AUTOMATICALLY EXTRACTING UNDOCUMENTED PROGRAMMING RULES AND DETECTING VIOLATIONS USING PR-MINER

K SUNITHA 1*, S V N SRINIVASU 2*

1. Research Scholar, Dept of CSE, ACHARYA NAGARJUNA UNIVERSITY, AP.
2. HOD, Dept of CS, PNC & KR PG COLLEGE, NARASARAOPET, AP.

Abstract: From the past three decades, the most widely using word is Data mining; it is the procedure to determine the different patterns from the large data sets, which involves methods of the junction like statistics, machine learning and other data base sets. While dealing with large data sets like big data, hadoop and some other applications, programmers should tend to pay the attention of their programs which have some implicit programming rules. If a programmer was unaware of that rules and violate the methods automatically it leads to system violation. If programmers have adequate knowledge on programming solving this is not a deal. To overcome this sort of unexpected violations many methods were introduced. One of the methods is PR-mining using the frequent item set. This paper deals with the technique of PR-mining which efficiently deals with the extracting of implicit programming rules from the large data sets like Hadoop, big data, C, Postgre SQL and some other applications. By using frequent item set Mining (FIM) we can easily trace out the unexpected violations.

Key Words: Data Mining, Frequent Item set Data base constraints, support count and patterns.

I. INTRODUCTION
Generally, Most of the programmers will follow the inherent programming rules. Programmers will develop the programs which include functions, algorithms, different methods and constraints. Every program is related to set of data base which includes some functions, tables or relevant data to that particular program. During the development of programs sometimes bugs will be detected and it distracts or interrupts the whole system. To overcome this unexpected violation and for the bug detection researchers will developed many methods. This paper deals with a common method PR-Miner (Programming Ruler Miner), it uses one of the data mining technique named as Frequent Item set Mining (FIM). FIM is used to remove effectively the implicit programming rules from the huge soft ware codes, which were written in machine language or industrial programming languages. While dealing with the large databases and large software codes like Postgre SQL, linux, big data. Apache HTTP server, for each they have 84k to 3M lines of code. Within 2minutes, PR-miner will efficiently takeout the thousands of programming rules and identify the violations. PR-miner reported almost 60 top violations, now all these bugs are fixed by developers.

PR-Miner is a tool used to frequent item set mining to take out the implicit programming rules from large source codes like C. Every program has a programming rule for call to lock and followed by unlock; For example if we take Linux, Postgre SQL and other applications can observe how some functions will be repeated and the impact of those functions will violate the system; Postgre SQL is popular open source database server, having implicit programming rule like “Search Sys Cache” followed by “Release Sys Cache” proceeds a copy of cache for a specific sequences. Once caller finished their work with tuple, it calls another function ‘Release Sys Cache’ to liberate it; thus, in cache, this copy will be replaced by other data. In figure 1 in Postgre SQL, this rule will be appeared for 209 times, if any code
breaches this rule, it leads to memory leak in PostgreSQL.

Figure 1: In PostgreSQL a function pair rules extracted by PR-Miner, which appears 209 times in code.

If we take example in Linux, sometimes some composite rules could indicate variable correlations, means these can be modified and accessed together in a reliable manner. In Linux code two variables named as “ic. Command” and “ic. Driver” should be accessed together. This rule will be appeared ninety eight times in Linux.

Figure 2: In Linux, variable correlation of programming rule which appears 98 times in code.

In this paper, a propose method called PR-Miner (programming rule Miner) used by data mining technique to extract the common programming rules from the software code written in C language. PR-Miner can extract thousands of programming rules from the software like Linux contains three thousand five hundred files and a sum of three million lines of code in one minute. When compared with the previous techniques [1] can extracts only function pair based rules. Whereas by using PR-Miner can take out more rules, cause PR-Miner significantly generalized by two ways. One is general method and another one is general rules. In general method, without any basic knowledge about software or without any templates, marginal or foot notes, explanations by programmers , PR-Miner will extracts the rules automatically; and it can be modified to employ along with the programs like java, C, etc… In general rules, the rules extracted by PR-Miner are general. Programming rules includes variables, functions, and data types. So, PR-Miner will not only extract the simple pair wise but also the complex rules.

2. BACK GROUND OF DATA MINING

PR-Miner is depends on a technique is called Frequent Item set Mining [2],[3] widely used in so many applications like gaming, business, statistics, DNA sequences etc. In FIM, the base data will be taken in the form of transaction (sets of instances) having number of features also named as items. If there is a huge number of item set in data base, contains a sub item set more than a specific threshold (min-support) of item sets is considered as Frequent. The number of occurrences of sub item set A is defined its support, the items in set A is called as supporting item sets.

Let’s take an example of a 4 sets; A = { (p,q,r,s,t) (p,q,r) (q,r,s) (p,r,t) (p,r,s,u,v) } the supporting item sets are 4; (p,q,r,s,t) (p,q,r) (q,r,s) (p,r,t) (p,r,s,u,v) here p: 4; q: 3; r:5 , s: 3, t:2; u:1,v:1 ;To overcome the frequent item set problem, few algorithms been developed, one of them is FP- Tree based mining algorithm named as FP Close [3]. In general it produces the complete set of frequent item sets; here FP-Close do only closed sub item sets; this will improve the time and performance of the space, it avoids producing exponential number of frequent sub items. Association rules will be generated after completing the procedure of closed frequent item set. This rule of association is denoted with X ⇒ Y; with confidence ‘C’ and support ‘S’ where X and Y are item sets. The meaning of this rule is if item set contains X, it also contains Y with the possibility of C(2,3). This rule allows the violation detection. If confidence is very high (99%) , means the item set contains only X, but not Y which violates the rule indicates prospective outlier.
3. LITERATURE REVIEW

Pan bian and et.al [5] to improve the efficiency of code mining they proposed a new method called EAntMiner. In this approach they reduced the noises from the statements which are not related to the rules and implementation forms for the identical language. To tackle the bugs which are difficult to be identified by mining frequent patterns as rules in this paper they developed a KNN based method and implemented EAnt Miner to evaluate large scale C systems. EAnt miner detects 105 bugs confirmed by development communities this approach improves the precision of data mining code. By Zhenmin Li and Yuanyuan Zhou [6] proposed a method called PR-miner used a data mining approach called frequent item set method to extract the programming implicit rules and violations in large software sources and codes. They worked on Postgre SQL, Apache server, Linux, C etc… they gave a good research work on this PR-Miner. Martin Monperrus, Mira Mezini et.al [7] dealt with the deviant codes and implicit rules, to detect the missing method call a new characteristic of the deviant code is generated here. This is happens basing on type-usages in first section they concentrated on problems related to the particular missing method secondly they focused on source code repository. They found a technique called DMMC to detect the missing method calls in object oriented software to find and support to solve the missing methods entire software life time. Bernard, Vincent et.al [8] proposed in a paper called Requirements Modelling and Formal Analysis using Graph Operations describes about the systems life cycle and complexity taking place in data mining, in this paper a method called formal approaching method was proposed to take the better representations of reality of the basing on the system environmental characteristics. Which is takeout the proper specifications, mechanisms, constraints to give the better results. It captures the system implicated and given specifications correctly and generates the accurate information, this was mostly used in enterprise modeling. In this paper Akito Monden et.al [9] presents, while dealing with the software sometimes unexpected implicit coding rules will be generated which are rarely written in specified or designed documents. If programmer unaware of those things violation occurs adding to this new functionality and some more other faults will be taken place. To overcome this authors proposed a method, defines that first identify the reason, conditions and code fragments for every fault shown in bug reports to identify the implicit coding rules. Bug patterns briefed pattern description language after that faulty code fragments will be taken out by a pattern matching technique.

4. PR-MINER OVER VIEW

PR-Miner had two key functionalities one is automatically takeout the implicit programming rules and the other one is detecting the violations. This rule extracting will be done among the elements which includes variables, function, data, etc all will focused on source code. To find out the program element correlations PR-Miner will convers the problem in to frequent item set mining problem, here it parse the source code of the software. So each program will be hashed by number, defined in a row and turned into item set of the data base. it contains many item sets. Here FP close algorithm will be used to find the frequent sub item sets; these will be used to interfere the programming rules to identify this a method called Programming pattern was introduced. After the extraction of programming patterns from FIM, PR-Miner will generate programming rules from the patterns; main theme of this one to identify the number of cases which contains the items on left side but not the right side. After producing the P-Rules PR miner will store them in particular files for later reference. PR-Miner had numerous benefits one is generality and others are time efficiency and space efficiency; In Generality close frequent item set algorithms are not limited to number of items and also does not need ant rule templates. In time efficiency they avoid scanning the data for too many times which reduces the time. In space efficiency they include many other rules for further process and accuracy.
To extract the Patterns in PR-Miner users generally go for two methods:

a) Parsing Source code: In order to convert programming pattern extraction into frequent itemset mining problem a item set will be built in database. This will modified by using GCC compiler and this will work on only C, and extended for other programming languages which replaces the GCC front ends.[10]

b) Mining for programming patterns: If a set of numbers appear together in any item sets for more than a specific threshold number of times this sub item set is called frequent. It applies on closed frequent sets. FP Close algorithm is used here. After knowing the closed programming patterns and its supported values occurs, programmer should record the functions. The original algorithm FP close and any other sets were not designed for the purpose. So, we enhance FP close to address this problem by maintained and supported item sets during the mining process.

c) Generating programming rules: Sometimes patterns lead to several different rules. Consequently, to produce rules from patterns depends on the conditional probabilities. There are two types of methods

- **A Naïve Method**: To produce the programming rules from taken out patterns to divide the items in each closed frequent sub itemset into 2 parts and then calculate the confidence. The main problem is that it needs to scrutinize the possible rules from every mined pattern. A programming pattern with K elements can produce up to \((2^K-2)\) rules which is not possible for long patterns.

- **Generating closed patterns**: To minimize the number of output outcomes and to fasten the generation as well as the detection of violation process, PR-Miner keeps the closed rules in condensed format. It is denoted with

\[
I:s|[C_1:s_1\geq s] \ldots [C_m:s_m\geq s]
\]

where \(C_1, \ldots, C_m\) are all subsets of \(I\) which supports \(S_1, \ldots, S_m\) are different from \(I\)’s and larger than \(S\). Such condensed format can represent all the closed rules derived from \(I\) equals to \(C_i\) (i.e. a subset of \(I\) with a support larger than \(I\)), the confidence of the rules is \(s/s_i\); otherwise, the confidence of the rule is 100%.

The problem arises here is that how to find the all from subset \(C_i\) which support \(s_i\) larger than \(s\). Since that support of \(C_i\) is larger than \(s\), it indicates that \(C_i\) should be contained in another closed frequent sub-itemset. The solution is here Pr-Miner will once again converts the problem back to the frequent sub item set mining where FP Close will be used one more time to find common frequent sub item sets which were generated by the first pass of FP close. Closed Rules will perform better when compared to naïve rule algorithm in terms of time and space because it need not require to testing the all possible rules which are generated from extracted programming patterns.

V. DETECTING VIOLATIONS TO EXTRACTED RULES & LIMITATIONS

Basing on the generated programming rules, PR-Miner will find out the impending bugs by detecting violations to the rules. The main theme of these programming rules normally holds for most cases and violations which happen occasionally.

a) Detecting Violations: Naïve method is used to detect the violations in programming rules, where it generates all possible programming rules and it checks the source code line by line. But it takes lot of time, so first, if rule has low confidence means it already trims in the rule generation step, secondly, if rule has confidence of 100% means no violation takes place in that rule. So we require check the rules with confidence having the range of \([t, 100\%]\). PR-Miner stores produced programming rules in condensed format which explicitly indicates the rules have less confidence but higher than specific threshold \(t\), so we can easily trace out the rules which contain violations, by calling closed rules, PR-Miner will do this action even more efficiently.
b) Pruning False Violations: Sometimes the violation detection will lead to false positives if elements in the particular programming rule duration transversely multiple functions because it uses only intra procedural analysis. To reduce the false violations, PR-Miner performs the inter procedural checking, where firstly it check the all callee’s paths for every function which possess violations. If the missed items found in any callee’s path it is a false violation, for time efficiency it limited to checking depth. During checking it is easy to follow the calling path. To check in backward direction PR-Miner maintains a caller list for each function, F denotes as the index of functions if any missing item pointed for functions in the paths of all the callers it means it is a false violation.

c) Ranking and reporting bugs: After detecting the violations and prunes false positives it ranks all the left violations and it reports to the programmers. Basing on the confidence it will give the ranks. PR-Miner will grouped the violations basing on the same functions, if the violation have highest confidence means it has bugs. The potential of these bugs are correlated. In future it may be solved by introducing some other approaches like correlation ranking for better accurate ranking.

Limitations: Even though PR-Miner is very effective in extract the implicit programming rules and identifies the violations automatically, it has some drawbacks.

- Due to copy and paste most of the times violations will occurs, due to this PR-Miner could not identify this error, to overcome this, the previous technique will be used called CP-Miner, which recognizes the copy-pasted codes for each group and supports PR-Miner.
- The noisy effect of the Macros is another drawback of PR-Miner, In C, Macros defined as false programming rules or false negatives in identifying the violations. Here GCC will pre process the source code by expanding the macros. Due to this several duplicate macros will be formed every macro will be taken as one rule, so failing report takes place, to remove this kind noisy effects of macros, have to consider each macro as a individual element using the method named Refactoring.
- PR-Miner used the compiled information from front end GCC, where some of the functions use the same name due to this PR-Miner could not identify them and it leads to violations, to conquer this, during PR-Miner converts the source code of itemset database, programmer should link the information. So PR-Miner can differentiate the function names having same names.
- When PR-Miner converts the whole functional definition to itemset, it could not identify the programming rules straddling across multiple function definitions. To recognize this PR-Miner should combine along with inter procedural analysis similar to the false positive pruning.
- Due to some control paths sometimes PR-Miner will go with the false negatives in identifying the violations. To rectify this, need to access some methods from model checking to ensure distinct control paths.

VI. CONCLUSION

This article deals with the method called PR-Miner which used frequent itemset mining. From the large software codes written in C with minor efforts from the developers, we deal how to extract the undocumented and implicated programming rules and identify the violations automatically? Here we discussed about the rules taken out by PR-Miner with two general forms one is simple pair wise and another one is complex with multiple elements of several types. Using this PR-Miner method we identify the many implicit programming rules and identified the violations in Postgre-SQL, Linux. PR-Miner is an effective and usable tool. In future, it can be applied to many programming languages like java, and others.
REFERENCES

[8] Requirements Modelling and Formal Analysis using Graph Operations by Bernard kamso-fogen and Vincent https://hal.archives-ouvertes.fr/hal-00353794