Investigating Role of Data Mining in Software Engineering

Himangi

Assistant Professor,
Department of Computer Science and Engineering,
Guru Jambheswar University of Science and technology,
Hisar, Haryana, India
Email: himangiverma2021@gmail.com

Article History

Article Submission
19 June 2021
Revised Submission
14 September 2021
Article Accepted
13 October 2021
Article Published
31 December 2021

Abstract

Companies that focus on software development produce vast volumes of data. Every stage of software development, from gathering requirements to ongoing upkeep, generates its own set of data. To better the software, efforts are undertaken to collect and store data produced in software repositories. Data mining techniques are used to the massive amounts of data found in software repositories in order to extract previously unseen patterns and insights. Researchers from the fields of Software Engineering and Data Mining have lately made this area of study a top priority. This research aims to examine the many uses of data mining in software engineering, the many types of software engineering data that can be mined, and the many data mining techniques that are available and have been used by researchers to solve the problems that this research focuses on. The next step is to use this classification to determine which subfield within software engineering has the highest scholarly interest.
Keywords: *Software Engineering, Data Mining, Hidden Patterns, Software Design.*

I. Introduction

Over the last several decades, the field of software engineering has captured the attention of many academics. Although software engineering's significance has diminished over time, the demand on software engineers to provide high-quality products has increased in today's cutthroat market. Scholars and practitioners alike are interested in many different facets of software engineering, including but not limited to software quality, metrics, and configuration management. For the last several years, more and more researchers and professionals have been focusing on multidisciplinary topics, with data mining and software engineering being the most notable example. It might be particularly useful in the search for novel and significant information hidden inside a mountain of gathered data. It's a way to find connections between seemingly unrelated datasets stored in different places, such as operational, distributed, external, and hypermedia databases. Using different Data Mining techniques, researchers may unearth previously unknown connections between previously collected data. This data extraction is useful to businesses since it provides more information with which to make decisions. The software development process generates a large amount of data that, in addition to standard databases, may be mined using data mining techniques to improve the quality of the programme being produced. SE There is great value in the information collected over the course of software development and stored in central locations known as repositories.

1.1 Software Engineering

Using a set of engineering principles and best ractice , software engineers create, test, and release software applications that address issues in the real world. Software engineering is the ractice of using scientific methods to software development in order to increase the reliability of the final product, the speed with which it can be completed, the cost-effectiveness of the project, and the quality of the testing done on it .It is unclear who originally used the phrase “software engineering,” however in 1968 NATO hosted a symposium on the topic. The conference discussed the need for improved QA and dependability in software development, as well as the problems associated with inconsistent and unreliable development. Experts from all around the globe met for this meeting, and they all agreed that the systematic approach of physical world engineering should be applied to software development.

ISSN: 2250-0839

Challenges of software engineers

Software engineering, like any other discipline, has its share of difficulties. Some software engineering difficulties are unique to a person's skill set and level of maturity, while others are shared by all software engineers.

A few typical difficulties in software engineering are as follows.

- **Rapidly changing technology.** Software developers are under constant scrutiny to set them apart from the competition and develop innovative computer software solutions in response to the ever-increasing frequency with which new technologies are introduced. They need to keep their skills fresh in order to accommodate new technologies.
- **Time limitations**. A significant challenge for software developers is dealing with time restrictions. Engineers can't produce a product that meets quality standards if they're rushed.
- **Changing requirements**. When requirements change in the middle of a software project, it might provide unexpected difficulties for the developers. If they can't adapt quickly to changing circumstances, the initiative might fail.
- **Data privacy**. There are a number of privacy regulations that affect organisations today, including the Health Insurance Portability and Accountability Act (HIPAA), the CCPA, and GDPR of the European Union. Software development teams' access to data for product development may be constrained by ever-changing regulations.
- **Unclear customer requirements.** Customers don't always know exactly what they want out of a piece of software. As a consequence, software developers may face problems once the programme is put into action since they are unaware of the whole set of functions that users often want from it.
- **Risk of software failure**. The effects of flawed software may be disastrous at times. For instance, in safety-critical sectors like aerospace, aviation, and nuclear power plants, the cost of software failure may be quite high.

1.2 Software design

Software design is the process through which the needs of the end user are translated into a form that can be understood by the programmer. The focus here is on translating the client's stated needs from the SRS paper into a format that can be readily implemented in code.

In the first stage of the SDLC, software design, the focus shifts from the issue domain to the solution domain. In software engineering, the system is seen as a collection of modules, each of which has its own set of predetermined behaviours and limitations.

1.3 Software Design Levels

There are three distinct outcomes from software development:

- **Architectural Design** - The architectural plan is the most simplified and generalized representation of the whole system. It recognizes the programme as a system consisting of interconnected parts. The designers now have a basic understanding of the suggested solution domain.
- **High-level Design**- The high-level design deconstructs the 'single entity-multiple component' idea of architectural design into a less-abstracted perspective of sub-systems and modules, and shows how they interact with one another. The goal of a high-level design is to create a modular system that can be used to build the whole system. Each component's modular structure, as well as its interdependence, are taken into account.
- **Detailed Design-** The focus of the third design stage is on the actual construction of the system and its constituent parts, as envisioned in the preceding two designs. The focus is on modules and how they are implemented in more depth. Each module's interfaces and logical structure are spelt out in this document.

1.4 Data Mining

Data Mining is a technique used by businesses to find useful information in large datasets. Data is transformed from its raw form into information that may be put to good use. Data Mining is analogous to Data Science in that it is performed by a human in a defined context using a defined data set to achieve a defined goal. Text mining, online mining, audio/video mining, graphical data mining, and social media mining are all part of this process. It's done using either generic or specialised pieces of software. Data mining may be completed quickly and at a reduced cost if outsourced. Data that would be very difficult to track down without the aid of modern technology is now within the reach of specialised businesses. While there is a wealth of data available across several mediums, only a fraction of that data is really usable. The most difficult part is sifting through all that data to find insights that can be utilised to improve processes or expand the business. There are several potent tools and methods for data mining accessible today.

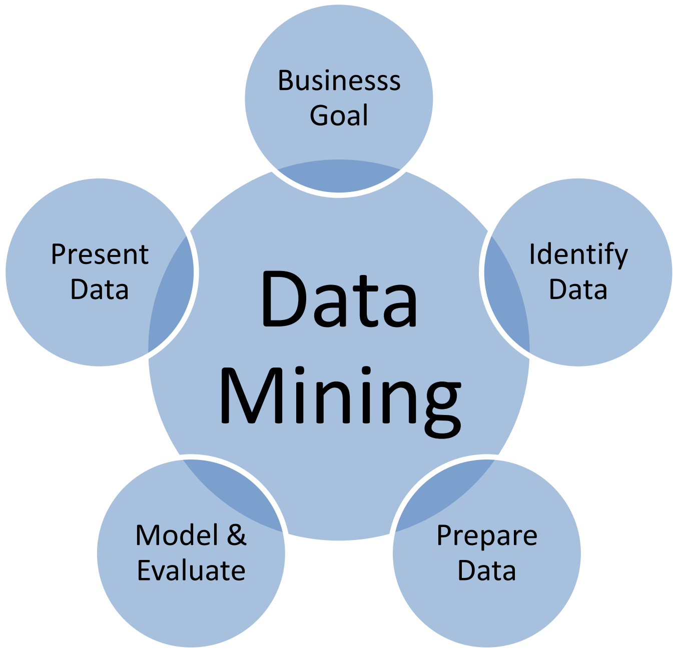

Figure 1: Data Mining

1.5 Hidden Patterns

The first stage in finding hidden patterns or trends is cleaning the raw data and evaluating it with the aid of a data mining application. At this stage, standard statistical procedures should be used to verify any findings that look significant. The primary reason we collect data is to seek for patterns and relationships not immediately apparent. Depending on how the data and patterns are presented, this pattern may be seen even in a simple tabular representation of the data. Finding a particular subsequence inside a text (rather than a string regarded as a sequence of successive symbols) is the "hidden pattern" problem, also known as the "sequence comparison" problem. The term "DATA" is used to describe any and all quantifiable information. When data is clustered, it becomes more similar to points in the same cluster than to points in other clusters, a cube-shaped data array having several dimensions.

II. Literature Review

Software Engineering, and Software Design have all been the subject of research in the past, and that work is summarized here. We'll talk about these things first:

German et al. [1] proposed a tool called SoftChange that retrieves and summarises textual data from open source software and then validates that textual data.Jensen et al. [2], who proposed a technique for detecting software processes, also focused on open source software. Text extraction made advantage of entity resolution and social network analysis.

Williams and Hollingsworth [3] came up with a novel approach to improving static analysis techniques for finding bugs by mining the repository's change history. Using methods from the field of association rule extraction. A software project generates a large amount of organised and unstructured data during the course of its life cycle. All of these different types of data may be extracted using a variety of data mining techniques. According to M. Halkidi et al. [4], the most important data sources for software engineering are documentation, software configuration management data, source code, generated code and execution traces, issue tracking and bug databases, and mailing lists. According to A.E. Hassan et al. [5], there are three distinct kinds of SE data repositories: historical repositories, run time repositories, and code repositories. Data repositories like Git, GitHub, and Junit, in addition to CVS logs, include a wealth of useful information that may be mined. [6] In their studies, Chaturvedi and his team addressed several data sources.Huffman et al. [7] provided a technique for retrieving information. Their strategy involved mapping the problem of requirements tracing into locating the similarity between the vector space representations of high and low level requirements, thereby reducing it to an IR task, and then using that to improve the extraction of high and low level requirements. Ankit Dhamija and Sunil Sikka conducted complementary research in [8] to better understand what variables affect software developers' use of data mining tools.

Morisaki et al. [9] assessed a set of defect data. Software flaws may take many forms, including bugs, incorrect specifications, and outdated designs. Using an extended association rule mining strategy based on defect data, crucial information and rules linked with defect rectification attempts were extracted.

Finding conditional rules in a code base and extracting rule breaches that draw attention to forgotten conditions may be done with the help of the graph mining techniques described by Chang et al [10].Conditional rules were discovered using dependency graphs together with frequent item set mining and frequent sub graph mining techniques.

To choose appropriate executions for conformance testing, Dickinson et al. [11] suggested a clustering technique-based strategy for narrowing down profiles with shared characteristics.Execution profiles are then derived from the remaining clusters.Last et al.[12] suggest automating the processing of input and output data using an infofuzzy network (IFN) with a tree-like topology.

Classifiers of software behaviour are built by analysing a dataset of programme executions using the technique proposed by Bowring et al. [13]. The execution patterns of the projects were encoded using a Markov model. After that, the Markov models from individual programme iterations are clustered using an agglomerative clustering method.

To aid in manual debugging, Liu et al. [14] created a mining strategy that used a statistical approach.

A data-mining-based method for assessing logical mistakes was proposed by Liu et al. [15]. They devised this method by classifying programme executions using closed graph mining and support vector machines.

In order to comprehend an object-oriented system and evaluate its maintainability, Kannelopoulos et al. [16] suggested a clustering-based technique for obtaining information from source code.

Citation	**Author/year**	**Objectives**	**Methodology**	**Limitation**
[1]	D. German/2003	Open-source projects	Automation on data mining	Less accurate
[2]	C. Jensen/2004	Software process discovery	Data mining	Narrow focus
[3]	Williams/2005	Automation on data mining	Data mining	Less performance
[4]	D. Spinellis/2011	Data mining in software engineering	Data mining	Lacks accuracy
[5]	R. C. Holt/2004	Predicting change propagation in software systems	Predicting, software systems	Less performance
[6]	Singh V.B/2013	Tools in Mining Software Repositories	Data Mining	Less focus on efficiency
[7]	Huffman/2003	Improving requirements tracing via information retrieval	Information retrieval	Focus only on statistical tests
[8]	Sundaram/2005	Analyst feedback based results	Text mining	Less efficiency

ISSN: 2250-0839

[9]	Morisaki/2007	Defect data analysis	Data analysis	Less focus on efficiency
[10]	Chang /2008	Tracing neglected conditions in software	Data mining	Less real life application

Citation	**Software Engineering**	**Data Mining**	**Hidden Patterns**	**Software Design**
[1]	Yes	No	No	No
[2]	No	Yes	No	Yes
[3]	Yes	No	No	No
[4]	Yes	Yes	No	No
[5]	No	No	No	Yes
[6]	No	Yes	No	No
[7]	Yes	No	No	No
[8]	No	Yes	No	Yes
[9]	No	Yes	No	Yes
[10]	Yes	Yes	No	No
[11]	Yes	No	No	No
[12]	No	Yes	No	No

III. Problem Statement

However, the authors here believe there is an immediate need to put all the work done by earlier researchers into a consistent format that covers all aspects, from the different SE repositories and the data that is available from SE repositories to the different data mining techniques available, the different tools that are available, and the tools that are best to use on different kinds of mining techniques and on what kind of SE data such tools can be applied. This review article explores the connections between software engineering and data mining. The data miner's goals include learning which data types may be mined, which tools are best for doing so, and how these processes can be applied to various software engineering datasets. Finally, the author uses this classification system to try to identify the most important area in SE.

IV. Need of Research

Researchers in this study categorized the different types of tools and offered a summary analysis of different Data Mining Tools and their focus area in Software Engineering data sources to provide a three-way analysis of heterogeneous Software Engineering Data. Data mining and its techniques can prove to be quite helpful in uncovering important patterns that may in turn help developers, testers, maintenance teams, and other people associated with Software Engineering in spotting issues, as shown by the extensive literature review and tools analysis.

V. Scope of Research

When it comes to software engineering data mining operations, scientists are more likely to make use of current tools than to develop their own. Researchers used these tools to mine data repositories for insights, identify patterns, acquire new knowledge, and make predictions.

References

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

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

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

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

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

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

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

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

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

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

[11] W. Dickinson, D. Leon and A. Podgurski, Finding failures by cluster analysis of execution profiles, International Conference on Software Engineering (ICSE), 2001.

[12] M. Last,M. Friedman and A. Kandel, The Data Dimining Approach to Automated Software Testing, In Proceeding of the SIGKDD Conference, 2005.

[13] J. Bowring, J. Rehg and M.J. Harrold, Acive learning for automatic classification of software behavior, International Symposium on Software Testing and Analysis (ISSTA), 2004.

[14] C. Liu, X Yan, and J. Han. Mining control ow abnormality for logical errors. In Proceedings of SIAM Data Mining Conference (SDM), 2006.

[15] C. Liu, X. Yan, H. Yu, J. Han and P. Yu, Mining behavior graphs for 'backtrace' of noncrasinh bugs. In SIAM Data Mining Conference (SDM), 2005.

[16] Y. Kannelopoulos, Y. Dimopoulos, C. Tjortjis and C. Makris, Mining source code elements for comprehending object oriented systems and evaluating their maintainability, SIGKDD Explorations 8(1), 2006.

[17] D. Engler, D. Chen, S. Hallem et al., "Bugs as deviant behavior: A general approach to inferring errors in systems code," ACM SIGOPS Operating Systems Review, vol. 35, no. 5, pp. 57-72, 2001.

[18] Z. Li, and Y. Zhou, "PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code," in Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, 2005, pp. 306-315.

[19] W. Qu, Y. Jia, and M. Jiang, "Pattern mining of cloned codes in software systems," Information Sciences, 2010.

[20] Talukdar, Veera, Dharmesh Dhabliya, Bhupendra Kumar, Suryansh Bhaskar Talukdar, Shahanawaj Ahamad, and Ankur Gupta. "Suspicious Activity Detection and Classification in IoT Environment Using Machine Learning Approach." In *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 531-535. IEEE, 2022.

[21] Veeraiah, Vivek, P. Gangavathi, Shahanawai Ahamad, Suryansh Bhaskar Talukdar, Ankur Gupta, and Veera Talukdar. "Enhancement of meta verse capabilities by IoT integration." In *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1493-1498. IEEE, 2022.

[22] Gupta, Neeraj, S. Janani, R. Dilip, Ravi Hosur, Abhay Chaturvedi, and Ankur Gupta. "Wearable Sensors for Evaluation Over Smart Home Using Sequential Minimization Optimization-based Random Forest." *International Journal of Communication Networks and Information Security* 14, no. 2 (2022): 179-188.

[23] Keserwani, Hitesh, Himanshu Rastogi, Ardhariksa Zukhruf Kurniullah, Sushil Kumar Janardan, Ramakrishnan Raman, Vinod Motiram Rathod, and Ankur Gupta. "Security Enhancement by Identifying Attacks Using Machine Learning for 5G Network." *International Journal of Communication Networks and Information Security* 14, no. 2 (2022): 124-141.

[24] H. A. Basit, and S. Jarzabek, "A data mining approach for detecting higherlevel clones in software," IEEE Transactions on Software Engineering, pp. 497-514, 2009.

[25] Z. Li, S. Lu, S. Myagmar et al., "CP-Miner: A tool for finding copy-paste and related bugs in operating system code," in Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, 2004, pp. 20.

ISSN: 2250-0839

[26] S. Lu, S. Park, C. Hu et al., "MUVI: automatically inferring multi-variable access correlations and detecting related semantic and concurrency bugs," ACM SIGOPS Operating Systems Review, vol. 41, no. 6, pp. 103-116, 2007.
[27] B. Baker, "On finding duplication and near-duplication in large software systems," in Second IEEE Working Conf on Reverse Eng.(wcre), 1995, pp. 86- 95.
[28] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic tokenbased code clone detection system for large scale source code," IEEE Transactions on Software Engineering, pp. 654-670, 2002.

ISSN: 2250-0839