Data Mining Tools and Techniques for Mining Software Repositories: A Systematic Review

Tamanna Siddiqui and Ausaf Ahmad

Abstract A software repository contains a historical and valuable wealth of information about overall development of software system (project's status, progress, and evolution). Mining software repositories (MSR) are one of the interesting and fastest growing fields within software engineering. It focuses on extracting and analyzing the heterogeneous data available in software repositories to uncover interesting, useful, and actionable information about software system and projects. Using well-established data mining tools and techniques, professionals, practitioners, and researchers can explore the potential of this valuable data in order to better understand and manage their complicated projects and also to produce high reliable software system delivered on time and within estimated budget. This paper is an effort to discover problems encountered during development of software projects and the role of mining software repositories to resolve these problems. A comparative study of data mining tools and techniques for mining software repositories has been presented.

Keywords Mining software repositories (MSR) · Heterogeneous data · Software mining

1 Introduction

Nowadays, software plays a very important role in the life of human and according to day-to-day needs and requirements it is going to enhance and evolve. Software evolution has been introduced as one of the most vital theme in software engineering and maintenance. It generally deals with spacious amounts of data that engenders from different sources such as source code repositories, bug tracking

T. Siddiqui (✉) · A. Ahmad
Department of Computer Science, Aligarh Muslim University, Aligarh, India
e-mail: ja_zu_siddiqui@hotmail.com

A. Ahmad
e-mail: ausafahmad.cs@gmail.com

© Springer Nature Singapore Pte Ltd. 2018

systems, version control system, issue tracking system, mailing and project discussion lists. One of the key points of software development and evolution is to design theories and models that make it possible for understanding the past and present, as well as help in predicting future characteristics related to software maintenance, and therefore support software maintenance tasks [1]. In this concern mining software repositories (MSR) contribute important role. Mining software repositories is a new emerging research field that attempts to gain a deeper understanding of the development process in order to build better prediction and recommendation systems [2]. Historical and valuable information stored in software repositories provide a great opportunity to acquire knowledge and help in monitoring complex projects and products without interfering with development activities and deadlines. Source code control system store source code and changes in it as development progress, bug tracking system keeps records of reported software defects in software development projects, issues tracking system manages and maintains lists of issues, and archive communication between project personnel record rationale for decision throughout the entire life of a project. Such wealth of information helps researchers and software project personnel to understand and manage the development of complex projects within estimated budget and time deadline. For example, historical information can assist developers in understanding the rationale for the current structure of the software system [3].

2 Software Repositories

The software repositories are a storage location maintained online or offline by several software development organizations where software packages, source code, bugs, and many other information related to software and their development process are maintained. Due to open source, the number of these repositories and its uses is increasing at a rapid rate. Anyone can extract many types of data from here, study them, and can make changes according to their need. The examples of software repositories are the following

2.1 Historical Repositories

Such repositories contain a heterogeneous and huge amount of software engineering data generated between long periods of time; some of them are following:

Source control repositories (SCR). SCR record the development history of a project. These repositories keep all the changes made to the source code together with metadata about each change during maintenance, e.g., developer name who made the change, the time at which change was made and a short message describing the intention of change, and the changes performed. The most commonly available, accessible, and used repositories for software projects are source control repositories. Some widely used source control repositories are Perforce, ClearCase, CVS (Concurrent Versions System), subversion, etc.

Bug repositories (BR). BR keep the track of bugs/faults encountered in different phase of software development life cycle maintained by the developers of different organizations. E.g., Jira and Bugzilla are bug repositories maintained by Atlassian and Mozilla community respectively. The benefits of maintaining these repositories are to improve communication, increase product quality, ensure accountability, and increase productivity.

Archived communications (AC). These repositories record the discussion carried out between developers and project managers about various features of a project throughout its entire life cycle. Some examples of archive communication are instance messages, emails, mailing lists, and IRC chats.

2.2 Run-Time Repositories

These repositories contain information related to the execution and the usage of an application at a single or multiple different deployment site such as deployment logs.

Deployment Logs (DL). Software deployment is a complete set of activities that make a software product available for use [4]. These activities can occur at the producer or consumer or both sides. The information related to execution of a particular deployment of a software project or heterogeneous deployments of the same projects are recorded in this type of repositories. For example, the error message expressed by an application at different single or multiple deployment sites may be recorded in deployment logs. 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 [5].

2.3 Code Repositories

CR repositories are maintained by the collection of source code of a large number of heterogeneous projects. Sourceforge.net, GitHub, and Google code are example of CR.

3 Related Work

Mining software repositories is a new emerging field and focus on extracting of both elementary and valuable information regarding software attributes from heterogeneous extant software repositories [6]. These types of repositories are mined to extract the hidden facts by different contributors with regards to accomplishing the different targets. The use of data mining is gaining continuous popularity in software engineering environments due to satisfactory results since last decade [7, 8] and its application include area as bugs prediction [9], Co-evolution of production and test code [10], impact analysis [11], effort prediction [12, 13], similarity analysis [14], prediction of software architectural change [15], software intelligence [5], also used to reduce complexities in Global Software Development [16] and many more.

According to Hassan and Xie [5], "Software Intelligence (SI) is a software developed to analyze source code to clearly understand the Information Technology scenarios. It offers a set of software tools and methods for extracting the valuable information. It keeps software experts up to date and relevant extracted information used to enhance the decision making." Before decade, mining software repositories (MSR) investigations were essentially subjected to industrial level. Consequently, research efforts were reasonably limited to a select few software systems and application domains, or restricted due to lack of historical software data that were not openly available like open source. Recently, there has been a rapid trained shift with respect to the above-noted situation; all these are because of vogue and development of open source software. Consequently, availability of the open source paradigm has been successful in producing excellent and high-quality products that continue to use and evolve.

As we reviewed many research papers, we observed that comparison of techniques used by different researchers is very hard because different researchers used different frameworks, techniques, and data sets. Some use their own private datasets that not accessible to all. Some papers are discussed in Table 1.

4 Challenges in Software Development

Some challenges faced by project managers and programmers are following.

4.1 Faults Encountered

At present time, all software companies essentially focus on handover high-quality products to their customers because of quality leads to customer satisfaction. To achieve the end user satisfaction, these companies targeting to produce high reliable and quality products which must be free from bugs. Therefore, for every software manager, delivering bags free software products to the customers has become the first priority. For example, United States Department of Defense (DOD) losses nearly four billion dollars per year basis because of software failures [17].

A study carried on 104 software projects developed by many developers in different parts and location indicate that end user reports 0.15 faults per 1 KLOC

Table 1 Techniques used in software mining by the various authors

S. No.	Task	Technique used	Data sets	Authors	Refs.
			Private/public		
1	Fault prediction	Statistical prediction rule	Private	Xuemei Zhanga, Hoang Phamb	[24]
2	Fault prediction	K-means clustering method	Private	Shi Zhong, Khoshgoftaar T.M., Seliya N.	[25]
3	Fault prediction	J48 & linear regression	Public	Menzine T., Di Stefano J., Chapman R.M	[26]
4	Fault prediction	CRB & data clustering	Private	Taghi M. Khoshgoftaa, Naeem Seliya	[27]
5	Fault prediction	AntMiner + classification	N.A.	Olivier V., David M., Bart B., Mues C., Manu B.	[9]
6	Detecting half-done change	Apriori association rule	Public	Qinbao S., Martin S., Michelle C., Carolyn M.	[28]
7	Detecting half-done change	FP-growth association rule	Public	Ying A.T., Murphy G.C., Chu-Carroll M.C.	[29]
8	Effort prediction	Linear regression rule	Public	Rahul P., Martin S., Barbara K., Pekka F.	[30]
9	Effort prediction	Decision tree (M.M.)	Public	Martin S., Michelle C.	[31]

(thousand lines of code) during the initial year after delivering of projects to customers; which means for a 100 KLOC project, 15 faults may occur that cannot be ignored [18, 19]. High reliable software to meet user requirements can be achieved by introducing the well-organized development process and structured testing process. In many cases, the results of testing process at development site may differ from user site where the project actually run which also leads to faults.

4.2 Half-Done Changes

A requirement also leads to changes in already developed projects, mostly in source code. Typically, a software project is developed and maintained by a group of developers. Especially in the case of open source software (OSS) these developers are not from the same geographical location. For example, a case study conducted on open source Apache Server, state that the core of this project, including new functionalities is developed by approximately 15 developers [19].

However, for changes in large project is done by a group of programmers. More programmers cooperate in carrying out changes to the source code also create a problem. As programmer that make changes in one module probably do not predict its effect on other modules that changed together, points to half-done change or incomplete change. Due to source code written in different programming languages, it is probably difficult to discover coherence between parts [20] which is also a cause of the half-done change.

5 Software Mining: A Solution

Software mining involves the concept of data mining to build a model or extract information that support and improve the continuous development process of software projects. In this, software artifacts are used as input data and output gives a systematic pattern that helps in prediction of further changes and help practitioners in addressing the challenges discussed above section.

5.1 A Solution of Fault Prediction

The end user always expects that the software system always fulfil the requirement and being free of bugs. While we know that software testing in an exhaustive way is infeasible. Testing phase of development process takes a lot of time and effort. In this concern software mining (SM) provides a solution by modelling software quality predictions and classification. For software quality prediction, software metrics and regression techniques are used to predict the various types of bugs in modules and in quality classification, modules are classified as a fault-prone and non-fault-prone by using classification mining techniques [19].

Therefore, based on the results of SM algorithm, managers allocate budget to the module that contains more bugs and programmers gives more concentration on it rather that module that contain no or some errors. Definitely, this is a better and efficient approach for budget allocation than traditional approach to allocate the budget to whole projects.

5.2 A Solution of Half-Done Change

Half-done change is also a major cause of bugs/faults. Pattern extracted from change histories repositories helps in fault prevention caused by half-done change in source code. Association mining techniques are applied on these repositories that gives warning like "Developers who change the function A, along with the suggested change in related function B, C [21] etc." This task is done by using mining techniques as a plugin in the programing framework by programmers that suggest the related change in source code at once. Source code of some software projects are nearly written in different languages, especially in open source software, to make the changes in such types of source code could be very challenging for a normal programmer, such types of changes can made easily and successfully by software mining.

A notable point is that classification techniques are used for fault prediction and concern of management while association techniques used for half-done change and a concern of programmers.

6 Comparative Study of MSR Tools

A number of mining tools have been developed for the extraction of information or pattern from the different heterogeneous datasets stored in various type software repositories discuss in Sect. 2. Some tools specially used for software mining are Softchange, Hipikat, Dynamine, Kenyon, Chianti, and Apfel. The dimensions of comparison of these tools are intent, information, infrastructure, effectiveness, interaction, input data, and language dependency (Table 2).

Parameter	Description
Intent	It describes expected user like managers, programmers, testers, maintainers, researchers, etc. [22]
Information	The specific data that tool extract for analysis as change management, source code, defect tracing, informal communication, etc. [22]
Infrastructure	A requirement needed to run the tool such as operating system, IDE, store backend, etc. [22]
Effectiveness	Describe the feasibility of proposed such as status, cost, evaluation [23]
Interaction	Describes interactivity and the life tools [23]
Input data	Refers to the input data require by a particular tool
Language independency	Describes the nature, whether the tool is depend on programing language or not

Table 2 A comparative study of MSR tool based on their characteristics and support

List of tools ⟶		DME	SCG	CHI	HKT	AFT	KYN
↓ List of parameters							
User	Manager	…	YES	…	…	…	…
	Programmer	YES	YES	YES	YES	YES	…
	Tester	YES	…	YES	…	…	…
	Maintainer	YES	YES	YES	YES	YES	…
Time	Past	YES	YES	YES	YES	YES	YES
	Present	YES	…	YES	…	…	…
	Future	…	…	…	…	…	YES
Information source	CVS	YES	YES	YES	YES	YES	YES
	Issue tracking	…	YES	…	YES	…	YES
	S/W release	…	YES	…	…	…	…
Change management		YES	…	YES	…	YES	YES
Defect tracking		…	YES	YES	YES	…	YES
Archive mailing lists		…	…	…	YES	…	…
Infrastructure requirement		YES	YES	YES	YES	YES	…
Online/offline		Both	Offline	Offline	Offline	Offline	Both
Storage required		…	YES	YES	…	YES	YES
Input data		SMC	MCB	EP	CVSV	EP	PMVC
Language dependency		…	YES	YES	YES	YES	…

Notations
SCG Softchange; *HKT* Hipikat; *DME* Dynamine; *KYN* Kenyon; *CNI* Chianti; *AFL* Apfel; *EP* Eclipse; *MCB* Metadata from CVS & Bugzilla; *SMC* Set of methods & calls, programs; *CVSV* CVS Data version; *PMVC* Program metadata from various kind of system

7 Conclusion

The present era is the era of software, everything is going computerizing. This is not possible without software. Development of high-quality software on time, within the estimated cost and effort is very challenging task for the software organizations. In this concern, mining software repositories play an important role. In this paper, we have discussed some problem faced by the project personnel and researcher and their expect solution in terms of software mining. The technique used to resolve the challenges is mentioned in this paper and a comparative study of software mining tools has done. This paper helps the project personnel and researchers to understand the basic working of mining software repositories (MSR).

References

1. Novais, R.L., Torres, A., Mendes, T.S., Mendonca, M., Zazworka, N.: Software evolution visualization: a systematic mapping study. Inf. Softw. Technol. **55**, 1860–1883 (2013)
2. Connor, A.M.: Mining Software Metrics from the Jazz Repository. ARPN J. Syst. Soft. **1**(5), ISSN 2222-9833 (2011)
3. Hassan, A.E., Holt, R.C.: Using development history sticky notes to understand software architecture. In: Proceeding of the 21st International Workshop on Program Comprehensive, Italy, June (2004)
4. Software deployment, https://en.wikipedia.org/wiki/Software_deployment
5. Hassan, A.E., Xie, T.: Software intelligence: the future of mining software engineering data. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ACM, pp. 161–166 (2010)
6. Keivanloo, I.: A linked data platform for mining software repositories. In: 9th IEEE Working Conference on Mining Software Repositories (2012)
7. Halkidi, M.: Data mining in software engineering. Intell. Data Anal. **15**(3), 413–441 (2011)
8. Xie, T., Pei, J., Hassan, A.E.: Mining software engineering data. In: 29th International Conference on Software Engineering Companion, ICSE (2007)
9. Vandecruys, O.: Mining software repositories for comprehensible software fault prediction models. J. Syst. Softw. **81**(5), 823–839 (2008)
10. Zaidman, A., Van Rompaey, B., Demeyer, S., van Deursen, A.: Mining software repositories to study co-evolution of production & test code. In: 1st International Conference on Software Testing, Verification, and Validation, pp. 220–229 (2008)
11. Canfora, G., Cerulo, L.: Impact analysis by mining software and change request repositories. In: 11th IEEE International, Symposium on Software Metrics (2005)
12. Moser, R.: A model to identify refactoring effort during maintenance by mining source code repositories. In: Product Focused Software Process Improvement, pp. 360–370. Springer, Berlin (2008)
13. Weiss, C.: How long will it take to fix this bug? In: Fourth International Workshop on Mining Software Repositories, ICSE Workshops MSR ’07 (2007)
14. Sager, T.: Detecting similar Java classes using tree algorithms. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, ACM, pp. 65–71 (2006)
15. Ratzinger, J.: Mining software evolution to predict refactoring. In: First International Symposium on Empirical Software Engineering and Measurement (2007)

16. Kandjani, H., Tavana, M., Bernus, P., Wen, L., Mohtarami, A.: Using extended axiomatic design theory to reduce complexities in global software development projects. Comput. Ind. **67**, 86–96 (2015)
17. Dick, S., Meeks, A., Last, M., Bunke, H., Kandel, A.: Data mining in software metrics databases. Fuzzy Sets and Systems, pp. 81–110. (2003)
18. Cusumano, M., MacCormack, A., Kemerer, C., Crandall, W.: Software development worldwide: the state of practice. IEEE Comput. Soc. **20**(6), 28–34 (2004)
19. Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., Haesen, R.: Mining software repositories for comprehensible software fault prediction models. J. Syst. Softw. **81**, 823–839 (2008)
20. Ying, A., Murphy, G., Ng, R., Chu-Carroll, M.: Predicting source code changes by mining change history. IEEE Trans. Softw. Eng. **30**(9), 574–586 (2004)
21. Zimmerman, T., Weissgerber, P., Diehl, S., Zeller, A.: Mining version histories to guide software changes. In: Proceedings of International Conference on Software Engineering (ICSE), pp. 429–445 (2005)
22. German, M.D., Cubranic, D., Storey D.: A Framework for describing and understanding mining tools in software development. ACM, MSR'05 (2005)
23. Sunday, O.O., Syed, U.I., Yasser S.A., Jarallah S.A.: A mining software repositories—a comparative analysis. Int. J. Comput. Sci. Netw. Secur. **10**(8), 161–174 (2010)
24. Zhanga, X., Phamb, H.: Software field failure rate prediction before software deployment. J. Syst. Softw. **79**, 291–300 (2006)
25. Zhong, S., Khoshgoftaar, T.M., Seliya, N.: Analyzing software measurement data with clustering techniques. Intell. Syst. IEEE (2004)
26. Menzine, T., Stefano, D.J., Chapman, R.M.: Mining repositories to assist in project planning and resource allocation. In: 26th International Conference on Software Engineering, Workshop. doi:10.1049/ic:20040480 (2004)
27. Khoshgoftaar, T.M., Seliya, N.: Analogy-based practical classification rules for software quality estimation. Empirical Softw. Eng. **8**, 325–350 (2003)
28. Song, Q., Shepperd, M., Cartwright, M., Mair, C.: Software defect association mining and defect correction effort prediction. IEEE Trans. Softw. Eng. **32**(2) (2006)
29. Ying, A.T., Murphy, G.C., Ng, R., Chu-Carroll, M.C.: Predicting source code changes by mining change history. IEEE Trans. Softw. Eng. (2004)
30. Rahul, P., Martin, S., Barbara, K., Pekka, F.: An empirical analysis of software productivity over time. In: 11th IEEE Int. Softw. Metr. Symp. (2005)
31. Shepperd, M., Cartwright, M.: Predicting with sparse data. IEEE Trans. Softw. Eng. **27** (2001)
32. What is data preparation, http://www.datapreparator.com/what_is_data_preparation.html
33. Mockus, A., Fielding, R., Herbsleb, J.: A case study of open source software development: The apache server. In: Proceedings of the ICSE Conference, pp. 263–272 (2000)