**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**II SEMESTER 2020-21**

## DSE CL ZG628T DISSERTATION

**Dissertation Outline**

1. Student details:

Name : Abhilash A

Student ID : 2018AB04638

WILP email id : 2018ab04638@wilp.bits-pilani.ac.in

Mobile : 9986900517

1. Date

03 April 2021

1. Dissertation Title

Commit Bug Probability

1. Supervisor details:

Name : Gaurav Ashok Batra

Role in organization : Specialist Programmer

Qualification : BE in IT

Experience : 16 years

email id : gaurav-batra03@infosys.com

Mobile : 8879490380

1. Problem statement (what is the problem being addressed)

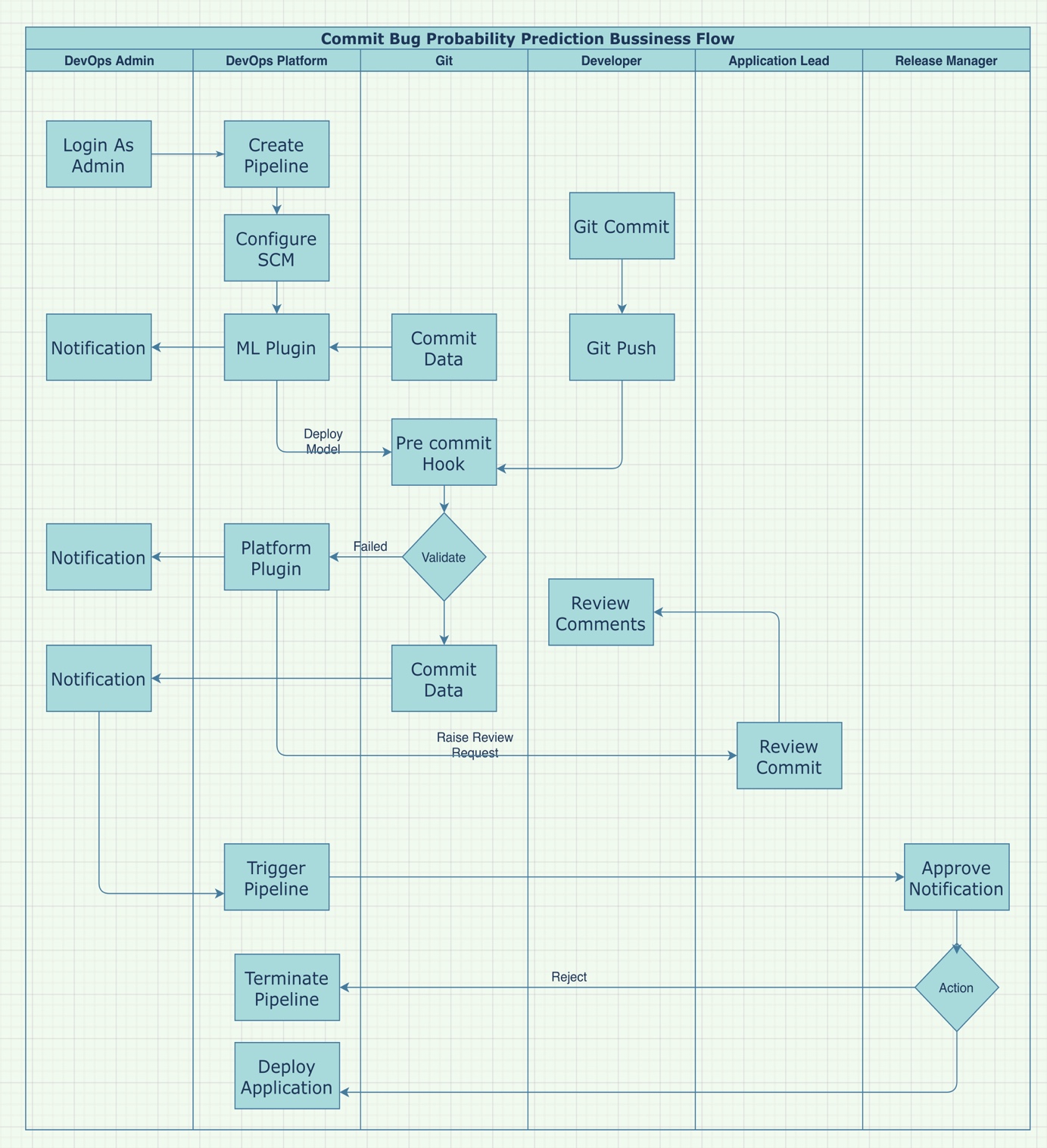
With the growing complexities of the software, the number of potential bugs is also increasing rapidly. These bugs hinder the rapid software development cycle. Bugs, if left unresolved, might cause problems in the long run. Software bug prediction at the initial stages of software development improves the important aspects such as software quality, reliability, and efficiency and minimizes the development cost. In majority of software projects which are becoming increasingly large and complex programs,

bugs are serious challenge for system consistency and efficiency.

Unfortunately, bug trackers tend to record when bugs were fixed but not when they were committed. Finding and fixing software bugs is difficult, and many developers put significant effort into finding and fixing them. A project’s software change history records the change that introduces a bug and the change that subsequently fixes it.

This bug-introducing and bug-fix commits can be used to predict the probability of bugs in future commits.

1. Business process flow, if any



1. Objective of the project (Expected outcome)

* Improve Mean Time to Failure up to 20%
* Reduce no of Bugs detected by QA up to 30%
* Improve software quality, reliability, efficiency and reduce the software cost

1. Uniqueness of the project

Most of the existing approaches try to predict bugs at code level[file /class/function] level. This makes it difficult to arrive at optimized set of metrics. These approaches require the software repository to be analyzed as a whole for producing meaningful results. This makes the integration of these approaches with production systems difficult in the world of CICD/DevOps and Agile. In this project we are try to predict the bug probability at a commit level which makes it easy to extract relevant set of metrics and can be plugged into existing CICD pipelines easily. Most of the bug prediction models use regression for prediction to arrive at the no of bugs , we are proposing a classification model to arrive at the probability of bug at commit level

1. Benefit to the organization

* Improve software quality, reliability, efficiency and reduce the software cost
* Improvement in Mean Time to Failure
* Effort reduction in QA

1. Scope of work

Create a micro frame work which can extract training data from a given GitHub repository , train a ML model for predicting bug probability and deploy the model as a pre commit hook to the given repository. This micro frame work can be easily integrated with the existing DevOps platform as plugins.

1. Resources needed for the project, including people, hardware, software, etc.

* Software requirements are included in the architecture diagram
* 8 core 32 GB Linux server
* GPU requirement depends in the complexity of selected model and the amount of data available in git repository

1. Potential challenges & risks in doing the project

* Bug fix commits needed to be extracted from git repository for constructing training dataset. This depends on the process followed by developers. Will be able to extract data only if bugfix commits are tagged
* Data quality depends on the agile maturity of the team.

1. Background of previous work done in the chosen area

Many bug prediction techniques have been proposed in the literature in the last decade.

Such techniques mainly differ for the speciﬁc predictors they use, and can roughly be classiﬁed in those exploiting product metrics(e.g., lines of code, code complexity, etc), those relying on process metrics (e.g., change- and fault-proneness of code components), and those exploiting a mix of the two.

The ﬁrst set of techniques exploits product metrics (i.e., metrics capturing intrinsic characteristics of the code components, like their size and complexity) [1], [2], [3] while the second one focuses on process metrics (i.e., metrics capturing speciﬁc aspects of the development process, like the frequency of changes performed to code components) [4], [5], [6]. While some studies highlighted the superiority of these latter with respect to the product metric based techniques [5], [6]

The studies in [7], [8] analyzed the applicability of various ML methods for fault prediction. Sharma and Chandra [7] added to their study the most important previous researches about each ML techniques and the current trends in software bug prediction using machine learning. This study can be used as ground or step to prepare for future work in software bug prediction. R. Malhotra in [5] presented a good systematic review for software bug prediction techniques, which using Machine Learning (ML).The paper included a review of all the studies between the period of 1991 and 2013, analyzed the ML techniques for software bug prediction models, and assessed their performance, compared between ML and statistic techniques, compared between different ML techniques and summarized the strength and the weakness of the ML techniques.

[1] N. Ohlsson and H. Alberg, “Predicting fault-prone software modules in telephone switches,” Software Engineering, IEEE Transactions on, vol. 22, no. 12, p. 886894, 1996.

[2] N. Nagappan and T. Ball, “Static analysis tools as early indicators of pre-release defect density,” in Proceedings of the 27th International Conference on Software Engineering, ser. ICSE’05. New York, NY, USA: ACM, 2005, pp. 580–586. [Online].Available: http://doi.acm.org/10.1145/1062455.1062558

[3] T. Zimmermann, R. Premraj, and A. Zeller, “Predicting defects for eclipse,” in Proceedings of the Third International Workshop on Predictor Models in Software Engineering, ser. PROMISE ’07.Washington, DC, USA: IEEE Computer Society, 2007, pp. 9–.[Online]. Available: http://dx.doi.org/10.1109/PROMISE.2007.10

[4] A. N. Taghi M. Khoshgoftaar, Nishith Goel and J. McMullan, “Detection of software modules with high debug code churn ina very large legacy system,” in Software Reliability Engineering. IEEE, 1996, pp. 364–371.

[5] J. S. M. Todd L. Graves, Alan F. Karr and H. P. Siy, “Predicting fault incidence using software change history,” Software Engineering, IEEE Transactions on, vol. 26, no. 7, pp. 653–661, 2000.

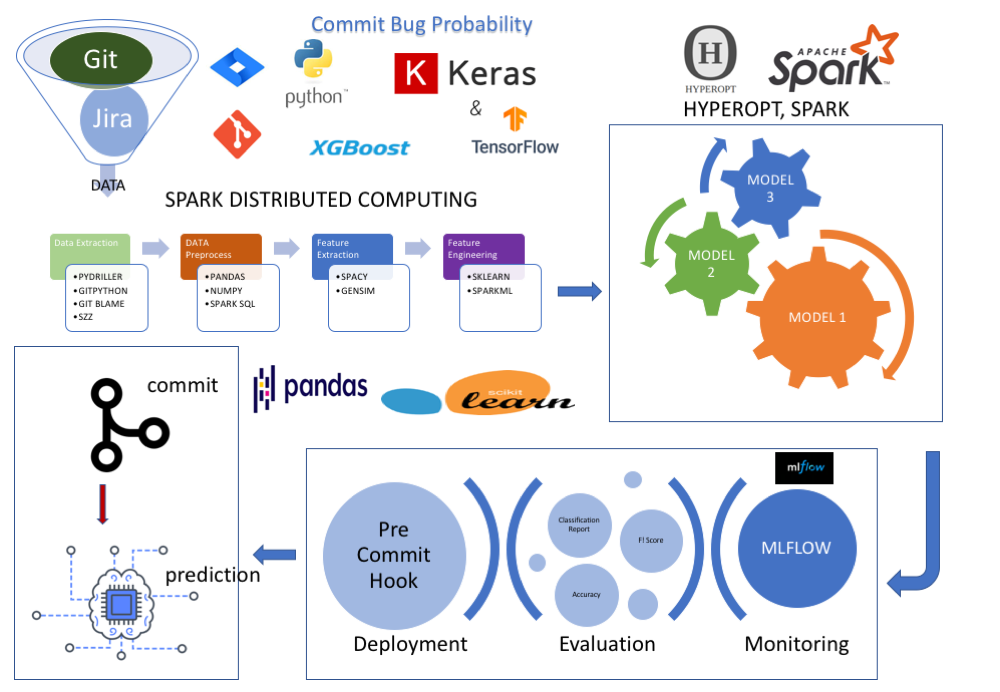
[6] R. Moser, W. Pedrycz, and G. Succi, “Analysis of the reliability of a subset of change metrics for defect prediction,” in Proceedings of the Second ACM-IEEE International

Symposium on Empirical Software Engineering and Measurement, ser. ESEM ’08. New York, NY, USA: ACM, 2008, pp. 309–311. [Online]. Available: http://doi.acm.org/10.1145/1414004.1414063

[7] D. Sharma and P. Chandra, "Software Fault Prediction Using Machine-Learning Techniques," Smart Computing and Informatics. Springer, Singapore, 2018. 541-549.

[8] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297

1. Solution architecture



1. Detailed Plan of Work (as follows)

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Task** | **Expected date of completion** | **Names of Deliverables** |
| 1 | Data Extraction | 10/04/2021 | Commit Data |
| 2 | Data Pre Processing | 24/04/2021 | Processed Data |
| 3 | EDA | 30/04/2021 | Features |
| 4 | Modelling | 21/04/2021 | Model Framework |
| 5 | Model Deployment | 12/06/2021 | Pre Commit Hook, Plugins for model training and deployment |