Large-Scale Patch Recommendation at Alibaba

Xindong Zhang
zxd139922@alibaba-inc.com
Alibaba Group

Chenguang Zhu
cgzhu@utexas.edu
University of Texas at Austin

Yi Li
yi_li@ntu.edu.sg
Nanyang Technological University

Jianmei Guo
jianmei.gjm@alibaba-inc.com
Alibaba Group

Lihua Liu
lihua.llh@alibaba-inc.com
Alibaba Group

Haobo Gu
haobo.haobogu@alibaba-inc.com
Alibaba Group

ABSTRACT

We present Precfix, a pragmatic approach targeting large-scale industrial codebase and making recommendations based on previously observed debugging activities. Precfix collects defect-patch pairs from development histories, performs clustering, and extracts generic reusable patching patterns as recommendations. Our approach is able to make recommendations within milliseconds and achieves a false positive rate of 22%. Precfix has been rolled out to Alibaba to support various critical businesses.

KEYWORDS

Defect detection, patch generation, patch recommendation.

ACM Reference Format:
Xindong Zhang, Chenguang Zhu, Yi Li, Jianmei Guo, Lihua Liu, and Haobo Gu. 2020. Large-Scale Patch Recommendation at Alibaba. In *42nd International Conference on Software Engineering Companion (ICSE '20 Companion), October 5–11, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377812.3390902

1 INTRODUCTION

Patch recommendation is the process of identifying errors in software systems and suggesting suitable fixes. Automated patch recommendation can significantly reduce developers' debugging efforts and the overall development costs, improving software quality and system reliability. Recommending patches automatically is a challenging task, especially for large-scale industrial codebase. Many state-of-the-art techniques from the literature make assumptions on the existence of auxiliary development artifacts such as complete test suites and detailed issue tracking as well as debugging reports, which may not be readily available in the day-to-day development environment.

Through our study of the Alibaba development practices and interviews with the developers, we identified three key challenges for existing fault localization and automated patch generation techniques to be successfully applied on our codebase.

Insufficient Labeled Data. Due to the widespread legacy code in the codebase, a large number of software projects only have partial debugging reports and very limited test cases. The commit messages may be succinct and do not follow any standard template either. Therefore, it is challenging to label defects and the associated fixes manually, given the size and complexity of the codebase.

High Responsive Standard. The application scenario of patch recommendation in Alibaba is highly interactive. Patch recommendation needs to be run whenever new commits are submitted by developers for code review. The recommended patches are then checked by developers, who may decide to incorporate the suggestions into the commits. Therefore, the responding time for patch recommendation is supposed to be reasonably low in order to be integrated into the development routine.

Generalizability Requirement. The software projects in Alibaba's codebase are diverse. These software applications cover a variety of domains including e-commerce, finance, cloud computing, and artificial intelligence, many of which are used by millions of users on a daily basis. Thus, the patch recommendation techniques should be generalizable to cover all different projects and defect types.

These specific challenges render most of the existing automated patch generation approaches inappropriate for our application scenario, since they either rely on a large amount of labeled data or require expensive compilation and test execution for each identified defect. To address this, we propose a pragmatic patch recommendation approach Precfix, with improvements in both the *precision* and *efficiency* when applied on large-scale industrial codebase.

2 APPROACH

Fig. 1 overviews the workflow of Precfix. Precfix consists of an *offline patch discovery* component and an *online patch recommendation* component. The patch discovery component extracts potential defect-patch pairs from commits in the version controlled history, clusters defect-patch pairs based on their similarity, and finally extracts generic patch templates to be stored in a database. The patch recommendation component recommends patch candidates to developers and collects their feedback, which can be used to improve the patch database.

Offline Patch Discovery. The offline patch discovery component performs three steps to generate patch templates.
(1) *Extracting Defect-Patch Pairs.* The first step is to extract a large number of defect-patch pairs from the existing codebase. This is largely based on the SZZ algorithm [3] with a number of strategies customized for the Alibaba codebase. For example, we constrain the number of files modified in a potential bug-fixing commit, filtering out any commit that exceed the threshold. This is based on our observation that such commits are often "multi-purpose" and can affect the precision of patch discovery significantly.

ICSE '20 Companion, October 5–11, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7122-3/20/05.
https://doi.org/10.1145/3377812.3390902

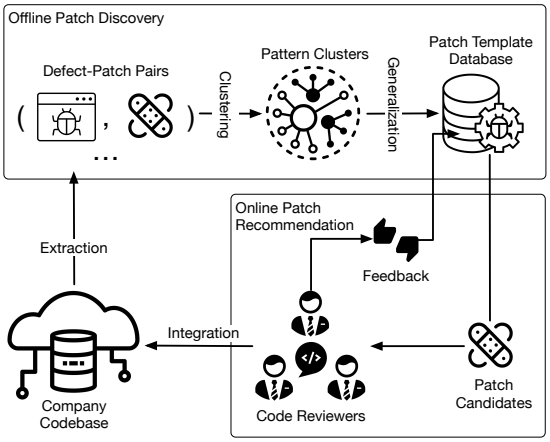


Figure 1: Overview of the PRECFIX workflow.

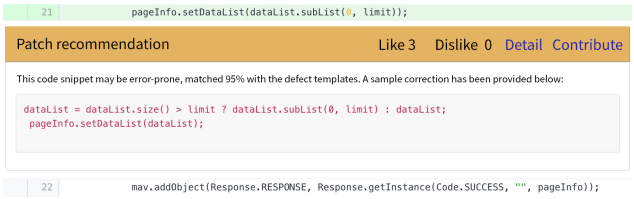
(2) *Clustering defect-patch pairs.* To obtain common defect patterns from the codebase, we group all the extracted defect-patch pairs into a set of clusters. We use DBSCAN [1] as the clustering algorithm and make several customizations. For instance, we exploit the information from API call sequences to avoid unnecessary comparisons. The intuition behind this is that if two code snippets contain two completely different API call sequences, then they are obviously not belonging to the same patch pattern, thus should not be compared during the clustering. We improve the clustering accuracy by normalizing code snippets with canonical representations of the most commonly seen statements and expressions.
(3) *Collecting generic patch templates.* After the clustering step finishes, for each pattern cluster, PRECFIX extracts a generic patch template, which summarizes the common pattern of defect-patch pairs in that cluster. The goal of template generalization is to keep the patch abstract and generally applicable in different contexts. This is achieved by first converting patches into token lists, and then applying the longest common substring (RLCS) algorithm [2] to perform matching between them. This helps separate context-specific "parameters" (unmatched) from context-independent "templates" (matched), with the latter serving as patterns.

Online Patch Recommendation. Online patch recommendation is triggered whenever developers commit new code changes. During code review, PRECFIX matches each of developers' newly committed code snippet with the template database. During the patch validation process, in addition to feedback on the recommended patches, PRECFIX also accepts patch templates created by developers and is able to integrate them into the template database. This contribution mechanism enriches the template database in the long run.

Fig. 2 shows the PRECFIX user interface with which developers interact during code reviews. Fig. 2(a) is the inline view of an identified defect and the corresponding patch recommended. Fig. 2(b) is the detailed view of the defect-patch pairs (left) and the developer template suggestion form (right), where developers can devise and submit their own patches. Each defect-patch pair can be expanded, viewed, and voted for or against.

3 IMPLEMENTATION AND EVALUATION

PRECFIX is implemented on top of the cloud-based data processing platform, MaxCompute, developed by Alibaba. The commit history



(a) Inline view of recommended patch.

(b) Detailed view of defect-patch pairs and the template suggestion form.

Figure 2: PRECFIX patch recommendation user interface.

data is preprocessed and stored in data tables on the Alibaba Cloud storage. The defect-patch pair extraction is implemented as a set of SQL scripts and user-defined functions (about 900 LOC). The clustering of defect-patch pairs (about 1 KLOC Java code) is highly parallelized and handled by the MaxCompute's MapReduce engine.

PRECFIX has been deployed in Alibaba for about one year so far. Every week, it recommends about 400 patches to developers on average, and receives about two to three false positive reports. PRECFIX managed to identify 30K defects from 7K projects in total and provided corresponding patches. Our approach is able to make recommendations within milliseconds and achieves a false positive rate of 22% confirmed by manual review. We also conducted a small-scale user study and the majority (10/12) of the interviewed developers appreciated PRECFIX, which has been rolled out to Alibaba to support various critical businesses.

4 CONCLUSION

We present PRECFIX, a patch recommendation technique designed for large-scale industrial codebase. PRECFIX does not rely on labeled defects or patches, which are difficult to obtain in practice. Instead, it automatically mines a large number of *defect-patch pairs* from historical changes, and clusters them to extract high-quality generic bug fix templates. PRECFIX has been implemented and deployed as an internal web service in Alibaba. It is also integrated as a part of the code review process and provides patch recommendations whenever developers commit new changes to the codebase.

REFERENCES

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *International Conference on Knowledge Discovery and Data Mining*. 226–231.
[2] Daniel S. Hirschberg. 1977. Algorithms for the Longest Common Subsequence Problem. *Journal of the ACM* 24, 4 (1977), 664–675.
[3] Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. 2005. When do Changes Induce Fixes?. In *International Conference on Mining Software Repositories*. 1–5.