A Comparison of the Efficiency of Applying Association Rule Discovery on Software Archive using Support – Confidence Model and Support – New Confidence Model

Sunchai Pitakchonlasup, and Assadaporn Sapsomboon, *Member, IACSIT*

***Abstract*—Software archives contain historical information about the development process of a software system. Using Association rule discovery, the development patterns can be extracted from these archives. This information is useful to support software modification activities, as indicated to software developers which modules are usually modified together during software maintenance or evolution. All previous works focused on mining associations with classical interestingness measure, support-confidence, where some disadvantage existed. The new Interestingness Measure, named as support-new confidence, was proposed by Liu *et al.* to improve the classical method. In this research, we present the comparison of the efficiency of applying association rule discovery on software archive using classical model and Liu *et al.*'s model. The experiments were conducted on software archive of KMyMoney software, an open source financial software project. The results show that the efficiency of the rules obtained in new model is higher than the rules obtained in classical model in navigation scenario.**

***Index Terms*—Association rule discovery, measures, version control system, software archives interestingness.**

I. Introduction

Version control systems automatically generate sources of software change history, software archive. They can be mined to identify associations between software module modifications [10][11]. This information is the basis to identify useful patterns of software module defect occurrences, associations, modification, evolution and decay. For this reason, software engineering research has been exploring software archives to understand software projects [2][8]. With these archives, it is possible to detect logical evolution coupling [3][4], extract function call usage patterns [5], find common error patterns [7], predict and suggest likely changes and prevent errors due to incomplete changes [10].

All association rule discovery in software archive researches using support-confidence framework, $conf\ (x \rightarrow y)$ is used to measure the interestingness of rule $x \rightarrow y$. However, sometime this model retrieves the uncorrelated rules and misleads to define interesting rule [5][7][8].

In 2009, Liu *et al.* analyzed some problem of interestingness measure on the classical association rules model, support-confidence model, and then proposed a new interestingness measure for mining association rules based on sufficiency measure of uncertain reasoning, called "new confidence", to improve the classical method of mining association rules [6]. Liu *et al.* showed that the new measure not only captures correlation but can also detect negative implication [6].

For the above reasons, we are interested in applying Liu *et al.*'s support-new confidence model to association rule discovery in software archive. This paper presents the comparison of the efficiency of applying association rule discovery on software archive using classical model and new model. The efficiency test of this research follows an approach similar to what was proposed by Zimmermann *et al.*, as in [11]. It mines software archive to characterize software entities modification associations for: (1) predicting and suggesting further changes to be made in files entities, named as navigation scenario; (2) preventing incomplete changes, named as error prevention scenario; and (3) preventing erroneously recommendations of (1) and (2), named as closure scenario [11]. The results indicate that the efficiency of the rules obtained in new model is higher than the rules obtained in classical model in navigation scenario.

This paper is organized as follows. Section 2 discusses related works. Section 3 presents the experiment planning and definition. Section 4 describes the experiment. Section 5 describes, analyzes and discusses the validity of the obtained results. And finally, Section 6 concludes the research result and proposes future work.

II. Related Works

This section discusses the studies in mining data from version control repositories. Those researches may diverge on the purpose of analysis, mining technique employed, amount of analysed data and algorithms used.

In 2003, reference [9] developed an approach that used association rule mining on software archives. It especially evaluated the usefulness of the results, considering a recommendation most valuable if it could not be determined by program analysis, and found several such recommendations in the MOZILLA and ECLIPSE projects.

Reference [4] was the first to use software history to detect logical coupling between program modules in 1998. Reference [3] proposed a method to identify and visualize classes and class coupling that are the most change prone.

Reference [5] proposed a general method that used a data mining technique to extract implicit programming rules from large software code written in a C programming language. Reference [7] analysed source code check-ins to find highly

Manuscript received July 1, 2012; revised July 29, 2012.

Sunchai Pitakchonlasup is with the Appsphere Group Co.,Ltd., Bangkok Thailand. (e-mail:dz.yoez@gmail.com).

Assadaporn Sapsomboon is with Business Software Devlopment Program, Faculty of Commerce and Accoutancy, Chulalongkorn University, Bangkok, Thailand. (e-mail:assadaporn@ acc.chula.ac.th).

correlated method calls as well as common bug fixes in order to automatically discover application-specific coding patterns.

Reference [11] proposed a tool to mine software modification association rules, called ROSE. Their tool mined associations between files and finer-grained entities. It also supported mining on the fly, by presenting the developers the suggestions that apply to the software modules being modified at a given time.

All the researches mentioned above have one thing in common. They all mined software archives with support-confidence model. Two of them [5][7] showed that the support-confidence model discovered the false positive rules which lead to false outcome if applied.

In 2009, Reference [6] analysed some deficiency of interestingness measure on the support-confidence framework and then proposed a new interestingness measure for mining association rules based on sufficiency measure of uncertain reasoning, named as "new confidence", to decrease the false positive rules from mining association rules with the classical method.

This research is motivated by the work of [11] and [6]. Our work is concerned with mining for software module modification association rules using Liu *et al.*'s support-new confidence model [6].

III. Experiment

This paper presents our work as an experimental research. This section will focus on the experiment definition and planning. The following sections will present the experiment execution and data analysis.

A. Goal Definition

Our main goal is to compare the efficiency of applying association rule discovery on software archive using support-confidence model and Liu *et al.*'s support-new confidence model in three main scenarios to support programmers in software coding. The three scenarios are:

- Navigation [11]: Given a single changed entity, can the system point programmers to entities that should typically be changed, too?
- Error prevention [11]: Can the system prevent errors? Say, the programmer has changed many entities but has missed to change one entity. Does the system find the missing one?
- Closure [11]: Suppose a transaction is finished—the programmer made all necessary changes. How often does the system erroneously suggest that a change is missing?

B. Planning

Context selection: For our experiment, we analysed the archive of one large open-source project which can retrieves changes and transactions from CVS, the most popular open-source version control system, archive. We chose the software project named KMyMoney [14], which is the popular open-source personal financial manager software. This project is developed with C++ Programming Language in Qt framework since 2000. Its CVS archive contains more than 30000 transactions on more than 2000 files.

Hypothesis Formulation: We are interested in the efficiency of file association rules application in KMyMoney project. We selected 60 transactions as a test set. Each of transactions (△) were selected from the statistical characteristics, size and frequency of appearance, to be the representatives of transactions. These test sets is used for checking whether its modified files can be predicted from earlier action:

- A test case q = (Q, E) was created, consisting of a query $Q \subset \triangle$ and an expected outcome $E = \triangle - Q$. We referred to a set of all queries as Z. In our work we created 451 test cases for Navigation scenario, 451 test cases for Error Prevention scenario and 60 test cases for Closure scenario from 60 transactions in the test set;
- Only the top ten rules of the all rules R were considered: $R_{10} \subset R$ ranked by confidence / new confidence;
- The union of consequent item sets of R_{10} of test case q was defined as A_q.
- $A_q \cap E_q$ will be the files that matched the expected outcome and will be considered correct;
- $A_q - E_q$ will be unexpected recommendations which are wrong.

After that, we used two information retrieval measures [12]. The first measure is the precision P_q. It describes which fraction of the returned files was actually modified by the programmer. The second measure is recall R_q. It indicates the percentage of modified files that were returned [8].

$$P_q = \frac{|A_q \cap E|}{|A_q|}, \quad R_q = \frac{|A_q \cap E|}{|E|} \tag{1}$$

In the case that no item was returned (A_q is empty), we defined the precision as $P_q = 1$. In the case that no item was expected, we defined the recall as $R_q = 1$

The harmonic mean between precision and recall, the F-measure$_q$ [13], assures the balance of the model characterization.

$$\text{F-measure}_q = \frac{2 * P_q * R_q}{P_q + R_q} \tag{2}$$

F-measure$_q$ can be calculated to measure the efficiency of file association rules in navigation and error prevention scenarios [8][11]. For closure scenario, we measured the efficiency by calculating the percentage of queries where the system makes at least one recommendation. We referred to this percentage as the Feedback.

$$\text{Feedback} = \frac{|Z^*|}{|Z|} \tag{3}$$

Z^* is a subset of Z which makes a non empty recommendation set.

Hence, the hypotheses we were trying to confirm are:

Hypothesis H^{NAV}: Software archive mining based on files association rules using support-new confidence model (2) has Navigation F-measure higher than using support-confidence model (1).

$$H^{NAV}: \mu_2(\text{F- measure}) > \mu_1(\text{F- measure}) \tag{4}$$

Hypothesis H^{PRE}: Software archive mining based on files association rules using support-new confidence model (2) has

Error Prevention F-measure higher than using support-confidence model (1).

$$H^{PRE}: \mu_2(\text{F- measure}) > \mu_1(\text{F- measure}) \quad (5)$$

Hypothesis H^{CLO}: Software archive mining based on files association rules using support-new confidence model (2) has Closure Feedback lower than using support-confidence model (1).

$$H^{CLO}: \mu_2(\text{Feedback}) < \mu_1(\text{Feedback}) \quad (6)$$

IV. Experiment Operation

The classical approach to compute association rules is the *Apriori Algorithm* [1]. The Apriori Algorithm takes a minimum support count (or minimum support) and a minimum confidence and computes the set of all association rules that are above both thresholds.

All experiments were given a minimum support count of 3 and minimum confidence / new confidence of 0.1.

Formally, we define:

- The *frequency* of a set x in a set of transactions D as

$$\text{freqD}(x) = |\{t \mid t \in T, x \subseteq t\}| \quad (7)$$

- The probability of a set x in a set of transactions D as

$$\text{pD}(x) = \frac{freq_D(x)}{|D|} \quad (8)$$

- The *support count* of rule $x \rightarrow y$ define by a set of transactions D as

$$\text{supcD}(x \rightarrow y) = \text{freqD}(x \cup y) \quad (9)$$

- For file association rule mining using support-confidence model, the *confidence* is used to determine the interestingness of the rule. The confidence of rule $x \rightarrow y$ define by a set of transactions D as

$$\text{conf}(x \rightarrow y) = \frac{p_D(x \text{ and } y)}{p_D(x)} \quad (10)$$

- For file association rule mining using Liu *et al.*'s support-new confidence model [6], the *new confidence* is used to determine the interestingness of the rule. The new confidence of rule $x \rightarrow y$ define by a set of transactions D as

$$\text{nconf}(x \rightarrow y) = \frac{p_D(x \text{ and } y)}{p_D(y)} - \frac{p_D(x \text{ and } \bar{y})}{p_D(\bar{y})} \quad (11)$$

The set of suggestions for a situation S and a set of rules R are defined as the union of the consequents of all matching rules:

$$\text{applyR}\{S\} = \bigcup_{(S \rightarrow y) \in R} y \quad (12)$$

V. Results

The experimental results are shown in Table 1. These results suggest that the association rules obtained from the new framework are better than those obtained from the classical framework in navigation scenario, but are worse than the classical framework in error prevention scenario and are equal in closure scenario. Next, we tested this evidence using inferential statistics.

TABLE I: The Experimental Results

Model	Efficiency of applying association rule discovery		
	Navigation (F-measure)	Prevention (F-measure)	Closure (Feedback)
Classical	0.3013	0.1353	$\frac{41}{60}$
New	0.3245	0.1135	$\frac{41}{60}$

A. Analysis and Interpretation

First, we analyzed navigation and error prevention scenario. For the statistical testing, we established a significance level (α) of 0.05. The Kolmogorov-Smirnov Test can be used to ensure that the sample is normally distributed. As seen in Table 2, we found p-values (1-tailed) of 0.013, 0.006, 0.000 and 0.000 for the navigation's and error prevention's classic and new model F-measure, respectively. It can be concluded that the data is not normally distributed.

TABLE II: Kolmogorov-Smirnov Test for Navigation and Prevention

	Navigation		Prevention	
	Classic	New	Classic	New
N	451	451	451	451
Kolmogorov-Smirnov Z	1.481	1.600	6.644	6.914
Asymp. Sig. (2-tailed)	0.025	0.012	0.000	0.000

Thus, we applied the Wilcoxon Signed Rank Sum Test for the Matched Paired Difference shown in Table 3. In navigation scenario, we obtained the p-value (1-tailed) of 0.000, significantly lower than 0.05, and the Z of -4.374 based on negative ranks accepting our hypothesis that software archive mining based on files association rules using support-new confidence model has higher navigation efficiency than using support-confidence model. In error prevention scenario case, we obtained the p-value (1-tailed) of 0.000, significantly lower than 0.05, and the Z of -6.055 based on positive ranks concluding that software archive mining based on files association rules using support-new confidence model has no difference in error prevention efficiency from using support-confidence model.

Finally, we applied Two Proportion Z Tests for closure scenario. The result is shown in Table 4. As the p-value is the lowest possible significance with which it is possible to reject the null hypothesis H_0^{CLO}. We found $Z = \sqrt{\text{Pearson Chi} - \text{Square}} = 0.000$ and p-value (1-tailed) of 0.5, larger than 0.05, that mean software archive mining based on files association rules using support-new confidence model is not different in closure efficiency from using support-confidence model.

TABLE III: Wilcoxon Signed Rank Sum Test for Navigation and Prevention

	Navigation	Prevention
	M_{NEW}-$M_{CLASSIC}$	M_{NEW}-$M_{CLASSIC}$
Z	-4.374[a]	-6.055[b]
Asymp. Sig. (2-tailed)	0.000	0.000

a. Based on negative ranks.
b. Based on positive ranks.

We can infer that the effectiveness of file modification association rules using support-new confidence model is higher than using support-confidence model only in navigation scenario.

TABLE IV: TWO PROPORTION Z TEST FOR CLOSURE

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	0.000[a]	1	1.000		
Continuity Correction	0.000	1	1.000		
Likelihood Ratio	0.000	1	1.000		
Fisher's Exact Test				1.000	0.578
N of Valid Cases	120				

a. 0 cells (0%) have expected count less than 5. The minimum expected count is 19

B. Validity and Reliability

In spite of having achieved statistical significance in the study, one must consider the following validity and reliability of our study:

- The software project that we used its archive for analysis is a popular open-source personal financial manager software, having a number of transactions per week and stable for over 10 years;
- We selected the test sets from the historical transactions, represented real actions in the past, and generated queries and expected results from them [11];
- The efficiency evaluation that we used are the calculated F-measure and Feedback, which are widely used in information retrieval research [12].

VI. CONCLUSIONS AND FUTURE WORK

This article shows that software module modification association rules derived by using the support-new confidence model were more effective than those derived by using the support-confidence model in navigation scenario. We found that in navigation scenario, the classical model can generate a lot of false positive rules in high rank position while the new model can generate only a bit of false positive rules. But in error prevention and closure scenarios, the classical and new model can derive near or same number of false positive rules. Our results indicates that association rules discovery using support-new confidence model can also be effectively applied in software archives such as the one we studied.

We believe that this finding may stimulate further studies in the support-new confidence model. And hopefully, motivate them to invest on association mining to support software maintenance and evolution activities.

The future work, we want to detect the ordering of changes or sequence rules such as "Given a single change, can system point programmer to entities that should typically be changed in ordering, too"? Moreover, we believe that the new association rule mining model can effectively applied for fine granularity associations, analyzing variables, methods and classes.

REFERENCES

[1] R. Agrawal and R. Srikant. "Fast Algorithms for Mining Association Rules" in *Proc. 20th Very Large Data Bases Conf. (VLDB).* 1994, pp. 487-499.

[2] T. Ball, J. M. Kim, A. A. Porter, and H. P. Siy. "If Your Version Control System Could Talk" in *Proc. ICSE Workshop on Process Modelling and Empirical Studies of Software Eng*., 1997.

[3] J. M. Bieman, A. A. Andrews, and H. J. Yang.. "Understanding Change-Proneness" in *OO Software through Visualization. Proc. 11th Int'l Workshop Program Comprehension,* 2003, pp. 44-53.

[4] H. Gall, K. Hajek, and M. Jazayeri. "Detection of Logical Coupling Based on Product Release History" in *Proceedings of the 26th International Conference on Software Maintenance (ICSM '98)*, 1998, pp. 190-198.

[5] Z. Li, and Y. Zhou.. "PR-Miner: Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code"in *Proceedings of 13th International Symposium on Foundations of Software Engineering (ESEC/FSE'05),* 2005, pp 306-315.

[6] J. Liu, F. Xiaoping, and Q. Zhihua. "A New Interestingness Measure of Association Rules. Genetic and Evolutionary Computing" in. *WGEC '08. Second International Conference*, 2008, pp. 393-397.

[7] B. Livshits, and T. Zimmermann.. "DyanMine: Finding Common Error Patterns by Mining Software Revision Histories" in *Proceedings of 13th International Symposium on Foundations of Software Engineering (ESEC/FSE'05)*, 2005, pp 296-305.

[8] C. J. Methania, M. Manoel, and R. Francisco. "Mining software change history in an industrial environment" in. *XXIII Brazilian Symposium on Software Engineering,*. 2009

[9] A. T. T. Ying. "Predicting source code changes by mining revision history". Master's thesis, University of British Columbia, Canada, Oct. 2003.

[10] T. Zimmermann and P. Weißgerber.. "Preprocessing CVS Data For Fine-Grained Analysis" in *Proc. Mining Software Repositories*, 2004, pp. 2-6.

[11] T. Zimmermann, P. Weißgerber, S. Diehl, and A. Zeller.. "Mining Version Histories to Guide Software Changes" in *Proceedings of the 26th International Conference on Software Engineering*, 2005, pp. 563-572.

[12] C. J. van. Rijsbergen. *Information Retrieval*, 2nd edition. Butterworths, London. 1979

[13] M. Grossman and R. Katz. "A new approach to means of two positive numbers" *International Journal of Mathematical Education in Science and Technology*, vol. 17, no. 2, 1986, pp. 205 – 208.

[14] KMyMoney project. Available: http://kmymoney2.sourceforge.net/

S. Pitakchonlasup was born in Ratchaburi, Thailand. He received his bachelor degree in Computer Science from Chulalongkorn University, Bangkok Thailand in 2007. He received his master degree in Business Software Development from Chulalongkorn University, Bangkok Thailand in 2011.

He is currently a software developer in iOS department at Appsphere Group Co.,Ltd., Bangkok Thailand.

A Sapsomboon was born in Bangkok, Thailand. She received her bachelor degree in Statistics from Chulalongkorn University, Bangkok Thailand in 1982. She continued her study and received her Master degree in Computer Science from the Pennsylvania State University, University Park, PA, and Ph.D. in Information Science from the University of Pittsburgh, PA, in 1984 and 1989 respectively.

She is currently an assistant professor in Information Technology at the Faculty of Commerce and Accountancy, Chulalongkorn University, Bangkok THAILAND. Her research interests are software development process and information retrieval.