Aspect Mining Using Clustering and Association Rule Method

Lili He,[†] *and Hongtao Bai*[††]*,*

College of computer science and technology, Jilin University, Changchun, China
New Technology Center, Shenzhen Tele. Com, China

Summary
Aspect oriented programming offers a unique module, an aspect, to encapsulate scattered and tangled code, which makes it hopeful to solve the problem of crosscutting concerns. Identification and encapsulation of crosscutting concerns is the key problem in the migration from OO system to AO system. A novel aspect mining method which combines clustering and association rule technology is provided in this article. Clustering analysis based on the execution traces is provided to find out candidate aspects; while association rule mining based on the execution traces with ordered call is used to find out the crosscuts. Both the aspect code (advice body) and the crosscuts (pointcuts) are gotten after the above two processes, which constitute the aspect mining process. An actual application on real system provides the validity of our methods.

Key words:
AOP, Aspect mining, Clustering method, Association Rule Method, reverse engineering

Introduction

Separation of concerns is an important principle of the software design. Concern is a canonical solution abstraction that is relevant for a given problem. [1]. Most of the concerns in large applications are well modularized, but there are other concerns, whose implementation codes are scattered across several modules and even mixed with the functionalities that implement the responsibilities of the modules. These concerns are hard to be modularized with traditional programming method, and they are known as crosscutting concerns.

Aspect Oriented Programming[2]: AOP sets up on the basis of existing technology and provides a special modular unit——Aspect. Aspect has ability to encapsulate scattered codes corresponding to some crosscutting concern and appoint where to carry out them in the source code. Translating OO system to AO system will raise system intelligibility, thus lengthen the life cycle of the software system.

The transformation from OO to AO system including the following two steps (1) *Aspect Mining*: Identifying the candidate crosscutting concerns from the source code; (2) *Aspect Refactoring*: Constructing Aspects according to confirmed concerns, the scattered code would be replaced by aspects in an AOP language such as AspectJ.

Aspect Ming is mainly discussed in this paper, which usually includes two steps: (1) *Identification of crosscutting concerns codes*: finding out those codes which are the implementation of some functionality and but they are scattered across the whole OO program; (2) *Location of crosscuts*: finding out the relationship between the base code and aspect code.

A novel aspect mining method is proposed in this paper. Clustering analysis on program execution traces is used to identify crosscutting concerns from legacy system. Execution traces are obtained by running an instrumented version of the legacy system under a set of scenarios. Scenarios, which have similar behavior, will be in the same cluster that implicates the candidate aspect set. Then association rule mining on static methods invocation relations is used to locate crosscuts.

2. Aspect Mining Process

Firstly, some definitions on aspect mining are provided.

Definition 1: Crosscutting concerns: the requirements that fail to be modularized in OO system; correspondingly, those are well encapsulated are regarded as the base concerns.

Definition 2: Crosscutting concern codes: the codes that implement the crosscutting concern; correspondingly, the codes that implement the base concern are base concern codes. In programs with good programming style, base concerns codes are well modularized. Crosscutting concerns codes are scattered across the modules of base concerns. But they are well modularized in each local module, too.

Definition 3: Aspect Mining: a reverse engineering process that aim to find out the potential crosscutting concerns from the existing OO program.

Definition 4: Crosscuts: the cut-in relationship between crosscutting concern codes and base concern codes. This relationship will be either invocation or execution, because the aspect mining method is based on execution traces in this paper.

Manuscript received February 25, 2006.
Manuscript revised February 28 , 2006.

2.1 Identification of Crosscutting Concerns

There are some well-known functions such as authentication, log and the exception/mistake in large software systems whose codes are always scattered across a lot of classes and tangled with functions of these classes. These crosscutting concern codes are often well encapsulated as methods or procedures in each local module and are used here and there.

In OO system, the behavior of a program is realized by the invocation relationship between methods. Similar execution sequences may imply the existence of crosscutting concerns. Clustering on execution traces may get some interesting unfound concerns.

Execution traces are obtained as follows: OO system is instrumented, and then executed at specified Scenarios and inputs. Every Scenario corresponds to a called-method sequence. If there were a group of codes that has similar action, i.e. similar called-method sequence, and appears frequently in execution traces, then a crosscutting concern may exist. Similar called-method sequences are possible crosscutting concerns code.

Using clustering analysis technology[3,4,5] on Scenarios, we regard Scenarios in OO system as data items (objects[1]), methods executed by Scenario as objects' attributes, and the number of times each method are called as values of attributes. Let n, m respectively be the quantity of all Scenarios and methods, then the object set $O=(o_1,o_2,\ldots,o_m)$, and every object is an m-dimensional vector $(a_{i1},a_{i2},\ldots,a_{im})$ (i =1,2,… n). The objects' data matrix is formed as follows: for any object o_i and method a_j, if method a_j (j =1,2, … m) is invoked by object o_i, for k times, then the value of the jth dimension of object o_i is k, otherwise zero.

The objects' dissimilarity matrix storing the dissimilar measure of each couple of objects is an $n*n$ matrix. If the values of the w-th dimension of object o_i and o_j both are non-zero, it means o_i and o_j both are invoked by o_w. That is to say, o_i and o_j have the similar behaviors in terms of o_w(the only nuance is the value k). If both values are zero, it proves neither is invoked by o_w. If one is zero and another is non-zero, it proves the dissimilar behaviors in terms of o_w.

Clustering on objects' matrix and object dissimilar matrix will produce a group of object sets. Common attributes elements in every set, namely the similar codes of Scenarios, are the candidate crosscutting concerns codes.

2.2 Location of Crosscuts

The next problem needs to be solved after the identification of candidate crosscutting concerns code is the location of crosscuts, the relationship between crosscutting concerns code and base concerns code.

Two mechanisms are provided by Aspect to interact with base concerns code: introduction and pointcuts. Introduction modifies the source code by adding fields or methods to class, interface or aspect directly; while Pointcuts intercept the normal execution flows at given join points, and aspect codes are carried out after or before this point, even replace this point.

The relationship between crosscutting concern codes and base concern codes is obtained by dynamic analysis to the execution traces of the legacy system. And the pointcuts can be identified are call() and execution().

Let $M=(m_1,m_2,m_3)$ is a typical called-method sequence and m_3 is recognized as crosscutting concern code, crosscuts may be showed that m_2 is followed by m_3, which is expressed as m_2 ->m_3.

We use association rule mining [6,7,8] to get the crosscuts as follows: Each method m is regarded as one transaction of association rule and called methods sequence by method m constitute its item set. Especially, in order to locate the beginning and the ending positions more easily, we defined two common functions: first () and end (). First () represents a virtual call at the start of m, while end () represents a virtual call at the end of m. This location relationship can be provided automatically as rule mode by association rule mining. The elements of association rule are defined as follows:

Itemset $I=(i_1, i_2,\ldots, i_m)$ is constructed by m methods in source code; every method is regarded as one item. There are two functions, first () and end (), are introduced for locating. Every method calls first () before the execution itself and calls end () after that. Let i_0 and i_{m+1} denote first () and end (), then the extended itemset is $I^*=(i_0,i_1,i_2,\ldots,i_m,i_{m+1})$.

Let each Scenario's name be a TID, the primary sign of one transaction, due to one transaction is determined by one scenario of source code. The transaction set $T=\{T_1,T_2,\cdots\cdots,T_m\}$ include all transactions, and for each T_j, j=1,2, $\cdots\cdots m$, $T_j \subseteq I^*$.

Confidence degree and support degree are important concepts of an association rule. In this paper, support degree can assure that candidate Aspects are "frequent", while confidence degree specifies the stability of location of crosscuts.

Association rules are obtained after the execution of association rule miming algorithms. The transaction set, itemset, the values of confidence degree and support degree are the inputs. Frequent 2-itemsets are calculated during the process of association rule mining, because the aim is to find out the relationship such that "method B is executed right after method A is executed", that is to say IF A THEN B. The association rule can be merged. Let IF A THEN B and IF B THEN C be the two rules produced

[1] The "object" here is the notation from clustering analysis without special explanation instead of the notation from Object-oriented technology.

by mining algorithm, they can be merged into IF A THEN BC, which shows method B and C are called after A. Pointcut would choose execution join point, if pre-condition or post-condition of association rule is "*first*" or "*end*", otherwise "*call*" join point would be chosen.

A series of rules similar with IF A THEN B are produced after the above steps, but only proves that A and B appear at the same time under confidence degree and support degree. It can't confirm whether method B is called closet to A or not, which is key point in "*advice*" of Aspect. Adjoining attributes between items of transactions are used to filter out the invalid rule. The rule IF A and B is valid only when method B appears right after method A.

Candidate Aspect set is mined through two steps above. *Advice* body is obtained at 2.1 and Advice at 2.2.

3. Case study

A banking application developed by OO technology is used as case study in this paper. Two main modules are mainly discussed: counter business and day balance at control center. Counter business includes creating new accounts (KH), depositing (CK), and withdrawing the money (QK). Day balance includes independence balance (DLRJ) and central balance (ZJRJ). We are trying to find out crosscutting concerns at the two modules using approaches discussed above.

A new account created can be a current account (KH1) or a fixed account (KH2). Every choice corresponds to one Scenario with own execution traces. For example, the execution traces of KH1 includes receiving message, encryption, analyzing message, opening accounts, locking and message returning etc. Each Scenario (KH1, KH2 …) is an object of clustering and its execution traces (called methods) are this objects' attributes. Objects and their attributes are shown as table 1 and the number in parentheses after every attribute denotes the invocation times of that method. Execution traces may contain deep nested calls. We specified the nested level constraint to be 3, which is enough for effectiveness.

Two clustering result classes: {KH_1, KH_2,…, QK_1, QK_2, …} and {$DLRJ_1$, $DLRJ_2$, …, $JZRJ_1$,$JZRJ_2$,…} are gotten through special clustering software[2].The common attributes set of {KH_1,KH_2,…,QK_1,QK_2,…} is {Trans.Accept, Encry.TsMac, Ans.Process, Trans.Send}, which implies the existence of crosscutting concerns such as communication, encryption or decryption and message. Similarly, {$DLRJ_1$, DLRJ2…, $JZRJ_1$, JZRJ2…} indicates the existence of two crosscutting concerns: access of database and table locking or unlocking.

Excutions	Executed Methods
KH_1	Trans.Accept(1), Encry.TsMac(1), Ans.Process(2), KH.hz_kh(1), ..,Error.Lock(1), Trans.Send(1)
KH_2	Trans.Accept(1), Encry.TsMac(1), Ans.Process(2), KH.zl_kh(1), ..,Trans.Send(1)
CK_1	Trans.Accept(1), Encry.TsMac(1), Ans.Process(3), KH.hz_ck(2), .., Trans.Send(1)
CK_2	Trans.Accept(1), Encry.TsMac(1), Ans.Process(3), KH.zl_ck (2), .., Trans.Send(1)
QK_1	Trans.Accept(1), Encry.TsMac(1), Ans.Process(3), KH.hz_qk(2), .., Trans.Send(1)
QK_2	Trans.Accept(1), Encry.TsMac(1), Ans.Process(3), KH.zl_qk(2), .., Trans.Send(1)
$DLRJ_1$	DB.Open(1), Lock.Table(5), RJ.rjemnllidr(1), Lock.UnlockTable(5), .., DB.Close(1)
$DLRJ_2$	DB.Open(1), Lock.Table(5), RJ.rjmemlcmsj(1), Lock.UnlockTable(5), .., DB.Close(1)
$JZRJ_1$	DB.Open(1), Lock.Table(8), RJ.rj1zzjzzc(1), Lock.UnlockTable(8), .., DB.Close(1)
$JZRJ_2$	DB.Open(1), Lock.Table(8), RJ.rj1hzlsz(1), Lock.UnlockTable(8), .., DB.Close(1)
	. . .

Once crosscutting concerns are obtained by clustering on Scenarios, next step is finding out the location of crosscuts. We construct transaction set and item set according to the description of 2.2. The transactions are Scenarios without nested call, which are KH_1, KH_2, …, $JZRJ_2$, …., represented by T_0, T_1, T_2…T_9…. The items are first(), Trans.Accept(), Encry.TsMac(), …, DB.Close() , end(), represented by $i_0, i_1, i_2, ..., i_{19,} i_{20}$. Especially if the same method is called several times in one Scenario, alias is used for accurately locating. For example, two appearances of Ans.Process() in T_0, is represented by two items: $i_3^{'}$ and $i_3^{''}$.Then, transaction set is showed by Table 2:

Table 1: Scenario objects set

Table 2: transaction sets

T	Items set
T_0	$i_0, i_1, i_2, i'_3, i_4, i''_3, i_{10}, i_{11}, i_{20}$
T_1	$i_0, i_1, i_2, i'_3, i_5, i''_3, i_{10}, i_{11}, i_{20}$
T_2	$i_0, i_1, i_2, i'_3, i'_6, i''_3, i''_6, i'''_3, i_{10}, i_{11}, i_{20}$
T_3	$i_0, i_1, i_2, i'_3, i'_7, i''_3, i''_7, i'''_3, i_{10}, i_{11}, i_{20}$
T_4	$i_0, i_1, i_2, i'_3, i'_8, i''_3, i''_8, i'''_3, i_{10}, i_{11}, i_{20}$
T_5	$i_0, i_1, i_2, i'_3, i'_9, i''_3, i''_9, i'''_3, i_{10}, i_{11}, i_{20}$
T_6	$i_0, i_{12}, i_{13}, i_{14}, i_{18}, i_{19}, i_{20}$
T_7	$i_0, i_{12}, i_{13}, i_{15}, i_{18}, i_{19}, i_{20}$
T_8	$i_0, i_{12}, i_{13}, i_{16}, i_{18}, i_{19}, i_{20}$
T_9	$i_0, i_{12}, i_{13}, i_{17}, i_{18}, i_{19}, i_{20}$
	. . .

Association rule mining algorithm get four rules as follows through filtration and combination under the min support degree and confidence degree (all are 40%).

(1) if i_0 then $i_1 \wedge i_2 \wedge i'_3$

(2) if $i_{10} \wedge i_{11}$ then i_{20}

(3) if i_0 then $i_{12} \wedge i_{13}$

(4) if $i_{18} \wedge i_{19}$ then i_{20}

The rule (1) proves that crosscutting concerns codes (Trans.Accept(), Encry.TsMac() and Ans.Process()) are called in turn at the entrance of business main functions (KH(), CK(), QK(), …). In the same way, the rule (3) proves that crosscutting concerns codes (Lock.UnlockTable() , DB.Close()) are called in turn at the exit of business main functions (DLRJ (), JZRJ()…). That is mining position of crosscutting concerns codes in source code.

Above all, this case study proves the effectiveness of clustering and association rule mining on execution traces.

4. Case study

Mining and refactoring Aspect from OO system is becoming hot research area, along with the maturation of AOP technology. In the existing literature, there are some works on aspect mining based on source code exploration and static code analysis [9, 10, 11, 12, 13, 14]. The Aspect Mining Tool AMT, described in [10], supports aspect identification by matching textual patterns against the names used in the code and by looking for repeated uses of the same types. The Aspect Browser tool also uses textual patterns to match the aspects [9]. Their location is improved by adopting a map-based display where aspects are shown in colors. The code-browsing tool JQuery is presented in [11], which provides hierarchical navigation and query facilities, which are useful while executing aspect extraction tasks.

Similarly to our paper, in [15] dynamic information and concept lattice are used for aspect mining. However, their approach is different at two following points compared with ours.

- One Scenario may call the same method several times. This times degree can be distributed by clustering analysis but be regarded the same by concept lattice.
- For large OO program, forming concept lattice structure needs longer time and choosing candidate Aspects manually. While, clustering gives result directly under scheduled constraints.

In addition, the automatic degree is a very important standard for Aspect Mining. At present, most methods on mining are based on static or dynamic analysis, and only crosscutting concerns codes can be found. While our technique, which is based on the dynamic analysis and association rule, may automatically locate crosscuts after identifying crosscutting concerns codes. This is advancement as compared to the earlier works. Refactoring of OO system to AO system will be easily processes based on our works.

Acknowledgments

This research benefits from the support of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University.

References

[1] Tzilla Elrad, Mehmet Aksit, Gregor Kiczales, Karl Lieberherr and Harold Ossher. “Disscutting Aspects of AOP”, Communications of the ACM, ACM Press, October 2001, 44(10), pp. 33-38.

[2] Tzilla Elrad, Robert E. Filman, Atef Bader. “Aspect Oriented Programming”, Communications of the ACM, ACM Press, October 2001, 44(10), pp. 29-32.

[3] L. Kaufman, and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, 1990.

[4] T.A.Wiggerts. Using clustering algorithms in legacy systems modularization. In Proc of the 4th Working Conference on Reverse Engineering, pages. 33-43,1997.

[5] R.Srikant, Q.Vu, and R.agrawal, Mining association rules with items constraints, In Proc. 1997 Int. Conf. Knowledge Discovery and Data Mining, pages 67-73, Newport Beach, CA,Aug.1997

[6] R.Agrawal and R.Srikant. Fast algorithms for mining association rules. In Proc. 1994Int. Conf. Very Large Data Bases, pages 487-499, Santiago, Chile, Sept, 1994.

[7] J.Pei, J.Han, and R.Mao. CLOSET: An efficient algorithm for mining frequent closed items. In Proc. 2000 ACM-SIGMOD Int. Workshop Data Mining and Knowledge Discovery, pages 11-20, Dallas, TX, May 2000.

[8] K.Sartipi, and K.kontogiannis, Component clustering based on maximal association. In Proc of the 8th Working Conference on Reverse Engineering, pages 103-114,2001.

[9] W. G. Griswold, J. J. Yuan, and Y. Kato. Exploiting the map metaphor in a tool for software evolution. In Proc. of the 2001 International Conference on Software Engineering (ICSE), pages 265–274, Toronto, Canada, March 2001. IEEE Computer Society.

[10] J. Hannemann and G. Kiczales. Overcoming the prevalent decomposition of legacy code. In Proc. of Workshop on Advanced Separation of Concerns at the International Conference on Software Engineering (ICSE), Toronto, Canada, 2001.

[11] D. Janzen and K. D. Volder. Navigating and querying code without getting lost. In Proc. of the 2nd International Conference on Aspect-Oriented Software Development (AOSD), pages 178–187, Boston, Massachusetts, USA, March 2003. ACM press.

[12] N. Loughran and A. Rashid. Mining aspects. In Proc. of the Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (with AOSD), Enschede, The Netherlands, April 2002.

[13] M. P. Robillard and G. C. Murphy. Concern graphs: Finding and describing concerns using structural program dependencies. In Proc. of the 24th International Conference on Software Engineering (ICSE), pages 406–416, Orlando, FL, USA, May 2002. ACM press.

[14] A. van Deursen, M. Marin, and L. Moonen. Aspect mining and refactoring. In Proceedings of the 1st International Workshop on Refactoring: Achievements, Challenges, Effects (REFACE), with WCRE, Waterloo, Canada, November 2003.

[15] P. Tonella, M. Ceccato, Aspect Mining through the Formal Concept Analysis of Execution Traces, in: Proceedings of the Working Conference on Reverse-Engineering (WCRE), 2004.

Lili He received the B.S. and M.S. degrees in Computer Science and Technology from Jilin University in 1998 and 2001, respectively. She is now an instructor of college of computer science and technology, Jilin university, majors in software reverse engineering.