ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

Volume-8, Issue-5, October 2018

International Journal of Engineering and Management Research

Page Number: 20-23

DOI: doi.org/10.31033/ijemr.8.5.4

Application of Data Mining Techniques for Improving Continuous Integration

Meenakshi Kathayat
Assistant Professor, Department of Computer Science & Engineering, Birla Institute of Applied Sciences Bhimtal, Uttarakhand, INDIA

Corresponding Author: meenakshik.kathayat4@gmail.com

ABSTRACT

Continuous integration is a software development process where members of a team frequently integrate the work done by them. Generally each person integrates at least daily - leading to multiple integrations per day. Integration done by each developer is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach reduces integration problems and allows a team to develop cohesive software rapidly. Continuous Integration doesn't remove bugs, but it does make them dramatically easier to find and remove. This paper provides an overview of various issues regarding Continuous Integration and how various data mining techniques can be applied in continuous integration data for extracting useful knowledge and solving continuous integration problems.

***Keywords*--** Continuous integration, Agile development, Software development, Data mining techniques

I. INTRODUCTION

The traditional software development life cycle follows a Waterfall methodology that eventually morphed into the Agile SCRUM lifecycle. But for most organizations the current lifecycle resembles something like an Agile-SCRUM-Fall.

Continuous integration (CI) is a phase in agile development of software which allows developers to frequently submit their piece of work in the central repository (where the code resides and shared among developers). Each developer pulls code from central repository to their local machine and changes the code according to the requirement. After modifying the code, developer integrates that code to the central repository. Then each integration is verified by an automated build, so that teams can detect problems early. There is a CI server present which performs this automated build task; it is the duty of CI server to inform the respective developer if the build is successful or failed. By integrating regularly, you can detect errors quickly, and locate them more easily [1]. Figure.1 show Continuous integration workflow:

The CI lifecycle looks like this:

1. Check in code.
2. Pull code to be changed for build.
3. Run tests (CI server generates builds and then arrange releases): Test individual models, run integration tests, and run user acceptance tests.
4. Store artifacts and then build repository (It is for artifacts storage, results storage, and releases storage).
5. Deploy and release (release automation product to deploy applications or software).

ISSN (ONLINE): 2250-0758, ISSN (PRINT): 2394-6962

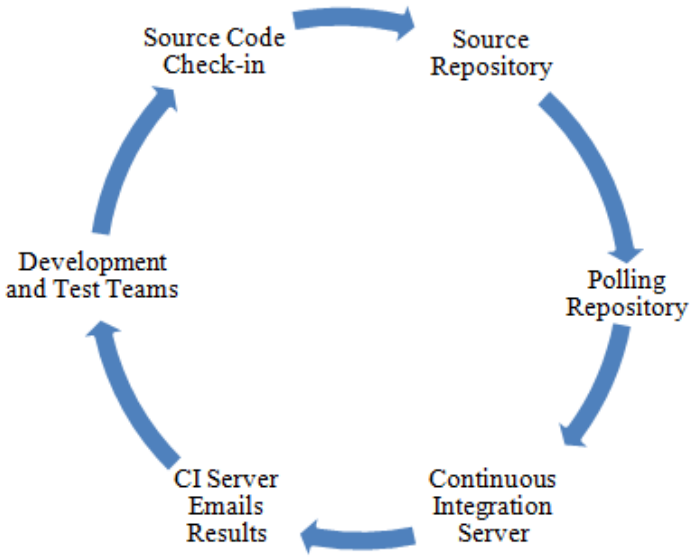

Figure 1. Continuous Integration Workflow

CI has many advantages and some of them are:

- There are no long and tense integrations.
- Visibility increases and so the communication between developers increases.
- It can identify issues or bugs very fast and fix them as soon as possible.
- More time can be given to adding features in software rather than debugging errors.
- No need to wait to find out if a developer's code is going to work or not.
- Reduce integration problems so that the software can be delivered more rapidly.
- Because integration is done so frequently, there is significantly less back-tracking to discover where things went wrong, so more time can be spent building features.

II. CONTINUOUS INTEGRATION DATA

There is a vast amount of data that is generated by continuous integration and is ignored by the organizations. First, the CI server contains data in the form of logs from each build performed by each developer. The logs are about which developer made which build at what time, what is the code coverage after each build, current status of the mainline code (Mainline code is the code present in the central repository), which module is getting changed more frequently, and so on.

Second, monitoring systems such as Zabbix, New Relic, Nagios, Icinga generates alarms for alerting various situations. For example, if a pipeline breaks then mail is sent by the monitoring system to a certain group of developers, if some code breaks due to a developer then mail goes to the team leader and the developer, how many test cases were done this mail goes to the QA (Quality Assurance) team. The monitoring system alerts by sending alarms to developers related to that situation.

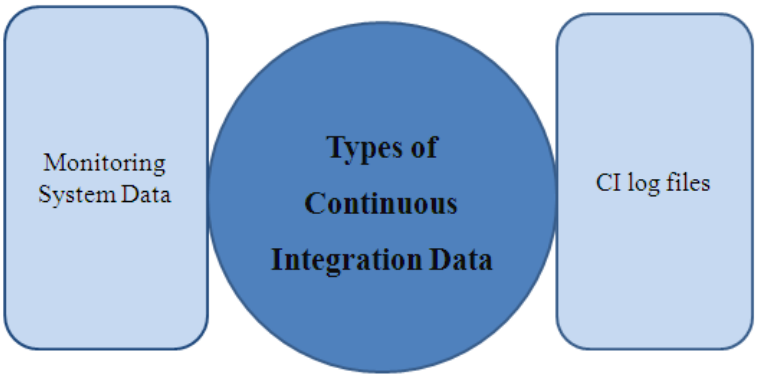

Fig. 2.1 Types of CI data to be mined

All of this data is present in textual form (logs, as well as monitoring system data) and is usually ignored by the developers. This data contains information about all the successful builds, all the failed builds and performance of all the developers. The need is to extract this information from this raw data, so that it can be useful for either the maintenance period of a project or for developing more future software.

III. MINING CONTINUOUS INTEGRATION DATA

One of the main objectives of continuous integration is to reduce the problems of "integration hell", i.e., different engineers working on the same code base at the same time, such that their changes have to be merged together [2]. Various aspects which should be considered for applying mining in CI data are:

- When a software product is composed of many of or even hundreds of components with complicated dependency relationship among each other, one component's change can affect lots of other components' behavior [4]. CI solves this problem by integrating as soon as possible and fixing errors.
- It requires each developer to finish a task, usually within a few hours, and submit the code to the version control repository regularly and early, then CI server launches integration build automatically by detecting the code change, run code static checking and automated testing, try to discover new potential defects introduced by this newly changed code and ensure the other functionalities are not affected, then CI server generates the final result report and sends the notification to corresponding developers and managers with final result automatically. The final result report contains build status, duplicate logic checking, code complexity degree report, compliance with coding standard, unit testing result, function testing result, etc [5].
- Different software metrics can be added with the log in CI server because software metrics are the quality measure of any software and can produce some useful result.
- Monitoring produces so many alarms with high frequency and everyone ignores them but instead of ignoring them we can device it in such a way that it will produce only genuine alarms. This will help everyone to be alert when the alarm produces.

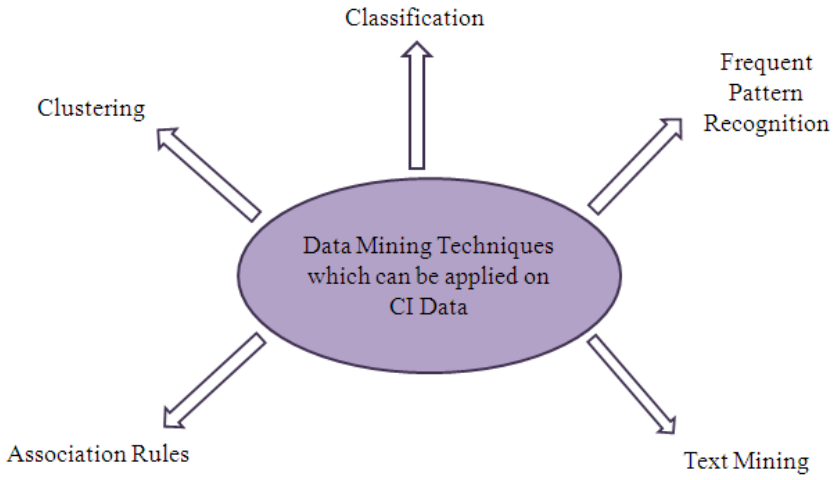

Fig. 3.1 Various methods of data mining for CI data

Following are various data mining techniques given with example of how we can apply them on continuous integration:

- ***Clustering*** is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called clusters. It can be applied on CI logs to group similar logs together with some distance with those logs which are different. It can also be used to cluster alarms which are genuine in one cluster and non-important alarms in another. Then produce only those alarms which are genuine from its cluster. There are various clustering techniques which can be used: KNN, K-means Clustering, etc.
- ***Classification*** is the process of sorting and categorizing data into various types, forms or any other distinct class. Classification can be achieved by two steps: training and testing. In the training

step, it will use machine learning to train a model on properties of logs; it will then create a machine learning model of those logs. In the testing step, the user will supply pre-computed model with properties of new logs and the model will predict results needed. Some of the classification methods are: Naïve Bayesian classifier, decision tree, ZeroR etc.

- ***Frequent patterns recognition and association rules.*** We can find out due to which reasons the failure of a build happens frequently, which developer and what changes are responsible for that build failure.
- ***Text mining*** is an area of data mining with extremely broad applicability. Rather than requiring data in a very specific format (e.g., numerical data, database entries, etc.), text mining seeks to discover previously unknown information from textual data [7].

Text mining can be directly applied to logs because logs are in textual form. Text frequency, classification, clustering are the main areas in text classification. We can mine logs in CI server to find out productivity of a team or a developer, can find out which module is altering with highest frequency and so on.

IV. CONCLUSION

Continuous Integration has become an established best practice of modern software development[6]. It is the practice of testing each change done to your codebase automatically and as early as possible; members of a team integrate their work frequently, usually each person integrates at least daily which leads to multiple integrations per day [3]. Since, CI generates a huge amount of data in the form of logs and alarms which is not very important from the view of organization. In this paper we have discussed that this data can be important for solving various issues related to CI. Then, we have proposed various data mining techniques which can be applied in CI data and how it can help in the maintenance and development of software.

REFERENCES

[1] Cios, K.J., Swiniarski, W.R., Pedrycz, W., & Kurgan, A. L. (2007). *Data mining: A knowledge discovery approach*. New York: Springer.

[2] S. Kotsiantis, C. Pierrakeas, & P. Pintelas. (2004). Prediction of student's performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, *18*(5), 411-426.

[3] Tiwari, Mahendra, Randhir Singh, & Neeraj Vimal. (2013 Feb). An empirical study of application of data mining techniques for predicting student performance in higher education. *International Journal of Computer Sciences and mobile Computing*, *2*(2), 53-57.

[4] Tsai, C.F., Tsai, C.T., Hung, C.S., & Hwang, P.S. (2011). Data mining techniques for identifying students at risk of failing a computer proficiency test requires for graduation. *Australian Journal of Educational Technology*, *27*(3), 481-498.

[5] Mardikyan, Sona & Badur, Bertan. (2011). Analysing teaching performance of instructors using data mining techniques. *Informatics in Education*, *10*(2), 245-257.

[6] Berry, J.A. Michael, & Linoff S. Gordon. (2004). *Data mining techniques for marketing, sales, and customer relationship management*. (2nd ed.). Wiley Publishing.

[7] Quinn Taylor & Christophe Giraud-Carrier. (2010). Applications of data mining in software engineering. *International Journal of Data Analysis Techniques and Strategies*, *2*(3), 243-257.