A Bug You Like: A Framework for Automated Assignment of Bugs

Olga Baysal Michael W. Godfrey Robin Cohen
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
{obaysal, migod, rcohen}@uwaterloo.ca

Abstract

Assigning bug reports to individual developers is typically a manual, time-consuming, and tedious task. In this paper, we present a framework for automated assignment of bug-fixing tasks. Our approach employs preference elicitation to learn developer predilections in fixing bugs within a given system. This approach infers knowledge about a developer's expertise by analyzing the history of bugs previously resolved by the developer. We apply a vector space model to recommend experts for resolving bugs. When a new bug report arrives, the system automatically assigns it to the appropriate developer considering his or her expertise, current workload, and preferences. We address the task allocation problem by proposing a set of heuristics that support accurate assignment of bug reports to the developers.

1 Introduction

Large development projects incur a sizable number of bug reports every day [7]. For example, in the Eclipse open source project there are on average 29 reports submitted each day [2]. Assignment of bugs is typically a manual, time-consuming, and tedious task. The person in charge of assigning bugs, who, in most cases, is the team lead of the project, spends an hour or more a day deciding which developer is most suitable for the task at hand [2]. When making such a decision, the team lead should consider not only the expertise of the developer [3, 6] but also his preferences in fixing bugs, his current workload, as well as the priority and the difficulty of the bug.

Currently, preference elicitation is one of the hot topics in the area of artificial intelligence (AI). Preference elicitation is concerned with acquiring the user's preferences (e.g., interests, tastes, goals) and using them to make decisions on behalf of the user (e.g., recommending what news to read or which digital camera to buy, helping the user to plan a family vacation).

In this paper, we present a framework for automated assignment of bug fixing tasks. The proposed framework is able to infer a developer's level of expertise by tracking the history of the bugs previously resolved by this developer. Our approach also employs preference elicitation methods [5] to determine the developer's preferences for fixing certain types of bugs. Assigning "favorite" bugs to developers can increase their self-motivation. When given a preferred task to complete, a developer would spend less time procrastinating since he would be familiar with the task at hand. Therefore, each developer is more productive when fixing preferred bugs.

2 Related Work

The problem of automated assignment of bugs has been previously addressed by the data mining research community [1, 2, 4]. Canfora and Cerulo [4] used an information retrieval approach to automate the bug assignment process. They used textual descriptions of fixed change requests stored in both bug tracking and source code change repositories to index developers and source files as documents in an information retrieval system. They reported recall levels of 20% for Mozilla projects. The work of Mockus and Herbsleb [6] also addressed the problem of recommending experts for certain parts of the system. However, they used source code change data from a version control repository to determine appropriate experts to work on given elements of software projects. Unlike this previous work, in our approach we determine developer expertise based on their history of past bug-solving tasks extracted from the bug repository data.

Anvik et al. [2] proposed an approach to automate bug assignment using machine learning techniques. They recommended potential resolvers for the bug by mining bug reports that the developers have been previously assigned to and resolved for the system. Their approach is semiautomated because the triager, the team lead for example, uses the recommended list to select a potential developer who can fix the bug and makes the final decision on assigning this bug to the appropriate developer considering the team's current workloads and schedules. They empirically evaluated their model on two software projects and were able to correctly assign appropriate developers to the tasks, with a precision of 57% and 64% for each project.

Our proposed framework not only makes recommendations on who is capable of fixing the bug, it considers the preferences of the developers, their workloads, and schedules, the properties of the report such as difficulty and priority when allocating problem reports to team members.

3 A Framework for Automated Assignment of Bugs

Our framework consists of three components.

1. Expertise recommendation

   When a new bug report arrives, the system creates a ranked list of the developers who have the most appropriate expertise to resolve the bug in question. This ranking is done by mining the bug repository to infer expertise profiles of the individual developers, based on the bug reports that they have resolved in the past. We employ a vector space model to infer information about the developer's expertise from the history of previously fixed bugs.

2. Preference elicitation

   Developers submit their preferences in fixing bugs by providing feedback for every bug they fix. The feedback represents a developer's rating of the bug; for example, if a developer did not struggle with fixing the bug, he would provide a positive feedback on this bug by labeling it "preferred". The feedback is stored in the bug report itself and can be retrieved for reasoning at any time.

3. Task allocation

   Knowing preferences and expertise levels of all the developers, the system then considers their schedules, the priority, and the difficulty of the report when making a decision on who should be assigned to this report. We suggest strategies, which are based on a set of three heuristics for addressing task assignment problems: workload, expertise, and preference. These heuristics provide the system with a guideline to assign reports to the most appropriate developer.

4 Conclusions and Future Work

In this paper, we presented a theoretic framework for automated bug assignment considering developer preferences, expertise, and workloads. Our work provides a novel model but lacks experimental evaluation due to the number of challenges in developing and validating the proposed model. The immediate future step is to include empirical validation of the proposed framework to provide better insights on the impact of preference elicitation in personalizing the bug assignment process.

One of the main challenges to address in the future is strategic developers. In time, developers could learn how the system assigns bug fixing tasks and try to manipulate task assignment. Thus, we should ensure that the assignment of bugs is a fair and manipulation-free process. A possible solution to prevent any misuse of the system would be to consider only most recent bug fixing experience. Another problem to tackle is to design an effective incentive mechanism to support truth-telling strategy and fairness of the task allocation. We can define a set of rewards to be given to the developers for their hard work, efficiency, productivity, and creativity.

References

[1] J. Anvik. Automating bug report assignment. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 937–940, New York, NY, USA, 2006. ACM.

[2] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 361–370, New York, NY, USA, 2006. ACM.

[3] J. Anvik and G. C. Murphy. Determining implementation expertise from bug reports. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 2, Washington, DC, USA, 2007. IEEE Computer Society.

[4] G. Canfora and L. Cerulo. How software repositories can help in resolving a new change request. In *Proceedings of the Workshop on Empirical Studies in Reverse Engineering*, 2005.

[5] L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL), 2004.

[6] A. Mockus and J. D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 503–512, New York, NY, USA, 2002. ACM.

[7] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller. How long will it take to fix this bug? In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 1, Washington, DC, USA, 2007. IEEE Computer Society.