS PROFILE

Author-1 Photo

Dr. Yasmin Shaikh is working as Assistant Professor at International Institute of Professional Studies (IIPS), Devi Ahilya Vishwavidyalaya (DAVV), Indore and carries teaching experience of over thirteen years. She has published several research papers in reputed journals and conference proceedings.

Author-2 Photo

Dr. Sanjay Tanwani is Professor & Head at School of Computer Science & IT, Devi Ahilya University, Indore. He has more than 30 years of teaching experience including three years of industry experience. He has published several research papers in reputed journals and conference proceedings. He has supervised several Ph. D. students. He has reviewed research papers of reputed journals. He has been a member of professional bodies like, IEEE and ACM since last more than 12 year

doi: 10.32622/ijrat.76201929