Improving Code Review Effectiveness through Reviewer Recommendations

Patanamon Thongtanunam[*] and

Raula Gaikovina Kula[†], Ana Erika Camargo Cruz[*], Norihiro Yoshida[*], Hajimu Iida[*]

[*]Nara Institute of Science and Technology (NAIST)

[†]Osaka University, Japan

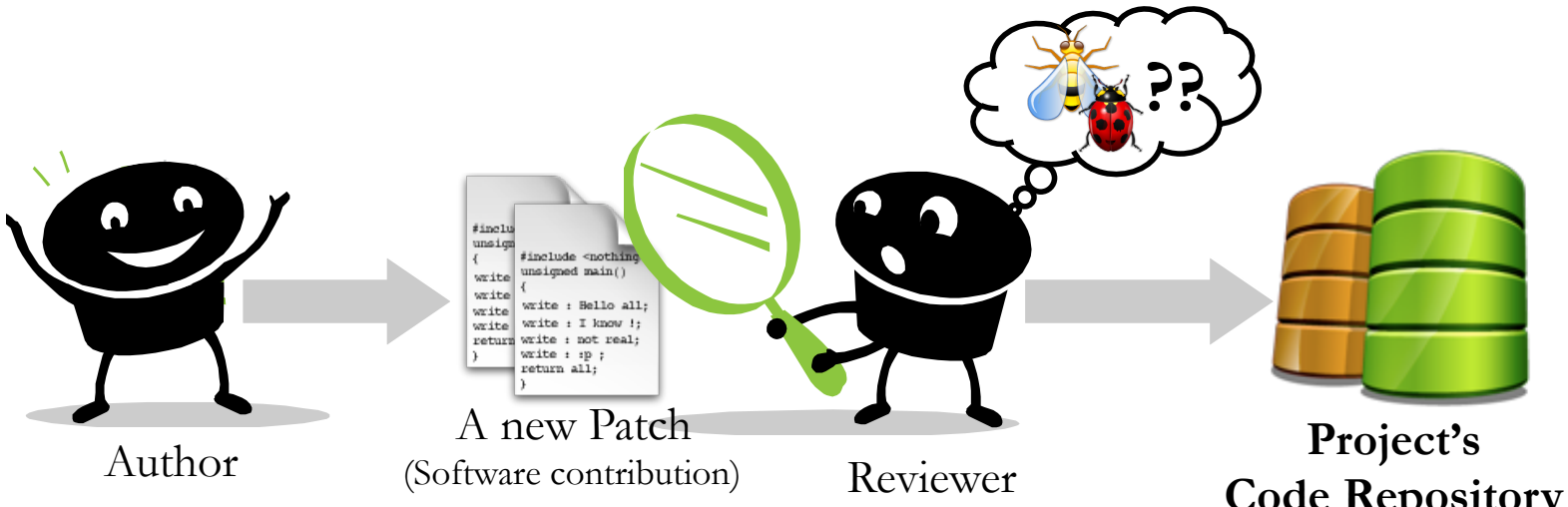
Laboratory for Software Design & Analysis

established in 2005

Nara Institute of Science and Technology NAIST ®

Introduction

- **Code Review** : A source code inspection performed by developers other than the author
  - Supported Tools: Gerrit, ReviewBoad, etc.
  - A patch reviewed by **developers with related knowledge** will have high quality and low defects



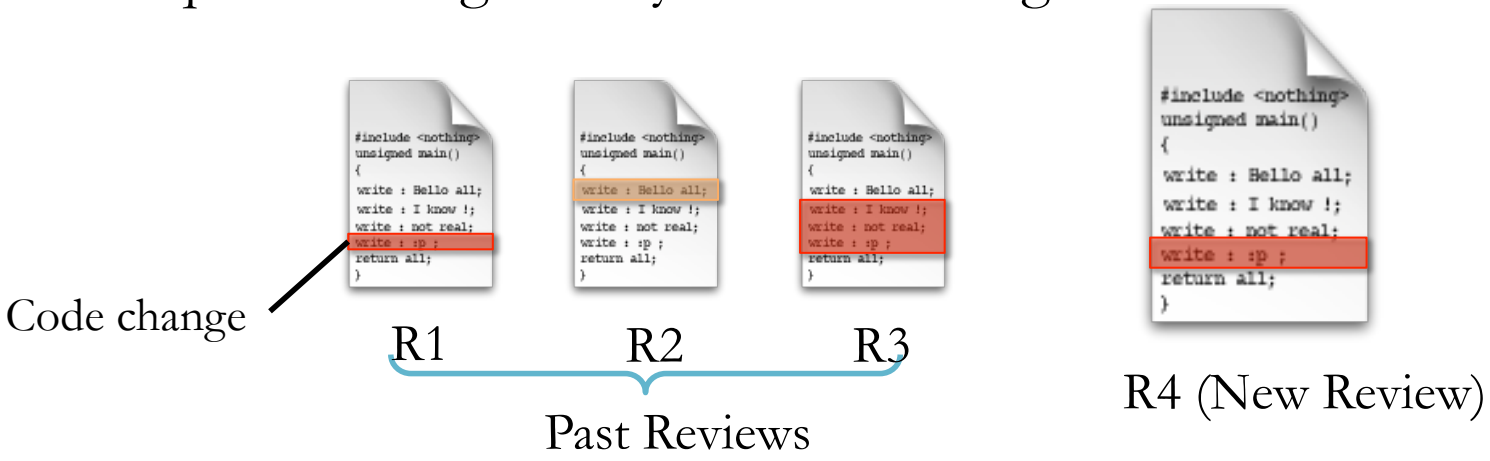
Reviewer Assignment Problem

- It is difficult and time-consuming to find an appropriate reviewers in large-scale projects

e.g. Globally distributed software development

Previous Work

- **Review Bot's algorithm**[1]
  - For industrial setting of VMware
  - Selects reviewers who have reviewed in the same section of code change.
  - Required a long history of code changes in code review

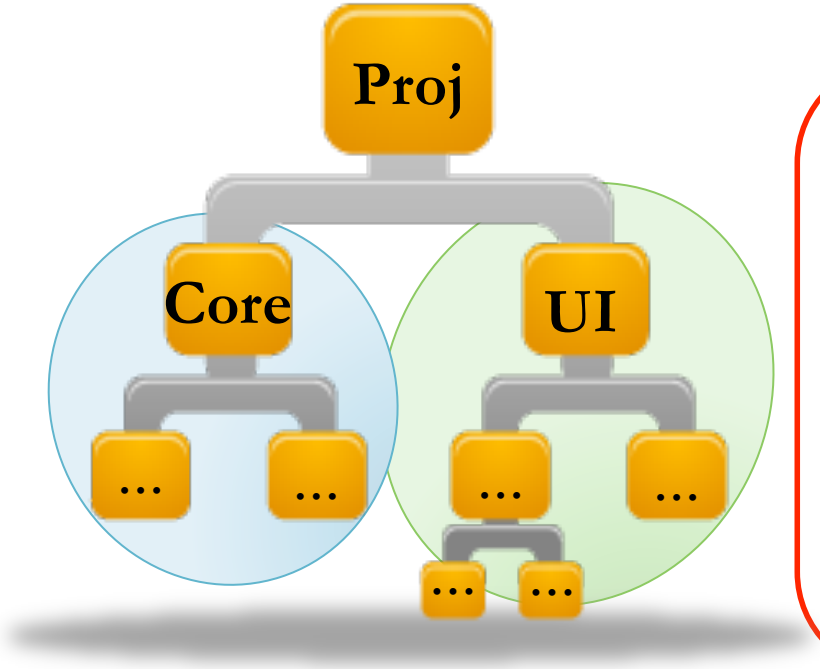


Reviewer Candidates = Reviewers ([R3, R1])

[1] V. Balachandran, "Reducing Human Effort and Improving Quality in Peer Code Reviews using Automatic Static Analysis and Reviewer Recommendation," in Proc. ICSE'13, 2013, pp. 931–940.

Reviewer Recommendation Algorithm

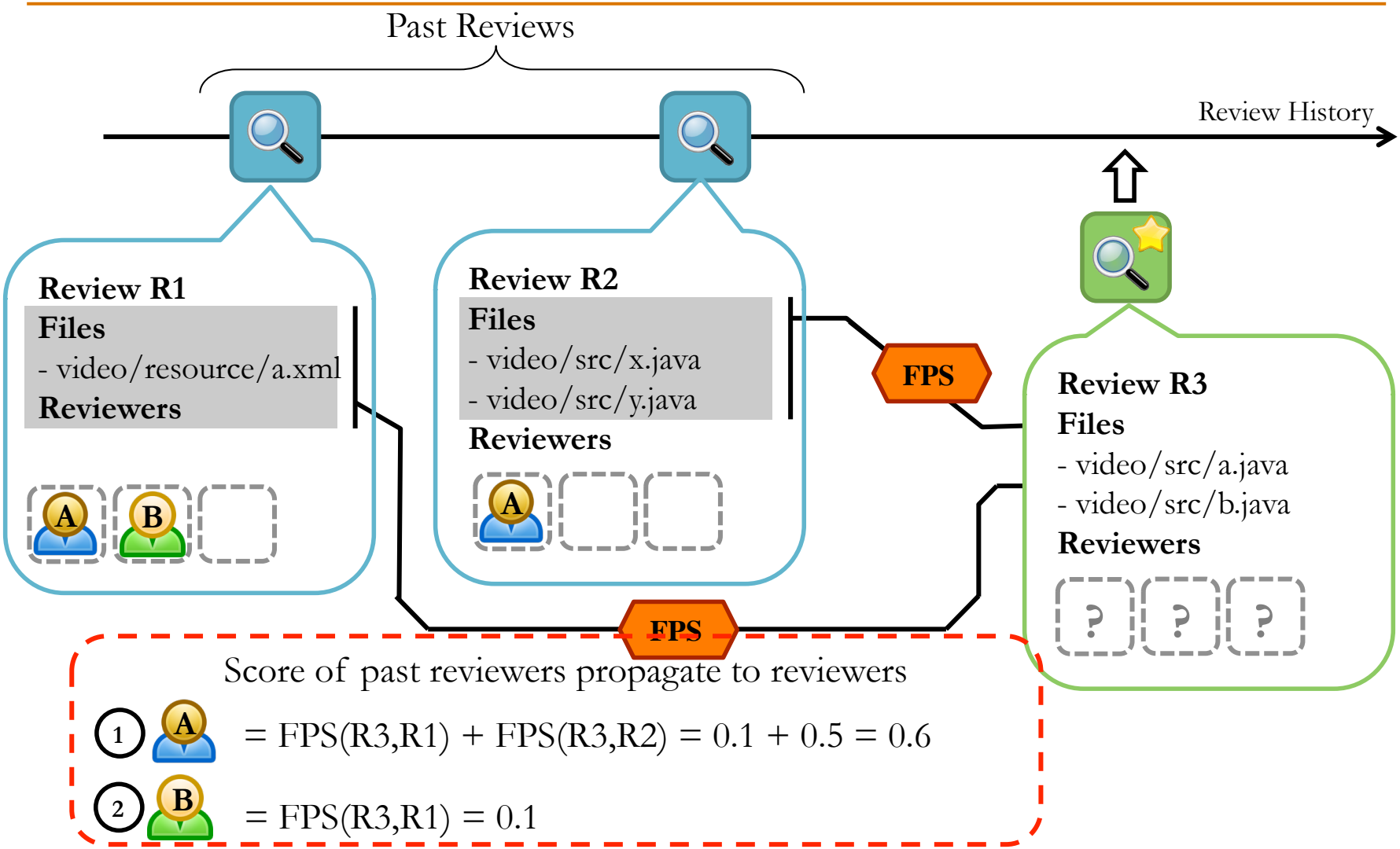
- Propose **File Path Similarity (FPS)** algorithm
  - Select reviewers by calculating similarity of file paths comparing with file paths of previous reviews.



- Files with similar functions are usually located under the same or near directories. [2]
- Reviewers having similar knowledge of system would have reviewed similar file.

[2] Ivan T. Bowman, Richard C. Holt, and Neil V. Brewster. Linux as a case study: Its extracted software architecture. In Proceedings ICSE '99, pp. 555–563, 1999.

Example of FPS algorithm



FPS score

- File Path Similarity score of new review $\boldsymbol{R_n}$ and past review $\boldsymbol{R_p}$

$$\text{FPS}(R_n, R_p, m) = \frac{\sum_{\substack{f_n \in \text{Files}(R_n),\\ f_p \in \text{Files}(R_p)}} \text{Similarity}(f_n, f_p)}{|\text{Files}(R_n)| \times |\text{Files}(R_p)|} \times \delta^m$$

- Similarity Function:

$$\text{Similarity}(f_n, f_p) = \frac{\text{commonPath}(f_n, f_p)}{\max(\text{Length}(f_n), \text{Length}(f_p))}$$

  - **commonPath(f_n,f_p) : Longest Common Prefix function** returns number of directory/file names that appear from the beginning of both file path

commonPath("**video**/**src**/a.java", "**video**/**src**/b.java") = 2

Evaluation

- **Accuracy** (same as the review bot research)

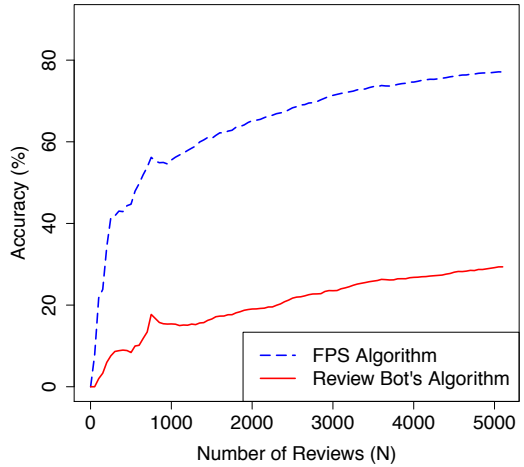
$$\text{Top-k Accuracy} = \frac{\text{Number of Correct Top-k recommendation}}{\text{Total number of reviews}}$$

- **Study Projects**
  - Using Gerrit code review

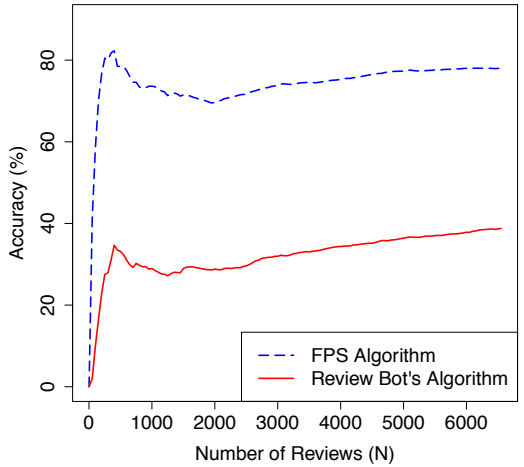
Projects	Study Period*	# of Reviews	# of Files
AOSP	Oct 2008 – Jan 2012 (~2 years)	5,126	26,840
OpenStack	Jul 2011 – May 2012 (~1 year)	6,586	16,953
Qt	May 2011 – May 2012 (~1 year)	23,810	78,401

*Study period started from the 1st year of using peer code review.

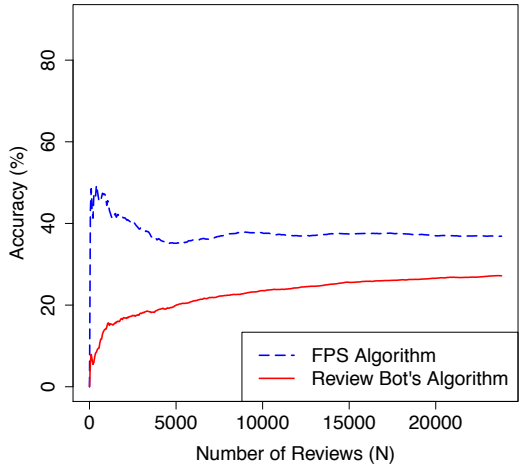
Results



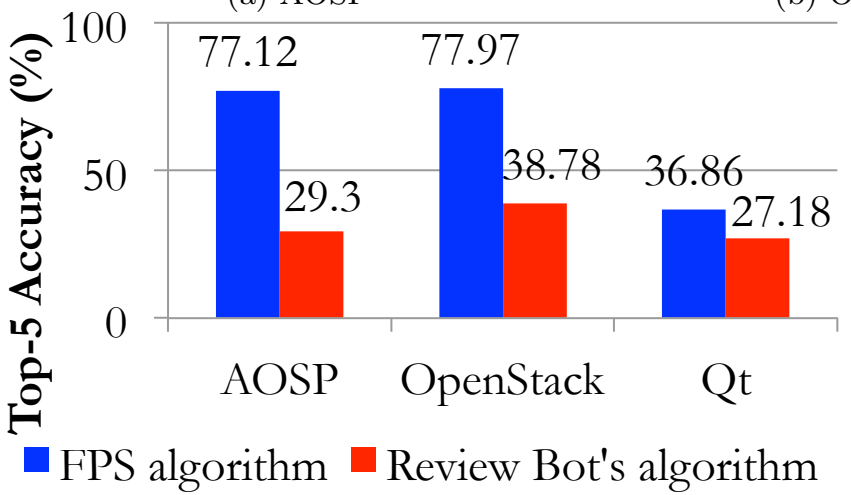
(a) AOSP



(b) OpenStack



(c) Qt



- FPS algorithm significantly outperformed Review Bot's algorithm

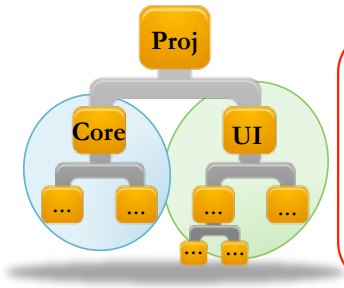
Reviewer Assignment Problem

- It is difficult and time-consuming to find an appropriate reviewers in large-scale projects

e.g. Globally distributed software development

Reviewer Recommendation Algorithm

- Propose **File Path Similarity (FPS)** algorithm
  - Select reviewers by comparing file paths with previous reviews



- Files with similar functions are usually located in the same or near directories. [1]
- Reviewers having similar knowledge of system would have reviewed similar files.

Evaluation

- **Accuracy** (same as the review bot research)

$$\text{Top-k Accuracy} = \frac{\text{Number of Correct Top-k recommendation}}{\text{Total number of reviews}}$$

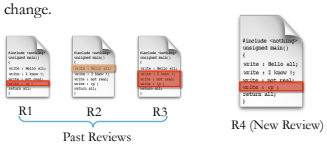
- **Study Projects**

Period*	# of Reviews	# of Files
– Jan 2012 years)	5,126	26,840
– May 2012 year)	6,586	16,953
– May 2012 year)	23,810	78,401

g peer code review.

Previous Work

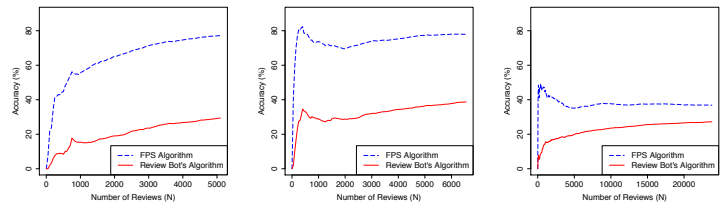
- **Review Bot's algorithm**[1]
  - For industrial setting of VMware
  - Selects reviewers who have reviewed in the same section of code change.

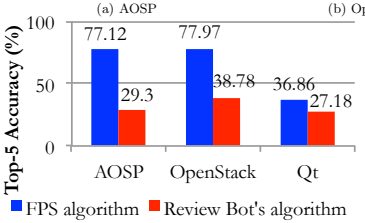


Reviewer Candidates = Reviewers ([R3, R1])

[1] V. Balachandran, "Reducing Human Effort and Improving Quality in Peer Code Reviews using Automatic Static Analysis and Reviewer Recommendation," in Proc. ICSE'13, 2013, pp. 931–940.

Results





- FPS algorithm significantly outperformed Review Bot's algorithm