I. Name and email of a project leader

Group	Jinfu Chen, Kundi Yao
Project leader	Jinfu Chen
Email	bajin886@gmail.com

II. A project title and short problem statement

Project Title
Empirical Study of Just-in-Time prediction of Performance regression introducing
Problem Statement
Software performance plays an essential role in software evolution and maintenance. Performance regression is more and more prevalent in software evolution and maintenance especially in large scale system. Very little code changes can degrade the performance and developers are not aware of the problem. In practice it is difficult to avoid introducing performance regression because there are hundreds of commits between every two releases. What is more each pair of consecutive commits contain a number of code changes. Most researches intend to to find the code changes which degrade performance from a large number of input sets.

III. Research Questions

RQ1
How well can we predict the existence of performance regression introducing change?
RQ2
How well can we predict the magnitude of performance regression introducing change?
RQ3
What are the important factors of performance regression introducing change?

IV. related papers

1 A Large-Scale Empirical Study of Just-in-Time Quality Assurance [1]
This paper presents some change measures and builds a logistic regression model to predict JIT software defect.
2 An Empirical Study of Just-in-Time Defect Prediction using Cross-Project Models [2]
This research constructs Just-in-Time defect prediction cross-project models to identify source code changes that have a high risk of introducing a defect.
3 ITowards Building a Universal Defect Prediction Model [3]
This study intends to build a universal defect prediction model for a large set of projects by combining context factors and clustering similar projects to determine the different software metrics sets.
4 Effort-Aware Just-in-Time Defect Prediction: Simple Unsupervised Models Could Be Better Than Supervised Models [4]
In order to find whether or not unsupervised models perform better than the supervised models on defect prediction, this paper considers fourteen change metrics and builds simple unsupervised models and supervised models to predict software defect.
5 Reducing Features to Improve Code Change Based Bug Prediction [5]
The more features (factors) prediction model learned the more insufficient performance the model predicts so this paper performs multiple feature selection algorithms to reduce the factors to predict software bug in a high performance.
6 Studying just-in-time defect prediction using cross-project models [6]
This paper constructs JIT model based on cross-project models, by using larger pool of training data and combining models from other projects to establish their JIT models, and the result shows performance of within-project model outperforms cross-project model.
7 The Impact of Human Discussions on Just-In-Time Quality Assurance An Empirical Study on OpenStack and Eclipse [7]
In this paper the authors build logistic regression models to study the impact of human discussion metrics on JIT predicting models, result shows a strong correlation between human discussion metrics and defect-prone commits.

8 An empirical study on software defect prediction with a simplified metric set [8]
In this paper, the authors study the feasibility of defect-predictor built with simplified metrics in different scenarios, and offer suggestions on choosing datasets and metrics, the result shows the predictor based on minimum metric subset, specific requirements of accuracy and complexity can provide satisfactory performance.

9 Defect Prediction: Accomplishments and Future Challenges [9]
This paper is an overview in defect prediction field, it intends to introduce and help readers understand previous studies on defect prediction, and highlight some important challenges for future works.

10 On Automatic Detection of Performance Bugs [10]
In this paper, the authors use four machine learning methods to build models to predict performance bugs and the most satisfying model is based on Logistic Regression with all attributes added.

References

[1] Kamei, Yasutaka, et al. ”A large-scale empirical study of just-in-time quality assurance.” IEEE Transactions on Software Engineering 39.6 (2013): 757-773.

[2] Fukushima, Takafumi, et al. ”An empirical study of just-in-time defect prediction using cross-project models.” Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[3] Zhang, Feng, et al. ”Towards building a universal defect prediction model.” Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[4] Yang, Yibiao, et al. ”Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models.” Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016.

[5] Shivaji, Shivkumar, et al. ”Reducing features to improve code change-based bug prediction.” IEEE Transactions on Software Engineering 39.4 (2013): 552-569.

[6] Kamei, Yasutaka, et al. ”Studying just-in-time defect prediction using cross-project models.” Empirical Software Engineering 21.5 (2016): 2072-2106.

[7] Tourani, Parastou, and Bram Adams. ”The impact of human discussions on just-in-time quality assurance: An empirical study on openstack and eclipse.” Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference. Vol. 1. IEEE, 2016.

[8] He, Peng, et al. ”An empirical study on software defect prediction with a simplified metric set.” Information and Software Technology 59 (2015): 170-190.

[9] Kamei, Yasutaka, and Emad Shihab. ”Defect prediction: Accomplishments and future challenges.” Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference. Vol. 5. IEEE, 2016.

[10] Tsakiltsidis, Sokratis, Andriy Miranskyy, and Elie Mazzawi. ”On Automatic Detection of Performance Bugs.” Software Reliability Engineering Workshops (ISSREW), 2016 IEEE International Symposium. IEEE, 2016.