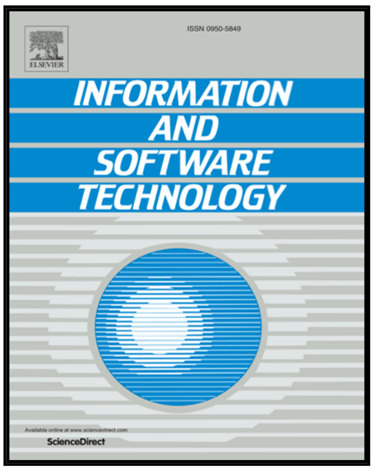

Who Should Comment on This Pull Request? Analyzing Attributes for More Accurate Commenter Recommendation in Pull-Based Development

Jing Jiang, Yun Yang, Jiahuan He, Xavier Blanc, Li Zhang

PII:	S0950-5849(16)30283-X
DOI:	10.1016/j.infsof.2016.10.006
Reference:	INFSOF 5780

To appear in: *Information and Software Technology*

Received date:	24 November 2015
Revised date:	22 October 2016
Accepted date:	24 October 2016

Please cite this article as: Jing Jiang, Yun Yang, Jiahuan He, Xavier Blanc, Li Zhang, Who Should Comment on This Pull Request? Analyzing Attributes for More Accurate Commenter Recommendation in Pull-Based Development, *Information and Software Technology* (2016), doi: 10.1016/j.infsof.2016.10.006

Who Should Comment on This Pull Request? Analyzing Attributes for More Accurate Commenter Recommendation in Pull-Based Development

Jing Jiang[a], Yun Yang[a], Jiahuan He[a], Xavier Blanc[b], Li Zhang[a]

[a] *State Key Laboratory of Software Development Environment*
Beihang University, Beijing, China
{jiangjing,lily}@buaa.edu.cn,ayonel@qq.com,lightbot.johnson@gmail.com
[b] *University of Bordeaux*
LaBRI, UMR 5800
F-33400, Talence, France
xavier.blanc@labri.fr

Abstract

Context: The pull-based software development helps developers make contributions flexibly and efficiently. Commenters freely discuss code changes and provide suggestions. Core members make decision of pull requests. Both commenters and core members are reviewers in the evaluation of pull requests. Since some popular projects receive many pull requests, commenters may not notice new pull requests in time, and even ignore appropriate pull requests.
Objective: Our objective in this paper is to analyze attributes that affect the precision and recall of commenter prediction, and choose appropriate attributes to build commenter recommendation approach.
Method: We collect 19,543 pull requests, 206,664 comments and 4,817 commenters from 8 popular projects in GitHub. We build approaches based on different attributes, including activeness, text similarity, file similarity and social relation. We also build composite approaches, including time-based text similarity, time-based file similarity and time-based social relation. The time-based social relation approach is the state-of-the-art approach proposed by Yu et al. Then we compare precision and recall of different approaches.
Results: We find that for 8 projects, the activeness based approach achieves the top-3 precision of 0.276, 0.386, 0.389, 0.516, 0.322, 0.572, 0.428, 0.402, and achieves the top-3 recall of 0.475, 0.593, 0.613, 0.66, 0.644, 0.791, 0.714, 0.65, which outperforms approaches based on text similarity, file similarity or social relation by a substantial margin. Moreover, the activeness based approach achieves better precision and recall than composite approaches. In comparison with the state-of-the-art approach, the activeness based approach improves the top-3 precision by 178.788%, 30.41%, 25.08%, 41.76%, 49.07%, 32.71%, 25.15%, 78.67%, and improves the top-3 recall by 196.875%, 36.32%, 29.05%, 46.02%, 43.43%, 27.79%, 25.483%, 79.06% for 8 projects.
Conclusion: The activeness is the most important attribute in the commenter prediction. The activeness based approach can be used to improve the commenter recommendation in code review.

Keywords: Commenter Recommendation, Reviewer Recommendation, Attribute Selection, Pull-based Software Development

1. Introduction

The pull-based software development is an emerging paradigm for distributed software development [1, 2]. Developers pull code changes from other repositories or the same repository in different branches, and merge them locally, rather than push changes to a central repository. Various open source software hosting sites, notably Github, provide support for pull-based development and allow developers to make contributions flexibly and efficiently. In GitHub, developers are allowed to fork repositories and make changes without asking for permission. Developers can submit pull requests when they want to merge their changes into the repositories they fork from. This pull-based software development separates making modification and integrating change, and makes contributing to others' repositories much easier than it has ever been [3].

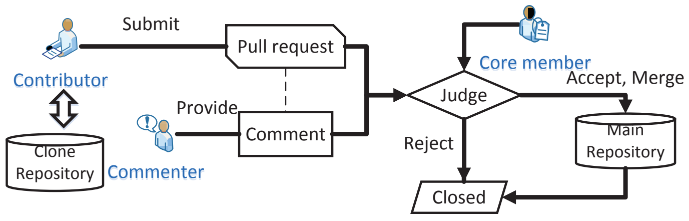

Figure 1: The overview of contribution process

As shown in Figure 1, the pull request process mainly includes three roles in GitHub, namely contributors, core members and commenters. Firstly, contributors modify codes to fix