International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 4 (2017), pp. 559-574

http://www.ripublication.com

A Review Paper on Software Engineering Areas Implementing Data Mining Tools & Techniques

Ankit Dhamija
Amity School of Engineering & Technology,
Amity University Gurgaon, India.

Sunil Sikka*
Amity School of Engineering & Technology,
Amity University Gurgaon, India.

Abstract

Software organizations produce data in huge amount while creating and developing software. Each stage of software development witnesses a different set of data that gets produced right from the requirements phase toll maintenance. Software organizations, in their continuous jest of improving the software quality put in considerable efforts to collect and maintain the data generated in software repositories. This huge data maintained in software repositories is analyzed using various Data Mining techniques which are used in extracting novel patterns and highlights from data. This area has attracted the attention of researchers in recent times and has become a preferred interdisciplinary research area of Software Engineering and Data Mining by researchers. In this review paper, the researcher tries to explore the various application areas of data mining in software engineering, different types of software engineering data where data mining is being and can be applied and various tools of data mining that are available and have been used by researchers to solve their related problems. Finally, based on this classification, the researcher tries to zero out the most important software engineering area that is garnering the researchers' attention.

Keywords: Data mining, software, engineering, hidden patterns, software engineering data

* *Corresponding author*

1. INTRODUCTION

The field of software engineering continues to remain the heartthrob for academic researchers from past several decades. The popularity of software engineering areas and their relevance has not reduced a bit, in fact, the onus on software developers to deliver a high quality software has increased with time in today's competitive world where software has become inseparable part of our lives in some form or the other. So many areas of software engineering are being explored by researchers and practitioners like software quality, metrics, software configuration management, testing & debugging etc to name a few.

Over the past few years, the focus of researchers & practitioners has shifted to interdisciplinary research areas and the most popular area in this context is the mix of Data mining and software engineering as Data mining is one such field which provides a lot of scope for research as the several techniques of It can be very useful in extracting novel & valuable information from the gathered data. It is the process of finding hidden patters from the data stored in several databases which can be from any of the databases-operational, distributed, external, hypermedia databases. Several techniques available in Data Mining allow researchers to fetch novel patterns from data stored in these databases. This data extraction helps organizations to make business decisions.

Apart from these regular databases, a lot of data is produced during software development that can be used to extract valuable information using data mining techniques which will further improve the quality of software under development. This Software Engineering Data (SE Data) generated from various stages of software development is very useful and it is stored in what is known as Software Repositories.

Halkidi et al [1] stated that the data in these SE repositories can come from any of the stages of software development such as from documentation, software configuration management data, program versions, source code, execution traces, compiled code, bug reports, discussion forums, mailing lists and email communications etc. The data in all these repositories becomes of a great value to software organizations because they realize the fact that by applying several data mining techniques, they can get great insights into the software under development. Popular Data Mining Techniques like clustering, which is an unsupervised learning technique and used for discovering groups & identifying interests & distribution, classification which is a supervised learning technique is used for mapping a data item into several predefined classes. Then there are techniques like Association rules, text mining, data characterization and change & deviation detection that can be used to mine different types of data available in several software repositories.

A lot of researchers have put considerable amount of efforts to categorize various Data Mining techniques that can be used in software engineering areas, some have tried to generalize various SE data on which such techniques may be applied to fetch valuable information, some have focused on various Data Mining tools that can be used to apply mining algorithms and techniques on SE data and so on. In other words,

a lot of quality work has happened in the past in this very interesting interdisciplinary field.

However, the authors here feels that there is an immediate need to put all the work done by earlier researchers into a consistent format that covers all aspects ranging from various SE repositories, data that is available from SE repositories, data mining techniques available, various tools that are available and tools which are best to use on different kinds of mining techniques and on what kind of SE data such tools can be applied. In this review paper, the author presents an overview of the connected research areas of software engineering and data mining. The author tries to explore the various software engineering areas where data mining can be applied, and what tools should be applied on what kind of SE data and what tools are best for using a particular data mining technique. Finally, based on this classification, the author tries to identify the most important SE area which is garnering the most attention. Section 2 of this paper describes the various Software Engineering data which can be mined into. Section 3 discusses mostly used tools by researchers. Section 4 presents a three way mapping of the various SE data available, various mining techniques/algorithms and various tools that can be used on that SE data by using those mining techniques/algorithms. Section 5 explores the best preferred area by the researchers.

II. SOFTWARE ENGINEERING DATA THAT CAN BE MINED

A software project that goes under development passes through different stages in its life cycle and generates a lot of data at every stage which can be structured or unstructured. All these different types of data can be mined using various mining techniques available. M. Halkidi et al [1] described documentation, software configuration management data, source code, compiled code & execution traces, issue tracking & bug databases and mailing lists as the most important software engineering data sources. A.E. Hassan et al [2] categorized sources of SE data as SE repositories namely historical repositories, run time repositories and code repositories. Chaturvedi et al [3] in their research paper, talked about CVS logs, test suites, s/w libraries, database repositories, Git repositories, GitHub, Junit and log4J as other important data sources that can be mined.

In the table given below, the authors have tried to give a brief description about each of these SE data sources.

Table I: Various SE Data Sources

SE Data	Description
Documentation	Includes external documentation(s/w description, startup & usage configuration, user guide, license issues), internal documentation(pdf's, html files), multimedia data (audio video instructions) etc.
SCM Data	It includes software codes, source documents, models used for software design, information related to accounting, defect

	tracking, revision and control data
Source code	It is the most important source of data where data mining techniques can be applied to analyze software components, to help maintenance tasks and much more.
Compiled code & Execution Traces	The object code for static analysis in software engineering and can be used for malicious software detection through data mining techniques. Execution traces is a chain of events that occurs when software components are tested and can be used by mining techniques.
Issue tracking & Bug Databases	These databases contain the low level details of the issues and bugs that are found in various software modules.
Mailing Lists	These contain communication history between several people who are involved with the software project like developers, testers, users etc. These can also be quite useful for text analysis.
Historical repositories	These cover achieved communications among project stakeholders, error listings and source control repositories.
Runtime repositories	It includes deployment logs which covers information about the software execution at different times during its development period.
Code repositories	Online repositories of software projects code like sourceforge.net, google code etc
CVS logs	Concurrent Version Systems logs are used to store a version control history—information about details of files changed.
Test Suites	Various test cases for testing software to depict the similar ways in which it behaves.
S/w Libraries	Include already available code segments written in various programming languages. Data mining may be applied to analyze the different types of libraries used for different software projects.
Database Repositories	A single centralized location where databases of various types and kinds and from different sources are kept and used for a specific data mining purpose.
Git Repositories	A version control system for software development and for remote file storage like GITHUB

III. TOOLS MOSTLY USED BY RESEARCHERS

To achieve mining tasks in software engineering, researchers prefer to use already existing tools; some researchers developed their own tool to work on software

engineering data. The researchers used these tools for the purpose of data extraction from repositories, to filter data, pattern finding, learning and prediction.

Chaturvedi et al [3], in their review paper categorized data mining tools in four categories such as newly created tools, traditional data mining tools, prototype developed/implemented and scripts

Table II: Tools According to Different Categories

Types	**Tools**
Newly Developed	Exempler, cloneTracker, sqminer, CallExtractor, J-REX on MapReduce, REXML, FPLearner, FPClassifier, EvoOnt, iSPARQL, DeepIntellisense, Operation Recorder, PopCon, Anchored Signature matching and rationalize, checkstyle, Spotweb, Apache Pig, Mamoset, Evolizer, Cp-Miner, ReSpam, SPIRIT(Sequential Pattern Mining with regular Expression constraint), ExceptionTracer, PR Miner ,Perracotta , PatchMiner,MAPO, XSnippet, Strathcona, BugLocator, BLIA**,** Binary Analysis Tool,HitoshiIO, EQMiner, PRMiner, CHRONICLER, PROSPECTOR, Deckard, NICAD, Boa,
Traditional Tools	CHANGEDISTILLER and EVOLIZER suite, WEKA toolkit,CodeMiner, CodeBroker, R Tools, RapidMiner, GNUPlot, Matlab, CCFinder, SourceMonitor, MALLET, CTAGS[], , UCINET, UNDERSTAND, Unix Diff utility, CVSAnalY tool, GIT and GitMining Tools,FLOSSmole, Simian, SAS Text Miner,Statistica, JBoss, Lex, Yacc, GrouMiner, Hipikat, GAR, KEEL
Prototype	Bug report assignment on eclipse, Automatic Labeling of Software Components in Java, How developers work together in Tomcat and JUnit, defect correction effort using the defect data collected from a Japanese multi-vendor information system development project COSE, which methods are cloned and which methods are changed in each transaction in DNSJava, Renaming recommendation system
Scripts	Eclipse, FreeBSD, GNOME, Apache and Apache Ant, Java Generics adoption, SeaHawk

IV. MAPPING OF SE DATA WITH THE TOOLS

In this section, the authors will provide a detailed mapping of the various types of software engineering data discussed one of the above sections and the various tools that can be applied on that type of SE data. This will be performed as below: Here we

classify the different types of SE data on which various mining approaches can be used and the best tools for those approaches.

Table III: Tools Mined with SE Data

SE Data	**Mining Approach Used**	**Purpose**
Documentation	Classification	Requirements elicitation
SCM Data	Information retrieval, Classification	Requirements elicitation Usage changes, Functionality analysis
	Clustering	Bug Tracking Maintenance(bug identification)
	Searching/matching	Extracting data patterns from historical data and data under development.
	Regression	Usage changes, Functionality analysis
Source code	Clustering	Software Processes Extracting significant Patterns from source code within similar classes, functions and data, System Modules Clone Detection and categorization
	Text retrieval	Bug tracking
	Decision trees, Frequent pattern mining and association rules	Detection of logical errors, bugs tracking, Decision tree of failed executions
	Classification	• Classes of commits FP and NFP modules classifier • Stability of prediction models
	Frequent pattern mining and association rules	• Detection of logical errors, bugs tracking • Failures Prediction • Correlation between entities,

		identification of update operations in code segments. • Design alternatives (source code of legacy systems) • Usage changes (instantiation code)
	Differencing, Mining based on Statistics	Syntactic and semantic changes
Compiled code & Execution Traces	Clustering	Clusters of execution profiles Software behaviour classifiers
	SVM classification	Debugging
	Classification	Software behaviour classifiers
Issue tracking & Bug Databases	Regression, Classification, Frequent Pattern Mining & Association Rules	Usage changes Functionality analysis
Historical repositories	Differencing, Mining based on Statistics	Syntactic and semantic changes
CVS logs	Data Summarization	Requirements elicitation
	Classification, Frequent Pattern Mining & Association Rules	• Stability of prediction models Failure Prediction • Correlation between entities, identification of update operations in code segments.
Test Suites	Probabilistic classification	Detection of logical errors
S/w Libraries	Frequent Pattern Mining & Association Rules	Reused patterns

Huffman et al [4] proposed technique based on information retrieval. They proposed an approach based on improving the extraction of high level and low level requirements by taking the documents' universe as being the union of the design elements and the individual requirements and they map the problem of requirements tracing into finding the similarities between the vector space representations of high level and low level requirements, thus reducing it into an IR task. Further, in

continuation to this, the authors in [5] worked upon identifying the elements that impact the behavior of analysts while they work on results produced by data mining tools in software engineering.

German et al [6] proposed tools called SoftChange, which uses textual data from various open source software and applies data validation on them which performs retrieval, summarization and validation functions.

Jensen et al [7] also worked on open source software and proposed a way to discover software processes. They used text extraction techniques, entity resolution and social network analysis.

Williams & Hollingsworth [8] proposed a method based on change history of source code repository to improve static analysis techniques used to find bugs.

Morisaki et al [9] proposed an approach based on association rules extraction techniques to analyze defect data. Software defects include bugs, specification and design changes. By collecting defect data, they applied an extended association rule mining method to extract useful information and proposed rules associated with defect correction effort.

Chang et al [10] proposed graph mining techniques to find out conditional rules in a code base and to extract rule violations that indicate neglected condition. Rules were represented using dependence graphs and frequent item set mining and frequent sub graph mining techniques were used to discover conditional rules.

Dickinson et al in [11] proposed a method based on clustering technique to filter profiles based on similar characteristics that exploit the cluster analysis methods to select the set of executions that will be evaluated for conformance to requirements. Execution profiles are then selected from the remaining clusters.

An approach proposed by last et al[12] propose to automate the input output analysis of execution data based on the *infofuzzy network* (*IFN*) which has an treelike structure.

Bowring et al in [13] approach analyze a collection of programs' executions and define classifiers of software behavior. They used Markov models to encode the execution of profiles of projects. Then the Markov models of individual program executions are clustered using an agglomerative clustering algorithm.

Liu et al [14] used a statistical approach to develop a mining algorithm to help programmers in manual debugging.

Liu et al [15] proposed another method based on data mining which analyze logical bugs. Here, they considered program executions as software behavior graphs and developed a method to integrate closed graph mining and SVM classification.

Kannelopoulos et al [16] presented an approach based on clustering techniques for knowledge acquisition from source code in order to comprehend an object oriented system and evaluate its maintainability.

A static verification tool called Checkers was developed by Engler et al [17] that was based on rule mining using compiler extentions and written in Meta Language. It compares frequently occurring programming errors with a code base by checking all function calls, pointer references, etc.

Li and Zhou[18] proposed PR-Miner which used itemset mining to extract general programming rules from C programs and detect violations. It used hashing to convert function definitions into numbers. FPclose algorithm was used for extracting rules from functions, data types and variables etc.

A tool called MUVI was proposed by Lu et al [19] that was based on frequent itemset mining. It mines variable pairing rules to detect bugs of two types: update bugs and concurrency bugs because of the possibility of inconsistent update of variables which are accessed together.

Baker[20] developed Dup for proposing an approach for code clone detection that detects code clones line by line. This tool finds variables, constants and other keywords which are used at several locations and then divides them into tokens through a lexical analyzer, replaces them into a parameter identifier, encodes them according to their occurance, represents all the prefixes using a suffix tree and finally, extracts the matches using suffix tree algorithm as a clone gets detected when two suffixes have a common prefix.

Kamiya et al [21] proposed CCFinder, a tool for identifying code clones. Their tool converts source code into tokens sequence through lexical analyzer.

Wahler et al [22] proposed a tool which converts source code into Abstract Syntax Tree (AST) which contains complete information about source code by using parser which is than transformed into XML representation.

Qu et al. [23] proposed a framework for pattern mining of clone code using spatial and graph base analysis of source code. Source code is first transformed into Program Dependency Graph (PDG), where original source code gets converted into encoded graphic sequence.

Basit and Jarabek [24] proposed *Clone Miner* that finds clones from the similar code fragments in source code. Frequent itemset mining is then used to detect again and again occuring groups of simple clones in different method and files. Then clustering is used to find File clone sets and method clone set

Li et al [25] proposed CP-Miner that uses mining on operating system code to find out copy paste defect. It used frequent subsequence mining and tokenization technique. It also used hash functions to compute a hash value and used CloSpan algorithm to find copypasted segments.

A tool called CodeWeb for discovering library usage patterns was proposed by Michail [26] that used association rule mining to identify classes anf dunctions that occur together frequently in library usage.

Sahavechaphan and Claypool [27] developedA tool called XSnippet was developed by Sahavechaphan and Claypool [27], which is a code assistant tool & plug-in and

compatible with Eclipse platform. It allows developers to search for required code segments from a code repository to find code segments relevant to their programming task.

Xie and Pei [28] developed MAPO which mines frequent API usage patterns through class inheritance. It works on API's access history and identifies methods calls.

Zimmermann et al.[29], developed a tool called *Rose* which works by mining historical changes in source code and use association rules as a technique to identify co-occuring changes..

Hassan and Holt [30] proposed a method for tracking changes of entities. Different types of Heuristics (developer based, history based, code layout based, file based and process based) are proposed that are used to predict the entities that need to be changes on the basis of another entity.

V. TOOLS USED IN DATA MINING

In this last section, we will present a summarized analysis of the tools that have been mostly used by the researchers according to the various SE areas discussed previously.

Table IV: Tools Summarization

Tool	Purpose	Where
Clone tracker	For mining code clones	Source code
Sqminer	to mine the sequences of changed-file and to process Subversion log entries.	Source code
CallExtractor	to analyze go programs. It extracts calls made to imported packages	Source code
J-REX	an optimized evolutionary code extractor,	Source code
FPClassifier	For training and classifying software modules	Source code
EvoONT	a software repository data exchange format based on the Web Ontology Language (OWL). EvoOnt includes software, release, and bug-related information.	Software repository mining projects
Isparql	Semantic Web query engine containing similarity joins for analyzing the extent of change between versions or to detect bad code smells.	Software repository mining projects
Deep	a tool for rehydrating evaporated information. It works as a Visual Studio plugin which	Source code,

Intellisense	consolidates and presents historical information about code.	documentation
SpotWeb	a web-based version of Spotnet and requires an existing HTTP server. Works on PHP5 (preferred is PHP 5.3 and up) to implement its functions, Compatible on Linux and Windows.	API
PopCon	POPcon™ Exchange POP3 Connector connects your Exchange server to your internet mailboxes. Over the time, it downloads emails from POP3 and IMAP mailboxes and distributes them to exchange mailboxes.	Emails, Mailbox searches
CP-Miner	A Tool for Finding Copy-paste and Related Bugs in Operating System Code which uses mining techniques to efficiently identify copy-pasted code in large software including operating systems, and detect copy-paste related bugs.	Source Code
PatchMiner	For frequent pattern mining, itemset mining	
Binary Analysis Tool	For analysis of binary files in source code	Source code
Checkstyle	Static code analysis tool to check java code for coding rules	Source code
Apache Pig	Tool that work on Big data, with a provision of multiple query approach, that results in less number of data scan iterations.	Big Data
Mamoset	Used on software repository mining, source code to find new bug patterns**.**	Source code
Evolizer	a platform for mining software archives	Software repositories
ChangeDistiller	a tool for change extraction and analysis	Source code
ReSpam	Sequential mining on strings, pre-processing tool to convert texts to sequences of integers	Text documents
SPIRIT	Sequential Pattern Mining with regular Expression constraint	
Exceptiontracer	an Eclipse plugin to resolve exceptions related to the stack trace in Java programs	
PR Miner	a tool that mines programming rules from source	Source code

	code	
Perracotta	a dynamic analysis tool for Mining Temporal API Rules from Imperfect Traces	API
MAPO	Mining API Usages from Open Source Repositories	API, software repositories
XSnippet	a tool for querying a sample repository for relevant code snippets for programming.	Source code
Strathcona	a system using multiple search to query for source code examples	Source code
BugLocator	an information retrieval based method for locating the relevant source code files for bug fixing based on an initial bug report.	Source code
BLIA	Bug Localization with Integrated Analysis. A Tool using stack traces in bug reports, information from source files, code change histories.	Source code
HotoshIO	attempts to overcome the problem of different data structures through a fuzzy equivalence matching. HitoshiIO compares the input/output of functions while observing their executions in-vivo.	
EQMINER	A C language tool that takes random input and dynamically executes relevant functions to detects clusters of similar code.	Source code
PR-MINER	Uses frequent itemset mining to extract implicit programming rules from large C code	Source code
CHRONICLER	a tool supporting remote debugging. by capturing program execution sounds. It guarantees accurate replay in the lab, with very low overhead.	Source code
PROSPECTOR	Analyzes Python code and offers information about errors, violations and complexity.	Source code
DECKARD	a tree-based, scalable, and An accurate code clone detection tool which is scalable and tree based and has provisions for uncovering clone related bugs.	Source code
NICAD	Automated Detection of Near-Miss Intentional Clones , code clone detection using sequential pattern mining	Source code

Boa	A Large Scale Software Repository.	Source code
KEEL	Knowledge Extraction based on Evolutionary Learning)It is an open source Java tool that can be used for different knowledge data discovery tasks	Software repositories
RapidMiner	an IDE for machine learning, data mining, text mining, predictive analytics and business analytics.	Text documents
SAS Text Miner	For text mining and analysis	Text documents
CCFinder	a C++ based token-based code clone analysis tool	Source code
SourceMonitor	A C++ tool to find out code quantity and modules complexity.	Source code
MALLET	*A* Java-based package for statistical natural language processing, document classification, clustering	Source code, text documents
CHANGEDISTILLER	enables fine-grained change type extraction and analysis to reason about coding conventions, control or exception flow, and even code and comment coevolution.	Source code

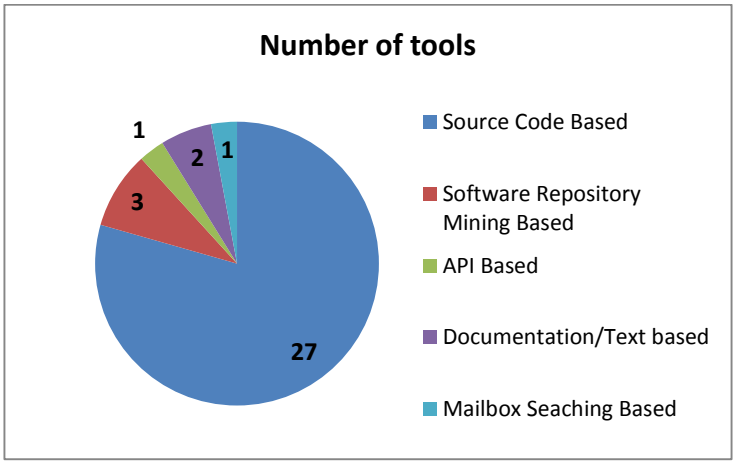

Fig 1: Pie Chart of Various Tools Categories

It can be deduced from the above table that more than 75% of the tools and techniques developed and proposed by researchers focus on the source code while remaining other tools are based on other SE data like text documents, mailboxes, API's, other software repositories and projects.

VI. CONCLUSION

In this paper, the researchers have tried to provide a three way analysis of various Software Engineering Data that are in focus, categorized the various types of tools and a summarized analysis of various Data Mining Tools and their focus area in Software Engineering data sources. The extensive literature survey and the tools analysis depicts that most of the work till today and most of the tools proposed by researchers till today has been on software code, thus there is a great scope of work in other areas of software engineering where data mining and its techniques can prove to be quite helpful in uncovering important patterns that may in turn assist developers, testers, maintenance teams and other people related to Software Engineering in some way or the other.

REFERENCES

[1]. M. Halkidi, D. Spinellis, G. Tsatsaronis *et al.*, "Data mining in software engineering," *Intelligent Data Analysis,* vol. 15, no. 3, pp. 413-441, 2011.

[2]. A. E. Hassan, and R. C. Holt, "Predicting change propagation in software systems," in *Proceedings of the 20th IEEE International Conference on Software Maintenance*, 2004, pp. 284-293.

[3]. Chaturvedi K.K, Singh V.B, Singh P, "Tools in Mining Software Repositories", 13th International Conference on Computational Science and Its Applications, pp. 89-98, 2013

[4]. J. Huffman Hayes, A. Dekhtyar and J. Osborne, Improving requirements tracing via information retrieval. In *Proceedings of the International Conference on Requirements Engineering*, 2003.

[5]. J. Huffman Hayes, A. Dekhtyar and S. Sundaram, Text mining for software engineering: How analyst feedback impacts final results. In *Proceedings of International Workshop on Mining Software Repositories* (*MSR*), 2005.

[6]. D. German and A. Mockus, Automating the measurement of open source projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering* (*ICSE03*), 2003.

[7]. C. Jensen andW. Scacchi, Datamining for software process discovery in open source software development communities. In *Proceedings of International Workshop on Mining Software Repositories* (*MSR*), 2004.

[8]. C.C.Williams and J.K.Hollingsworth,Automatingmining of source code repositories to improve bug finding techniques, *IEEE Transactions on Software Engineering* **31**(6) (2005), 466–480.

[9]. S.Morisaki,A.Monden andT.Matsumura, Defect data analysis based on extended association rulemining. In*Proceedings of International Workshop on Mining Software Repositories* (*MSR*), 2007.

[10]. R Chang, A. Podgurski and J. Yang, Discovering neglected conditions in software by mining dependence graphs, *IEEE Transactions on Software Engineering*, 2008.

[11]. W. Dickinson, D. Leon and A. Podgurski, Finding failures by cluster analysis of execution profiles, *International Conference on Software Engineering* (*ICSE*), 2001.
[12]. M. Last,M. Friedman and A. Kandel, *The Data Dimining Approach to Automated Software Testing*, In Proceeding of the SIGKDD Conference, 2005.
[13]. J. Bowring, J. Rehg and M.J. Harrold, Acive learning for automatic classification of software behavior, *International Symposium on Software Testing and Analysis* (*ISSTA*), 2004.
[14]. C. Liu, X Yan, and J. Han. Mining control ow abnormality for logical errors. In *Proceedings of SIAM Data Mining Conference* (*SDM*), 2006.
[15]. C. Liu, X. Yan, H. Yu, J. Han and P. Yu, Mining behavior graphs for 'backtrace' of noncrasinh bugs. In *SIAM Data Mining Conference* (*SDM*), 2005.
[16]. Y. Kannelopoulos, Y. Dimopoulos, C. Tjortjis and C. Makris, Mining source code elements for comprehending object oriented systems and evaluating their maintainability, *SIGKDD Explorations* **8**(1), 2006.
[17]. D. Engler, D. Chen, S. Hallem *et al.*, "Bugs as deviant behavior: A general approach to inferring errors in systems code," *ACM SIGOPS Operating Systems Review,* vol. 35, no. 5, pp. 57-72, 2001.
[18]. Z. Li, and Y. Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code,*" in Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, 2005, pp. 306-315.
[19]. S. Lu, S. Park, C. Hu *et al.*, *"MUVI: automatically inferring multi-variable access correlations and detecting related semantic and concurrency bugs,"* ACM SIGOPS Operating Systems Review, vol. 41, no. 6, pp. 103-116, 2007.
[20]. B. Baker, "On finding duplication and near-duplication in large software systems," in *Second IEEE Working Conf on Reverse Eng.(wcre)*, 1995, pp. 86-95.
[21]. T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, pp. 654-670, 2002.
[22]. V. Wahler, D. Seipel, J. Wolff *et al.*, "Clone detection in source code by frequent itemset techniques," in Fourth IEEE International Workshop on Source Code Analysis and Manipulation, 2004, pp. 128-135.
[23]. W. Qu, Y. Jia, and M. Jiang, "Pattern mining of cloned codes in software systems," *Information Sciences*, 2010.
[24]. H. A. Basit, and S. Jarzabek, "A data mining approach for detecting higher-level clones in software," *IEEE Transactions on Software Engineering*, pp. 497-514, 2009.
[25]. Z. Li, S. Lu, S. Myagmar *et al.*, "CP-Miner: A tool for finding copy-paste and related bugs in operating system code," in *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*, 2004, pp. 20.

[26]. A. Michail, "Data mining library reuse patterns using generalized association rules," in *Proceedings of 22nd International Conference on Software Engineering (ICSE'00)*, Limerick, Ireland, 2000, pp. 167-176.

[27]. N. Sahavechaphan, and K. Claypool, "XSnippet: mining for sample code," *ACM SIGPLAN Notices,* vol. 41, no. 10, pp. 413-430, 2006.

[28]. T. Xie, and J. Pei, "MAPO: Mining API usages from open source repositories," in *Proceedings of the international workshop on Mining software repositories*, 2006, pp. 54-57.

[29]. T. Zimmermann, P. Weisgerber, S. Diehl *et al.*, "Mining version histories to guide software changes," *IEEE Transactions on Software Engineering,31(6)*, pp. 429-445, June 2005.

[30]. A. E. Hassan, and R. C. Holt, "Predicting change propagation in software systems," in *Proceedings of the 20th IEEE International Conference on Software Maintenance*, 2004, pp. 284-293.