Systematic Literature Review on Mining Software Repositories

Arvinder Kaur, Kamaldeep Kaur, Deepti Chopra and Harguneet Kaur
University School of Information and Communication Technology (U.S.I.C.T), Guru Gobind Singh Indraprastha University (G.G.S.I.P.U), New Delhi, India
Email:arvinder@ipu.ac.in, kdkaur99@ipu.ac.in, dchopra27@gmail.com, harguneet.13316490018@ipu.ac.in

Abstract
Mining Software Repositories (MSR) explores the complex software systems to unfold valuable and interesting knowledge. The systematic literature review is performed on MSR studies published over a decade. The review is conducted on 300 selected papers. The study aims to identify – i) popular application areas, ii) popular tools, iii) datasets and software projects, along with their type and SDLC phase in which they are used, iv) popular techniques, v) top conferences and journals along with the types of studies and vi) current trends, of MSR research. The important results implied from the reviewed studies are: 1) Bug Prediction (25%) and Change Prediction (17%) are the most popular application areas of MSR, 2) Data Modelling and Statistical Tools, (19%) were employed in most of the reviewed studies, 3) About 71 % of reviewed studies use datasets from open source repositories, 4) There is a large number of mining techniques with Classification techniques (29%) dominating the field, 5) The most popular conference for MSR is the International Conference on Mining Software Repositories (MSR), and 6) Research gaps in evolving application areas of clone detection, code reuse and software evolution are identified.

Keywords: Systematic literature review, Mining software repositories, Software Repositories, Software Engineering

1. INTRODUCTION

Mining Software Repositories (MSR) is a dynamic research field that deals with extracting and analyzing the data available in software repositories. As the data is readily available with bug tracking repositories, source code repositories, control versioning systems(CVS) and mailing lists, this emerging field has extended its reputation since 2004, with the first MSR workshop and sustains to be a growing research area in software engineering.Most of the Research work done in software engineering uses the data available in various software repositories. Some frequently examined areas include software defect prediction, determining software change patterns, the study of code clones, code reuse, software evolution and software performance analysis.
The purpose of this study is to explore the various techniques and tools that have been majorly used for MSR, determine popular approaches for MSR, identify the commonly explored repositories of the software projects and find out the current trends in the MSR research area. Our study not only investigates the literature in the context of software evolution but also reviews papers on all application areas of MSR. Apart from examining the techniques, repositories and methodology, our study also identifies the popular tools used for mining data repositories and determines the Software Development Lifecycle (SDLC) phase at which data is extracted.

Huzefa Kagdi et al. lead “A survey and taxonomy of approaches for mining software repositories in the context of software evolution”. Their paper examines the MSR field based on the four views: type of software repository, the aim, the approach used, and the assessment method. All application areas are explained in detail by the survey. They also described the usage of metrics in MSR.

K.K. Chaturvedi et al. carried out a review of “Tools in Mining Software Repositories”. Quality papers from the Mining Software Repositories (MSR) conferences/journals from 2007 to 2012 were reviewed and analyzed by K.K. Chaturvedi et al. [272]. In their work [K.K. Chaturvedi et al.], techniques, data sets and tools used are identified. Their paper categorizes different tools on the basis of the application areas and the datasets it works on. This review considers papers published in the proceedings of MSR conference and other related conferences/journals from the year 2003 to 2018.

Hadi Hemmati et al., gathered comments from MSR proceedings between 2005 and 2012 and analyzed and categorized them. After analysis, four themes were determined namely: “data acquisition and preparation, synthesis, analysis, and sharing/replication.” The trends in specific theme were identified and explored. The review in [Hadi Hemmati et al.]

provides a public forum consisting of the extracted patterns for enhancing the interaction among the MSR community for evolving best methodologies. However, none of the above papers [Huzefa Kagdi et al., K.K. Chaturvedi et al., Hadi Hemmati et al.], follow the systematic approach for conducting the Systematic Literature Review (SLR) according to the principles established by Kitchenham and Charters [Kitchenham and Charters], commonly recommended in software engineering systematic reviews. Our current paper follows the SLR principles [Kitchenham and Charters] and identifies which techniques and tools are prominently used in previous research, determining what all approaches for MSR have been studied, identify the most commonly explored repositories of the projects, explore the recent trends and identifies the research gaps in various application areas of MSR.

Some of the issues which have not been identified by the above mentioned papers [Huzefa Kagdi et al., K.K. Chaturvedi et al., Hadi Hemmati et al.] are answered in our paper including the quality assessment (QA) [Kitchenham and Charters] of the reviewed papers. QA is an essential step to discover valid and high quality papers for our analysis. The previous reviewed studies [Huzefa Kagdi et al., K.K. Chaturvedi et al., Hadi Hemmati et al.] did not identify the SDLC phase at which the repository is mined where as our study examines MSR at each SDLC phase. It is important to determine the correct SDLC phase for mining the required data. This study correlates the application areas and datasets of MSR with the SDLC phase from which data needs to be extracted, thereby providing a framework for new researchers who want to work in MSR research area.

This paper is composed as follows. Section 2 describes the methodology used, Section 3 discusses the review results, Section 4 provides the implications for research and practice, Section 5 discusses the threats to validity and Section 6 provides conclusion and future work.

2. METHOD

The Systematic Literature Review (SLR) process given by 'Kitchenham and Charters' is followed in our current study. In the earliest planning phase of the SLR, the systematic review protocol is formulated. The formulated protocol consists of six major steps:

- Research questions
- Search protocol
- Study selection
- Quality assessment
- Data extraction
- Data synthesis.

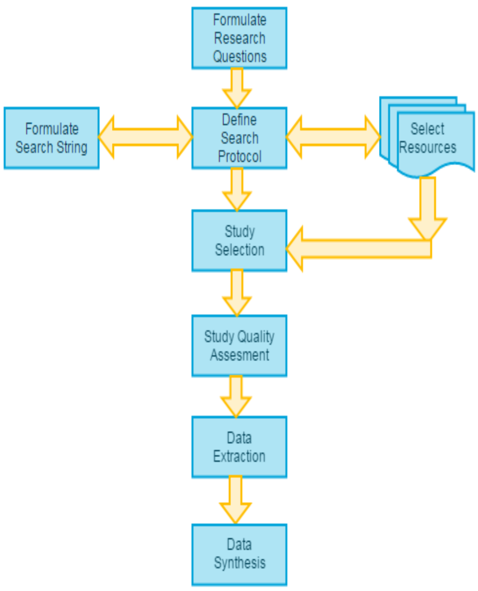


Figure 1 Steps of Review Model

Figure 1 depicts the six steps of the systematic review model. In the first step, research questions were developed. In the second step, the search strategy was planned, to find out the primary research papers related to the questions formulated in step 1. It involves identifying the search string and determining the databases to be searched. Then in the third step, the study selection criteria was planned to find the appropriate studies that contribute significantly to the research questions. Pilot study selection was also applied at this step to further filtering of the selected studies. In step 4, the selected papers were assessed for quality according to the defined quality checklist mentioned in section 2.3. Then at the data extraction step, data extraction card is designed and refined using pilot data extraction i.e. the form is tested on a representative sample of studies to be reviewed and is refined to collect desired information. The data extraction card is then planned to be used for extracting data from the selected studies. Finally, in the data synthesis step, the proper process for synthesizing the data to extract useful information based on the research questions is to be determined. A systematic review model is crucial for an SLR. The subsections 2.1, 2.2, 2.3, 2.4, 2.5 describe the review model in detail.

2.1. Research questions

The formulation of research questions (RQ) is done in such a way so that any researcher who wants to work in the field of MSR gain knowledge of popular application areas, the tools available for MSR, the datasets along with its types and software projects on which previous studies have been conducted. It also determines the SDLC phase in which dataset is used, popular MSR techniques implemented by researchers and conferences and journals in which different types of MSR papers are published. These RQ provides future directions and framework for new researcher. 6research questions (RQs) addressed in this study are:

i. **RQ1: Which application areas have been explored in MSR papers?**
RQ1 aims at identifying the application areas in which MSR has been used, for example bug prediction, change prediction, clone detection, code reuse, software performance analysis, software evolution, etc. It helps in examining the current trends in MSR research along with analyzing popular application areas for further research in MSR field.

ii. **RQ2: Which are the tools used to mine repositories?**
RQ2 tries to identify popular mining repository tools for the researchers and the software industry for extracting information from different repositories. Researchers in the field of MSR use different types of tools such as eclipse plugins, bug tracking tools, diff, parser, statistical tools, extraction tools and others. We also analyze the repositories on which these tools work, along with the application areas in which these tools are used.

iii. **RQ3: Which datasets and software projects are used for MSR studies? What type of datasets are used and in which SDLC phase dataset is available?**
Datasets such as production data, archived data between project personnel, source code data and bug tracking systems are utilized to record the evolution of various software systems. Researchers and developers identify the perks of mining the repositories for controlling defects and changes in future software development, improving design and development issues so as to increase the efficiency of projects. So the knowledge of popular datasets is essential for an effective MSR research. It is important to analyze which datasets are primary i.e. directly collected and which are secondary i.e. indirectly collected. We also identify the type of datasets used i.e. Open Source, Industrial or Partial This helps in analyzing which datasets are available freely to researchers specially those belonging to academics.
Popular software projects are also recognized in our study so that they can be used in other studies as a benchmark. It is easier to conduct comparative studies based on previously studied software projects. It is also significant to determine the software life cycle phase at which data sets are available. It will help researchers in analyzing at which phase of SDLC most datasets are available for extraction.

iv. **RQ4: What are the most investigated techniques for MSR?**
Researchers want to develop and identify mining techniques that can be used for extracting hidden and valuable information from unstructured software datasets. Our study investigates the popular mining techniques used by researchers. It also identifies which techniques are used in which application areas, thereby providing the researchers working in those areas with a list of useful mining techniques.

v. **RQ5: What are the different types of MSR studies? Which journals and conferences are popular in the field of MSR research?**

MSR papers are categorized as theory, survey, experiment, review, development of tool, development of mining technique and case study. MSR community is growing at fast rate which has an impact on software research. It is important to distinguish different types of studies so as to comprehend the current challenges which will reflect the future work for all leading research in this area.

This SLR also determines the popular journals and conferences for MSR. This is the most beneficial research question for the researchers working in the field of MSR as it provides them with the details of relevant MSR conferences and journals. Researchers in MSR field follow a quantitative methodology for answering research questions. In last few years, MSR field is gaining its ground and has become one of the expanding area of software engineering.

vi. **RQ6: What are the research gaps in various application areas of MSR?**

Research gaps recognize the missing elements in the current research in a particular field of study. Finding gaps in the literature is a difficult task but it enlightens the areas that have scope for exploration and research in future. These research gaps should be filled by new research in the field of MSR.

2.2. Search protocol

The search protocol of SLR includes selection of search string and database resources which are explained in the following subsections.

2.2.1. Search string

In order to determine the search string, we have used the following procedure:

- We searched majorly used terms from the research questions.
- Derived alternative synonyms and spellings of these terms.
- Identified the keywords in related papers.
- The Boolean OR operator is used to combine the synonyms of the main terms.
- Boolean AND operator is used to join the main terms.

The keywords and their synonyms which are used for searching the research papers are mining, mine, software, systems, repository, repositories, evolution, analysis, change, bug, fault, defect and prediction. Thus search string finally formalized is given below:

Search string:

[(‘‘Mine’’ OR ‘‘Mining’’) AND (‘‘Software’’ OR ‘‘Systems’’) AND (‘‘Repositories’’ OR ‘‘Repository’’)]

OR

[(‘‘Software’’ OR ‘‘Systems”) AND (‘‘Evolution’’ OR ‘Analysis’’ OR “Refactoring”)]

OR

[(“Change” OR “Bug” OR “Fault” OR “Defector “Clone”) AND (“Prediction” OR “Detection”]

OR

[(“Code” OR “Software” AND (“Reuse” OR “Reusability”]

2.2.2. Selection of resources

The six electronic databases which are searched are shown in Table 1. Studies are extracted from these databases using our search process. The search process comprises of the following phases:

Table 1 Selected Databases

Source	**Location**
IEEE Explore	http://ieeexplore.ieee.org
ACM Digital Library	http://portal.acm.org
Springer Link	http://www.springerlink.com
MSR	http://www.msrconf.org

Google Scholar	http://scholar.google.co.in
Elsevier(Science Direct)	http:// www.elsevier.com

Search phase 1: The six electronic databases were searched using the above defined search string to obtain papers related to the field of MSR. The inclusion and exclusion criteria were applied on the extracted studies to select relevant studies.

Search phase 2: We applied snowballing analysis for finding new papers through the analysis of the references of the papers extracted using the search string. The inclusion and exclusion criteria were applied on these related papers to further identify the relevant studies. The inclusion criteria (IC) and exclusion criteria (EC) of the systematic literature review are as follows:

Inclusion criteria:

- The studies on the field of MSR.
- The studies that propose or compare MSR techniques.
- The studies that utilize MSR techniques to develop efficient methods for different MSR application areas.
- The studies that explain the development or implementation of a software tool that can be used for MSR.

Exclusion criteria:

- The studies that are not related to MSR.
- The studies that are not written in English.
- Summaries of tutorials, panels, editorials, posters and prefaces.

562 papers were initially found on MSR. The after applying IC and EC, 321 studies are selected for further reading. This search process is also known as replication package.

Table 2 provides statistics of obtained studies, studies included after removing duplicate studies, and the rate index. Rate Index is defined as proportion of included studies to the total number of studies considered for the systematic review.

Table 2 Obtained and Included Papers

Database	Obtained	Included	Rate Index
IEEE xplore	195	137	42.68
ACM digital	155	67	20.87
Springer link	60	38	11.84
MSR	78	49	15.26
Elsevier	53	24	7.48
Google Scholar	30	6	1.87

2.3. Study quality assessment

The Quality Assessment (QA) criteria were enforced to the research papers so as to select the papers with acceptable quality, which were eventually used for data extraction. A number of questions were formulated to assess the quality and relevance of the selected papers. QA questions are defined on the basis of different parameters: research questions, application areas, tool, dataset, techniques and research methodology.

These QA questions are presented in Table 3. Each question has one of the three answers: ‘‘Yes’’, ‘‘Partly/Not Relevant’’, or ‘‘No’’ which have been given score as: ‘‘Yes’’ = 1, ‘‘Partly/Not Relevant’’ = 0.5, and ‘‘No’’ = 0.For a given paper, its quality score was calculated by summing up the scores of the answers to the QA questions. The studies which have the quality score greater than 8 are considered reliable with acceptable quality and further used for the subsequent data extraction and data synthesis. The summary of quality assessment is presented in Table 11 in Appendix A.

Table 3 Quality Assessment Questions

Parameter	No.	Questions
Research Questions	QA1	Are the research questions clearly stated?
	QA2	Are the research questions justified in the context?
	QA3	Are all the research questions answered?
Application Area	QA4	Does the paper contribute significantly to the application area?
Tool	QA5	Is the usage of tools adequately described?
	QA6	If the study involves tool development, then is the development of a tool explained clearly?
	QA7	If the study involves tool development, then is its performance assessed?
	QA8	Is the tool benchmarked against existing tools?
Dataset	QA9	If the study involves data collection, are the data extraction methods defined?
	QA10	If the study involves analysis, are the data sets adequately described?
	QA11	Is the data available or can be collected by the method defined?
Techniques	QA12	Is the purpose of the technique defined clearly?
	QA13	Is the result of the used technique clearly stated?
	QA14	Is the result of the technique compared with other techniques?
Research Methodology	QA15	Is research methodology clearly described?
	QA16	Are the threats to validity of the study clearly presented?
	QA17	Does the results of the study are further added to the literature?
	QA18	Does the study discuss the issues related to validity/reliability of their measures?

2.4. Data extraction

After the selection of papers, data extraction card is formed as shown in Table 4. It is used to gather relevant information from the selected papers. This table is formed through pilot data extraction, i.e., the data extraction card is tested on representative samples from the studies to be reviewed and refined to collect desired information. Items in Table 4 are written according to the associated research questions which help in data synthesis step.

During the data extraction, some data could not be extracted directly from the selected studies. Nevertheless, we could obtain them indirectly by processing the available data appropriately. Not every selected study provides answers to all the 6 research questions. For ease of tracing the extracted data, we explicitly labeled each study with the IDs of the research questions to which the study can provide the corresponding answers. Some of the research questions may have multiple answers, for example one paper may apply two techniques for mining the software repositories, and so to resolve this issue, reviewed paper has been counted in both the techniques.

Table 4 Data Extraction card

Article Title
Year of publication(2003-2018)
Name of authors
Source
Application areas explored in MSR papers (RQ1) Bug Prediction, Change Prediction, Software Evolution ,Software Performance Analysis, Clone Detection and Code Reuse
Tools used to mine repositories(RQ2)(Bug Tracking tools, Data Modeling and Statistical tool, Extraction tool)
Which data repositories are mined (NASA, PROMISE, GitHub, Jira, FOSS, GCC, Sourceforge, StackOverflow, Bugzilla) Which software project are mostly used(Eclipse, Firefox, GNOME desktop suite, Net Beans, Android, Apache, FreeBSD, PostgreSQL, Linux, ArgoUML and others) Distribution of Type of datasets used in MSR papers(Open Source, Industrial, Partial and None) Identify software life cycle phase at which data sets is gathered(Requirements, Design, Development, Testing and Maintenance) (RQ3)
Investigated techniques for MSR (RQ4)(Association Rule, Regression, Text Mining, Clustering, Frequent Pattern Matching, Classification, Machine Learning)
Type of MSR study (theory, survey, experiment, Implementation, review, development of tool, development of mining technique and case study) Identify the popular MSR journals and conferences(RQ5)(Journal of Empirical Software Engineering Springer, Software Quality Journal Springer, IEEE Transactions on Software Engineering Journal, International Conference on Mining Software Repositories (MSR), International Conference on Software Process (ICSP),International Conference on Software Engineering (ICSE),Elsevier Journal)
The research gaps in various application areas (RQ6)(Clone Detection, Code reuse, Software evolution)

2.5. Data Synthesis

The purpose of data synthesis phase is aggregation of information from the selected papers to provide response to the specified research questions. One piece of information alone might not lead to a conclusive answer, so we need to aggregate the extracted information.

The data synthesis strategy used for this review is the narrative synthesis method [275], i.e., the studies are organized into homogenous groups and the characteristics, quality, context and results of the studies are reported in a standardized format. In narrative synthesis, the extracted data is tabulated in accordance with the research questions. Visualization tools such as pie chart, line graph, bar chart and box plot, are used in our study to report and present the results.

3. Results and Discussion

An overview of the studies selected for the review is given in this section. Then the findings of the review are presented and discussed. Each research question is answered in a separate subsection. The results of the review are interpreted not only in the context of the respective research questions, but also in the wider context linked to all research questions.

3.1. Overview of selected studies

A total of 300 studies are reviewed in our SLR. They consist of papers spanning over entire decade (2003-2018). Figure 2 shows a graph indicating the number of publication in each year. The first workshop on MSR (held in 2004) helped in naming a field that was arising in software engineering research. MSR field continues to be rapidly growing area in software research thus we have included majority of MSR papers after 2004.

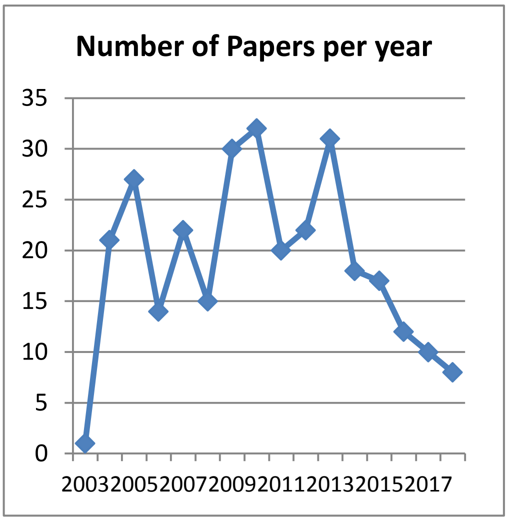


Figure 2 Number of Research Papers per year in review

After applying IC and EC, 321 studies were selected on which QA was performed. The studies with quality score greater than 8 were reviewed. 21 papers were rejected because their quality score was found to be less than 50%. Therefore majority of the studies selected have high quality. As mentioned in Table 5, about 61.05% (196 of 300) of the studies selected for review have a very high or high quality level.

Quality level	**Number of studies**	**Percent**
Very high (15≤score≤17)	66	20.56
High (12≤score≤14.5)	130	40.49
Medium (8≤score≤11.5)	104	32.40
Low (5.5≤score≤8)	12(rejected)	3.74
Very low (0≤score≤5)	9(rejected)	2.81
Total	321(300 included 21 rejected)	100

Table 5 Quality Levels of relevant studies

3.2. Application areas explored in MSR papers (RQ1)

The observations made after reviewing 300 papers in the field of MSR suggest that it has many applications. Below is a brief description of popular applications areas of MSR:

- **Bug Prediction**: A bug indicates an error or fault in the software system that may cause the system to behave unexpectedly. Bug Prediction uses the bug history of software system to predict bugs that may be introduced in future. Most of the work in bug prediction predicts bugs during design and coding phase.
- **Change Prediction:** Change prediction helps developers by predicting program entities that are likely to be

changed in next version or next phase of software development.

- **Clone detection:** Code clones are pieces of code that are identical to each other. Researchers employ mining techniques for detection of similar fragments in project. Clone detection is usually performed in design and development phase of a SDLC.
- **Code reuse:** As a large amount of code is freely accessible through the Internet, there is a chance that a given problem has been solved already, and its implementation is open source. It is essential to mine the software code repositories of previous versions during the design and development of the next reuse.
- **Software Evolution:** Software Evolution refers to the tasks and activities involved in addition of new functionality to existing software. Major organizations spend more on evolving existing software than on developing new one. In order to maintain the existing software, there is a need to mine software repositories during the maintenance phase.
- **Software Performance Analysis:** The growing complexity of the software and its profound use in day to day life has promoted interest in the field of software performance analysis. Researchers tend to analyze the attributes of the software by mining repositories in order to suggest improvements in the software development and maintenance phase.
- **Others**: All the application areas of MSR which have not been covered above are refactoring, software maintenance etc.

Figure 3 presents the distribution of reviewed papers focusing on different application areas. If a paper focuses on multiple application areas then the same paper is included in all the application areas it covers. This figure 3 suggests that:

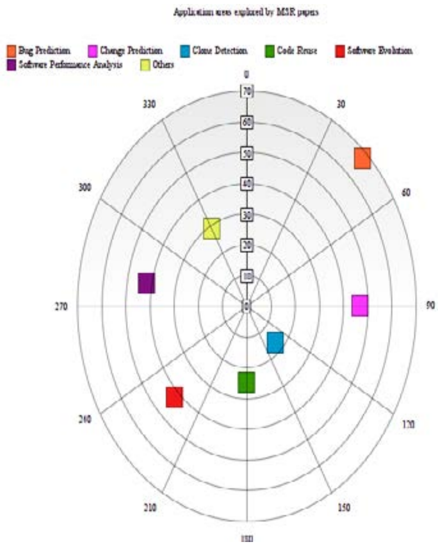


Figure 3 Application areas explored in MSR papers

The dominant application found in research papers are Bug prediction and change prediction. 25% of the papers are based on bug prediction, with the next-highest proportion being the change prediction area (17%). This shows that researchers are more concerned on improving the mining techniques of bug and change prediction.

- Researchers are also working on Software Performance Analysis (16%) and Software Evolution (16%) application areas. These are important research areas as they analyze the software repositories and suggest improvements in the existing software development practices which helps in increasing the productivity of the software.

- Code Reuse (nine percent) has been relatively less explored in research work. An important aspect of code reuse is that if used efficiently, it not only reduces the cost of development but also increases the productivity of software.

- Six percent of papers included in our study are on clone detection. Clones are useful for reengineering and can be advantageous in many ways. But it is difficult and costly to manage clones. Researchers are trying to develop effective clone detection approaches.

Table 6 provides a replication package of the reviewed studies for the different application areas of MSR.

Table 6 Replication Package for Application Area of MSR

S.No.	Application Area	No. of papers	Citations
1	Bug Prediction	68	15, 17, 18, 27, 30, 31, 45, 47, 67, 71, 72, 77, 83, 84, 89, 90, 94, 96, 99, 100, 104, 108 , 110, 111-116, 118, 122, 123, 126-130, 133, 134, 137, 151-153, 159, 161, 162, 166, 173, 174, 178, 179, 194, 198, 199, 201,214,229,230, 240, 245, 247, 248, 252, 256, 259, 261, 266, 267
2	Change Prediction	47	1, 2, 5, 10-12, 19, 21, 22, 30, 32, 36-39, 43, 44, 53, 56, 59, 62, 66, 68, 69, 75, 85, 92, 100, 101, 103, 104, 107, 113, 119, 121, 124, 131, 156, 170, 176, 195, 251, 260, 262-265, 269
3	Clone detection	17	29, 132, 135, 169, 178, 203,215-221 , 246, 249, 268, 270
4	Code reuse	25	51, 54, 55, 76, 80, 102, 103, 105, 106, 109, 125, 140, 146, 175, 184, 187, 233-239, 241, 255
5	Software Evolution	42	8, 14, 16, 20, 23-26, 28, 29, 38, 46, 63, 86-88, 95, 97, 119, 138, 143-145, 150, 177,205,211,212,221-225,227,228,232, 242-244, 250, 253, 268
6	Software Performance Analysis	42	8, 10, 11, 13, 14, 16, 33, 41, 50, 52, 61, 70, 73, 78, 79, 81, 93, 105, 117, 120, 129, 149, 155, 158, 160, 163-165, 167, 182, 183, 185, 188, 191, 192, 200,207,208, 254, 257, 258

3.3. Tools used to mine repositories (RQ2)

Researchers have used various types of tools to mine the datasets. Tools have been categorized into three categories: Bug Tracking tools, Data Modeling and Statistical Tools, Extraction Tools. There are some papers which do not use or develop any tool, so those papers are excluded for this research question. The distribution of tools is given in Fig. 4 indicates:

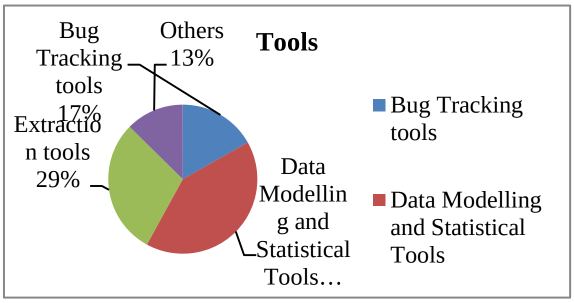


Figure 4 Tools Used to Mine Repositories

- Data Modeling and Statistical Tools (41 %) are the most widely used tools to evaluate the performance of MSR. The popular statistical tools used by researchers include WEKA, SPSS, Rapid Miner, DTreg, DB Miner , R Tool , checkstyle etc. These tools are used in majority of studies for generating and analyzing the results.
- Extraction Tools (29%) such as ckjm (Chidamber and Kemerer java metrics) extractor, ParseHub, Visual Scraper are used to extract desired information from software repositories and other online resources.
- Bug Tracking Tools (17%) such as Buginfo, trac, FOSSIL, Redmine etc. are used to identify patterns of bugs and changes from the data repositories.13% of the papers used other tools. The tools included in this category are used less than three times in different research papers. Some of these tools include MALLET, Doc Miner, etc.

Different tools work on different datasets and different applications. It is important that we choose the correct tool according to our dataset and application area. Table 7 lists the datasets and application areas for which the MSR tools can be used.

Tools	Dataset	Application Area
Data Modelling and Statistical Tools	GitHub, Sourceforge, NASA, Bugzilla	Bug Prediction, Change Prediction, Software Performance Analysis
Extraction Tools	PROMISE, StackOverflow, GitHub, FOSS, Bugzilla	Clone Detection, Bug Prediction, Change Prediction, Code Reuse
Bug Tracking Tool	Jira, Bugzilla	Bug Prediction

Table 7 Datasets and application areas of MSR tools

3.4. Frequently used datasets and software projects along with its type and SDLC phase in which it is used (RQ3)

Data is usually stored in one or more data centers or repositories from which it can be extracted for research work. Some datasets are directly used for example NASA, PROMISE, GitHub, Jira, Sourceforge and some are mined like Google Code, FreeBSD, StackOverflow, FOSS, Apache Software Foundation are some of the prominent repositories which are mined. Figure 5 shows the distribution of datasets mined. This figure indicatess that:

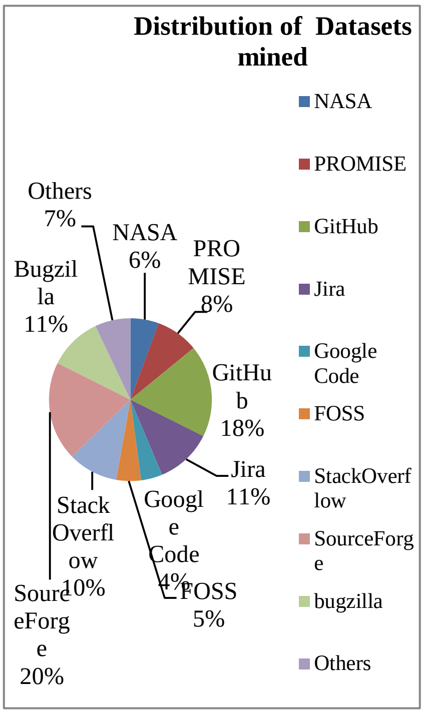


Figure 5 Distribution of Datasets mined

- 20% of the papers used Sourceforge [3] repository, which is a central online hub for software developers to manage and control open source software projects. It was the first service available for open source projects where developers are provided with bug tracking system, source code repository, free access for hosting, news bulletin board as a blog, user and developer comment lists etc. This repository is widely used by the researchers for mining data.
- 18% of papers collect data from GitHub repository [226] which is a popular open source repository consisting of code, commits and issues of various software projects. GitHub is a web based software repository which provides all versions of the software along with the source code management functionality. It offers access to bug tracking systems, task management and wikis for software projects etc. It also act as a storage area for private repositories as well as open source projects.
- StackOverflow (10%) [193, 209] is a popular socio technical network. It acts as a podium for end users to inquire and respond the questions on a wide range of topics related to software engineering. It also assign flag to each question based on its priority which is a useful feature for researchers in mining data.

- Bugzilla (11%) is a web-based general purpose bug tracking software repository. It was originally developed and used by Mozilla Project but now is used by many software projects as a bug tracking system.
- Promise repository (eight percent) is one of the research largest dataset repository used mainly for software engineering area. It is formed to promote verifiable, repeatable refutable and improvable prediction models. Datasets and tools are available in this repository to serve researchers for mining useful information from it.
- Jira (11%) is a proprietary issue tracking and bug tracking product which provides project management functions. It is written in Java language.
- Google code (four percent) is a source code repository where codes of various software or web pages are stored globally or confidentially and support version control, bug tracking, and mailing lists for researchers to mine data.
- FOSS [4] repository(five percent) contains data such as reports, presentations, conference proceedings, videos, photos, and digital collections of software which is produced by open source software society which is a great help for researchers
- NASA (six percent) is a collection of datasets that have been contributed by various universities, agencies and companies. This repository aims to contain datasets which are used for development of prognostic algorithms by researchers.

The reviewed papers were categorized based on not only the datasets but also on the basis of software projects used. Projects can be software or operating systems. Categorization is as follows: Eclipse, Firefox, GNOME desktop suite, Net Beans, Android, Apache, PostgreSQL, Linux, ArgoUML and others. Figure 6 shows the distribution of software projects used. The figure indicates that:

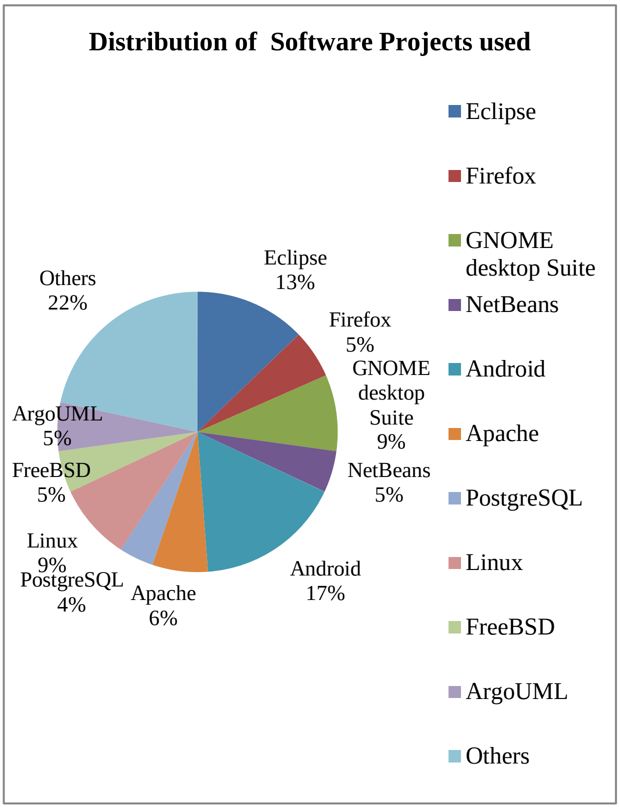


Figure 6Distribution of Software Projects

- The Android operating system [154, 181, 231] (17%) is the mostly used project by researchers. It is the mobile operating system developed by Google which is used in touch screen mobile devices such as smart phones and tablets. Its source code is released under open source license which is generally used by many researchers in their specific field.
- 13% of the papers used repositories are based on Eclipse software [35]. Eclipse is a java based Integrated development environment (IDE) used in software engineering. It uses plug-ins to provide different functionalities within and on top of the runtime system. It contains a base workspace where researchers can mine for the changes and bugs in the software.

- Datasets of Linux and GNOME [102, 109] desktop suite both contribute same proportion (nine percent) of papers reviewed. Linux is an open source Operating System. The main advantage of Linux is it can be used, modified and distributed by any researcher under the GNU General Public License. GNOME (GNU Network Object Model Environment) is a desktop environment that is based on Linux OS and composed of free and open source software. GNOME is an international project which aims to develop software frameworks and to coordinate researchers for accessing the details of software projects.
- Data from Apache software projects [74] is analyzed in six percent of the papers. Apache is a decentralized open source area of developers where project is managed by team of experts to provide legal protection and access software code by researchers of apache projects.
- Firefox (five percent) and ArgoUML (five percent) software projects are also used in the papers reviewed in our study. ArgoUML is UML diagram software based on java which is developed under open source Eclipse public license. It is basically used by researchers to mine the software in design phase of SDLC. Firefox is an open source web browser invented by Mozilla Foundation based on Windows, MacOS and Linux operating system. Firefox provides an environment for web developers or researchers to use built in tools for extracting useful information for example bugs of a web page.
- Netbeans (five percent) and PostgreSQL (four percent) are open source software projects. While FreeBSD (five percent) [48] is a free UNIX like operating system which is also relatively less evaluated software project.

It is further important to find that whether research in the field is verifiable and repeatable. For studies to be verifiable, data should be available openly to other researchers. A reproducible research promotes new researchers to conduct research in this field. But most industrial companies do not make their data repository public pertaining to confidentiality. In this case, alternatives to share confidential data should be devised by researchers and companies to promote research without compromising with the confidentiality.

The selected papers were classified on the basis of type of data repositories used: open source, industrial, partial and none. Data repositories which are freely accessible by everyone are referred to as open source data repositories. The industrial repositories are privately owned by a software company or researcher. Some papers use both industrial and open source repositories, the repositories in such of papers are considered under partial data repositories. If no information about the repository is mentioned in the paper, then that is considered as none. Figure 7 depicts the distribution of type of data repositories indicating that:

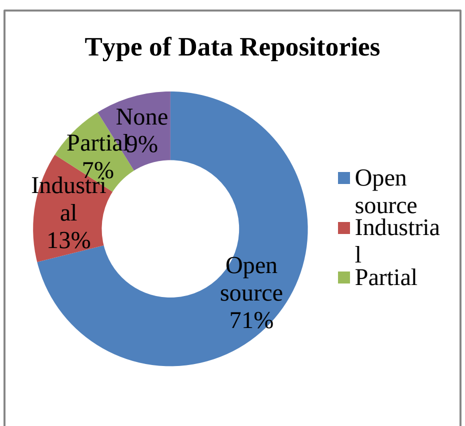


Figure 7 Type of Data Repositories

- The percentage of papers that make use of open source repositories [139, 204] to assess the performance of the proposed mining technique is 71%. The proportion of research papers that evaluated industrial data repositories is less (13%)—this trend shows that more researchers prefer open source repositories to carry out their research as they are easily available to them. The correctness and appropriateness of papers that report results on the basis of evaluation of data from industrial data repositories is worthy of retrospection as the results of these papers may not be repeatable.

- Only a few papers (seven percent) used open source as well as industrial data repositories. Partial repositories provides assurance to people planning to apply the results, as using both of open source repositories and industrial data repositories together in a paper makes the evaluation results more dependable and credible.
- Nine percent of the papers did not use any data repository. These papers may include survey papers, review papers, theory papers that do not rely on data repositories for validation.

The datasets are also categorized as primary and secondary depending on whether they are directly collected (primary) or indirectly collected (secondary). Some of the datasets discussed in this study are primary while some are secondary irrespective of whether it is open source, industrial and partial. . Table 8 gives the classification of datasets on basis of whether they are primary or secondary, and type of dataset, i.e., open source, industrial or partial.

Dataset	**Type (Open Source / Partial / Industrial)**	**Primary/Secondary Dataset**
Sourceforge	Open Source	Primary
GitHub	Open Source	Primary
StackOverflow	Open Source	Secondary
Promise	Open Source	Primary
Jira	Partial	Primary
Google Code	Open Source	Secondary
FOSS	Open Source	Secondary
NASA	Industrial	Primary
Bugzilla	Partial	Primary

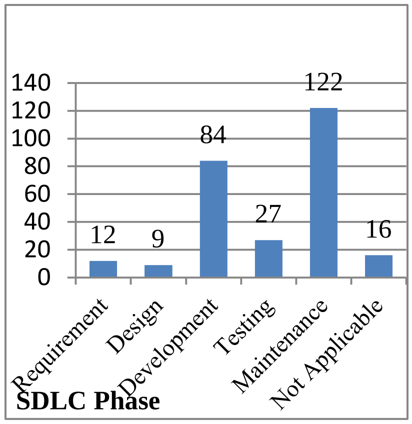
Table 8 Classification of Datasets

The 'Software Development Life Cycle' (SDLC) property is used to discover the relationship between the SDLC phase and data set . It investigated the effect of SDLC phase on MSR. The studies were categorized according to SDLC at which data is collected into: Requirements, Design, Development, Testing and Maintenance.The SDLC lays down a foundation outlining the tasks to be carried out during the development of software. SDLC defines a formal organization structure to be pursued by the software development team.

SDLC comprises of following activities:

- **Requirements:** In this phase requirements for the software product are finalized and analyzed. It is a vital step in the process of software development and requires the expertise of proficient and knowledgeable software engineers. It is important to mine data from requirements gathering phase in order to understand the implications of decisions made in initial stage of the software development.
- **Design:** It is a critical phase in the SDLC where approved requirements are taken as an input and logical system is converted into physical system design. Hence making it necessary to analyze the design phase of the software development to comprehend fully the evolution of software.
- **Implementation/Development**: In this phase the software code is actually written on the basis of the requirements gathered in the Requirements phase. Code analysis, change prediction and bug prediction all make use of data from this phase of the software development.
- **Testing:** This is the process of discovering and fixing bugs in the software code. It involves systematically finding bugs in the software and debugging them. The information from testing phase is usually mined for bug prediction and software performance analysis.
- **Maintaining:** Software maintenance is required for any software that needs to continuously improve its performance after initial release. It involves fixing existing bugs, improvement and addition of new features. Hence it is generally a longer phase than the development phase. This phase provides a lot of data that needs to be mined not only to understand the software evolution but also for bug prediction and change prediction. Figure 8 shows the phases of SDLC at which data sets are gathered by the researchers.

Figure 8 SDLC Phase of Dataset



It suggests:

- Majority of the data sets are collected at the maintenance phase [40]. 45% of the research papers have collected the data at maintenance phase.
- The other dominant phase, constituting 31% of the papers reviewed, is the development phase followed by testing (10%).
- It is noticed that only eight percent of papers extract the data from requirement (five percent) and design (three percent) phase. It implies that it is difficult to extract data at requirement and design phase.

Table 9 gives the classification of application areas according to the SDLC phase.

Table 9 Classification of Application Areas

Application Area	SDLC Phase
Bug Prediction	Design , Implementation, Testing, Maintenance
Change Prediction	Implementation, Maintenance
Code Reuse	Design, Implementation
Software Evolution	Design, Maintenance
Software Performance Analysis	Implementation, Testing, Maintenance
Clone Detection	Design, Implementation

3.5. The most investigated techniques for MSR (RQ4)

The objective of this research question was to categorize and document the existing mining techniques [9]. Figure 9 depicts the distribution of different mining techniques. This distribution indicates that there exist a wide range of techniques used for MSR:

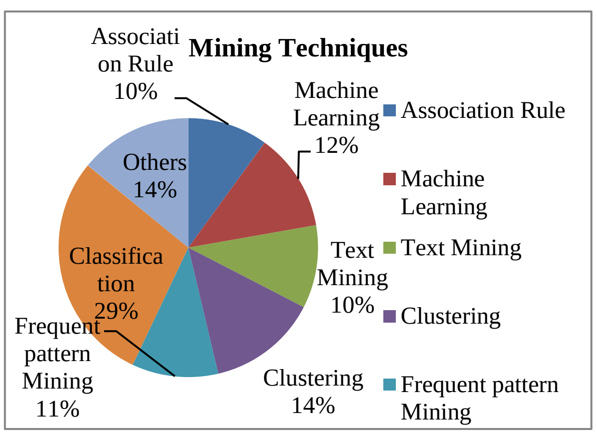


Figure 9 Distribution of Mining Techniques

- Classification [26, 92, 98, 131, 146, 172, 203] based techniques form the highest proportion (29%) of the techniques used in the studied papers. Classification algorithms like Naïve Bayes classifier, Decision Trees etc. are quite popular for classifying bugs, changes, code clones etc.

- Clustering technique [8, 191] (14%) is also commonly used in studies reviewed in this SLR. Clustering algorithms like K-means, DBSCAN etc. are used for recognizing patterns and similarities from the vast amount of data available in repositories.
- Machine Learning [180] (12%) is also used in some of the studies reviewed in this SLR. Machine Learning provides a range of popular predictive modeling techniques used for finding out the association between dependent and independent variables in a study. Regression techniques such as Linear Regression, Logistic Regression, Polynomial Regression etc. have been widely used for prediction in MSR research.
- Frequent pattern matching [49, 53, 56, 190, 202] (11%), Association Rule mining [19, 101, 121] (10%) and Text Mining [6, 34, 52, 60, 82, 104, 106, 136, 137, 147, 148, 152, 160, 171, 186] (10%) are evolving techniques for MSR. Frequent Pattern matching and Association rule mining has been used by researchers to uncover useful patterns from repositories. For instance it might be interesting to find out relation between different kinds of changes in the software and the frequency of these changes. Text Mining is also being widely used for extracting information from rich but unstructured text available from various repositories and artifacts produced during software development.

It is important to understand which techniques are useful for which applications. Table 10 provides a list of application areas for which the MSR techniques can be employed.

3.6. Different types of MSR studies and popular Mining Software Repositories journal and conferences (RQ5)

The research studies are segregated based on the research method employed, i.e. theory, survey, experiment, implementation, review, development of tool, development of mining technique and case study. Figure 10 depicts the distribution of research methods. This figure suggests the following:

Table 10 Application Areas for MSR Techniques

Techniques	Application Areas
Classification	Bug Prediction, Change Prediction, Code Reuse, Clone Detection
Frequent Pattern Matching	Code reuse, Change Prediction
Association Rule Mining	Software Performance Analysis, Bug Prediction, Change Prediction, Software Evolution
Text Mining	Code Reuse, Bug Prediction, Change Prediction, Software Evolution
Machine Learning	Bug Prediction, Change Prediction, Software Performance Analysis
Clustering	Clone Detection, Software Evolution

- A majority of the papers (28%) reported results of an experiment [210, 213], with the next-highest proportion being the development of mining technique (15%) and case study (15%). This shows that researchers are keen on exploring new results by conducting experiments.
- Theory papers [1, 58, 64, 197] form 12% each of the total papers reviewed. Also 14% of the papers describe the development of mining tool [7, 57, 168]. Only 10% of the papers are found to be survey [91, 196] papers in our study.
- Only six percent of the papers are review papers [42, 65, 141, 142, 206] so there is a need to conduct more reviews in the field of MSR.

International Conference on Mining Software Repositories (MSR) [157, 189] is the dominant conference in the MSR field. International Conference on MSR provides a platform for researchers and practitioners in the field to discuss and collaborate on new research ideas. Research in the field of MSR has attained recognition since the year 2004 when the first International Conference on MSR took place. It is from then that MSR carries on to being a rapidly expanding research field. Figure 11 represents the distribution of popular journals and conferences.

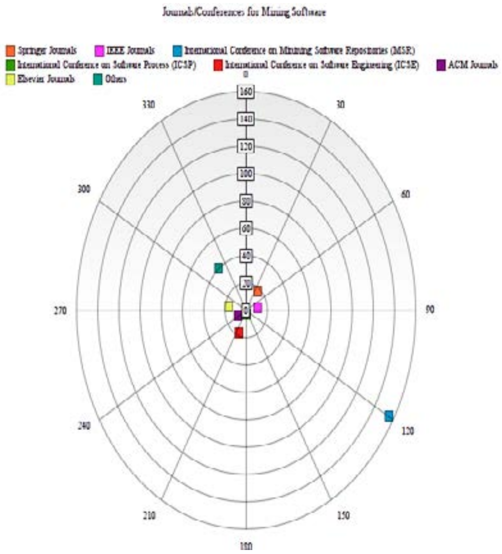


Figure 11 Journals/Conferences for MSR

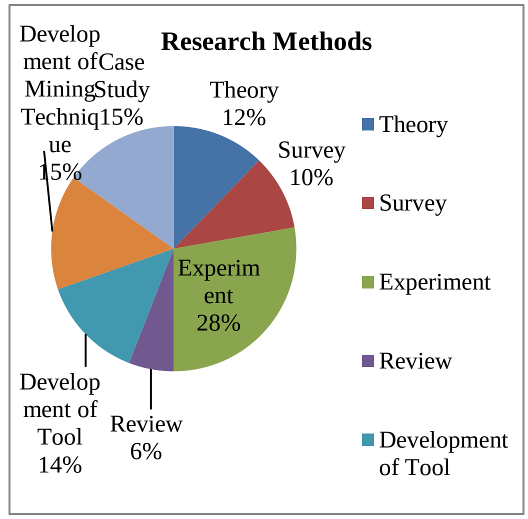


Figure 10 Distribution of research methods

3.7. Research gaps in various application areas of MSR (RQ6)

Among the various application areas as discussed in section 3.2, it was found that the proportion of the selected papers for code reuse, software evolution and clone detection areas were to be less in comparison to other application areas of MSR. This section tries to identify the research gaps in these areas.

3.7.1 Clone Detection

Clone detection aims at detecting the reused code fragments with or without minor revisions. It is important to detect code clones not only during software development, but also during the maintenance phase to speed up the process. For example if a bug is found inside a code fragment then all fragments similar to it should also be checked to detect the same bug.From the papers reviewed related to clone detection it is observed that detecting software clones have made a significant contribution, but it is still not identified whether clones are good or bad? It is required to carry out more studies which determine the impact of clones on software quality and maintainability.

It is still a challenge for researchers to determine the behavior and impact of code clones. It cannot be said for sure that non-cloned codes contain less defects than cloned code. Thus more empirical studies need to be conducted using automatic tools for clone detection to arrive at such general conclusions.

It is also observed that there is a need to record the history of clone detection to analyze the reason for occurrence of code clones. Clone management studies should be encouraged more to understand economic tradeoff software clones in software development lifecycle. There is an urgent need to have empirical and analytical studies that compare and evaluate various clone detection techniques and tools.

3.7.2 Code Reuse

Code Reuse involves the usage of already existing software for developing new software with the help of reusability principles. It aims to save resources and time by using already created software in development phase of software life cycle. For example software library, subroutines are the basic practices used in code reuse area. The research in the code reuse field is at a growing stage. Therefore, there is a requirement of automated tools that help in enhancing the reusability of code. In the software development area, there is a necessity to understand the role of reusability. Thus new and improved methods need to be developed for organizing, classifying and retrieving software code for reuse. Code reuse should not only consider the software routines or libraries rather take the whole system in context of maintaining or building new systems.

3.7.3 Software Evolution

Software evolution is the process of developing software initially using software engineering principles and then updating till it met the user's requirements. Once the desired software is developed, advancing technology and requirements force to change the software accordingly. Thus recreating software from the scratch is not feasible, so it's economical to update the existing software to meet the new requirements. The laws of software evolution faced many challenges since their introduction, and now it has been shown that they are not universal. Similar to the changes in software, these laws also need to be changed. These laws need to be adapted not only according to the changes in software, but should also account for the changes in software development and maintenance processes.
With time, it is necessary to carry out empirical studies that assist in the formulation of new software evolution laws which can be shown valid according to the current trends.

4. Implications for research and practice

The results of the review suggest that there are a wide range of techniques used for MSR with no single technique dominating the research. Classification techniques were used in maximum of the studies but still it constitutes only 29% of the reviewed research. The proportion of papers employing machine learning are increasing, but at a slow rate. Researchers should explore this technique. Association Rule Mining, Frequent Pattern Mining, Text Mining, Clustering are other mining techniques that are applied by the researchers. Hence, researchers should be encouraged to conduct more studies and experiment on these lesser used techniques to strengthen evidence regarding their performance. Additionally, researchers must continue exploring the possibilities of developing new techniques and frameworks for MSR.
Partial repositories are known to provide assurance to people planning to employ the results of a study, as using both open source repositories and industrial data repositories makes the evaluation of results more credible. But only seven percent of the papers used partial data repositories for their research and majority of studies (71%) relied only on Open source data repositories. This trend is noticed because open source repositories are easily accessible unlike industrial repositories. It is encouraged to validate the findings of an experiment both with open source and industrial data repositories.

5. Threats to validity

The major threats to validity of the review model are examined on the basis of the following aspects: publication bias, study selection bias and probable imprecision in data extraction.
Due to publication bias, positive findings in the MSR field are published more than the negative findings. Also researchers have a tendency to claim that their technique or methodology perform better than others'. This may result in overestimation of performance of MSR techniques. Luckily, one inclusion criterion (i.e., the third inclusion criterion) may help alleviate this threat. This criterion includes the studies that perform comparison of MSR techniques. These studies which perform a comparison of MSR techniques are not biased towards any technique and report the results of the comparison in an impartial way. Hence, these studies report the negative results in addition to the positive results. However, this review does not consider gray literature (i.e., white papers, technical reports, thesis and work in progress) as it is difficult to obtain, publication bias is unavoidable.

Papers have not been searched using issue-by-issue, manual reading of research titles and abstracts of all papers published in various journals and conferences. The reason for not preferring this approach has been related to practical concerns such as workload and limited query string. Therefore, some MSR papers might have been excluded from some less cited journals or conference proceedings.

Research in the field of MSR has gathered momentum since the year 2004 with the occurrence of the first International Conference on MSR. That is why we have included majority of papers after 2004 and excluded before 2003. The results and trends observed are based on all the papers reviewed during the entire time frame from 2003 to 2018. The review findings do not represent any increasing or decreasing trends over the time series.

Another limitation of this study is that even though it discusses the various application areas of MSR, it does not dwell upon the different metrics used by these application areas.

In order to decrease the threat of imprecision in data extraction, specialized data extraction cards have been elaborated. Any disagreement between researchers is resolved through discussion. Nevertheless, perfection in the extraction process cannot be guaranteed, as human errors are unavoidable.

6. Conclusion and future work

The present paper investigated 300 MSR papers published in journals and proceedings of conferences. The purpose of this systematic literature review is to help researchers to find out popular application areas, tools, repositories and techniques in the field of MSR. The research gaps and popular conferences and journals for publishing MSR studies are also described. Thus our review provides a roadmap for upcoming research in MSR field.

The following important observations are made from this systematic literature review:

- Bug Prediction (25%) is found to be the most popular application area of MSR followed by Change Prediction (17%), Software Performance Analysis (16%) and Software Evolution (16%).

- Researchers have developed and used many tools for mining and analysis of data available in various software repositories. Data Modeling and Statistical tools (41%) are widely employed for analyzing the performance of MSR techniques.

- The proportion of papers that use open source repositories is 71%, while the proportion of studies that employ partial repositories is only seven percent. This trend does not appear to be satisfactory, and more papers need to employ partial datasets. Partial repositories gives assurance to people who intend to apply the results, as using both open source repositories and industrial data repositories makes the results more credible and dependable.

- Classification (29%) is the most popular technique for mining software repositories. Text Mining (10%) and Association Rule Mining (10%) are emerging techniques and should be explored more in future research.

- International Conference on Mining Software Repositories (MSR) is the most popular conference on MSR. It provides ideas and opportunities for researchers working in this field. Moreover most of the papers report the studies of an experiment (28%) and very few are review (six percent) or survey papers (10%).

- Research gaps of relatively less explored and evolving application areas like clone detection (six percent), code reuse (nine percent) and software evolution (16%) are identified and researchers should focus on filling these research gaps.

The future work includes comparing accuracy of various techniques for MSR, so that performance of MSR techniques can also be assessed. It is also recommended to discuss with the experts and existing literature in the field for summarizing the strengths and weaknesses of MSR techniques.

Appendix A. Summary of Quality Assessment Table 11 provides a summary of quality assessment of selected assessment

Table 11 Summary of Quality Assessment

Parameter	**No.**	Question	Yes	Partly/Not Relevant	No
Research Questions	QA1	Are the research questions clearly stated?	278(92.66%)	9(3%)	13(4.34%)
	QA2	Are the research questions justified in the context?	267(89%)	17(5.67%)	16(5.33%)
	QA3	Are all the research questions answered?	271(90.33%)	13(4.34%)	16(5.33%)
Application Area	QA4	Does the paper contribute significantly to the application area?	267(89%)	12(4%)	21(7%)
Tool	QA5	Is the usage of tools adequately described?	177(59%)	16(5.33%)	107(35.67%)
	QA6	If the study involves tool development, then is the development of a tool explained clearly?	65(21.67%)	62(20.67%)	173(57.66%)
	QA7	If the study involves tool development, then is its performance assessed?	76(25.33%)	59(19.67)	165(55%)
	QA8	Is the tool benchmarked against existing tools?	54(18%)	119(39.67%)	127(42.33%)
Dataset	QA9	If the study involves data collection, are the data extraction methods defined?	223(74.33%)	17(5.67%)	60(20%)
	QA10	If the study involves analysis, are the data sets adequately described?	258(86%)	18(6%)	24(8%)
	QA11	Is the data available or can be collected by the method defined?	270(90%)	0(0%)	30(10%)
Techniques	QA12	Is the purpose of the technique defined clearly?	289(96.33%)	8(2.67%)	3(1%)
	QA13	Is the result of the used technique clearly stated?	192(64%)	15(5%)	93(31%)
	QA14	Is the result of the technique compared with other techniques?	167(55.67%)	44(14.67%)	89(29.66%)

Research Methodology	QA15	Is research methodology clearly described?	273(91%)	10(3.33%)	17(5.67%)
	QA16	Are the threats to validity of the study clearly presented?	162(54%)	10(3.33%)	128(42.67%)
	QA17	Does the results of the study are further added to the literature?	277(92.33%)	1(0.33%)	22(7.34%)
	QA18	Does the study discuss the issues related to validity/reliability of their measures?	201(67%)	1(0.33%)	98(32.67%)

References for the 300 Reviewed Papers

[1] Andreas Jedlitschka and Markus Nick. 2003. Software Engineering Knowledge Repositories. R. Conradi and A.I. Wang (Eds.): ESERNET 2001-2003, LNCS 2765, pp. 55–80, 2003,Springer-Verlag Berlin Heidelberg 2003,55-80

[2] Thomas Zimmermann and Peter Weißgerber. 2004. Preprocessing CVS Data for Fine-Grained Analysis. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 2-6.

[3] James Howison and Kevin Crowston. 2004. The perils and pitfalls of mining SourceForge. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 7-11.

[4] Les Gasser, Gabriel Ripoche and Robert J. Sandusky. 2004. Research Infrastructure for Empirical Science of F/OSS. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 12-16.

[5] Daniel M. German. 2004. Mining CVS repositories, the softChange experience. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 17-21.

[6] Alexander Dekhtyar, Jane Huffman Hayes and Tim Menzies. 2004. Text is Software Too. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 22-26.

[7] Gregorio Robles, Jesus M. González-Barahona and Rishab A. Ghosh. 2004. GlueTheos: Automating the Retrieval and Analysis of Data from Publicly Available Software Repositories. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 28-31.

[8] Ying Liu, Eleni Stroulia, Kenny Wong and Daniel German. 2004. Using CVS Historical Information to Understand How Students Develop Software. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 32-36.

[9] Omar Alonso, Premkumar T. Devanbu and Michael Gretz. 2004. Database techniques for analysis and exploration of software repositories. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 37-41.

[10] Masao Ohira, Reishi Yokomori, Makoto Sakai, Ken-ichi Matsumoto, Katsuro Inoue and Koji Torii. 2004. Empirical Project Monitor: A Tool for Mining Multiple Project Data. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 42-46.

[11] Filip Van Rysselberghe and Serge Demeyer. 2004. Mining Version Control Systems for FACs (Frequently Applied Changes). In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 48-52.

[12] Jelber Sayyad Shirabad, Timothy C. Lethbridge and Stan Matwin. 2004. Mining the Software Change Repository of a Legacy Telephony System. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 53-57.

[13] Michael Godfrey, Xinyi Dong, Cory Kapser and Lijie Zou. 2004. Four Interesting Ways in Which History Can Teach Us About Software. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 58-62.

[14] Mohammad El-Ramly and Eleni Stroulia. 2004. Mining Software Usage Data. In Proceedings of 1st

International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 64-68.
[15] Chadd C. Williams and Jeffrey K. Hollingsworth. 2004. Mining Bug Driven Bug Finders. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 70-74.
[16] Tim Menzies, Justin S. Di Stefano and Chris Cunanan. 2004. Mining Repositories to Assist in Project Planning and Resource Allocation. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 75-79.
[17] Robert J. Sandusky, Les Gasser and Gabriel Ripoche. 2004. Bug Report Networks: Varieties, Strategies, and Impacts in a F/OSS Development Community. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 80-84.
[18] Thomas J. Ostrand and Elaine J. Weyuker. 2004. A Tool for Mining Defect-Tracking Systems to Predict Fault-Prone Files. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 85-89.
[19] Ranjith Purushothaman and Dewayne E. Perry. 2004. Towards Understanding the Rhetoric of Small Changes -- Extended Abstract --. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 90-94.
[20] Chris Jensen and Walt Scacchi. 2004. Data Mining for Software Process Discovery in Open Source Software Development Communities. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 96-100.
[21] Luis Lopez-Fernandez, Gregorio Robles and Jesus M. Gonzalez-Barahona. 2004. Applying Social Network Analysis to the Information in CVS Repositories. In Proceedings of 1st International Workshop on Mining Software Repositories (MSR 2004), Edinburgh, Scotland, United Kingdom, 101-105.
[22] Ahmed E. Hassan, and Richard C. Holt. "Predicting change propagation in software systems." In the Proceedings of 20th IEEE International Conference on Software Maintenance, 2004, pp. 284-293. IEEE, 2004.
[23] Zhenchang Xing, and Eleni Stroulia. "Understanding class evolution in object-oriented software." In Proceedings of 12th IEEE International Workshop on Program Comprehension, 2004, pp. 34-43. IEEE, 2004.
[24] Tom Mens, Michel Wermelinger, Stéphane Ducasse, Serge Demeyer, Robert Hirschfeld, and Mehdi Jazayeri. "Challenges in software evolution." In Eighth International Workshop on Principles of Software Evolution, pp. 13-22. IEEE, 2005.
[25] Gregorio Robles, Juan Jose Amor, Jesus M. Gonzalez-Barahona, and Israel Herraiz. "Evolution and growth in large libre software projects." In Eighth International Workshop on Principles of Software Evolution, pp. 165-174. IEEE, 2005.
[26] Iulian Neamtiu, Jeffrey S. Foster, Michael Hicks. Understanding Source Code Evolution Using Abstract Syntax Tree Matching. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 2-6.
[27] Chadd C. Williams, Jeffrey K. Hollingsworth. Recovering System Specific Rules from Software Repositories. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 7-11.
[28] Michael Fischer, Johann Oberleitner and Jacek Ratzinger, Harald Gall. Mining Evolution Data of a Product Family. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 12-16.
[29] Miryung Kim and David Notkin. Using a Clone Genealogy Extractor for Understanding and Supporting Evolution of Code Clones. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 17-21.
[30] Jacek S´ liwerski, Thomas Zimmermann and Andreas Zeller. When Do Changes Induce Fixes? In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 24-28.
[31] Peter Weißgerber and Carsten G¨org. Error Detection by Refactoring Reconstruction. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 29-33.
[32] Jaime Spacco, Jaymie Strecker, David Hovemeyer, and William Pugh. Software Repository Mining with Marmoset: An Automated Programming Project Snapshot and Testing System. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 36-40.
[33] Keir Mierle, Kevin Laven, Sam Roweis, Greg Wilson. Mining Student CVS Repositories for Performance Indicators. In Proceedings of 2nd International Workshop on Mining Software Repositories

(MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 41-45.
[34] Masaru Ohba and Katsuhiko Gondow. Toward Mining “Concept Keywords” from Identifiers in Large Software Projects. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 48-52.
[35] Annie T.T. Ying, James L. Wright and Steven Abrams. Source code that talks: an exploration of Eclipse task comments and their implication to repository mining. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 53-57.
[36] Jane Huffman Hayes and Alex Dekhtyar and Senthil Sundaram. Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 58-63.
[37] Sunghun Kim, E. James Whitehead, Jr., Jennifer Bevan. Analysis of Signature Change Patterns. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 64-68.
[38] Jacek Ratzinger, Michael Fischer and Harald Gall. Improving Evolvability through Refactoring. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 69-73.
[39] Giuliano Antoniol, Vincenzo Fabio Rollo and Gabriele Venturi. Linear Predictive Coding and Cepstrum coefficients for mining time variant information from software repositories. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 74-78.
[40] Michael Van Hilst, Pankaj K. Garg and Christopher Lo. Repository Mining and Six Sigma for Process Improvement. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 80-83.
[41] Shih-Kun Huang, Kangmin Liu. Mining Version Histories to Verify the Learning Process of Legitimate Peripheral Participants. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 84-88.
[42] Huzefa Kagdi, Michael L. Collard, Jonathan I. Maletic. Towards a Taxonomy of Approaches for Mining of Source Code Repositories. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 89-94.
[43] Daniel M. German, Davor ˇCubrani´c and MargaretAnne D. Storey. A Framework for Describing and Understanding Mining Tools in Software Development. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 95-99.
[44] Abram Hindle, Daniel M. German. SCQL: A formal model and a query language for source control repositories. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 100-104.
[45] Gregorio Robles, Jesus M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. In Proceedings of 2nd International Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, Co-located With International Conference on Software Engineering (ICSE 2005) 106-110.
[46] Ahmed E. Hassan, Member, IEEE, Audris Mockus, Member, IEEE, Richard C. Holt, Member, IEEE, and Philip M. Johnson, Member, IEEE. 2005. Guest Editors’ Introduction: Special Issue on Mining Software Repositories. In IEEE TRANSACTIONS ON SOFTWARE ENGINEERING journal (2005).
[47] Chadd C. Williams and Jeffrey K. Hollingsworth, Senior Member, IEEE. 2005. Automatic Mining of Source Code Repositories to Improve Bug Finding Techniques. In IEEE TRANSACTIONS ON SOFTWARE ENGINEERING journal (2005).
[48] Trung T. Dinh-Trong and James M. Bieman, Senior Member, IEEE. 2005. The FreeBSD Project: A Replication Case Study of Open Source Development. In IEEE TRANSACTIONS ON SOFTWARE ENGINEERING journal (2005).
[49] Benjamin Livshits, Thomas Zimmermann. DynaMine: finding common error patterns by mining software revision histories.2005. In the proceedings of the 10th European software engineering conference held jointly with 13 th ACM SIGSOFT international symposium on Foundations of software engineering, Pages 296-305.
[50] Yuefeng Zhang and Dhaval Sheth. Gonzalez-Barahona. 2006. Mining Software Repositories for Model-Driven Development. In IEEE SOFTWATRE Magazine (January/February 2006), IEEE Computer

Society, 82-90.

[51] G. Gui and P. D Scott. 2006. Coupling and Cohesion Measures for Evaluation of Component Reusability. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 18-21.

[52] Lucian Voinea and Alexandru Telea. 2006. An Open Framework for CVS Repository Querying, Analysis and Visualization. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 33-39.

[53] Huzefa Kagdi, Shehnaaz Yusuf and Jonathan I. Maletic. 2006. Mining Sequences of Changed-files from Version Histories. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 47-53.

[54] Tao Xie and Jian Pei. 2006. MAPO: Mining API Usages from Open Source Repositories. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 54-57.

[55] Tobias Sager, Abraham Bernstein, Martin Pinzger and Christoph Kiefer. 2006. MAPO: Detecting Similar Java Classes Using Tree Algorithms. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 65-71.

[56] Thomas Zimmermann, Sunghun Kim, Andreas Zeller and E. James Whitehead Jr. 2006. Mining Version Archives for Co-changed Lines. In Proceedings of 2006 International Workshop on Mining Software Repositories (MSR 2006), ACM, Shanghai, China, 72-75.

[57] Sunghun Kim, Thomas Zimmermann, Miryung Kim, Ahmed Hassan, Audris Mockus, Tudor Girba, Martin Pinzger, E. James Whitehead, Jr., and Andreas Zeller. TA-RE: An Exchange Language for Mining Software Repositories. In the Proceedings of the 2006 ACM International Working Conference on Mining Software Repositories (MSR 2006), Shanghai, China.

[58] Hassan, A.E. 2006. Mining Software Repositories to Assist Developers and Support Managers. In the Proceedings of the 2006 IEEE International Working Conference on Software Maintenance, Philadelphia, pp. 339 – 342

[59] Huzefa Kagdi, and Jonathan I. Maletic. "Software-change prediction: Estimated+ actual." In Second International IEEE Workshop on Software Evolvability, 2006. SE’06, pp. 38-43. IEEE, 2006.

[60] Yan Wu, Harvey Siy, Mansour Zand and Victor Winter. 2007. Construction of Ontology-Based Software Repositories by Text Mining. In Proceedings of The International Conference on Computational Science 2007 (ICCS 2007), Springer-Verlag Berlin Heidelberg, Beijing, China, 790-797.

[61] Vladimir Rubin, Christian W. Gunther, Wil M.P. van der Aalst, Ekkart Kindler, Boudewijn F. van Dongen and Wilhelm Schafer. 2007. Process Mining Framework for Software Processes. In Proceedings of The International Conference on Software Process 2007 (ICSP 2007), Springer-Verlag Berlin Heidelberg, Minneapolis, MN, USA, 169-181.

[62] Romain Robbes. Mining a Change-Based Software Repository. 2007. In the Proceedings of MSR '07 Proceedings of the Fourth International Workshop on Mining Software Repositories.

[63] Kagdi, H. ; Maletic, J.I. ; Sharif, B. Mining software repositories for traceability links. 2007. In the Proceedings of the 15th IEEE International Conference on ICPC '07,pages 145-154

[64] Lucian Voinea , Alexandru Telea, Visual data mining and analysis of software repositories.2007.In the proceedings of Computer and Graphics, Elsevier. Volume 31, Issue 3, June 2007, Pages 410–428

[65] Walt Scacchi . Free/Open Source Software Development: Recent Research Results and Methods.2007. In the proceedings of Advances in Computers, Elsevier. Volume 69, 2007, Pages 243–295

[66] Huzefa Kagdi and Jonathan I. Maletic. "Combining single-version and evolutionary dependencies for software-change prediction." In Proceedings of the Fourth International Workshop on Mining Software Repositories, p. 17. IEEE Computer Society, 2007.

[67] Hemant Joshi, Chuanlei Zhang, S. Ramaswamy and Coskun Bayrak. Local and Global Recency Weighting Approach to Bug Prediction. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[68] Israel Herraiz, Jesus M. Gonzalez-Barahona, Gregorio Robles. Forecasting the number of changes in Eclipse using time series analysis. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[69] Adrian Schroter. Predicting Defects and Changes with Import Relations. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[70] Erik Linstead, Paul Rigor, Sushil Bajracharya, Cristina Lopes, Pierre Baldi. Mining Eclipse Developer Contributions via Author-Topic Models. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[71] Lucas D. Panjer. Predicting Eclipse Bug Lifetimes. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[72] John Anvik and Gail C. Murphy. Determining Implementation Expertise from Bug Reports. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[73] Jesus M. Gonzalez-Barahona, Gregorio Robles, Israel Herraiz. Impact of the Creation of the Mozilla

Foundation in the Activity of Developers. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[74] Peter C. Rigby, Ahmed E. Hassan. What can OSS mailing lists tell us? A preliminary psychometric text analysis of the Apache developer mailing list. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[75] Huzefa Kagdi, Michael L. Collard, and Jonathan I. Maletic. Comparing Approaches to Mining Source Code for Call-Usage Patterns. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[76] Abram Hindle, Michael W. Godfrey, Richard C. Holt. Release Pattern Discovery via Partitioning: Methodology and Case Study. In Proceedings of 29th International Conference on Software Engineering Workshops (ICSEW'07), IEEE Computer Society.

[77] K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra. "Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems." Journal of Object Technology 6, no. 10 (2007): 127-141.

[78] Shen Zhang, Yongji Wang, Ye Yang, and Junchao Xiao. 2008. Capability Assessment of Individual Software Development Processes Using Software Repositories and DEA. In Proceedings of The International Conference on Software Process 2008 (ICSP 2008), Springer-Verlag Berlin Heidelberg, Leipzig, Germany, 147-159.

[79] Junya Debari, Osamu Mizuno, Tohru Kikuno, Nahomi Kikuchi and Masayuki Hirayama. 2008. On Deriving Actions for Improving Cost Overrun by Applying Association Rule Mining to Industrial Project Repository. In Proceedings of The International Conference on Software Process 2008 (ICSP 2008), Springer-Verlag Berlin Heidelberg, Leipzig, Germany, 51-62.

[80] Ahmed E. Hassan. 2008. The Road Ahead for Mining Software Repositories. In Proceedings of the 2008 Frontiers of Software Maintenance (FoSM 2008), IEEE 2008, 48-57.

[81] Lucian Voinea and Alexandru Telea. 2008. Visual querying and analysis of large software repositories. In the Journal of Empirical Software Engineering (2009), Springer Science + Business Media, LLC 2008, 316-340.

[82] Miha Grcar, Marko Grobelnik, and Dunja Mladenic. 2008. Using Text Mining and Link Analysis for Software Mining. In Proceedings of the 3rd International Workshop on Mining Complex Data (MCD 2007), 2008. Springer-Verlag Berlin Heidelberg 2008 1-12.

[83] Yomi Kastro and Ayse Basar Bener. 2008. A defect prediction method for software versioning. In Software Quality Journal (2008), Springer Science + Business Media, LLC 2008, 316-340.

[84] Olivier Vandecruysa, David Martensa, Bart Baesensa, Christophe Muesb, Manu De Backera, RafHaesena, 2008. Mining software repositories for comprehensible software fault prediction models in Journal of Systems and Software in Elsevier Volume 81, Issue 5, May 2008, Pages 823–839.

[85] Van Rompaey, Bart, and Serge Demeyer. "Estimation of test code changes using historical release data." Reverse Engineering, 2008. WCRE'08. 15th Working Conference on. IEEE, 2008.

[86] Andy Zaidman, Bart Van Rompaey, Serge Demeyer, and Arie Van Deursen. "Mining software repositories to study co-evolution of production & test code." In 1st International Conference on Software Testing, Verification, and Validation, 2008, pp. 220-229. IEEE, 2008.

[87] Michael W. Godfrey, and Daniel M. German. "The past, present, and future of software evolution." In Frontiers of Software Maintenance, 2008. FoSM 2008. pp. 129-138. IEEE, 2008.

[88] Erik Linstead, Cristina Lopes, and Pierre Baldi. "An application of latent dirichlet allocation to analyzing software evolution." In Seventh International Conference on Machine Learning and Applications, 2008. ICMLA'08, pp. 813-818. IEEE, 2008.

[89] Arvinder Kaur, and Ruchika Malhotra. "Application of random forest in predicting fault-prone classes." In International Conference on Advanced Computer Theory and Engineering, 2008. ICACTE'08, pp. 37-43. IEEE, 2008.

[90] Olivier Vandecruys, David Martens, Bart Baesens, Christophe Mues, Manu De Backer, and Raf Haesen. "Mining software repositories for comprehensible software fault prediction models." Journal of Systems and software 81, no. 5 (2008): 823-839.

[91] Shen Stephan Diehl & Harald C. Gall & Ahmed E. Hassan.2009. Guest editors introduction: special issue on mining software repositories. In the Journal of Empirical Software Engineering (2009), Springer Science + Business Media, LLC 2009, 257-261.

[92] Pierre van de Laar. On the Transfer of Evolutionary Couplings to Industry. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 187-190.

[93] Joel Ossher, Sushil Bajracharya, Erik Linstead, Pierre Baldi and Cristina Lopes. SourcererDB: An Aggregated Repository of Statically Analyzed and Cross-Linked Open Source Java Projects. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 183-186.

[94] Nathaniel Ayewah, William Pugh. Learning from Defect Removals. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC,

Canada, 179-182.
[95] Adrian Kuhn. Automatic Labeling of Software Components and their Evolution using Log-Likelihood Ratio of Word Frequencies in Source Code. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 175-178.
[96] Prasanth Anbalagan, Mladen Vouk. On Mining Data across Software Repositories. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 171-174.
[97] Gregorio Robles, Jesus M. Gonzalez-Barahona, Israel Herraiz. Evolution of the core team of developers in libre software projects. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 167-170.
[98] Kai Tian, Meghan Revelle, and Denys Poshyvanyk. Using Latent Dirichlet Allocation for Automatic Categorization of Software. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 163-166.
[99] Kim Herzig and Andreas Zeller. Mining the Jazz Repository: Challenges and Opportunities. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 159-162.
[100] Sarah Rastkar and Gail C. Murphy. On What Basis to Recommend: Changesets or Interactions?. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 155-158.
[101] Zeeger Lubsen, Andy Zaidman and Martin Pinzger. Using Association Rules to Study the Co-evolution of Production & Test Code. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 151-154.
[102] Emad Shihab, Zhen Ming Jiang and Ahmed E. Hassan. On the use of Internet Relay Chat (IRC) meetings by developers of the GNOME GTK+ project. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 107-110.
[103] Lile Hattori and Michele Lanza. Mining the History of Synchronous Changes to Refine Code Ownership. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 141-150.
[104] Sushil Bajracharya Cristina Lopes. Mining Search Topics from a Code Search Engine Usage Log. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 111-120.
[105] Georgios Gousios, Diomidis Spinellis. A Platform for Software Engineering Research. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 31-40.
[106] Eric Enslen, Emily Hill, Lori Pollock and K. Vijay-Shanker. Mining Source Code to Automatically Split Identifiers for Software Analysis. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 71-80.
[107] Methanias Colaço Júnior, Manoel Mendonça, Francisco Rodrigues. Mining Software Change History in an Industrial Environment. In the Proceedings of the 2009 XXIII Brazilian Symposium on Software Engineering, 2009 IEEE Computer Society. 54-61.
[108] Dominique Matter, Adrian Kuhn, Oscar Nierstrasz. Assigning Bug Reports using a Vocabulary-Based Expertise Model of Developers. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 131-140.
[109] Mircea Lungu, Jacopo Malnati, Michele Lanza. Visualizing Gnome with The Small Project Observatory. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 103-106.
[110] Jayalath Ekanayake, Jonas Tappolet, Harald C. Gall, Abraham Bernstein. Tracking Concept Drift of Software Projects Using Defect Prediction Quality. In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 51-60.
[111] Yonghee Shin, Robert Bell, Thomas Ostrand, Elaine Weyuker. Does Calling Structure Information Improve the Accuracy of Fault Prediction?.In the Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories (MSR 2009), Vancouver, BC, Canada, 51-60.
[112] Burak Turhan, Gozde Kocak, Ayse Bener. Data mining source code for locating software bugs: A case study in telecommunication industry.2009. In the proceedings of Expert Systems with Applications,Elsevier. Volume 36, Issue 6, August 2009, Pages 9986–9990
[113] Ahmed E. Hassan, "Predicting faults using the complexity of code changes." In Proceedings of the 31st International Conference on Software Engineering, pp. 78-88. IEEE Computer Society, 2009.
[114] K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra. "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study." Software process: Improvement and practice 14, no. 1 (2009): 39-62.
[115] Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra. "Application of support vector machine to predict fault prone classes." ACM SIGSOFT Software Engineering Notes 34, no. 1 (2009): 1-6.

[116] Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra. "Software fault proneness prediction using support vector machines." In Proceedings of the world congress on engineering, vol. 1, pp. 1-3. 2009.

[117] Diomidis Spinellis, Georgios Gousios, Vassilios Karakoidas, Panagiotis Louridas, Paul J. Adams, Ioannis Samoladas, and Ioannis Stamelos. "Evaluating the quality of open source software." Electronic Notes in Theoretical Computer Science 233 (2009): 5-28.

[118] Tracy Hall, David Bowes, Gernot Liebchen, and Paul Wernick. 2010. Evaluating Three Approaches to Extracting Fault Data from Software Change Repositories. In Proceedings of the International Conference on Product-Focused Software Development and Process Improvement 2010 (PROFES 2010), Springer-Verlag Berlin Heidelberg, Limerick, Ireland, 107-115.

[119] Sheng-Kuei Hsu and Shi-Jen Lin. 2010. Mining Source Codes to Guide Software Development. In Proceedings of 2nd Asian Conference on Intelligent Information and Database Systems(ACIIDS 2010), Springer-Verlag Berlin Heidelberg, Hue, Vietnam, 445-454.

[120] Lucas Nussbaum and Stefano Zacchiroli. 2010. The Ultimate Debian Database: Consolidating Bazaar Metadata for Quality Assurance and Data Mining. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 52-61.

[121] Romain Robbes, Damien Pollet, Michele Lanza. 2010. Replaying IDE Interactions to Evaluate and Improve Change Prediction Approaches. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 161-170.

[122] Marco D'Ambros, Michele Lanza and Romain Robbes. 2010. An Extensive Comparison of Bug Prediction Approaches. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 31-41.

[123] Adrian Schroter, Nicolas Bettenburg and Rahul Premraj. 2010. Do Stack Traces Help Developers Fix Bugs?. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 118-121.

[124] Meiyappan Nagappan and Mladen A. Vouk. 2010. Abstracting Log Lines to Log Event Types for Mining Software System Logs. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 114-117.

[125] Massimiliano Di Penta, Daniel M. German and Giuliano Antoniol. 2010. Identifying Licensing of Jar Archives using a Code-Search Approach. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 151-160.

[126] Antonio Vetro, Marco Torchiano and Maurizio Morisio. 2010. Assessing the Precision of FindBugs by mining Java Projects developed at a University. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 110-113.

[127] Ahmed Lamkanfi, Serge Demeyery, Emanuel Gigery and Bart Goethalsz. 2010. Predicting the Severity of a Reported Bug. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 1-10.

[128] Ariadi Nugroho, Michel R.V. Chaudron and Erik Arisholm. 2010. Assessing UML Design Metrics for Predicting Fault-prone Classes in a Java System. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 21-30.

[129] Adrian Bachmann and Abraham Bernstein. 2010. When Process Data Quality Affects the Number of Bugs: Correlations in Software Engineering Datasets. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 62-71.

[130] Julius Davies, Hanyu Zhang, Lucas Nussbaum and Daniel M. German. 2010. Perspectives on Bugs in the Debian Bug Tracking System. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 86-89.

[131] Andreas Mauczka, Christian Schanes, Florian Fankhauser, Mario Bernhart and Thomas Grechenig. 2010. Mining Security Changes in FreeBSD. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 90-93.

[132] Jens Krinke, Nicolas Gold, Yue Jia and David Binkley. 2010. Cloning and Copying between GNOME Projects. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 98-101.

[133] Gargi Bougie, Christoph Treude, Daniel M. German and Margaret-Anne Storey. 2010. A Comparative Exploration of FreeBSD Bug Lifetimes. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 106-109.

[134] Bart Luijten, Joost Visser and Andy Zaidman. 2010. Assessment of Issue Handling Efficiency. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 94-97.

[135] Yusuke Sasaki, Tetsuo Yamamotoy, Yasuhiro Hayase and Katsuro Inoue. 2010. Finding File Clones in FreeBSD Ports Collection. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 102-105.

[136] Methanias Colaço Júnio, Manoel Mendonç, Mario Farias and Paulo Henrique. 2010. OSS Developers Context-Specific Preferred Representational Systems: A Initial Neurolinguistic Text Analysis of the

Apache Mailing List. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 126-129.
[137] Michael Gegick, Pete Rotella and Tao Xie. 2010. Identifying Security Bug Reports via Text Mining: An Industrial Case Study. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 11-20.
[138] Shane McIntosh, Bram Adams and Ahmed E. Hassan. 2010. The Evolution of ANT Build Systems. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 42-51.
[139] Roozbeh Nia, Christian Bird, Premkumar Devanbu and Vladimir Filkov. 2010. Validity of Network Analyses in Open Source Projects. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 201-209.
[140] Marcel Bruch, Mira Mezini and Martin Monperrus. 2010. Mining Subclassing Directives to Improve Framework Reuse. In Proceedings of 2010 IEEE Working Conference on Mining Software Repositories (MSR 2010), IEEE Computer Society, Cape Town, South Africa, 141-150.
[141] Barbara Kitchenham , Rialette Pretorius, David Budgen, O. Pearl Brereton, Mark Turner, Mahmood Niazi, Stephen Linkman. Systematic literature reviews in software engineering – A tertiary study.2010.In the proceedings of Information and Software Technology, Elsevier. Volume 52, Issue 8, August 2010, Pages 792–805
[142] Robles, G. Replicating MSR: A study of the potential replicability of papers published in the Mining Software Repositories proceedings. In the Proceedings of the 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), Cape Town. Pages 171 – 180.
[143] Stephen W. Thomas, Bram Adams, Ahmed E. Hassan, and Dorothea Blostein. "Validating the use of topic models for software evolution." In 10th IEEE Working Conference on Source Code Analysis and Manipulation (SCAM), 2010, pp. 55-64. IEEE, 2010.
[144] Huzefa Kagdi, Malcom Gethers, Denys Poshyvanyk, and Michael L. Collard. "Blending conceptual and evolutionary couplings to support change impact analysis in source code." In 17th Working Conference on Reverse Engineering (WCRE), 2010, pp. 119-128. IEEE, 2010.
[145] Marco D'Ambros, and Michele Lanza. "Distributed and collaborative software evolution analysis with churrasco." Science of Computer Programming 75, no. 4 (2010): 276-287.
[146] Maen Hammad, Michael L. Collard and Jonathan I. Maletic. 2011. Automatically identifying changes that impact code-to-design traceability during evolution. In Software Quality Journal (2011), Springer Science + Business Media, LLC 2010, 35–64.
[147] Petr Knoth, Vojtech Robotka and Zdenek Zdrahal. 2011. Connecting Repositories in the Open Access Domain Using Text Mining and Semantic Data. In Proceedings of International Conference on Theory and Practice of Digital Libraries 2011 (TPDL 2011), Springer-Verlag Berlin Heidelberg, Berlin, Germany, 483-487.
[148] Agustin Casamayor, Daniela Godoy and Marcelo Campo. Mining textual requirements to assist architectural software design: a state of the art review. Published online: 28 May 2011 at Springer Science+Business Media B.V. 2011, 173-191.
[149] Wouter Poncin, Alexander Serebrenik and Mark van den Brand. 2011. Process mining software repositories. In Proceedings of the 15th European Conference on Software Maintenance and Reengineering , IEEE Computer Society, 5-13.
[150] Kalpana Johari, and Arvinder Kaur. "Effect of software evolution on software metrics: an open source case study." ACM SIGSOFT Software Engineering Notes 36, no. 5 (2011): 1-8.
[151] Muhammad Asaduzzaman, Michael C. Bullock, Chanchal K. Roy and Kevin A. Schneider. Bug Introducing Changes: A Case Study with Android. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 116-119.
[152] Lee Martie, Vijay Krishna Palepu, Hitesh Sajnani, Cristina Lopes. Trendy Bugs ,Topic Trends in the Android Bug Reports. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 120-123.
[153] Victor Guana, Fabio Rocha, Abram Hindle, Eleni Stroulia. Do the Stars Align? Multidimensional Analysis of Android's Layered Architecture. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 124-127.
[154] Wei Hu, Dan Han, Abram Hindle, Kenny Wong. The Build Dependency Perspective of Android's Concrete Architecture.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 128-131.
[155] Vibha Singhal Sinha, Senthil Mani, and Monika Gupta. MINCE: MINing ChangE History of Android Project. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 132-135.
[156] Laura Arjona Reina, Gregorio Robles. Mining for Localization in Android. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 124-127.
[157] Margaret-Anne St. The Evolution of the Social Programmer.In the Proceedings of the 2012 IEEE

International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 140.
[158] Anh Cuong Nguyen and Siau-Cheng Khoo. Discovering Complete API Rules with Mutation Testing. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 151-160.
[159] Emad Shihab, Yasutaka Kamei and Pamela Bhattacharya. Mining Challenge 2012: The Android Platform. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 112-115.
[160] Mark Harman, Yue Jia, and Yuanyuan Zhang. App Store Mining and Analysis: MSR for App Stores. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 108-111.
[161] Lucia, Ferdian Thung, David Lo, and Lingxiao Jiang. Are Faults Localizable?.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 74-77.
[162] Rodrigo Souza and Christina Chavez. Characterizing Verification of Bug Fixes in Two Open Source IDEs.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 70-73.
[163] Christian Rodríguez-Bustos and Jairo Aponte. How Distributed Version Control Systems Impact Open Source Software Projects. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 36-39.
[164] Iman Keivanloo, Christopher Forbes, Aseel Hmood, Mostafa Erfani, Christopher Neal, George Peristerakis, Juergen Rilling. A Linked Data Platform for Mining Software Repositories. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 32-35.
[165] Yuan Tian, Palakorn Achananuparp, Ibrahim Nelman Lubis, David Lo, and Ee-Peng Lim. What Does Software Engineering Community Microblog About?.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 247-250.
[166] Maximilian Steff, Barbara Russo. Co-evolution of Logical Couplings and Commits for Defect Estimation.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 213-216.
[167] Aigerim Issabayeva, Ariadi Nugroho, and Joost Visser. Issue Handling Performance in Proprietary Software Projects.In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 209-212.
[168] Andrea Capiluppi, Alexander Serebrenik and Ahmmad Youssef. Developing an H-Index for OSS Developers. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 251-254.
[169] Alexander Breckel. Error Mining: Bug Detection through Comparison with Large Code Databases. In the Proceedings of the 2012 IEEE International Working Conference on Mining Software Repositories (MSR 2012), Zurich, Switzerland, 175-178.
[170] Gerardo Canfora, Luigi Cerulo, Marta Cimitile and Massimiliano Di Penta. 2012. How changes affect software entropy: an empirical study. In Emperical Software Engineering Journal (2012), Springer Science+Business Media LLC 2012.
[171] Xiang Wang, Xiaoming Jin, Meng-En Chen, Kai Zhang, and Dou Shen. 2012. Topic Mining over Asynchronous Text Sequences. In IEEE TRANSACTIONS ON SOFTWARE ENGINEERING journal(2012).
[172] Changyun Huang, Kazuhiro Yamashita, Yasutaka Kamei, Kenji Hisazumi and Naoyasu Ubayashi. 2013. Domain Analysis for Mining Software Repositories: Towards Feature-Based DSL Construction. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 41-44.
[173] Ramin Shokripour, John Anviky, Zarinah M. Kasirun and Sima Zamani. 2013. Why So Complicated? Simple Term Filtering and Weighting for Location-Based Bug Report Assignment Recommendation. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 2-11.
[174] Hoda Naguib, Nitesh Narayan, Bernd Brugge and Dina Helal. 2013. Bug Report Assignee Recommendation using Activity Profiles. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 22-30.
[175] Siddharth Subramanian and Reid Holmes. 2013. Making Sense of Online Code Snippets. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 85-88.
[176] Kim Herzig and Andreas Zeller. 2013. The Impact of Tangled Code Changes. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 121-130.
[177] Bogdan Dit, Andrew Holtzhauer, Denys Poshyvanyk and Huzefa Kagdi. 2013. A Dataset from Change

History to Support Evaluation of Software Maintenance Tasks. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 131-134.
[178] Mehdi Amoui, Nilam Kaushik, Abraham Al-Dabbagh, Ladan Tahvildari, Shimin Li and Weining Liu. 2013. Search-Based Duplicate Defect Detection: An Industrial Experience. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 173-182.
[179] Anahita Alipour, Abram Hindle and Eleni Stroulia. 2013. A Contextual Approach towards More Accurate Duplicate Bug Report Detection. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 183-192.
[180] Debdoot Mukherjee and Malika Garg. 2013. Which Work-Item Updates Need Your Response?. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 12-21.
[181] Ryan Stevens, Jonathan Ganz, Vladimir Filkov, Premkumar Devanbu and Hao Chen. 2013. Asking for (and about) Permissions Used by Android Apps. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 31-40.
[182] Claudia Iacob and Rachel Harrison. 2013. Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 41-44.
[183] Murtuza Mukadam, Christian Bird and Peter C. Rigby. 2013. Gerrit Software Code Review Data from Android. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 45-48.
[184] Kazuki Hamasaki, Raula Gaikovina Kula, Norihiro Yoshida, A. E. Camargo Cruz, Kenji Fujiwara and Hajimu Iida. 2013. Who Does What during a Code Review? Datasets of OSS Peer Review Repositories. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 49-52.
[185] Miltiadis Allamanis and Charles Sutton. 2013. Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 53-56.
[186] Joshua Charles Campbell, Chenlei Zhang, Zhen Xu, Abram Hindle and James Miller. 2013. Deficient Documentation Detection A Methodology to Locate Deficient Project Documentation using Topic Analysis. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 57-60.
[187] Wei Wang and Michael W. Godfrey. 2013. Detecting API Usage Obstacles: A Study of iOS and Android Developer Questions. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 61-64.
[188] Patrick Morrison and Emerson Murphy-Hill. 2013. Is Programming Knowledge Related To Age? An Exploration of Stack Overflow. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 69-72.
[189] Vibha Singhal Sinha, Senthil Mani, and Monika Gupta. 2013. Exploring Activeness of Users in QA Forums. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 77-80.
[190] Carlos Gomez, Brendan Cleary and Leif Singer. 2013. A Study of Innovation Diffusion through Link Sharing on Stack Overflow. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 81-84.
[191] Amiangshu Bosu, Christopher S. Corley, Dustin Heaton, Debarshi Chatterji, Jeffrey C. Carver and Nicholas A. Kraft. 2013. Building Reputation in StackOverflow: An Empirical Investigation. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 89-92.
[192] Mario Linares-Vásquez, Bogdan Dit and Denys Poshyvanyk. 2013. An Exploratory Analysis of Mobile Development Issues using Stack Overflow. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 93-96.
[193] Muhammad Asaduzzaman, Ahmed Shah, Mashiyaty Chanchal, K. Roy and Kevin A. Schneider. 2013. Answering Questions about Unanswered Questions of Stack Overflow. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 97-100.
[194] Yujuan Jiang, Bram Adams and Daniel M. German. 2013. Will My Patch Make It? And How Fast? Case Study on the Linux Kernel. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 101-110.
[195] Abdulkareem Alali, Brian Bartman, Christian D. Newman and Jonathan I. Maletic. 2013. A Preliminary Investigation of Using Age and Distance Measures in the Detection of Evolutionary Couplings. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 169-172.

[196] Tao Xie , Thomas Zimmermann and Arie van Deursen. Introduction to the special issue on mining software repositories. In the Journal of Empirical Software Engineering (2013), Springer Science+Business Media New York 2013 ,1043-1046.
[197] Mathieu Goeminne, and Tom Mens. "A comparison of identity merge algorithms for software repositories." Science of Computer Programming 78, no. 8 (2013): 971-986.
[198] Saeed Parsa, Mojtaba Vahidi-Asl and Maryam Asadi-Aghbolaghi. 2014. Hierarchy-Debug: a scalable statistical technique for fault localization. In Software Quality Journal (2014), Springer Science+Business Media New York 2013, 427–466.
[199] Bora Caglayan, Ayse Tosun Misirli, Ayse Basar Bener and Andriy Miranskyy. 2014. Predicting defective modules in different test phases. In Software Quality Journal (2014), Springer Science+Business Media New York 2014.
[200] Mark D. Syer, Meiyappan Nagappan, Bram Adams and Ahmed E. Hassan. 2014. Studying the relationship between source code quality and mobile platform dependence. In Software Quality Journal (2014), Springer Science+Business Media New York 2014.
[201] Lech Madeyski and Marian Jureczko. 2014. Which process metrics can significantly improve defect prediction models? An empirical study. In Software Quality Journal (2014), Springer Science+Business Media New York 2014.
[202] Matías Martínez and Martin Monperrus. 2013. Mining software repair models for reasoning on the search space of automated program fixing. In Emperical Software Engineering Journal (2013), Springer Science+Business Media New York 2013.
[203] Jiachen Yang & Keisuke Hotta&Yoshiki Higo & Hiroshi Igaki& Shinji Kusumoto. 2014. Classification model for code clones based on machine learning. In Emperical Software Engineering Journal (2014), Springer Science+BusinessMedia 2014.
[204] Bajracharya, Sushil, Joel Ossher, and Cristina Lopes. "Sourcerer: An infrastructure for large-scale collection and analysis of open-source code." Science of Computer Programming 79 (2014): 241-259.
[205] Thomas, Stephen W., et al. "Studying software evolution using topic models." Science of Computer Programming 80 (2014): 457-479.
[206] Borg, Markus, Per Runeson, and Anders Ardö. "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability." Empirical Software Engineering 19.6 (2014): 1565-1616.
[207] Linares-Vásquez, Mario, et al. "Mining energy-greedy api usage patterns in android apps: an empirical study." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[208] Hindle, Abram, et al. "Greenminer: A hardware based mining software repositories software energy consumption framework." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[209] Ponzanelli, Luca, et al. "Mining StackOverflow to turn the IDE into a self-confident programming prompter." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[210] Bajaj, Kartik, Karthik Pattabiraman, and Ali Mesbah. "Mining questions asked by web developers." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[211] McIntosh, Shane, et al. "The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[212] Kononenko, Oleksii, et al. "Mining modern repositories with elasticsearch." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[213] Mittal, Megha, and Ashish Sureka. "Process mining software repositories from student projects in an undergraduate software engineering course." Companion Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.
[214] Martinez, Matias, Westley Weimer, and Martin Monperrus. "Do the fix ingredients already exist? an empirical inquiry into the redundancy assumptions of program repair approaches." Companion Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.
[215] Rattan, Dhavleesh, Rajesh Bhatia, and Maninder Singh. "Software clone detection: A systematic review." Information and Software Technology 55.7 (2013): 1165-1199.
[216] Canfora, Gerardo, Luigi Cerulo, and Massimiliano Di Penta. "Identifying Changed Source Code Lines from Version Repositories." MSR. Vol. 7. 2007.
[217] Lozano, Angela, Michel Wermelinger, and Bashar Nuseibeh. "Evaluating the harmfulness of cloning: A change based experiment." Proceedings of the Fourth International Workshop on Mining Software Repositories. IEEE Computer Society, 2007.
[218] Thummalapenta, Suresh, et al. "An empirical study on the maintenance of source code clones." Empirical Software Engineering 15.1 (2010): 1-34.
[219] Krinke, Jens. "Is cloned code older than non-cloned code?." Proceedings of the 5th International Workshop on Software Clones. ACM, 2011.
[220] Kim, Miryung, and David Notkin. "Program element matching for multi-version program

analyses." Proceedings of the 2006 international workshop on Mining software repositories. ACM, 2006.
[221] Sager, Tobias, et al. "Detecting similar Java classes using tree algorithms." Proceedings of the 2006 international workshop on Mining software repositories. ACM, 2006.
[222] Sharif, Khaironi Y., et al. "An empirically-based characterization and quantification of information seeking through mailing lists during open source developers' software evolution." Information and Software Technology 57 (2015): 77-94.
[223] Laguna, Miguel A., and Yania Crespo. "A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring." Science of Computer Programming 78.8 (2013): 1010-1034.
[224] Dyer, Robert, et al. "Demonstrating programming language feature mining using Boa." Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity. ACM, 2015.
[225] Abate, Pietro, et al. "Mining component repositories for installability issues." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[226] Hauff, Claudia, and Georgios Gousios. "Matching GitHub developer profiles to job advertisements." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[227] Ponzanelli, Luca, Andrea Mocci, and Michele Lanza. "Summarizing complex development artifacts by mining heterogeneous data." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[228] Serebrenik, Alexander, and Tom Mens. "Challenges in software ecosystems research." Proceedings of the 2015 European Conference on Software Architecture Workshops. ACM, 2015.
[229] Ghotra, Baljinder, Shane McIntosh, and Ahmed E. Hassan. "Revisiting the impact of classification techniques on the performance of defect prediction models." Proceedings of the 37th International Conference on Software Engineering-Volume 1. IEEE Press, 2015.
[230] Altinger, Harald, et al. "A novel industry grade dataset for fault prediction based on model-driven developed automotive embedded software." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[231] Linares-Vásquez, Mario, et al. "Mining android app usages for generating actionable gui-based execution scenarios." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[232] Rajan, Hridesh, et al. "Inferring behavioral specifications from large-scale repositories by leveraging collective intelligence." Proceedings of the 37th International Conference on Software Engineering-Volume 2. IEEE Press, 2015.
[233] Hsu, Sheng-Kuei, and Shi-Jen Lin. "MACs: Mining API code snippets for code reuse." Expert Systems with Applications 38.6 (2011): 7291-7301.
[234] Selby, Richard W. "Enabling reuse-based software development of large-scale systems." IEEE Transactions on Software Engineering 31.6 (2005): 495-510.
[235] Hummel, Oliver, Werner Janjic, and Colin Atkinson. "Code conjurer: Pulling reusable software out of thin air." IEEE software 25.5 (2008): 45-52.
[236] Lu, Kangjie, et al. "deRop: removing return-oriented programming from malware." Proceedings of the 27th Annual Computer Security Applications Conference. ACM, 2011.
[237] Hill, Emily, Lori Pollock, and K. Vijay-Shanker. "Automatically capturing source code context of NL-queries for software maintenance and reuse." Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009.
[238] Linstead, Erik, et al. "Sourcerer: mining and searching internet-scale software repositories." Data Mining and Knowledge Discovery 18.2 (2009): 300-336.
[239] Hummel, Oliver, and Colin Atkinson. "Using the Web as a Reuse Repository." International Conference on Software Reuse. Springer Berlin Heidelberg, 2006.
[240] Knab, Patrick, Martin Pinzger, and Abraham Bernstein. "Predicting defect densities in source code files with decision tree learners." Proceedings of the 2006 international workshop on Mining software repositories. ACM, 2006.
[241] Livshits, Benjamin, and Thomas Zimmermann. "DynaMine: finding common error patterns by mining software revision histories." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 5. ACM, 2005.
[242] Mens, Tom, and Mathieu Goeminne. "Analysing the evolution of social aspects of open source software ecosystems." IWSECO@ ICSOB. 2011.
[243] Zaidman, Andy, et al. "Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining." Empirical Software Engineering 16.3 (2011): 325-364.
[244] Karus, Siim, and Harald Gall. "A study of language usage evolution in open source software." Proceedings of the 8th Working Conference on Mining Software Repositories. ACM, 2011.
[245] Jermakovics, Andrejs, Alberto Sillitti, and Giancarlo Succi. "Mining and visualizing developer networks from version control systems." Proceedings of the 4th International Workshop on Cooperative and

Human Aspects of Software Engineering. ACM, 2011.
[246] Hemel, Armijn, et al. "Finding software license violations through binary code clone detection." Proceedings of the 8th Working Conference on Mining Software Repositories. ACM, 2011.
[247] Giger, Emanuel, Martin Pinzger, and Harald C. Gall. "Comparing fine-grained source code changes and code churn for bug prediction." Proceedings of the 8th Working Conference on Mining Software Repositories. ACM, 2011.
[248] Zaman, Shahed, Bram Adams, and Ahmed E. Hassan. "Security versus performance bugs: a case study on firefox." Proceedings of the 8th working conference on mining software repositories. ACM, 2011.
[249] Davies, Julius, et al. "Software bertillonage: finding the provenance of an entity." Proceedings of the 8th working conference on mining software repositories. ACM, 2011.
[250] Thomas, Stephen W., et al. "Modeling the evolution of topics in source code histories." Proceedings of the 8th working conference on mining software repositories. ACM, 2011.
[251] Brun, Yuriy, et al. "Proactive detection of collaboration conflicts." Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. ACM, 2011.
[252] Kim, Miryung, Dongxiang Cai, and Sunghun Kim. "An empirical investigation into the role of API-level refactorings during software evolution." Proceedings of the 33rd International Conference on Software Engineering. ACM, 2011.
[253] Hindle, Abram, et al. "Automated topic naming to support cross-project analysis of software maintenance activities." Proceedings of the 8th Working Conference on Mining Software Repositories. ACM, 2011.
[254] Alves, Vander, et al. "Requirements engineering for software product lines: A systematic literature review." Information and Software Technology 52.8 (2010): 806-820.
[255] Cleland-Huang, Jane, et al. "Automated classification of non-functional requirements." Requirements Engineering 12.2 (2007): 103-120.
[256] Mittal, Megha, and Ashish Sureka. "Process mining software repositories from student projects in an undergraduate software engineering course." Companion Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.
[257] Davril, Jean-Marc, et al. "Feature model extraction from large collections of informal product descriptions." Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, 2013.
[258] Hassan, Ahmed E., and Tao Xie. "Software intelligence: the future of mining software engineering data." Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, 2010.
[259] Menzies, Tim, et al. "The inductive software engineering manifesto: principles for industrial data mining." Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering. ACM, 2011.
[260] Asundi, Jai. "The need for effort estimation models for open source software projects." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 4. ACM, 2005.
[261] Moeyersoms, Julie, et al. "Comprehensible software fault and effort prediction: A data mining approach." Journal of Systems and Software 100 (2015): 80-90.
[262] Sun, Xiaobing, et al. "MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks." Information and Software Technology 66 (2015): 1-12.
[263] Khalid, Hammad, et al. "What do mobile app users complain about?." IEEE Software 32.3 (2015): 70-77.
[264] Ortu, Marco, et al. "The jira repository dataset: Understanding social aspects of software development." Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 2015.
[265] Dyer, Robert, et al. "Boa: Ultra-Large-Scale Software Repository and Source-Code Mining." ACM Transactions on Software Engineering and Methodology (TOSEM) 25.1 (2015): 7.
[266] Martinez, Matias, and Martin Monperrus. "Mining software repair models for reasoning on the search space of automated program fixing." Empirical Software Engineering 20.1 (2015): 176-205.
[267] Thongtanunam, Patanamon, et al. "Investigating code review practices in defective files: An empirical study of the qt system." 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015.
[268] Sureka, Ashish, Atul Kumar, and Shrinath Gupta. "Ahaan: Software process intelligence: Mining software process data for extracting actionable information." Proceedings of the 8th India Software Engineering Conference. ACM, 2015.
[269] Choetkiertikul, Morakot, et al. "Characterization and prediction of issue-related risks in software projects." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.
[270] Ding, Wei, et al. "Understanding the Causes of Architecture Changes Using OSS Mailing Lists." International Journal of Software Engineering and Knowledge Engineering 25.09n10 (2015): 1633-1651.

[271] Kamei, Y., Fukushima, T., McIntosh, S., Yamashita, K., Ubayashi, N., & Hassan, A. E. (2016). Studying just-in-time defect prediction using cross-project models. Empirical Software Engineering, 21(5), 2072-2106.

[272] Kamei, Y., Fukushima, T., Wang, S., Liu, T., & Tan, L. (2016, May). Automatically learning semantic features for defect prediction. In Proceedings of the 38th International Conference on Software Engineering (pp. 297-308). ACM.
[273] Zhou, Y., Tong, Y., Gu, R., & Gall, H. (2016). Combining text mining and data mining for bug report classification. Journal of Software: Evolution and Process, 28(3), 150-176.
[274] Di Nucci, D., Palomba, F., De Rosa, G., Bavota, G., Oliveto, R., & De Lucia, A. (2018). A developer centered bug prediction model. IEEE Transactions on Software Engineering, 44(1), 5-24.
[275] Herzig, K., Just, S., & Zeller, A. (2016). The impact of tangled code changes on defect prediction models. Empirical Software Engineering, 21(2), 303-336.
[276] Zhang, F., Zheng, Q., Zou, Y., & Hassan, A. E. (2016, May). Cross-project defect prediction using a connectivity-based unsupervised classifier. In Proceedings of the 38th International Conference on Software Engineering (pp. 309-320). ACM.
[277] Li, W., Huang, Z., & Li, Q. (2016). Three-way decisions based software defect prediction. Knowledge-Based Systems, 91, 263-274.
[278] Di Sorbo, A., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., & Gall, H. C. (2016, November). What would users change in my app? summarizing app reviews for recommending software changes. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 499-510). ACM.
[279] Saini, M., & Kaur, K. (2016). Fuzzy analysis and prediction of commit activity in open source software projects. IET Software, 10(5), 136-146.
[280] Choudhary, A., Baghel, A. S., & Sangwan, O. P. (2016, January). Software reliability prediction modeling: a comparison of parametric and non-parametric modeling. In Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference (pp. 649-653). IEEE.
[281] Lou, J., Jiang, Y., Shen, Q., Shen, Z., Wang, Z., & Wang, R. (2016). Software reliability prediction via relevance vector regression. Neurocomputing, 186, 66-73.
[282] Koroglu, Y., Sen, A., Kutluay, D., Bayraktar, A., Tosun, Y., Cinar, M., & Kaya, H. (2016, May). Defect prediction on a legacy industrial software: A case study on software with few defects. In Conducting Empirical Studies in Industry (CESI), 2016 IEEE/ACM 4th International Workshop on (pp. 14-20). IEEE.
[283] Malhotra, R., & Khanna, M. (2017, July). Software change prediction using voting particle swarm optimization based ensemble classifier. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 311-312). ACM.
[284] Zhang, F., Hassan, A. E., McIntosh, S., & Zou, Y. (2017). The use of summation to aggregate software metrics hinders the performance of defect prediction models. IEEE Transactions on Software Engineering, 43(5), 476-491.
[285] Bansal, A. (2017). Empirical analysis of search based algorithms to identify change prone classes of open source software. Computer Languages, Systems & Structures, 47, 211-231.
[286] Yang, X., Lo, D., Xia, X., & Sun, J. (2017). TLEL: A two-layer ensemble learning approach for just-in-time defect prediction. Information and Software Technology, 87, 206-220.
[287] Kaur, K., & Kaur, P. (2017, September). Evaluation of sampling techniques in software fault prediction using metrics and code smells. In Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on (pp. 1377-1387). IEEE.
[288] Gousios, G., & Spinellis, D. (2017, May). Mining software engineering data from GitHub. In Proceedings of the 39th International Conference on Software Engineering Companion (pp. 501-502). IEEE Press.
[289] Le, K., Chua, C., & Wang, R. (2017, December). Mining Software Engineering Team Project Work Logs to Generate Formative Assessment. In Software Engineering Conference Workshops (APSECW), 2017 24th Asia-Pacific (pp. 78-83). IEEE.
[290] Bagnato, A., Barmpis, K., Bessis, N., Cabrera-Diego, L. A., Di Rocco, J., Di Ruscio, D., ... & Korkontzelos, I. (2017, July). Developer-Centric Knowledge Mining from Large Open-Source Software Repositories (CROSSMINER). In Federation of International Conferences on Software Technologies: Applications and Foundations (pp. 375-384). Springer, Cham.
[291] McIlroy, S., Shang, W., Ali, N., & Hassan, A. E. (2017). Is It Worth Responding to Reviews? Studying the Top Free Apps in Google Play. IEEE Software, 34(3), 64-71. (software performance analysis)
[292] Rebouças, M., Santos, R. O., Pinto, G., & Castor, F. (2017, May). How does contributors' involvement influence the build status of an open-source software project? In Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on (pp. 475-478). IEEE.
[293] Wang, X., Peng, Y., & Zhang, B. (2018). Comment Generation for Source Code: State of the Art, Challenges and Opportunities. arXiv preprint arXiv:1802.02971.
[294] Agrawal, A., Fu, W., & Menzies, T. (2018). What is Wrong with Topic Modeling?(and How to Fix it Using Search-based Software Engineering). Information and Software Technology.

[295] Arora, R., & Garg, A. (2018). Analysis of Software Repositories Using Process Mining. In Smart Computing and Informatics (pp. 637-643). Springer, Singapore.
[296] Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2018). Gitana: A software project inspector. Science of Computer Programming, 153, 30-33.
[297] Sahu, K., Lilhore, U. K., & Agarwal, N. (2018). Survey of Various Data Reduction Methods for Effective Bug Report Analysis.
[298] Hora, A., Silva, D., Tulio, M., & Robbes, R. (2018). Assessing the Threat of Untracked Changes in So ware Evolution.
[299] Fatema, K., Syeed, M. M., & Hammouda, I. (2018). Demography of Open Source Software Prediction Models and Techniques. In Optimizing Contemporary Application and Processes in Open Source Software (pp. 24-56). IGI Global.
[300] Moonen, L., Rolfsnes, T., Binkley, D., & Di Alesio, S. (2018). What are the effects of history length and age on mining software change impact? Empirical Software Engineering, 1-36.

References

Huzefa Kagdi, Michael L. Collard and Jonathan I. Maletic. 2007. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. In JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE, Wiley InterScience, 77-131.

K.K. Chaturvedi, V.B. Singh and Prashast Singh. 2013. Tools in Mining Software Repositories. In Proceedings of 13th International Conference on Computational Science and Its Applications, IEEE CPS, Ho Chi Minh City, Vietnam, 89-98.

Hadi Hemmati, Sarah Nadi, Olga Baysal, Oleksii Kononenko, Wei Wang, Reid Holmes and Michael W. Godfrey. The MSR Cookbook Mining a Decade of Research. In Proceedings of 2013 IEEE Working Conference on Mining Software Repositories (MSR 2013), IEEE Computer Society, San Francisco, CA, USA, 343-352.

Kitchenham, B. A., Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report EBSE-2007.1-65.

Popay, Jennie, Helen Roberts, Amanda Sowden, Mark Petticrew, Lisa Arai, Mark Rodgers, Nicky Britten, Katrina Roen, and Steven Duffy. "Guidance on the conduct of narrative synthesis in systematic reviews." A product from the ESRC methods programme. Version 1 (2006).

Arvinder Kaur- Dr.(Mrs.) Arvinder Kaur is a Dean and Professor in the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi. She obtained her B.E(Electrical Engineering) & M.E. (Computer Engineering) from Thapar Institute of Engg. & Technology, Patiala, Punjab. She Obtained Ph.D. from GGS Indraprastha University Delhi . She was 4th topper in B.E.(Electrical Engineering) and ranked 2nd in M.E.(Computer Engineering) with C.G.P.A. of 9.81. Prior to joining Guru Gobind Singh Indraprastha University , she worked with Dr. B.R. Ambedkar Regional Engineering College, Jalandhar (1993-2000) and Thapar Institute of Engg. & Tech., Patiala. She has also worked for Gabriel India Ltd., Parwanoo(H.P) as Engineer (Research & Development). She has published more than 100 papers in International/ National journals & Conferences. Her seven research papers have been published as Book chapters in various Springer publications. Her Arnetminer Index is 7. Her research interests include Software Engineering, Object-Oriented Software Engineering, Software Metrics, Applications of Artificial Intelligence in Software Engineering, Software Project Management and Software Metrics. She was awarded Career award for young Teachers from All India Council for Technical Education (AICTE), Delhi for three years from 2006 of Rs. 10.5 lakh. She has been awarded Research Award consecutively for four years (2010, 2011 ,2012, 2013) by Guru Gobind Singh Indraprastha University . She has visited, & delivered lectures in International conferences at Kingston College , London (U.K.), University of Tor Vergata Rome (Italy), Las Vegas (U.S.A.), Hongkong & Phuket (Thailland). She is a lifetime member of ISTE & CSI. She has chaired technical sessions in National as well as International conferences. She was co-ordinator for Ph.D programme of University School of Information & Communication Technology, & chairperson Staff Deverlopment Cell, Guru Gobind Singh Indraprastha University, Delhi.

Kamaldeep Kaur is with University School of Information, Communication and Technology, Guru Gobind Singh Indraprastha University. She has 20 years of experience in software development, consulting and teaching computer science and software engineering. She obtained her doctorate and M.Tech degree in Information Technology from Guru Gobind Singh Indraprastha University. Her research interests include software engineering, software quality management, software metrics, software testing, Internet of Things, Cloud computing and Computer Architecture. She is a lifetime member of ISTE. She has published more than 20 research papers in scopus indexed journals and IEEE conferences. She has successfully completed UGC major project “Design and development of Techniques for measurement and prediction of software quality” in 2016. She is also Head of Microprocessor Lab and also Head of IEEE Women in Engineering Chapter at University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University.

Deepti Chopra is pursuing her Ph.D. from University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University. She has been working as Assistant Professor in the Department of Computer Science at Indraprastha College for Women, University of Delhi since 2016. She obtained her M.Tech (I.T.) and B.Tech (CSE) degrees from Guru Gobind Singh Indraprastha University. Her research interests include software engineering, data mining, software metrics, mining software repositories, text mining and software quality.

Harguneet Kaur is pursuing Ph.D as a Full Time Scholar with University School of Information, Communication and Technology, Guru Gobind Singh Indraprastha University. She had 2.5 years of experience as a Assistant Professor in Department of Information Technology in Guru Tegh Bahadur Institute of Technology. She had 1.5 year of industrial experience in Infosys Technology Limited, Bangalore as a System Engineer. She obtained her M.Tech and B.Tech degree in Information Technology from Guru Gobind Singh Indraprastha University. Her research interests include software engineering, software quality management, software metrics, software testing. She has published research papers in scopus indexed journals and IEEE conferences