Information and Software Technology 55 (2013) 1479–1495

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Predicting failure-proneness in an evolving software product line

Sandeep Krishnan [a,*], Chris Strasburg [a,b], Robyn R. Lutz [a], Katerina Goseva-Popstojanova [c], Karin S. Dorman [d]

[a] Department of Computer Science, Iowa State University, Ames, IA 50011-1041, United States
[b] Ames Laboratory, US DOE, Iowa State University, Ames, IA 50011-3020, United States
[c] Lane Department of Computer Science and Electrical Engineering , West Virginia University, Morgantown, WV 26506-6109, United States
[d] Department of Statistics, Iowa State University, Ames, IA 50011-1210, United States

ARTICLE INFO

Article history:
Received 14 February 2012
Received in revised form 29 September 2012
Accepted 28 November 2012
Available online 12 December 2012

Keywords:
Software product lines
Change metrics
Reuse
Prediction
Post-release defects
Failure-prone files

ABSTRACT

Context: Previous work by researchers on 3 years of early data for an Eclipse product has identified some predictors of failure-prone files that work well. Eclipse has also been used previously by researchers to study characteristics of product line software.
Objective: The work reported here investigates whether classification-based prediction of failure-prone files improves as the product line evolves.
Method: This investigation first repeats, to the extent possible, the previous study and then extends it by including four more recent years of data, comparing the prominent predictors with the previous results. The research then looks at the data for three additional Eclipse products as they evolve over time. The analysis compares results from three different types of datasets with alternative data collection and prediction periods.
Results: Our experiments with a variety of learners show that the difference between the performance of J48, used in this work, and the other top learners is not statistically significant. Furthermore, new results show that the effectiveness of classification significantly depends on the data collection period and prediction period. The study identifies change metrics that are prominent predictors across all four releases of all four products in the product line for the three different types of datasets. From the product line perspective, prediction of failure-prone files for the four products studied in the Eclipse product line shows statistically significant improvement in accuracy but not in recall across releases.
Conclusion: As the product line matures, the learner performance improves significantly for two of the three datasets, but not for prediction of post-release failure-prone files using only pre-release change data. This suggests that it may be difficult to detect failure-prone files in the evolving product line. At least in part, this may be due to the continuous change, even for commonalities and high-reuse variation components, which we previously have shown to exist.

1. Introduction

A software product line displays a high degree of commonality among the products that comprise it. The products differ one from another via a set of allowed variations. The commonalities are implemented in files reused in every product, while the variations are implemented in files available for reuse in the subset of products requiring those options or alternatives.

The high degree of commonality and low degree of variations lead us to investigate whether we can learn something about predicting failure-prone files in the product line from information about changes and failures experienced previously by the same or other products in the product line.

We perform classification of files as failure-prone and not failure-prone (two-class classification) using supervised learning methods. We define a *failure-prone file* to be a file with one or more non-trivial post-release bugs recorded. File-level predictions are then grouped at the component level to examine whether the level of reuse has an impact on the prediction of failure-proneness at the component level. For the Eclipse product line studied in this work, we classify the components based on their level of reuse: *Common* components reused in all products, *High-reuse variation* components reused in more than two products and *Low-reuse Variation* components reused in at most two products.

File-level predictions are also grouped at the product level to investigate whether the classification capability improves for different products in the product line. Data at the product level is an aggregation of data at the component level, i.e., the files in a

* Corresponding author. Tel.: +1 515 451 2338.
E-mail addresses: sandeepk@iastate.edu (S. Krishnan), cstras@iastate.edu (C. Strasburg), rlutz@iastate.edu (R.R. Lutz), Katerina.Goseva@mail.wvu.edu (K. Goseva-Popstojanova), kdorman@iastate.edu (K.S. Dorman).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.11.008

product are the files of the components that belong to that particular product. Each file is in one and only one component, but may be in multiple products.

Ongoing change is typical in product lines, including the one studied here. Change proceeds along two main dimensions. The first dimension is evolution of the product line in which, as the product line matures, more products are built. These additional products typically include new features (e.g., units of functionality [1]). The changes also may propagate to other, previously built products [2]. When the changes are incorporated into the product line, the product line asset repository is updated so that future products can reuse them.

The second dimension of product line evolution is change in an individual product from one of its releases to another. This is similar to the evolution and maintenance of a single system, except that it may happen to each system in the product line.

In previous work [3], we found that even files implementing commonalities experience change on an on-going basis and that, as the product line evolves, fewer serious failures occur in components implementing commonalities than in components implementing variations. We also found that the common components exhibit less change than the variation components over time. This led us to explore, beginning in [4], whether the stabilizing behavior of the commonalities as the product line evolves supports prediction of failure-prone files.

The following research questions motivate the work reported in this paper:

- Are there any change metrics that serve as good predictors for which files are failure-prone as a product matures over releases?
- Do any of these change metrics also serve as good predictors across all the products and components in a product line over time?
- Does our ability to predict the failure-prone files improve over time across products as the product line matures?
- Does the ability to predict failure-prone files differ across components belonging to different categories of reuse?
- How do datasets with different data collection and prediction periods affect prediction performance?
- Do datasets with incrementally increasing data collection periods yield better results?

To investigate these questions, we explore here whether accurate and meaningful predictions of failure-prone files can be made, both across the sequential releases of a single product and across the various products in a product line, taking into consideration the periods of data collection and prediction. We study whether there are good predictors of failure-prone files for individual products in the product line, whether there are good predictors across the product line, and how they are related. We study whether predicting failure-prone files over shorter time gaps is easier as compared to the standard prediction of failure-prone files six months after release.

The results reported in this paper extend our previous work to evaluate failure prediction for the Eclipse product line at the product level to also consider the component level. In brief, the new contributions first reported here include: (1) results from an investigation into whether any specific learner performed significantly better than the J48 learner we previously used for classifying failure-prone files using change data on Eclipse, (2) a quantitative evaluation of differences in defect prediction performance with respect to alternative time periods for change data collection and prediction, (3) findings from analysis of defect prediction for the three categories of reuse levels described above (commonalities, high-reuse variations, and low-reuse variations) across these periods, and (4) results from experiments using incrementally increasing data collection periods.

Our data-driven investigation uses the large open-source project Eclipse. Following Chastek et al. [5], we consider Eclipse to be a product line. We distinguish evolution of a single Eclipse product from evolution of the Eclipse product line and the evolution of its components. We also build on previous work by Zimmermann et al. [6] and by Moser et al. [7]. The authors in [6] studied defects from the bug database of three early releases of an Eclipse product at both the file and package level. They built logistic regression models to predict post-release defects. At the file level, the models had mixed results, with low recall values less than 0.4 and precision values mostly above 0.6. The authors in [7] found that change metrics performed better than code metrics on a selected subset of the same Eclipse dataset, and that the performance of the J48 decision tree learner surpassed the performance of logistic regression and Naïve Bayes learners.

Following [7], we use 17 change metrics collected over different periods of Eclipse's release cycle. Existing studies have used different types of metrics for predicting failure-prone files, including code metrics [8–12], change metrics [7,13–15] and previous defects [16]. Such metrics are used either to classify files as defective or not (binary), or to predict the number of defects per file. In general, it is easier to perform classification than to predict the number of defects. In this study, we seek to classify files as failure-prone or not with the goal being to predict whether files have one or more post-release failures.

From a product line perspective, we are most interested in observing whether predictive ability improves as the product line evolves and whether the set of prominent predictors, identified by a feature selection method based on gain ratio, changes both between products and as the product line evolves over time. In the work described in this paper, we first replicate the decision tree portion of the study presented in [7] to validate previous results and then extend it by including four more recent years of data.

In our previous work [4], we used the J48 tree-based learning algorithm for prediction. Our effort in this paper is not to identify the most optimal machine learner; rather it is to investigate improvement in prediction ability in an evolving product line. However, to validate if the J48 learner is a good choice, we perform a preliminary comparison analysis of the performance of 17 machine learners. Consistent with Menzies et al. [9,17] and Lessmann et al. [18], we observe that there is no statistically significant difference between the performance of most machine learners. As a result, in this work we continue our analysis with the J48 machine learner as implemented in Weka [19].

We look at the evolution of one particular product, Eclipse Classic, over a period of 9 years. We observe the classification results during its early evolution (versions 2.0, 2.1, and 3.0), as in [7], but also look at its more recent evolution (versions 3.3, 3.4, 3.5, and 3.6). We find some overlaps and some differences between the most prominent predictors (identified based on gain ratio) over the shorter and longer time periods for these components.

We then repeat the effort for three additional products in the Eclipse product line, Eclipse Java, Eclipse JavaEE and Eclipse C/C++, across the last four years of their evolution. We perform this analysis for three types of datasets, distinguished by their data collection and prediction periods. This is new work that has not been reported previously. We observe mixed results, with very high recall and low false-positive rates when no distinction is made between pre-release and post-release defects. However, we find that the recall rates drop significantly, if we use pre-release change data to predict post-release defects. We also observe that classifying failure-prone files using incrementally increasing data collection periods does not give better results even for commonality components. All our data and results are available at [20].