Advanced Literature Review
Mining Software Engineering Repositories

Bhavana Pallepati, Venkata Sarath Kumar Galimelu

Abstract: Mining Software Engineering Repositories describes a broad class of inquiry into the examination of software repositories. This field analyzes the rich amount of data available in software industry such as Metadata information about the software change: user-ids, comments and timestamps. Various differences between the software project versions: adding up, deleting or modifying the code, Classification of different software versions (artifacts) by mining those repositories. In our Literature Review on this topic, we mainly focus on what types of software engineering data are available for mining, methodology being followed in mining the software repositories, general problems being faced in the software Project development cycle, Preprocessing of data and the techniques used for preprocessing, Algorithms mainly used for mining, software engineering tasks that can benefit from mining software engineering data, the data mining techniques that are used in software engineering, the advantages of mining in Software Repositories, challenges in applying the techniques, tools used in mining software repositories and determining the defective software modules etc. The final expected outcome of this review will showcase the advantages of mining software repositories, new approaches to be used for mining, analysis of results from different tools used for mining and the determine some of the best techniques that are published and discussed in the conference papers.

I. INTRODUCTION

Software Industry has large amounts of data produced daily from various sources of the project. Generally Software companies collect data from software projects such as number of staff hours, customer satisfaction, records of product size, meeting deadlines and improving quality in the development of software etc., Software companies may use this data for estimating developer efforts, bug density, productivity and determining a wide range of development standards. The failure data is also maintained using different bug reporting and Tracking system. This data is very useful in conducting research on software reliability finding developer expertise, quality of software, resource utilization effort, cost and time estimation, duplicate detection, dependency analysis, guiding co-change analysis, change prediction and many other analysis. To conduct all these tasks, we need the accessibility to the software repositories and analyzing it is called mining software repositories (MSR). Hence Software developers and researchers have identified the importance of Data Mining in Software Engineering Repositories as this information is used to support the maintenance of software systems, improve software design/reuse, and empirically validate novel ideas and techniques as it helps to comprehend the development and evolution of the software to support predictions to build the Prediction models, Pattern Identification tools and techniques by using large amount of Information Technology in various domains and to support future software development cycle, and to utilize this information in planning future development.
Working with Nokia, Gall et al. showed that exploiting software repositories can provide support to developers in changing legacy systems by pointing out hidden code dependencies. Working with Bell Labs and Avaya, Graves et al. and Mockus et al. demonstrated that using historical change information can support management in building reliable software systems by predicting bugs and effort. Working on open source projects, Chen et al. showed that using historical information can assist developers in understanding large systems.

Although software repositories are readily available for most large software projects, the data stored in these repositories has not been the focus of software engineering research until recently primarily because of two reasons.
Limited Access to Repositories. - Companies in many cases are not willing to give researchers access to such detailed information about their software systems
Complexity of Data Extraction. -Most repositories are not designed with automated bulk data-extraction and mining in mind, so they provide limited support for automated extraction.

With the advent of open source systems, easy access to repositories of large software systems became a reality
In this paper we will discuss the following mentioned topics

1. **Software Engineering:**
a) The types of software engineering repositories that are available for mining
b) The software engineering tasks that can get benefitted from mining software engineering repositories
c) The approach by how the techniques of data mining are used in software engineering.
2. **Data mining**
a) The challenges that are faced in applying the DM techniques to SE repositories.
b) The DM techniques that are most appropriate for specific types of SE repositories.

3. Future Directions:

The challenges and the future directions in applying new methodologies to be followed for the data mining and software engineering communities

The significance of Mining Software Repositories has been widely increasing with its first workshop on MSR started since 2004 and is considered a highly rated international conference on software engineering. Most of the researchers and practitioners are focusing on identifying about the new techniques and development of new tools to uncover the ways in which mining these repositories can help to understand software development and software evolution, to support predictions about software development, and to exploit this knowledge concretely in planning future development.

Mining Software Repositories research mainly focuses on two aspects:

The first aspect is the creating new techniques to improvise the data extraction of huge amount of software repositories and it focusses on to ease the adoption of MSR techniques by others.

The second aspect is the discovery and validation of new approaches to mine valuable information from those extracted repositories and it focusses on the importance and value of information stored in software repositories and encourages others to adopt MSR techniques.

II. SOFTWARE ENGINEERING DEVELOPMENT CHALLENGES

Understanding Software Systems:

Understanding huge software projects remains a challenge for many software companies as the Documentations for large projects have no proper documentation and even if they exist they are often not up-to-date. The experts and senior developers are usually too busy to help new or novice developers, or may no longer be part of an organization. The data that is stored in historical software repositories, such as bug repositories and mailing lists represent a group memory for a project and is very valuable for current team members of a project.

Propagating Changes

Propagation is the process of propagating the code changes to other entities of a software system so that the entire project maintains the consistency of assumptions in the entire Project after making the changes to any particular entity. For example any change in a component might require the change to be propagated to all the other components which are using that particular component in order to maintain code equivalence and to avoid any future bugs from those components. Generally the change propagation is handled either by emails or any other means of communications between humans and many defects would occur due to the inconsistency that occurred through improper propagation of changes because of unnoticed dependencies between the entities.

Predicting and Identifying Bugs

Predicting the instance of bugs and defects in a large software systems is one of the most prominent areas in software engineering research. By using this information, managers can allocate testing resource appropriately, developers can review risky code more closely, and testers can prioritize their testing efforts.

Understanding Team Dynamics

Many large projects communicate through mailing lists, IRC channels, or instant messaging. These discussions cover many important topics such as future plans, design decisions, project policies, and code or patch reviews. These discussions represent a rich source of historical information about the inner workings of large projects. These discussions could be mined to better understand the dynamics of large software development teams. For example, a psychometric text analysis tool is used to analyze the mailing lists discussions to capture the overall morale of a development team before and after release time for the Apache web server project.

Reusing Code

Code reuse is an important activity in modern software development. Unfortunately, large code libraries are usually not well-documented, and have flexible and complex APIs which are hard to use by non-experts. Several researchers have proposed tools and techniques to mine code repositories to help developers reuse code.

III. SOFTWARE REPOSITORIES

Software data has been mainly classified as

Historical Repositories

The data from source control repositories, bug repositories, and archived communications record several information about the evolution and progress of a project.

Run-Time Repositories

The data from deployment logs contain information about the execution and the usage of an application at a single or multiple deployment sites.

Code Repositories

The data from Sourceforge.net and Google code contain the source code of various applications developed by several developers.

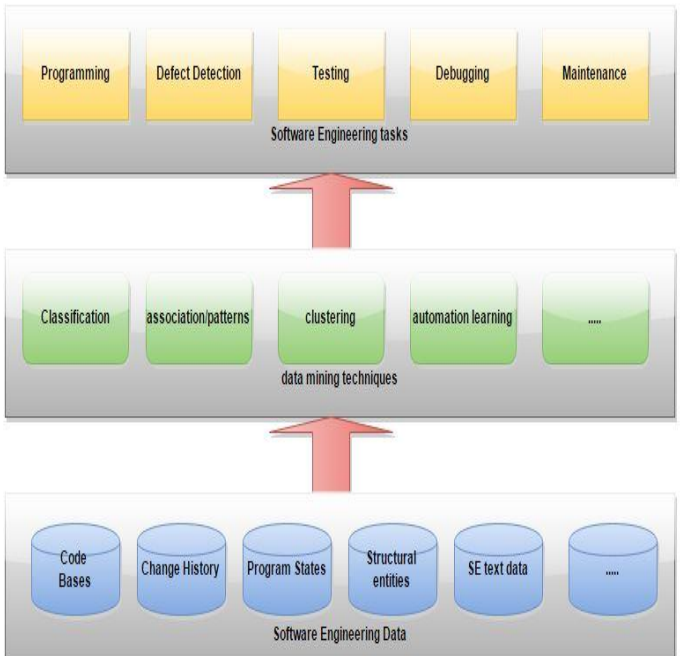
To improve software development cycle and productivity of the software being developed, Researchers and practitioners from software Engineering and Data mining are trying to apply the Data Mining Algorithms in order to utilize the rich information available in those software repositories to gain the benefits of resolving the issues or the general problems or challenges that are seen in the Software engineering project cycle in the product development.

Below is the table listed with various types of data such as source control repositories, Bug Repositories, Archived communications, deployment logs and code repositories. All these data can be used for applying the effective mining techniques that suits the data accordingly in order to perform various software engineering tasks that can benefit over the obtained information. Generally, people used various types of data from MSR Challenge and popularly known source code repositories as Linux (linux.org) Kernel, Eclipse (eclipse.org), GNOME (gnome.org), Mozilla (Mozilla.org), FreeBSD (freebsd.org), NASA's PROMISE (promisedata.org), MySQL (mysql.com), Postgres (postgresql.org), Python (python.net), AgroUML (agrouml.tigris.org), Apache (apache.org), Wikipedia (Wikipedia.org), OpenOffice (openoffice.org),

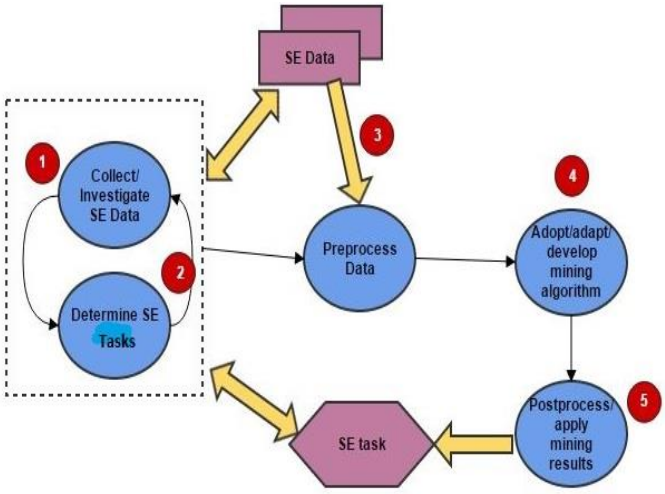
Repository	Description
Source control repositories	These repositories record the development history of a project. They track all the changes to the source code along with meta-data about each change, e.g., the name of the developer who performed the change, the time the change was performed and a short message describing the change. Source control repositories are the most commonly available and used repository in software projects. CVS, subversion, Perforce, ClearCase are examples of source control repositories which are used in practice.
Bug repositories	These repositories track the resolution history of bug reports or feature requests that are reported by users and developers of large software projects. Bugzilla and Jira are examples of bug repositories.
Archived communications	These repositories track discussions about various aspects of a software project throughout its lifetime. Mailing lists, emails, IRC chats, and instant messages are examples of archived communications about a project.
Deployment logs	These repositories record information about the execution of a single deployment of a software application or different deployments of the same applications. For example, the deployment logs may record the error messages reported by an application at various deployment sites. The availability of deployment logs continues to increase at a rapid rate due to their use for remote issue resolution and due to recent legal acts. For instance,the Sarbanes-Oxley Act of 2002 stipulates that the execution of telecommunication and financial applications must be logged.
Code repositories	These repositories archive the source code for a large number of projects. Sourceforge.net and Google code are examples of large code repositories.

Examples of Software Repositories

IV. MINING METHODOLOGY



The above figure shows the general methodology of Mining Software Repositories. Various forms of Software repository data is available in the Software projects as shown in the Layer1 and the appropriate Data Mining Techniques as in the Layer2 are applied to the data and the obtained data from mining is used for Software Tasks. But the existing model has got certain drawbacks for which developers face problem. It is difficult to set up experiments that can be extended or replicated as a result of which they invariably face hurdles which the researcher must overcome in order to perform experiments with huge amounts of data. So here is another better approach with five steps in mining the Software Engineering data. Software engineers and researchers can begin with either a problem-driven approach by knowing what Software Engineering task to assist or a data-driven approach by knowing what Software Engineering data to be mined, but in practice they commonly adopt a mixture of the first two steps: collecting/investigating data to mine and determining the SE task to assist.



The three remaining steps are, in order, preprocessing data, adopting/adapting/developing a mining algorithm, and post processing/applying mining results. The above figure shows the diagrammatical representation of the flow of the Mining Methodology.

For applying the data mining algorithm to the available software engineering data we need to first preprocess the data in order to extract the relevant useful data from the raw SE data. For example call sequences or call graphs from code sources.

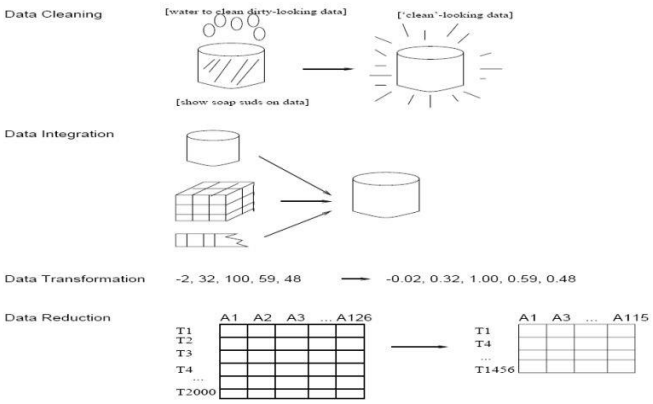
The below table shows the some of the sample Software

engineering data sources and the relevant Data mining algorithms applied to those software data sources and the example software Tasks that benefit from these data and application of algorithms.

Examples of Software Data Mining Algorithms and SE tasks		
SE Data	**Mining Algorithms**	**SE Tasks**
Sequences: execution/static traces, co-changes	Frequent item set/sequence/partial-order mining, sequence matching/clustering/classification	Programming, maintenance, bug detection, debugging
Graphs: dynamic/static call graphs, program dependence graphs	Frequent subgraph mining, graph matching/clustering/classification	Bug detection, debugging
Text: bug reports, e-mails, code comments, documentation	Text matching/clustering/classification	Maintenance, bug detection, debugging

V. DATA PREPROCESSING

Most of the data present in the software artifacts is dirty not clear or some data might be missing or contains errors. Usually incomplete data will lack the attributes and might have only the aggregate data, noisy data will have errors and the outliers, inconsistent data contains the discrepancies in the code or the names. So this data is irrelevant for applying the data mining algorithms to SE tasks



Various Forms of Data Preprocessing

For example, around 2,852 Open Source Java Projects have been identified as a source for SourcererDB and used as Rich dataset of source code to facilitate the sharing of extracted data. Eclipse project repositories are used in Data Preparation and analyzing large software Repositories. Sourceforge.net datasets are used in finding highly relevant software projects from a large archive of executable applications Exemplar (Executable examples Archives) for finding highly relevant software projects & information retrieval and program analysis in developing Exemplar, CodeFinder and Codebroker. Eclipse, BIRT and datatoolsare used in J-REX on MapReduce tool which helps in MapReduce to support Research in MSR.

Data Cleaning: Acquisition refers to the data in a flat file should be in a fixed column format or delimited format: tab, comma or any other symbol. For example data supported in Weka "arff" and c4.5 uses comma- delimited data. *Data Reformatting* refers to converting the data to a standard format (Example. Csv or arff), handling the missing values, Binning of the numbers, Fixing the errors and outliers, converting the nominal fields whose value have order to numeric. Sourceforge and googlecode is used as a source for amassing and indexing a large sample of version control systems.

Missing Data techniques are also widely used Software engineering repositories may have missing data. There are three approaches to this problem. (i) Missing data ignoring techniques. (ii) Missing data toleration techniques (iii) missing data imputation techniques.

The missing data ignoring techniques simply delete the cases that contain missing data Because of their simplicity, they are widely used. This approach has two forms:

List wise deletion also referred as case wise deletion omits the cases containing missing values.

Pair wise deletion also referred as available case method considers each feature separately. For each feature, all recorded values in each observation are considered and missing data are ignored. This means that different calculations will utilize different cases and will have different sample sizes, an undesirable effect.

Missing data toleration techniques use a probabilistic approach to handle missing data. They do not predict missing data but assign a probability to each of the possible values. Thus they are internal missing data treatment strategies, which perform analysis directly using the data set with missing values.

The missing data imputation techniques estimate missing values for the missing cases and insert estimates obtained from other reported values to produce an estimated complete case. The common forms are as follows:

Mean imputation method imputes each missing value with the mean of reported values. The disadvantage is that it leads to underestimation of the population variance.

Regression imputation model is built using the complete observations. It tends to perform better than MI, but still underestimates variance.

Hot-deck imputation method Fill in missing data by taking values from other observations in the same data set. The choice of which value to take depends on the observation containing the missing value. Randomly choosing observed values from donor cases is the simplest hot-deck method.

Unified Date Formatting is another data cleaning form in which we need to transform all dates to the same format internally and some systems accept dates in many formats.

Converting nominal to numeric can deal with nominal values internally and ordered attributes can be converted to numbers preserving natural order. For Ex A-> 4.0, A- -> 3.7, B+ -> 3.3, B -> 3.0. Identify outliers and smooth out noisy data (random error or variance in a measured variable). In *Clustering* we

detect and remove the outliers. Combined computer and human inspection detect suspicious values and check by human. In *Regression* we smooth by fitting the data into regression functions. And we correct inconsistent data by making consistent by removing unnecessary data.

Data Integration technique combines data from multiple sources into a coherent store by detecting and resolving data value conflicts. For the same real world entity, attribute values from different sources are different. Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality.

Data Transformation technique includes Smoothing which removes the noisy data, Aggregation does the summarization and data cube construction, Generalization concentrates on concept hierarchy climbing and finally Normalization transforms al the variables in the data to a specific range.

Data Reduction is another Data Preprocessing techniques obtains a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results. Data reduction strategies include *Data cube aggregation* where in the data is aggregated into an individual entity of interest, *Dimensionality reduction* is important for modeling and DM algorithms work better if dimensionality(number of attributes) is reduced. This can lead to more understandable model. *Numerosity reduction* reduces the data volume by choosing alternative, smaller forms of data representation.

VI. MINING CHALLENGES

Requirements unique to SE: Most Software Engineering, data mining studies rely on well-known, publicly available mining tools. Software companies and researchers compromise on the requirements that are unique to the Software Engineering by fitting them to the tools with undesirable features. Software researchers may lack the expertise to adapt or develop new mining algorithms or tools, while data mining researchers may lack the background to understand mining requirements in the Software Engineering domain. One best and promising way to reduce the gap is to provide close collaborations between the Software Engineering communities i.e. the requirement providers and the data mining community or the solution providers.

Complex data and patterns**:** Software researchers generally mine only individual data types to perform a certain Software Engineering task. However, Software Engineering tasks increasingly demand the mining of multiple correlated data types, including both sequence and text data, together to achieve the most effective result. Even for a single data type, rich information is commonly associated not only with an individual data item but also with the linkage among multiple data items. In addition, pattern representation in the SE domain can be complex. There might be no existing mining algorithms that produce desired pattern representations, and developing new algorithms for such representations can be difficult. Overall, ensuring a scalable yet expressive mining solution is difficult.

Large-scale data**:** Software Engineering researchers often mine only a few local repositories. However, there may be too few relevant data points in these repositories to support the mining of desirable patterns. One way to address this problem is to mine Internet-scale software repositories—for example, via a code search engine. Mining can then be applied to the entire open source world or too many software repositories within an organization or across organizations. Further, execution traces collected from even an average-sized program can be very long, and dynamically or statically extracted call graphs can be enormous. Analyzing such large-scale data poses a challenge to existing mining algorithms.

Just-in-time mining: SE researchers usually conduct offline mining of data already collected and stored. However, in modern integrated SE environments, especially collaborative environments, software engineers must be able to collect and mine SE data on the fly to provide rapid just-in-time feedback. Stream data mining algorithms and tools could be adapted or developed to satisfy such challenging mining requirements.

VII. SE TASKS THAT BENEFIT FROM DATA MINING

Programming: Benefit the novice Developers in extracting and understanding the source code developed by previous developers.

Static defect detection: Helps in detecting the bugs based on the previous defects. Generally previous bugs are good predictors of future bugs.

Testing: Helps in assigning bugs and knowing the expertise of the developers.

Management Tasks: Assists the managers of large projects to prevent the introduction of faults, ensure quick discovery and immediate repair while the software evolve gracefully to handle new requirements of the customers.

VIII. MINING ALGORITHMS CLASSIFICATION

Based on the mining requirements derived by collecting the SE data and determining the SE task, In general, mining algorithms fall into four main categories:

Estimation and Prediction: Estimation consists of examining attributes of a set of entities (products, processes, and resources) and, based on these attribute values, assigning values to an unknown attribute that one wants to quantify.

Classification: consists of examining the attributes of a given entity and, based on these attribute values, assigning it to a predefined category or class. Knn Classification, Decision tree Algorithm, Naïve Bayes Classifier are examples for Classification algorithms. Firefox and chrome repositories are used and classification tool like GNU and R tool are used in the application of Mining Usage Data how software systems are actually used in practice. Eclipse and NetBeans bug repositories are used for the verification of Bug Fixes, Software product and process quality by using the Rodrigorgs2 classification tool. DnsJava (implementation of a domain name system) dataset is used for calculating the Number (amount), density (amount per time unit)of co-change, change in containing clones & Tool measures which methods are cloned and which methods are changed in each transaction. Fedora, Ubuntu, Suse, RedHat, and Firefox repositories are also used in automatically mine repositories and link information across repositories & Bugzilla

bug tracker and Launchpad bug tracker.

Association: Association discovery consists of identifying which attributes are associated with each other in a given environment. It is capable of discovering interesting attributes of large databases. JUNIT and TOMCAT repositories are mined using Association techniques to find out how developers work together. Android repositories are mined in Mapping bug reports to the changes for fixing the bugs. These fixes identified buggy code lines Association (bug introducing changes) using the Git and Diff Association tool. Apache, Python, Postgres SQL, and MySQL repository data is used in mining Patch submission and acceptance into the codebase Built in tools. KDE, Apache, jEdit, and GCC data is used in mining Software co-change prediction using association (Frequent pattern mining) using Association Sqminer tool.

Clustering: Clustering is the task of segmenting a heterogeneous population into a set of more homogeneous subgroups. It differs from classification, as it does not rely on predefined classes. It splits a population into classes on the basis of self-similarity between the class members. Clustering techniques to software engineering data relate to the discovery and localization of program failures. The focus on comparing procedures for filtering and selecting data, each of which involves a choice of sampling strategy and a clustering metric. Failure proximity metric which pairs failing execution traces and regards them as similar if they suggest roughly the same fault location.These traces are created when a crash is detected and are sent back to developers of the software. Their approach improves on previous methods that group traces which exhibit similar behaviors, although the same fault may be triggered by different sets of conditions.K Means Clustering, Em Algorithm are most frequently used algorithms for Clustering.

Text Mining: We may not have the required data in a very specific format (e.g., numerical data, database entries etc.,) text mining seeks to discover high quality unknown information from textual data. It is widely used in Testing. Common types of text mining algorithms include text clustering, classification, and matching. Example text clustering applications include clustering bug reports to detect duplicate bug reports and thereby reduce inspection efforts, and assigning reports to specific developers to fix the bugs. Example text classification applications include recommending assignment of a new bug report to a specific developer based on the past assignment of old bug reports. Example text matching applications include searching keywords in code comments, API documentation, or bug reports, and detecting duplicates of a given bug report among old reports.

IX. SOFTWARE INTELLIGENECE

Software Intelligence (SI) is defined as a new approach to offer software practitioners up to date and pertinent information to support their daily decision-making processes. This is considered as the future of mining software engineering data, within modern software engineering research, practice, and education. Recent advances in the Mining Software Repositories (MSR) field show great promise and provide strong support for realizing SI in the near future.

We now focus on the areas where we need special attention of researchers in Mining Software Engineering Repositories.

SI throughout the lifecycle of a Project: Current State in this scenario is that 80% of the published papers focus on source code and bug-related repositories. Reasons could be that the used bug repositories or source control repositories are commonly available, well structured, facilitating automated data analysis and processing. Documentation repositories (e.g., requirements) are rarely studied, likely due to their limited availability. Strong emphasis in assisting software engineering tasks in the coding phase of a project's lifecycle, benefiting primarily developers. Future Directions: MSR work should look beyond the coding phase as this phase represents a small portion of the lifecycle of a project. Managers, testers, deployers, and support teams are all stakeholders of a software system and they all need SI support from the software engineering community.

SI Using Non-Historical Repositories: Current State of MSR started with strong focus on Historical data(Source Code & Bug Repositories).MSR is about mining any type of software engineering data (e.g., execution logs, code snippets scattered throughout the Internet, and API documents), even when these data are not stored in an explicit "repository".

Future Directions: Storing all kind of data beyond traditional data, Developer interaction data with tools within IDEs, developer meeting notes (even voice recording and recognition with advances in natural/spoken language processing), recordings of support calls, and online posting about software products, Large Volume and Privacy Concerns, Special attention in research required collecting the data, MSR work should make proactive suggestions and influences on improving the repository or IDE design to ease the collection of data.

SI use of Effective Mining Techniques: Current State of MSR work heavily exploited basic off-the shelf data mining (DM) algorithms(such as association rule mining and frequent item set mining) or tools (such as Weka).Commonly compromised their mining requirements to over fit what these basic off-the-shelf algorithms or tools could provide.

Future Directions: Follow a problem-driven methodology in advancing the field, empirically investigate problems in the software engineering domain, identify mining requirements for addressing those problems, adopt or adapt advanced mining algorithms from the DM community, or develop new mining algorithms for satisfying the mining requirements. Finally SI and DM fields should work together.

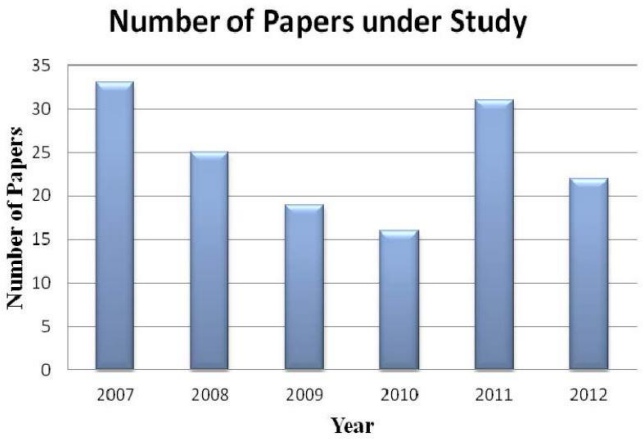
SI Adoption in Practice: Current State: Some of the companies integrate ideas and innovations in their products based on mining software data. (Eg. Coverty and Pattern Insight). The barrier and cost for experimentation with SI innovations are considerably low compared to other software engineering innovations and techniques (e.g., extreme programming or agile development). If your company has a repository, you can mine it with minimal effort.

Future Directions: To enable the widespread adoption of SI, we must first consider the level at which SI support is being provided. The lower and more focused the challenges or questions that an SI technique provides, the more likely it will

be adopted. Make sure that SI techniques are intuitive and SI results are easy to explain & describe.

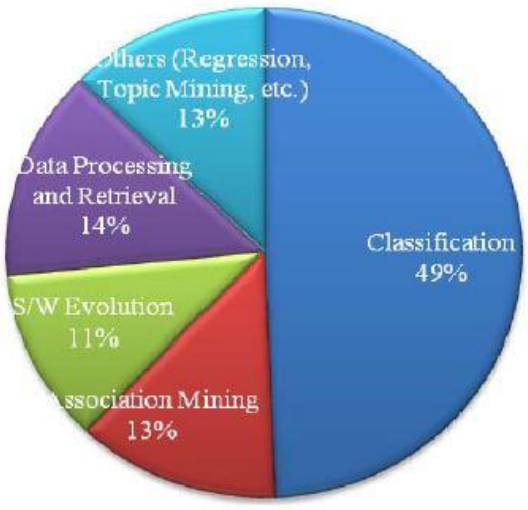
In the MSR field, several start-up companies (founded by academia based on MSR research), such as Coverity, Pattern Insight, and Tasktop, already demonstrated high promises in providing substantial SI capabilities to industrial practices.

Tools in Mining



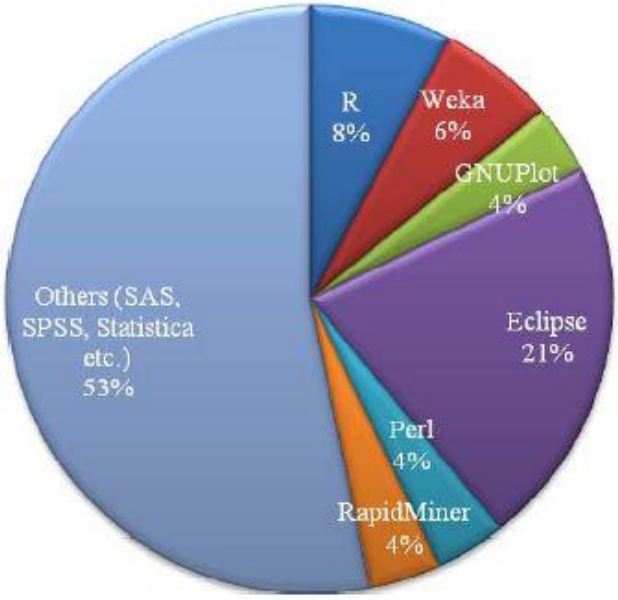
Trends of Number of papers considered under study

The trends in number of papers considered under the study of data mining has been increasing tremendously as shown in the above graph. This says the increasing importance of Mining Software Repositories. It is found that major tasks that are mentioned in the Mining Software papers are association, classification, data processing & data retrieval, software evolution and others as shown in figure 2. We have also found that data retrieval, pre-processing and post processing of the data are most important and mentioned in most of the papers.

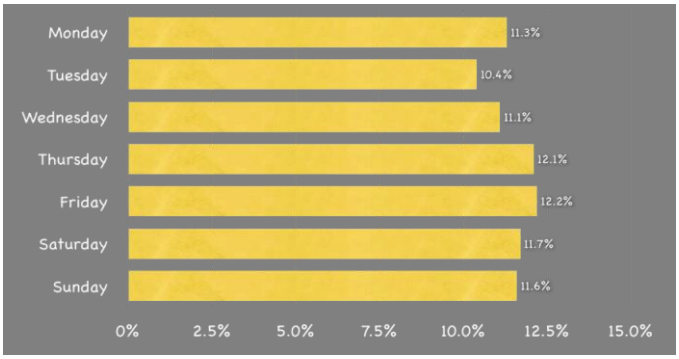


Mining Tasks in Mining Software Repositories

Most of the tools used in Mining are Eclipse, Weka, R and others such as Statistics etc. The use of tools depend on the data mining task being performed on the extracted data and availability of techniques used to accomplish the required task available in the data mining tools. The major contribution of tools is in others category. The others category includes the tools from open as well as commercial software. The commercial tools used for mining software repositories are mainly SAS, SPSS, Statistics, Matlab and many other proprietary tools. The techniques mainly used are Naïve Bayes, Support Vector Machine, Frequent item set mining etc.



Most commonly used applications in MSR are modeling and predicting changes, detection of defect prone modules, finding bugs and their lifetimes, dependency and co-change prediction, developer expertise, predicting type of code changes, severity, assignment, effort estimation, automated task of mining repositories and link information across these repositories etc.



Percentage of Bug –introducing changes for Eclipse.

It is inferred from the above figure that Eclipse is a major tool and mining the repositories of Eclipse has shown that coding on Friday's has resulted in more number of bugs. So this shows that mining is a major 0

X. CONCLUSION

In this paper we have discussed the importance of mining software engineering repositories, various preprocessing techniques, Algorithms widely used for Data mining in Software Repositories, industry tools and techniques used for mining, challenges faced in mining Software Repositories, tasks that benefit from MSR, future approaches for better mining and the way to finding new techniques and brief survey of the tasks and tools in software repositories.

XI. REFERENCES

[1] K.K. Chaturvedi, V.B. Singh and Prashast Singh, "Tools in Mining Software Repositories", 2013 13th International Conference on Computational Science and Its Applications.

[2] Marco D'Ambros and Romain Robbes, "Effective Mining of Software Repositories", 2011 27th IEEE International Conference on Software Maintenance (ICSM).

[3] S. Alam, "Association Rule Mining for Re-planning Decisions of Software Product Releases", IDoESE, 2013

[4] Ahmed E. Hassan and Tao Xie, "Software Intelligence: The Future of Mining Software Engineering Data"2010

[5] Wahidah Husain, Pey Ven Low, Lee Koon Ng, Zhen Li Ong, "Application of Data Mining Techniques for Improving Software Engineering", ICIT 2011 The 5th International Conference on Information Technology

[6] Quinn Taylor and Christophe Giraud-Carrier, "Applications of data mining in software engineering", Int. J. Data Analysis Techniques and Strategies, Vol. 2, No. 3, 2010

[7] Ms Anupama Das, Ms.Kaberi Das, Prof (Dr) B.Puthal, "Improving Software Development Process through Data Mining Techniques Embedding Alitheia Core Tool", Anupama Das et al, / (IJCSIT)

[8] International Journal of Computer Science and Information Technologies, Vol. 2 (2) , 2011, 629-632