Indian Journal of Science and Technology, *Vol 7(5), 559–565, May 2014*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Feasibility Study of Aspect Mining at Requirement Level

Hema Subramaniam[1,2*], Hazura Zulzalil[1], Marzanah A. Jabar[1] and Saadah Hassan[1]

[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Malaysia; hazura@fsktm.upm.edu.my, marzanah@fsktm.upm.edu.my, saadah@fsktm.upm.edu.my
[2]Faculty of Computer Science and Information Technology, Universiti Selangor (UNISEL), Malaysia; hema@unisel.edu.my

Abstract

Modularity is categorized as quality characteristic which can increase the maintainability of a software program. Although modularity is gaining popularity, yet it is hard to be realized since there are many crosscutting concerns scattered and tangled in object oriented programs. Thus, more efforts are needed to maintain the software program which uses object oriented approach. On the other hand, aspect oriented approach has been viewed as it can encourage modularization. Since majority of the existing application is using object oriented approach, restructuring process known as refactoring become essential in increasing the program modularity. Refactoring means the process of restructuring the internal section without changing the system behaviour. Even though refactoring becomes the solution for this yet it does not seem to increase the modularity of a software program. This is due to lack of a comprehensive aspect mining method which helps in extracting the crosscutting concern from the existing applications before the refactoring takes place. At the present time, software practitioner preferred to conduct aspect mining at coding level which resulted in incomplete crosscutting concern extraction. Since the requirement stage being the initial stage before coding, it is believed to have the ability to extract more crosscutting concerns. Thus, it creates a space for aspect mining at requirement level as an alternative to aspect mining at coding level. In that case, the feasibility of aspect mining at requirement level becomes a need. This study aims to demonstrate the opportunity of conducting aspect mining at requirement level. Interview conducted among the Certified Professional Requirement Engineers (CPRE) has revealed that aspect mining at the requirement level is feasible and needed. The result of this study represented in SWOT analysis matrix is useful in justifying the alternative method of aspect mining. This alternative analysis also highlighted on the frequency of crosscutting concern that used among the CPRE indicating the worthiness of aspect mining at the requirement level.

Keywords: Aspect Mining, Feasibility Study, Refactoring

1. Introduction

Product quality is the desired outcome of an item. Meanwhile, software quality is viewed in terms of how well the software fits to the customer needs[1]. Therefore, software quality models which stress on the customer satisfaction characteristics along with their internal quality characteristics have been produced from time to time. Among the software quality models, modularity has been stressed as a characteristic of maintainability which indicates the needs to modularize the software program while maintaining the software quality[2]. Modularization is the action of breaking a large program into more manageable components. In which this would promote reusability of the components[3]. However, it is hard to achieve comprehensive modularity in object-oriented program due to existence of crosscutting concern[4]. Concern is anything that exists in the program that apart from the business functionality. It can be non-functional characteristics, error handling characteristics or security

*Author for correspondence

characteristics that do exist in the program coding[5]. In most of the cases, concern appears in program either in scattered or tangled manner[6]. Tangling means the program logic (or also known as functional requirement) has been mixed up with the concern. Meanwhile, scattering refer to the condition where by the same concern being repeatedly used in the program[7]. Refer Figure 1 on the sample of scattering and tangling concept at coding level.

From the sample given, the concern that scattered and tangled in this program is logging concern. The purpose of this concern is to log the activity. Since it is not the core part of the program it is considered as crosscutting concern which crosscut within and between the methods[8]. To emphasize the program modularity, this crosscutting concerns need to be extracted out from the base code/ program logic and produce the aspect[8]. Since the outcome of this process is an aspect, so the process is known as aspect mining[8]. Aspect mining at coding level becomes a challenge due to the increasing amount of Line Of Code (LOC) for large programs[5]. For this reason, aspect mining at requirement level has been viewed as an alternative method[9]. Thus, a study is needed to investigate the feasibility of aspect mining at requirement level. Based on the feasibility study, strength, weakness, opportunity and threat of conducting aspect mining at requirement level is analyzed and represented in SWOT analysis matrix.

Related work on aspect mining at coding, design and requirement level is discussed in the following sections. Feasibility study methods and result are presented in Section 4. Lastly the future work is suggested in Section 5.

2. Related Work on Aspect Mining

Aspect mining is the process of separating the crosscutting concern from the base code[10]. The base code is referred as functional requirement of an application. Since the aspect mining deals with the system which is already completed, so the developer has the option to do it either at requirement, design or coding level[11]. Among those software processes, coding level aspect mining seems to be popular and preferable stage. However, there are number of researchers who have suggested the aspect mining at coding level has several fundamental problems even it has been widely used[12]. According to them, earlier extraction would promote comprehensive aspect detection rather than at the source code level. These insist the need for an aspect mining activity at the requirement level. For this purpose, following sub section explains and compares the available aspect mining methods at three different software processes.

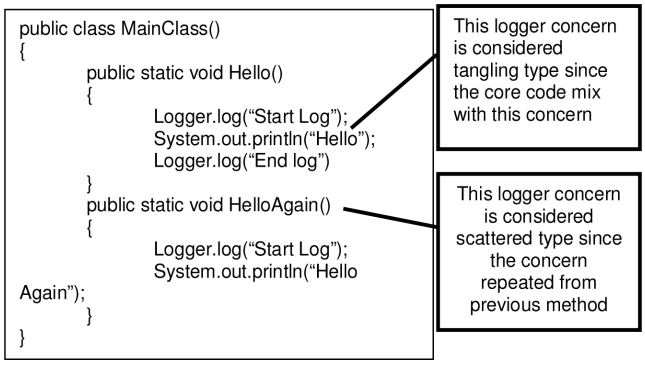

Figure 1. Scattering and tangling sample.

2.1 Existing Aspect Mining Activity at Coding Level

The concept of aspect mining has started since the crosscutting concern highlighted as the major problem in modularizing applications. Moreover, aspect mining is considered as initial activity during refactoring process[10]. One of the most significant discussions on the aspect mining at the coding level has taken place in the year of 2007. Whereby, the identification of aspect and their corresponding joinpoint at the coding level is arranged in cluster manner[13]. However, cluster level identification would possibly create unfavorable situation for comprehensive crosscutting concern extraction[13]. Apart from that, dependency of base code and crosscutting concern code are used as another method of aspect mining at the coding level. Then the dependency is represented in a graph[14]. Although it seems an effective method but it is not possible to draw dependency graph for a legacy system which contain huge number of LOC. On the other hand, code mining cascaded with traceability framework is used as the way of aspect mining at the coding level. This method seems applicable only for scattered crosscutting concern without involving tangled type of crosscutting concern[15]. Likewise, code mining using aspect idioms technique also seems to reveal scattered concern only. This indicate the situation whereby tangled concern missed out and resulted in incomplete searching results[16].

2.2 Existing Aspect Mining Activity at Design Level

One of the most significant discussions on aspect mining at design level explains about the impact of refactoring

at the software design stage[17]. Discussion has highlighted the proposed method namely Software Design Quality Index (SDQI) to estimate the quality of a software design. Moreover the index is used to determine the impact of refactoring at software design. Even though SDQI is used at the software design level, it does not propose any aspect mining methods. Otherwise, it only proposes the index to evaluate the refactoring at the design level.

2.3 Existing Aspect Mining Activity at Requirement Level

Refactoring at the requirement level started when the early aspect concept becomes popular among the software practitioner. The term early aspect integrates the aspect-oriented requirement engineering and the architecture design approach[18]. Since the term early aspect significantly relate with requirement, there are number of aspect mining techniques have been proposed at the requirement level rather than architecture design. Among the requirement representation, scenario modeling has been used as the space to extract the crosscutting concern[19]. During this process of extraction, aspectual scenarios are identified earlier and the isolated crosscutting concern is represented in state machine diagram. Moreover, scenario based aspect mining at the requirement level continue to gain popularity using concept of lattices analysis. With the interpretation of lattices concept, the crosscutting concern class method has been extracted out[20]. Beside, a method known as dominant decomposition is used at the requirement level for the purpose of aspect mining. The method highlights the way crosscutting concern being specified in the requirement document[21].

3. Feasibility Study Result and Discussion

Feasibility study is an initial investigation about the subject matter and determines the do or don't decision regarding the subject under study[22]. As a result, the study would suggest appropriateness of subject to be explored further. Likewise, this feasibility study aims to understand the suitability of aspect mining to be conducted at the requirement level. Requirement level refers to the stage whereby the software requirement specification document is used as a reference for an aspect mining purpose. Since the domain of the study involves requirement, people who are familiar with the requirement stage has been identified as object of study. Therefore a number of certified professional requirement engineers (CPRE) interviewed for this purpose. Based on the interview session, the strength, weakness, opportunity and threat of implementing aspect mining at the requirement level has been derived.

3.1 Data Gathering Method

Feasibility study aims to get an initial opinion on applying aspect mining at the required level. Therefore, qualitative method especially interview session has been identified as the appropriate technique that can help in getting opinion from the expert. The interview session is categorized as non-structured interview whereby the opinion from expert in terms of suggestion and comments are taken into consideration as well. However, certain general questions regarding the subject matter has been prepared earlier. Interview session has started by getting to know about the interviewee and their background about the requirement engineering. Notably it helps in understanding and determining the appropriate person interviewed. Furthermore, questions regarding their knowledge on requirement representation also included in the interview session. Indeed it would help in understanding the variety and possibility of requirement representation that can be considered while conducting aspect mining. Moreover, the perception of CPRE holders regarding the aspect mining at the requirement level gathered by knowing about the possibility of concern isolation at requirement representation diagram such as use case scenario, viewpoint and etc.

There are 10 CPRE holders involved in the interview session. Among them 50% CPRE are from academician background while remaining 50% are from software developer, requirement engineer researcher and system analyst respectively. Figure 2 shows the working background of interviewee who involved in the feasibility study.

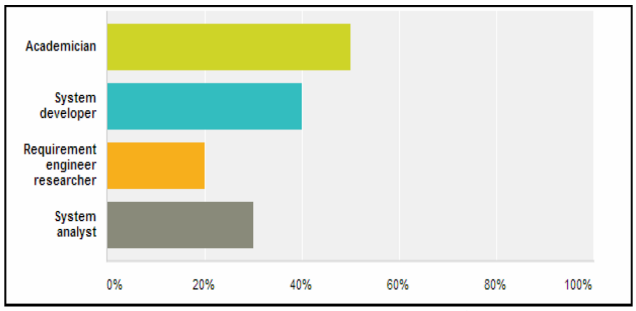

Figure 2. Interviewee working background graph.

3.2 Data Analysis Strategy

Feasibility study was conducted in a qualitative manner. In that case the analysis strategy should be on the qualitative manner rather than quantitative manner. There are number of qualitative analysis strategy has been proposed[23]. Among the qualitative analysis strategy the most appropriate one for feasibility study is SWOT analysis matrix[23]. Strength, Weakness, Opportunity and Threat analysis help in analyzing the initial information gathered from the interviewee. For instance strength of applying the aspect mining at requirement level were derived based on the opinion elicited from the interviewee. Similar kind of analysis applied to weakness, opportunity and threat. Table 1 shows the mapping between SWOT components towards the feasibility study outcomes.

3.2.1 Strength

SWOT components content for feasibility study was determined based on the mapping strategy (refer Table 1). The information gathered from the interview session were categorized into four categories. For instance, most of the interviewee generally felt that it is appropriate to have earlier isolation between the core business logic and crosscutting concern. They highlighted that concern normally mixed with the core functionality when there is an alternative flow involved in the use case. If the concern can be successfully extracted out, they clearly state there will be a possibility to reuse the concerns for a future application.

Table 1. Mapping between SWOT and Feasibility Study[24]

SWOT Components	Feasibility Study Outcomes
Strength	Strength of applying the aspect mining into requirement level. On the other hand, it can be said as anticipated benefit of implementing the subject matter into the new environment
Weakness	Weakness of applying the aspect mining technique at requirement level. This is referred as anticipated drawback that would be caused by the implementation.
Opportunity	This would refer to the external positive possibility available that increases the need for an aspect mining at the requirement level.
Threat	Threat refers to the challenges that are needs to be faced if the aspect mining will be implemented at requirement level. It is also referred as the external negative that influences towards the subject of study.

3.2.2 Weakness

The main internal factor that expected to give negative impact on the implementation of aspect mining into the requirement level is the textual representation of the requirement in requirement document. Interviewee felt it will give a huge impact since it is hard to identify the concern from text. This particular weakness also been supported by few researchers who concentrate about the early aspect[18,25]. Although textual representation is hard in generating concern from requirement, the appropriate text reading and extraction method would help in isolating concern from use case scenario[25].

3.2.3 Opportunity

There are number of opportunity has been highlighted by the interviewee for the aspect mining to be conducted at the requirement level. The opportunity here refers to the external characteristics that expected to give positive sign of aspect mining at requirement level. The extensive amount of Line Of Code (LOC) seems like giving opportunity for the aspect mining at requirement level since difficult to extract out concern from lengthy LOC. Apart from that, interviewee also felt that the strong binding between requirements, design and coding would create opportunity for the aspect mining at the requirement level. In fact, these point also supported by related works from previous researchers[26].

3.2.4 Threat

External factors that would give the negative possibility listed as the threat of aspect mining at the requirement level. In that case, interviewee felt that new non-functional requirement would be the challenges for the aspect mining at the requirement level. Notably new non-functional requirement will not be listed under the classification of concern and it would create a challenge to be extracted out from the requirement. Apart from that, interviewee also highlighted another challenge on handling the contra relationship between the concerns being extracted out from the requirement. For instance, in order to achieve high security, low response time will be recorded. In such case, there will be a challenge to force the bonding

between those concerns. However, these challenge can be overcome using the concept of observe the bonding between the concerns[27].

Refer Table 2 on the detail description on the strength, weakness, opportunity and threat of applying the aspect mining at the requirement level. The list generated based on the input from the interview session

3.3 Common Crosscutting Concern

Apart from the feasibility of aspect mining at the requirement level, this study also pointed out the opinion from the CPRE holders on the popularity of common crosscutting concern. The concern is anything apart from the core processes of a system. In such case, non-functional requirements also classified as one of the concerns[20]. During the interview session, interviewee has been asked to rate the frequency of non-functional requirements that commonly used in their requirement document. In particularly, it aims to understand the commonly used concern in requirement document. Consequently it will help in effective concern classification at the requirement phase itself. Table 3 shows the result of non-functional requirements usage frequency.

The frequency of non-functional requirement evaluated in terms of average rating. For instance, Error-Handling requirement average rating calculates with the following formula:

$$\left[1*(1)\right]+\left[3*(2)\right]+\left[0*(3)\right]=13/\left[\text{number of interviewee}\right]$$

$$=13/10=1.30$$

whereby, the values in the parentheses refer to the weighted values assigned to the each frequency level. Then the sum of that number divided by the number of interviewee. In this case it is divided by 10. Generally most of the rating falls right of the 'very frequent' and near to 'frequent'.

Based on the Table 3, security and performance have been rated as very frequently used by all the interviewees. Meanwhile, ability to handle error generally is rated as frequently used non-functional requirement by all

Table 2. SWOT analysis Matrix

	Positive	Negative
Internal	**Strength** • Earlier isolation of business logic from the main flow and alternative flow – this earlier separation process will cover overall system • Reusability of the concern definition into new application requirement specification without repeating the same process. • Easier mapping between the available group of concern into functional requirement • Increase the maintainability of the requirement specification	**Weakness** • Limitation in identifying concern since requirement deal with text.
External	**Opportunity** • Extensive amount of LOC-concern extraction become tedious due to extensive amount of LOC. This increase the opportunity of refactoring at requirement level • Strong relationship between design and requirement – the direct relationship between the software process would ease the refactoring process.	**Threat** • Contra relationship between the non-functional requirements – conflict between the requirements would happen since there is a negative relationship between the requirements. • Existence of new non-functional requirement in a new project

Table 3. Frequency of non-functional requirement used in requirement document

Non-functional Requirement /Concern	Very Frequent	Frequent	Less Frequent	Average Rating
Security	100% (10)	0% (0)	0% (0)	1.00
Performance	100% (10)	0% (0)	0% (0)	1.00
Error-Handling	70% (7)	30% (3)	0% (0)	1.30
Availability	50% (5)	50% (5)	0% (0)	1.50
Usability – Including User interface	60% (6)	40% (4)	0% (0)	1.40
Distribution- Platform Independent	60% (6)	40% (4)	0% (0)	1.40
Storage Capability	60% (6)	40% (4)	0% (0)	1.40
Data Consistency	90% (9)	0% (0)	10% (1)	1.20

the interviewees. They also stress error-handling normally mixed up with functional requirement whereby it specified in the alternative flow of use case scenario. Together with that, availability, usability and distribution also generally rated as frequently used non-functional requirements with average rating of 1.30 to 1.40. With the reference to Table 3 , the aspect mining at the requirement level would be much more manageable whereby most of the non-functional requirements are frequently used. Accordingly, it shows that concern / non-functional requirement could be isolated at the requirement level since it can be clearly identified at the requirement level itself rather go into coding level. Thereupon it supports the statement that requirement level aspect mining is feasible to do.

4. Conclusion and Future Work

Aspect mining which is known to be initial process of refactoring always correlated with source code. Due to the fact that coding level aspect mining fail to comprehensively extract crosscutting concern from an application, requirement level aspect mining becomes the alternative solution. A study that conducted among the CPRE holders has supported the statement that it is feasible to conduct aspect mining at the required level. Moreover aspect mining at the required level is also gaining popularity among the researcher since the concept of early aspect is being introduced[18]. In the event that aspect mining at requirement level is feasible, it increases the need for further investigation on the method or technique of conducting it. Further experimental investigation is needed in estimating the number of crosscutting concern classification at the required level. Consequently it will help in modularizing the application at the required level itself.

5. References

1. I. JTC 1/SC, ISO/IEC 9126-3: Software engineering-product quality-part 3: Internall Metrics.Canada; 2002.
2. Berander P, Damm L-O, Eriksson J, Gorschek T, Henningsson K, Jönsson P. Software quality attributes and trade-offs. Blekinge Institute of Technology; 2005.
3. Tambe S, Dabholkar A, Gokhale A, Kavimandan A. Towards a QoS modeling and modularization framework for component-based systems. 2008 12th Enterprise Distributed Object Computing Conference Workshops; 2008 Sep. p. 43–9.
4. Zhang Y, Zhang J, Chen Y, Wang Q. Resolving synchronization and analyzing based on aspect-oriented programming. 2008 International Symposium on Computer Science and Computational Technology; 2008. p. 34–7.
5. Marin M, Moonen L, Van Deursen A. An integrated crosscutting concern migration strategy and its application to JHOTDRAW. Seventh IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2007). 2007 Sep. p. 101–10.
6. Yang S. Research on recovering early aspects in aspect-oriented software reserve engineering. International Conference on Computer Application and System Modeling (ICCASM); 2010. p. 202–206.

7. Yang S. Understanding crosscutting concerns from various perspectives in software reverse engineering. Sixth International Conference on Networked Computing and Advanced Information Management (NCM); 2010. p. 145–150.
8. Zhang P, Su Y. Understanding the aspects from various perspectives in aspects-oriented software reverse engineering. 2010 International Conference on Computer Application and System Modeling (ICCASM 2010). 2010 Oct . p. V11–311–V11–314.
9. Mortensen M, Ghosh S, Bieman JM. Testing during refactoring: adding aspects to legacy systems. 17th International Symposium on Software Reliability Engineering. ISSRE '06, 2006. p. 221–30.
10. Huang J, Lu Y, Yang J. Aspect mining using link analysis. 2010 Fifth International Conference on Frontier of Computer Science and Technology. 2010 Aug. p. 312–17.
11. Mens K, Kellens A, Krinke J. Pitfalls in Aspect Mining. 2008 15th Working Conference on Reverse Engineering. 2008 Oct. p. 113–22.
12. McFadden RR, Mitropoulos FJ. Aspect mining using model-based clustering. 2012 Proceedings of IEEE Southeastcon, no. 978. 2012 Mar. p. 1–8.
13. Anbalagan P, Xie T. Automated inference of pointcuts in aspect-oriented refactoring. 29th International Conference on Software Engineering (ICSE'07). 2007 May. p. 127–36.
14. Lee S-H, Cho B-H, Song Y-J. A study on crosscutting refactoring using progam dependency relation. 2010 IEEE/ACIS 9th International Conference on Computer and Information Science. 2010 Aug. p. 684–89.
15. Majumdar D. Migration from procedural programming to aspect oriented paradigm. 2009 IEEE/ACM International Conference on Automated Software Engineering. 2009 Nov. p. 712–15.
16. Zhang C, Jacobsen H-A. Refactoring middleware with aspects. IEEE Transactions on Parallel and Distributed Systems. 2003; 14:1058–73.
17. Sharma T. Quantifying quality of software design to measure the impact of refactoring. 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops. 2012 Jul. 266–71.
18. Rashid A, Sawyer P, Moreira A, Araújo J. Early aspects: a model for aspect-oriented requirements engineering. IEEE Joint International Conference on Requirements Engineering; 2002. p. 1–4.
19. Whittle J, Arau J. Scenario modelling with aspects. IEE Proceedings – Software. 2004; 151(4).
20. Su Y, Zhou X-W, Zhang M-Q. Approach on aspect-oriented software reverse engineering at requirements level. 2008 International Conference on Computer Science and Software Engineering; 2008. p. 321–24.
21. Li G. Identifying crosscutting concerns in requirement specifications - a case study [Thesis]. Queen's University; 2009 Sep.
22. Geetha DE, Kumar TVS, Kanth KR. Predicting performance of software systems during feasibility study of software project management. 2007 6th International Conference on Information, Communications & Signal Processing; 2007. p. 1–5.
23. Bernard HR, Ryan GW. Analyzing qualitative data: systematic approaches. Sage Publications Inc; 2009.
24. Helms MM, Nixon J. Exploring SWOT analysis – where are we now?: A review of academic research from the last decade. 2010; 3(3):215–51.
25. Zheng X, Liu X, Liu S. Use case and non-functional scenario template-based approach to identify aspects. 2010 Second International Conference on Computer Engineering and Applications. 2010; p. 89–93.
26. Breu S. Extending Dynamic aspect mining with static information. Fifth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'05). 2005. p. 57–65.
27. Monteiro MP, Fernandes JM. Refactoring a Java code base to Aspect: an illustrative example. 21st IEEE International Conference on Software Maintenance (ICSM'05). 2005; p. 17–26.