ABSTRACT
Bugs are prevalent in software systems and improving time efficiency in bug fixing is desired. We performed an analysis on 11,115 bug records of Eclipse JDT and found that bug resolution time is log-normally distributed and varies across fixers, technical topics, and bug severity levels. We then propose FixTime, a novel method for bug assignment. The key of FixTime is a topic-based, log-normal regression model for predicting defect resolution time on which FixTime is based to make fixing assignment recommendations. Preliminary results suggest that FixTime has higher prediction accuracy than existing approaches.

1. INTRODUCTION
Bug fixing is an important activity in software development. With the need to develop high quality, competitive software products, maintain high customer satisfaction, and keep the development cost and schedule on track, software teams are always desired to improve the resolution time of software defects. A promising direction for improvement is to optimize the bug assignment process by finding the best fixer, in term of resolution time, for each bug.

CosTriage (Cost-aware Bug Triaging) is recently introduced to address that need [17]. It first classifies the bugs into different topics. Then, for each developer, it computes his/her profile as the average time that s/he has fixed the bugs of each topic in the history. If s/he has never fixed any bug in a topic, his/her profiled resolution time on that topic is derived from that of other developers with similar profiles. Finally, for a newly-reported bug, CosTriage recommends a ranked list of potential fixers based on the bug’s topics and those recovered profiles of developers on bug resolution time.

However, such prediction of bug resolution time might not be accurate. Using the average fixing time of historical bugs to make recommendation for new bugs, CosTriage implicitly assumes that the performance of a developer on future bugs of a topic stays the same. In practice, this assumption is not realistic, i.e. the bugs of the same topic are likely to be resolved differently. For example, more severe bugs (e.g. blocker) are likely to be resolved faster than the less critical ones due to their urgent nature. Other factors such as reporter reputation, fixer reassignments (bug tossing), or bug dependencies also affect bug resolution time [5, 8].

Several prediction models proposed in the literature for bug resolution time have used those features as predictors [5, 7]. Unfortunately, those models predict the resolution time of bugs without considering their assigned fixers and the topics of the bugs, as well as the characteristics of bug resolution time.

To study bug resolution time, we performed an analysis on 11,115 bug records of Eclipse JDT and found that bug resolution time is log-normally distributed and varies across fixers, technical topics, and bug severity levels. Based on these findings, we propose FixTime, a time-aware approach for bug assignment problem. The core of FixTime is a topic-based, log-normal regression model to predict the resolution time of a given bug if it is assigned to a given developer. FixTime uses that model to predict for all available developers and ranks them based on the predicted resolution time to recommend fixing assignment.

By those ideas, FixTime could take advantage of both worlds: taking into account fixers and bug topics (as in CosTriage [17]) as well as using a suitable regression model (as in the existing resolution time prediction models [5, 7]). Moreover, with a log-normal regression model, FixTime can address the nature of bug resolution time.

Section 2 reports several analysis results on the characteristics of bug resolution time that motivate our approach. Section 3 will present FixTime in detail and the results of a preliminary experiment comparing FixTime with CosTriage method. Related work will be discussed in Section 4 and conclusions appear last.

2. DATA ANALYSIS
This section presents our analysis on the bug resolution time of Eclipse JDT, an open-source project supporting Java application development with the functionality such as compiling, programming assistant (e.g. refactoring, code search and auto-completion), source code and resource management, etc.

2.1 Data collection
JDT bugs are managed in the Bugzilla issue tracking system of Eclipse project, which is available online. For each bug, we consider the last developer assigned to it as its fixer. We chose the last assigned developer because a bug might be reassigned (i.e. re-assigned to another developer) several times. The resolution time of that developer on that bug is computed as the elapsed time (in days) from the last assignment to the last time the bug is marked as FIXED. We chose the last ‘FIXED’ time because a bug might be re-opened when the earlier fix is incorrect. We ignored the bug records that are marked as invalid, duplicate, unconfirmed (i.e. WORKFORME), and unfixed (i.e. WONTFIX). The total number of collected bugs is 11,115.

2.2 Distribution of Bug Resolution Time
We first analyzed the distribution of the resolution time of the collected bugs. The histogram of the resolution time (not shown due to the space limit) suggests its distribution to be left-skewed, i.e. non-normal. For such a skewed distribution, the median is more representative than the mean. The median of resolution time is of 4 days, i.e. 50% of the bugs are resolved within 4 days.
Investigating various distributions for bug resolution time, we have learned from system engineering research that the time to repair a maintainable system could be modeled with a log-normal distribution [13]. Thus, we fit the resolution time of collected bugs to a log-normal distribution. To measure the goodness of fit, we computed the Kolmogorov-Smirnov metric which is within the commonly used critical value of 5%. Therefore, we conclude that log-normal distribution can be used to represent the bug resolution time. Figure 1 shows the density of the log-transformed resolution time which resembles the bell shape of the normal distribution.

### 2.3 Variations of Bug Resolution Time

#### 2.3.1 Topic recovery

To verify the use of fixers, topics, and bug severity for predicting on bug resolution time, we performed another analysis. We applied LDA on the summaries of the collected bugs with four topics (we choose such a small number of topics for this exploratory study to simplify the interpretation of the recovered topics). Table 1 shows the top words assigned for four recovered topics. The words suggest that, Topic 1 is about the functionality of user interface, via terms such as view, editor, or explorer. Topic 2 is about debugging support, e.g. code generation, execution, and testing, via terms such as junit, build, or launch. Topic 3 is about compiling, expressed in the terms such as compiler, type, or method. Topic 4 is about code assistant, e.g. code search, code completion, quick fix, with terms such as assist, code, manipulation, etc. Consulting JDT documentation, we were able to confirm the correctness of the topics with regard to the key technical functionality of JDT.

#### 2.3.2 Variation on fixers and topics

We computed the median resolution time of bugs across fixers and topics. We observed that the bug resolution time varies across the fixers and bug topics. For example, as seen in Table 2, for the top fixer in JDT, anonymized as J1, the bug resolution time on four topics is not the same. Bugs related to user interface are resolved the fastest (median of 3.2 days), while the bugs related to code assistant topic are the slowest (median of 8 days). However, also on code assistant bugs, the next most active developer, anonymized as J2, has much better resolution time (median of just 1.2 days). Several other developers would have resolution time faster than J1 but slower than J2, since on average for all developers, code assistant bugs are resolved within 5 days. This observation is consistent with the findings from our prior work on a commercial software project [14] where different developers also have different average resolution times on different bug topics.

#### 2.3.3 Variation on severity levels

Bugs in JDT are classified in several levels of severity. We also see that the bug resolution time varies across bug severity levels. For example, as seen in Table 3, high severe bugs (e.g. blocker, critical) are resolved faster than the less severe ones (e.g. normal, minor). This is true for all bugs in overall, and for bugs of a particular topic (code assistant) resolved by particular developer (J1). Interestingly, bugs classified as trivial, although less severe than normal or minor bugs, are resolved faster. This might be because they are rather simple to fix.

### 2.4 Implications

The analysis results suggest three important implications in designing prediction models of bug resolution time. First, predicting bug resolution time via averaging on fixers and topics is not sufficiently accurate due to the influences of other factors such as bug severity. Second, since bug resolution time is log-normally distributed, a log-normal regression model will be more suitable for prediction than an ordinary linear regression model (which has been used in [5]). Third, because bug resolution time varies across fixers, topics, and bug severity levels, those three factors should be used as inputs of a prediction model. In the next section, we will propose our prediction approach which takes them into account.

### 3. APPROACH

Let us present FixTime, a method to estimate bug resolution time and suggest efficient assignment. The core of FixTime is a topic-based log-normal regression model which could recover the topics of a newly reported bug and predict the resolution time of that bug for any candidate fixer. FixTime uses such prediction to suggest bug fixing assignments for developers.
3.1 Prediction Model

Figure 2 illustrates the prediction model of FixTime. Based on the analysis results in Section 2, this model assumes the resolution time of a bug is a random variable following a log-normal distribution $\text{LN}(\mu, \sigma)$ whose parameters $\mu$ and $\sigma$ are dependent on the assigned fixer $d$, the bug topics $\theta$, and the bug severity $e$. Since the assigned fixer and the bug severity are given as input, FixTime needs to recover only the bug topics from the bug descriptions in the bug reports to predict bug resolution time.

FixTime uses topic modeling to recover bug topics. It assumes a project to have $K$ technical topics containing words from a dictionary $V$. A topic is represented as a multi-nomial distribution $\psi$ on the words in $V$. That is, for a topic $k$ and a word $w$, $\psi_{k,w}$ is the probability that $w$ is used to describe topic $k$ in a bug description. $\psi$ is assumed to follow Dirichlet distribution $\text{Dir}(\beta, V)$. A bug description is considered to contain a mixture $\theta$ of $K$ topics, called topic proportion. That is, $\theta_k$ is the percentage of words mentioning topic $k$. This description has a topic assignment $z$, i.e. $z_i$ denotes the topic mentioned at location $i$. The word actually used at that location is denoted by $w_i$. Since $\theta$ is the topic proportion, $z_i$ is a random variable following the multi-nomial distribution parameterized by $\theta$, i.e. $\theta_k$ is the probability that $z_i = k$. $\theta$ is assumed to also follow Dirichlet distribution $\text{Dir}(\alpha, K)$.

3.2 Training and Predicting

The training data for this model contains a collection of the resolved bugs. For each bug, the values of variables $w$ (words in description), $d$ (fixer), $e$ (bug severity), and $t$ (resolution time) are provided. Hyper-parameters $K$ (number of topics), $\alpha$, and $\beta$ are pre-chosen. The training process will infer the latent variables $\theta$ and $z$ for each bug; $\psi$ for each topic, and determine the functions $f_\mu$ and $f_\sigma$ to compute $\mu$ and $\sigma$ from $d$, $e$, and $\theta$.

After training, for a new bug, FixTime first uses the trained model to infer its topic proportion $\theta$ from its description. Then, for each available fixer $d$, it computes the log-normal parameters $\mu$ and $\sigma$ and predicts the resolution time $t$ when $d$ is assigned to this bug. Finally, it ranks the predicted resolution time to recommend the best fixer for that bug.

3.3 Preliminary Results

We performed an experiment to compare the accuracy of FixTime with that of CosTriage. We implemented a simplified instance of our topic-based prediction model, in which the topic variables $\theta$, $z$ and $\psi$ are recovered using the standard topic modeling procedure in LDA. The resolution time is predicted as $t = e^t$ (i.e. as the median of the log normal distribution $\text{LN}(\mu, \sigma)$). The log normal parameter $\mu$ is computed as a linear combination of three factors: fixer, topic, and severity. That is, $\mu = w_0 + w_d + w_k + w_e$ where $k$ is the main topic of the given bug (the one having the largest proportion). The weights $w_0, w_d, w_k$, and $w_e$ are estimated using linear regression analysis (each developer, topic, and severity level has a corresponding weight).

The experiment data includes 2,393 bugs resolved by two developers J1 and J2 (see Section 2). Like CosTriage, we sorted those bugs chronically, and chose the first 2,000 bugs for training, and 393 remaining bugs for testing. CosTriage’s average-based profiling predicts the resolution time of those 393 bugs by computing the average resolution time of the corresponding fixer and topic in the first 2,000 bugs, resulting in the median absolute error of 23.8 days. In contrast, our prediction has the median absolute error of only 2.3 days. Boxplots in Figure 3 illustrate the differences of prediction errors between FixTime and CosTriage. As seen, FixTime has significantly smaller prediction error than CosTriage.

We also compared FixTime to an ordinary linear regression model used in [5] (denoted by Linear), which predicts the resolution time as a linear combination of the three factors fixer, topic, and severity, i.e. $t = v_0 + v_d + v_k + v_e$. The prediction error of this model is also illustrated in Figure 3. As seen, it has much larger prediction error than FixTime. This result is reasonable since the ordinary linear regression assumes the normally distributed data, thus, is not well-suited for the log-normally distributed nature of bug resolution time.

The results suggest that FixTime is more accurate than the existing approaches in predicting bug resolution time. Since our regression model uses only bug severity as a factor in addition to fixers and bug topics, and utilizes the main topic of a bug, we expect that adding more factors and utilizing more topic-based factors will gain better performance.

4. RELATED WORK

Most closely related to our work is CosTriage by Park et al. [17]. CosTriage obtains developer profiles and uses topic modeling on bug reports to improve bug triaging. CosTriage classifies the bugs into different topics. For each developer, it computes his/her profile as the average time that $s$/he has fixed the bugs of each topic in the history. If $s$/he has never fixed any bug in a topic, his/her bug resolution time on that topic is derived from those of others with similar profiles.

In comparison, FixTime has important advances. First, FixTime uses a log-normal regression model to profile bug resolution time across developers and topics, while CosTriage assumes that the
resolution time of a developer for any bug in a topic stays the same over time. That assumption might not be realistic because the resolution time of the same developer on the same topic would inherently vary due to different development factors. Our experiment shows that our regression model is more suitable than CosTriage. Second, with a regression model, FixTime is able to provide a manager with more in-depth statistical information regarding the bug resolution time, rather than just the average fixing time in CosTriage. For example, one can derive a confidence interval of the predicted resolution time and this can help the manager to find a more suitable fixer with stable fixing time.

To improve bug triaging, while FixTime and CosTriage recommend bug assignment via bug resolution time, existing approaches focus on finding who should fix the bug without concerning on the time cost. Machine learning (ML) models such as SVM, Naive Bayes, C4.5, Bayesian Network have been applied for bug triaging by analyzing bug reports’ contents and meta-data [6, 2, 4]. Vector space model (VSM) has been used in bug triaging where each developer’s expertise is represented by a set of terms in their bug reports [12, 3, 15]. Jeong et al. [10] use Markov-based model to learn the patterns of bug tossing to reduce the lengths of bug tossing paths. However, bug topics are not considered.

Another line of related research focuses on the analysis and prediction of the time to fix bugs [11, 5, 7]. Kim et al. [11] reported bug-fix time statistics such as average bug-fix time, and distributions of bug-fix time in ArgoUML and PostgreSQL. Giger et al. [7] use decision tree to predict whether a new bug should and will be fixed fast or will take more time for resolution. Other bug-fixing time prediction approaches have used the number of developers involved in bug fixing, bug severity, and the number of comments to build predictors. Guo et al. [8] found that the bug-fixing time is highly correlated to the bug openers. Hooimeijer et al. [9] used the reputation of bug openers and bug severity to build a linear prediction model. Bhattacharya and Neamtiu [5] used univariate and multivariate linear regression to test the prediction performance of several features. Their results showed that the predictive power of existing features is still under desired. In brief, none of the above approaches in this direction takes into account the assigned bug fixers, the bug topics, and the log-normality of bug resolution time.

The study and analysis on bug resolution time for each developer with respect to a technical topic (domain) has been performed in our prior work [14]. In comparison, this work first aims to estimate bug resolution time for a given assignment and to recommend an assignment of fixers that can help reduce bug resolution time. Second, we found that the resolution time can be modeled via log-normal distribution. Third, based on that finding, we propose a new topic-based, log-normal model to estimate the bug resolution time for a given bug if assigned to a given developer.

5. CONCLUSIONS
In this paper, we performed an analysis on 11,115 bug records of Eclipse JDT and found that bug resolution time is log-normally distributed and varies across fixers, technical topics, and bug severity levels. We then propose FixTime, a novel method for bug assignment. The core of FixTime is a topic-based, log-normal regression model for predicting defect resolution time which then is used recommend fixing assignments. Preliminary results suggest this model has higher prediction accuracy than existing ones. We expect that adding more factors and using more topic-based factors will provide better performance in predicting bug resolution time and recommending bug assignments.

In the future work, we will implement our model by its full specification and perform the evaluation on a larger dataset with more subject systems. This preliminary work also suggests similar approaches for analyzing and predicting other time information in the life cycle of bugs, such as the time since a bug is reported to the time it is confirmed, or the time since a bug is resolved to the time it is verified and closed. The insight from such analyses and predictions could help software engineers and managers optimize their bug fixing process.

6. REFERENCES