Triaging Incoming Change Requests: Bug or Commit History, or Code Authorship?

Md Kamal Hossen
Faculty: Dr. Huzefa Kagdi
Department of Electrical Engineering and Computer Science

Abstract- Combining information retrieval and source code authorship information as a novel approach to recommend expert developers to assist with software change request is presented. This approach is based on the presence of code authorship information in source code files and does not require software repository mining or training for past bugs/issues.
Our approach is compared with two other representative approaches 1) using machine learning on past bug reports and 2) based on commit logs, over three open source systems *ArgoUML*, *Jedit* and *MuCommander*. Without the need of any historical change information our approach provides recommendation accuracies equivalent or better than the two compared approaches.

1. Introduction

Software change request, such as bug fixes and new features, are an integral part of software evolution and maintenance. Automatic support for recommending expert developers to address changes request [1, 3, 18-20, 24-26, 28, 29, 34, 37] has gained lots of attention in last few years. Without efficient management of change request desirable software maintenance and evolution is not possible for large scale systems. Change requests coming from end user as well as from inside the development teams are typically specified in a free-form textual description using natural language (e.g., a bug reported to the Bugzilla system of a software project). It is not uncommon in such projects to receive tens of change requests daily that need to be resolved in an effective manner (e.g., within time, priority, and quality factors).

From previous studies [1, 3, 18-20, 24-26, 28, 29, 34, 37] it can be seen that there are mainly two board idea for automatic developer recommendation task: 1) building a model that trains from the past bug reports using their descriptions and developers who were assigned to them [1, 2], and 2) using a combination of a concept location technique to locate relevant source code to a bug request and then mine the source code (commit) repository to recommend developers [19, 21]. Both these approaches require extensive mining of software repositories.

The assumption behind our approach is that author of a source code file should have most relevant ideas about solving any problem related to that file and our approach does not require any kind of mining activities. We applied our approach to a number of bug reports sampled from ArgoUML, jEdit, and MuCommander and calculate precision, recall values of the developer recommendation. Also we empirically compared our approach two other approaches that requires software repository mining and our new approach performs equal or better than those approaches.

2. Code Authorship Based Approach For Developer Recommendation

Code authorship based approach for automatically recommending developers to incoming change requests works in following two steps:

A. *Locating Relevant Files With information Retrieval*

Given a change request description, we use Latent Semantic Indexing (LSI) [10] to locate a ranked list of relevant units of source code (*e.g.*, files, classes, and methods) that match the given description in a version of the software system. This version is typically the one in which an issue is reported or a snapshot of source code before the change request is implemented.

B. *Using Authorship to Recommend Expert Developers*

First all the relevant source code files obtained from the first step of information retrieval are converted to SrcML [9] representation to ease the process of extracting comments from source codes. Then using XML processing copyright, licensing, and authorship information are extracted from the header comments of all the source files. The content and format of the author listing in the header comments may vary across systems.

From a thorough manual examination of a number of open source projects, we devised regular expressions to extract the authors from the header comments. We also extracted and compiled all the entities of each developer from the project resources, and mapped them to a unique identifier.

After getting relevant source files for a change request and their author information we used a frequency based approach to rank authors. The hypothesis is the higher the occurrence of an author in the relevant files to a change request, the more knowledgeable that author is in handling that particular request. First a unique list of developers is created from the extracted list of developers and then for each of the developer in the unique list we count the number of relevant files in which the developer appeared. Once a frequency count of each author is obtained, all the authors are sorted in descending order of their file frequency counts. From this sorted list of authors, we recommend the top *m* ranked authors that are the most likely developers to assist with fixing the bug/change request in question. When two developers have same frequency count we break ties using the information of their file ranks and lexical positions in the source code file.

3. Empirical Study

To evaluate the efficiency of our Authorship approach we compared it with machine learning – *ML* approach [1] and *xFinder* approach [19] and addressed the following research question

RQ1: How does the accuracy of the Authorship approach compare to its two competitors that are based on software repository mining, namely ML and xFinder?

For *ArgoUML*, from top-2 to top-10, the other approaches outperformed the precision and recall reported by the *Authorship* technique with a significant difference from top-3 to top-10 (except for top-3 recall, top-8 recall, and top-10 precision). *For JEdit* xFinder exhibited a higher accuracy than ML and Authorship from top-1 to top-10 recommendations with a significant difference (except for top-5 and top-6 precision). Authorship exhibited higher accuracy than ML from top-2 to top-10 recommendations with a significant difference from top-3 to top 10. *For MuCommander* (Figure 2.c), the Authorship showed higher precision values than ML and xFinder from top-2 up to top-10 recommendations with a statistical significant difference (except for top-2).

The *Authorship* outperformed precision and recall of *ML* in *JEdit* and *MuCommander*. We found significant differences between the precisions of the two approaches in recommendations from the top-3 to top-10 developers, on *JEdit* and *MuCommande*r. Therefore, for **RQ1** we concluded that the precision of the *Authorship* outperformed *ML* on *JEdit* and *MuCommander* datasets. *Authorship* also outperformed precision of *xFinder* in *MuCommander*. We found significant differences between the precisions of the two approaches in recommendations from the top-3 to top-10 developers. Therefore, for **RQ1**, we concluded that the precision of the *Authorship* outperformed *xFinder* on *MuCommander*.

4. Conclusions

To the best of our knowledge, our approach is the only one to use a combination of a concept location technique and the source code authorship for assigning expert developers to change requests. It does not need to mine past change requests or source code change repositories
A single-version source code analysis of a system is only required. Our simple lightweight approach is 20% more accurate than an approach that uses machine learning on past bug reports in one of the systems.

5. References

[1] Anvik, J., Hiew, L., and Murphy, G. C., "Who should fix this bug?" inProceedings of 28th ICSE'06, pp. 361-370.
[2] Anvik, J. and Murphy, G., "Determining Implementation Expertise from Bug Reports", in Proceedings of MSR'07, Minneapolis, MN.
[3] Anvik, J. and Murphy, G. C., "Reducing the effort of bug report triage:Recommenders for development-oriented decisions", *ACM TOSEM*, vol.20, no. 3, 2011.
[10] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by Latent Semantic Analysis", *Journal of the American Society for Info Science*, vol. 41, no. 6, 1990, pp. 391-407.
[18] Jeong, G., Kim, S., and Zimmermann, T., "Improving Bug Triage withBug Tossing Graphs", in Proc. of 7th ESEC/FSE 2009.
[19] Kagdi, H., Gethers, M., Poshyvanyk, D., and Hammad, M., "Assigning Change Requests to Software Developers", *JSME*, vol. 24, no. 1, January 2012, pp. 3–33.
[20] Kagdi, H., Hammad, M., and Maletic, J. I., "Who Can Help Me with this Source Code Change?" in Proc. ICSM'08
[21] Kagdi, H. and Poshyvanyk, D., "Who Can Help Me with this ChangeRequest?" in Proc. ICPC'09, pp. 273-277
[24] Ma, D., Schuler, D., Zimmermann, T., and Sillito, J., "Expertise Recommendation with Usage Expertise", in Proc. of 25th ICSM'09.
[25] Matter, D., Kuhn, A., and Nierstrasz, O., "Assigning Bug Reports using a Vocabulary-Based Expertise Model of Developers", in Proc. of 6th IEEE MSR'09, pp. 131—140.
[26] McDonald, D. and Ackerman, M., "Expertise Recommender: A Flexible Recommendation System and Architecture", in Proc. of CSCW '00, pp. 231-240.
[28] Minto, S. and Murphy, G., "Recommending Emergent Teams", in Proc. of MSR '07.
[29] Mockus, A. and Herbsleb, J., "Expertise Browser: a Quantitative Approach to Identifying Expertise", in Proc. 24th ICSE '02, pp. 503-512.
[34] Robles, G., González-Barahona, J., "Developer Identification Methods for Integrated Data from Various Sources", MSR'05, pp.106-110.
[37] Surian, D., Liu, N., Lo, D., Tong, H., Lim, E. P., and Faloutsos, C., "Recommending People in Developers' Collaboration Network", in Proc. of 18th WCRE'11, pp. 379-388.