Expanding the Number of Reviewers in Open-Source Projects by Recommending Appropriate Developers

Aleksandr Chueshev[1], Julia Lawall[2], Reda Bendraou[1], and Tewfik Ziadi[1]

LIP6 [1]

SORBONNE UNIVERSITÉ [1]

Inria [2]

BENEVOL'2020

Why expand the number of reviewers?

- promote knowledge-sharing among the contributors
- balance the workload without putting too much burden on a few key persons

How to expand the number of reviewers?

Approaches:

- recommendations of external reviewers [Rahman et al., ICSE-C'16]

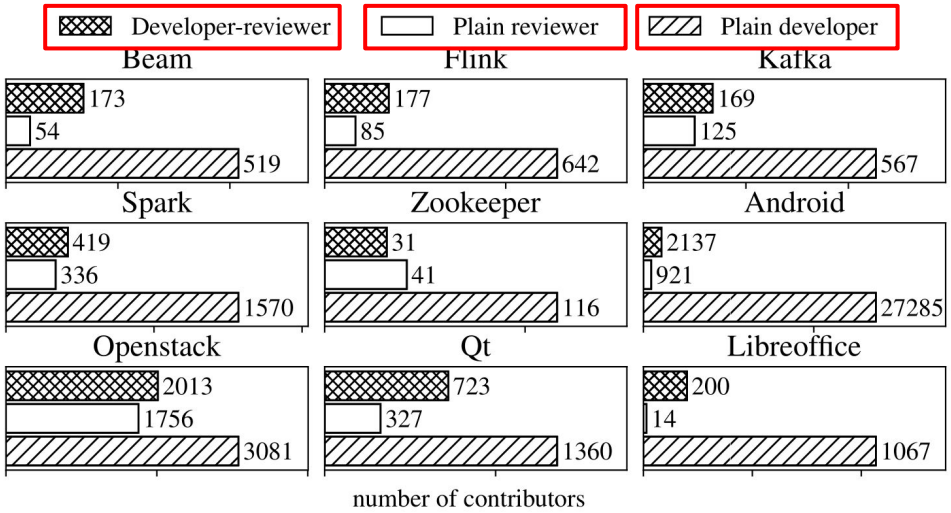
Limitations:

- external reviewers need time to get familiar with the project

How to expand by internal recommendations?

Exploratory study – part I

The roles of contributors and their distribution:



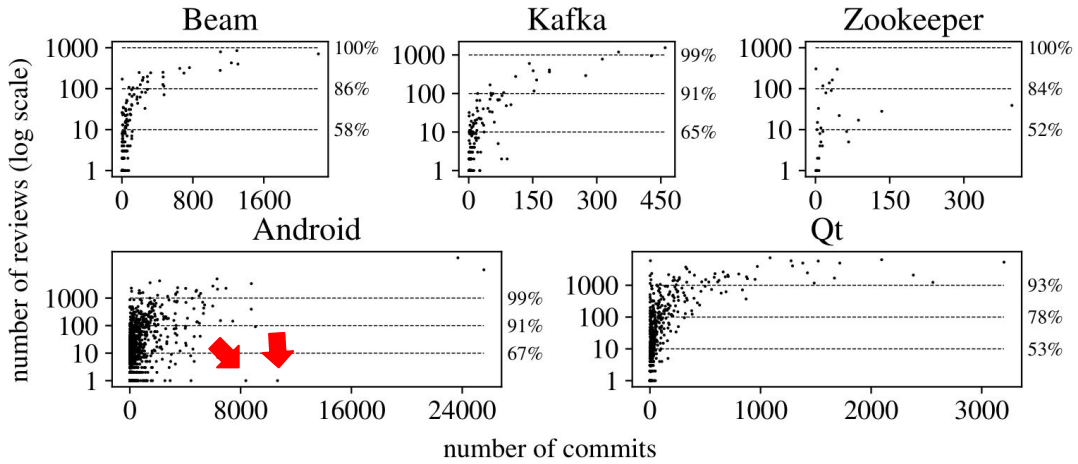
on average, 55% more plain developers than all reviewers

Summary:

Each project has an opportunity to increase the number of reviewers from among the plain developers

Exploratory study – part II

The distribution of efforts between development and reviewing:



Summary:

Each project contains low-intensity developer-reviewers with sufficient development experience to perform more reviews

Identified opportunities

- recommend plain developers
- recommend low-intensity developer-reviewers

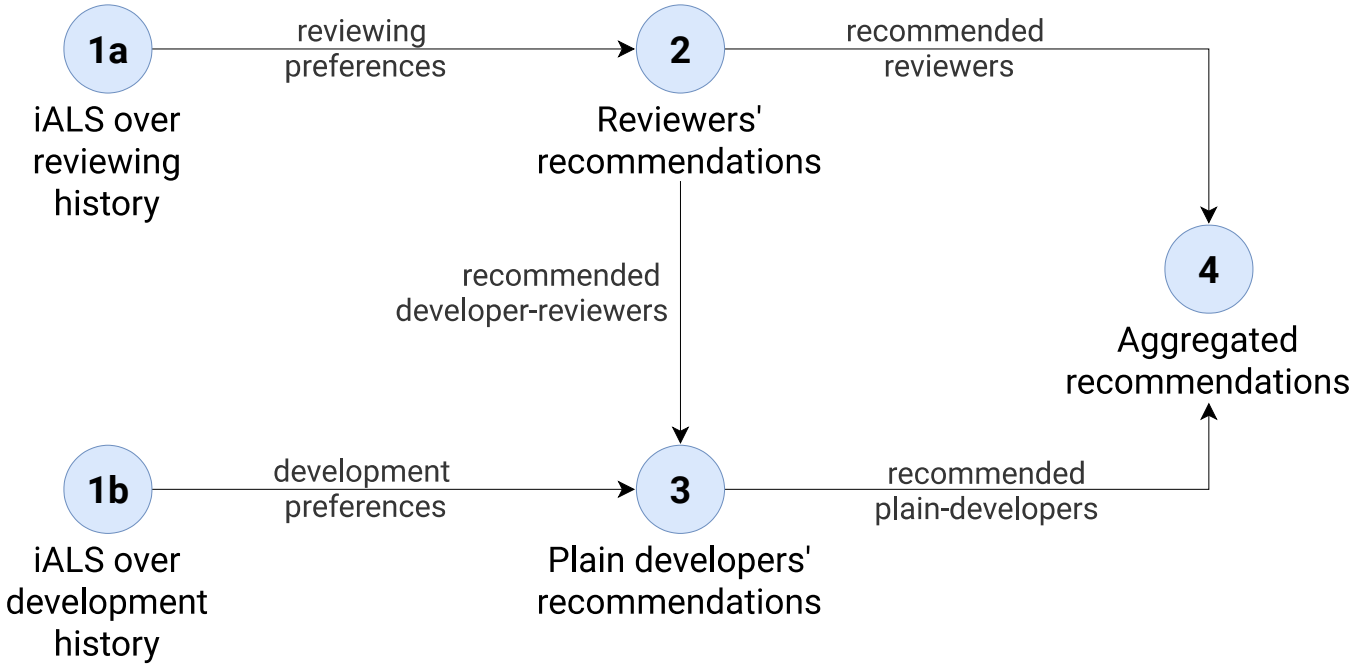
Who is appropriate?
How to identify appropriate candidates?

REx - Expanding Reviewers

Features:

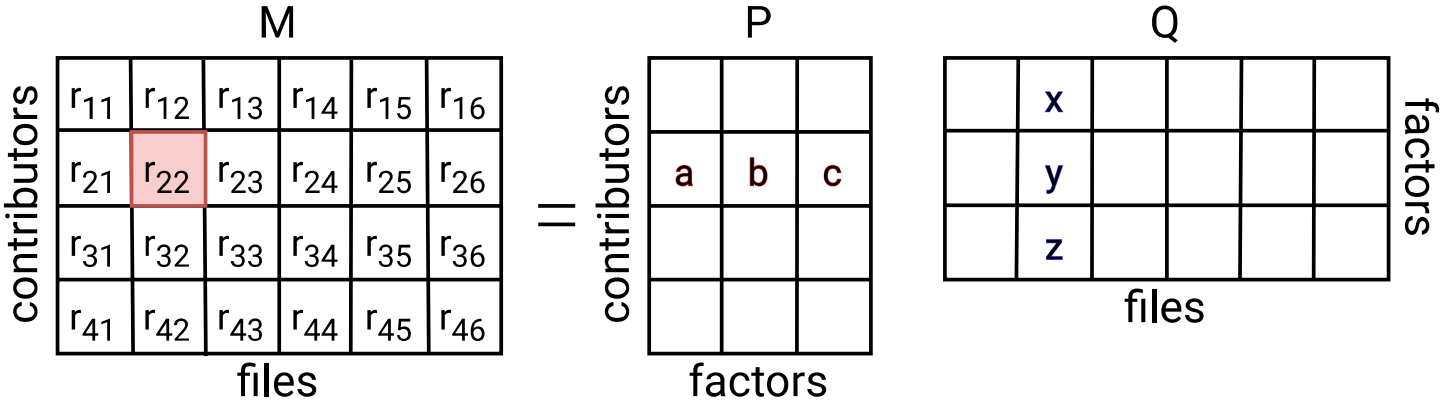
- no need for any explicit interactions from contributors
- development language independent
- considering varying levels of expertise in the different parts of the system
- recommendation of both previous and new possible reviewers

REx: Workflow

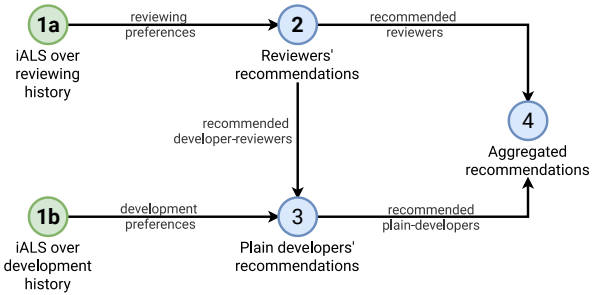


Step #1a,b: iALS to mine contributor's preferences

[Hu et al., ICDM'08]



- r_{ui} - the number of reviews (commits) of the ***u***-th contributor in the ***i***-th file
- ***P*** - the contributor-factor matrix, representing reviewing (development) preferences
- ***Q*** - the file-factor matrix, representing files properties



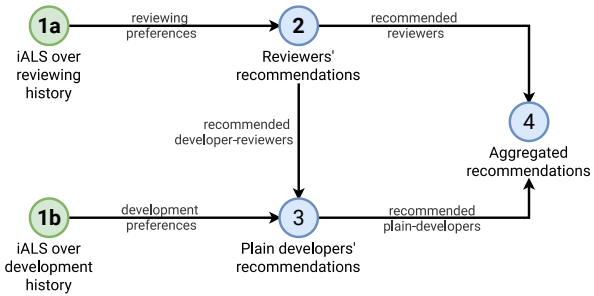
Step #1a,b: Results

Factorization of the **review history** $M^{rev} \approx P^{rev} Q^{rev\,T}$ results into:

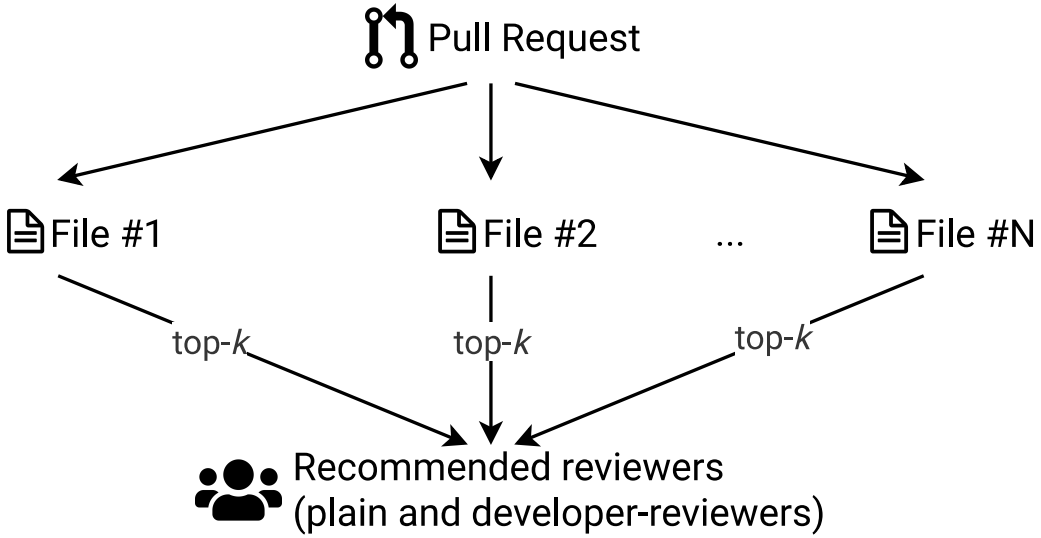
- $p_u^{rev} \in P^{rev}$ - the contributor's reviewing preferences
- $q_i^{rev} \in Q^{rev}$ - the reviewing file profile

Factorization of the **development history** $M^{dev} \approx P^{dev} Q^{dev\,T}$ results into:

- $p_u^{dev} \in P^{dev}$ - the contributor's development preferences
- $q_i^{dev} \in Q^{dev}$ - the development file profile

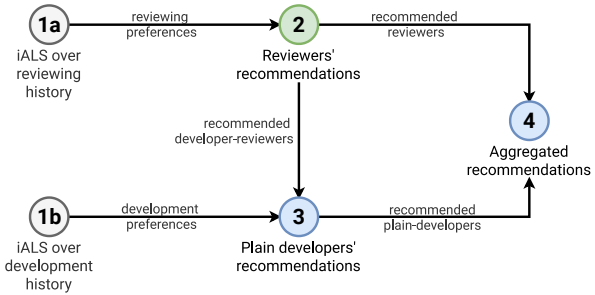


Step #2: Recommending reviewers



Calculating **reviewing score** to make recommendations:

$$\hat{r}_{ui}^{rev} = p_u^{rev} \cdot q_i^{rev}$$



Recommends **previous** reviewers, including **low-intensity** developer-reviewers

Step #3: Recommending plain developers

Recommendations. For each found developer-reviewer, recommend *top-k* similar plain developers.

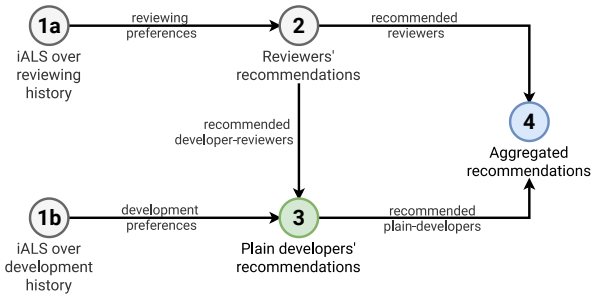
Estimating possible reviewing score: contributors with similar development preferences may also have similar reviewing preferences:

$$\hat{r}_{ui}^{rev} = \frac{\sum_{v \in U} cos(p_u^{dev}, p_v^{dev}) \ \hat{r}_{vi}^{rev}}{\sum_{v \in U} cos(p_u^{dev}, p_v^{dev})}$$

- U - the set of found developer-reviewers

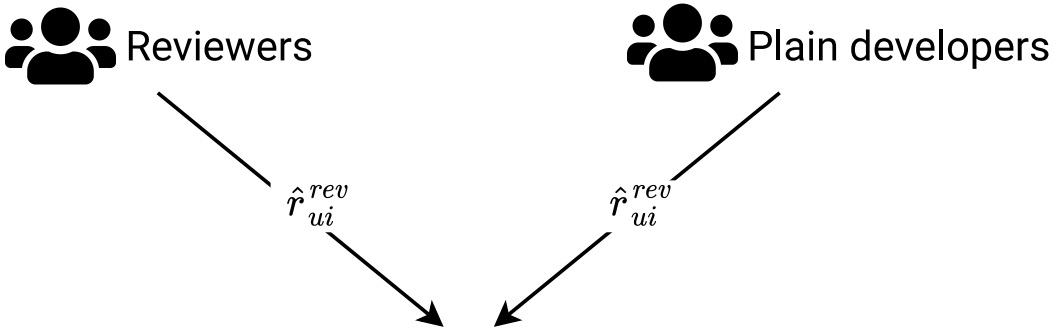
Calculating **similarity score** to make recommendations:

$$s_{uv}^{dev} = cos(p_u^{dev}, p_v^{dev})$$



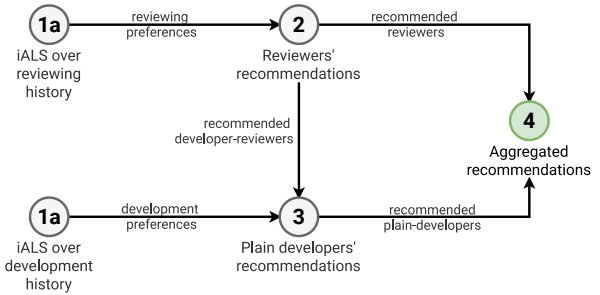
Step #4: Aggregating recommendations

Recommend ***top-N*** for a pull request, involving all reviewing scores over a set of files:



Calculating **overall reviewing score** to make recommendations:

$$\overline{r}_u = \sum_{i \in I_{pull}} \hat{r}_{ui}^{rev}$$



Experimental study

RQ1: How well is our system able to predict previous reviewers compared to existing solutions?

RQ2: Is our system able to expand the set of previous reviewers?

Results – RQ1

Projects	Reviewers	Top-3			Top-5			MRR		
		REx	Tie	Rev.	REx	Tie	Rev.	REx	Tie	Rev.
Beam	227	**0.45**	-	-	**0.60**	-	-	**0.32**	-	-
Flink	248	**0.50**	-	-	**0.61**	-	-	**0.34**	-	-
Kafka	285	**0.59**	-	-	**0.72**	-	-	**0.34**	-	-
Spark	572	**0.51**	-	-	**0.66**	-	-	**0.29**	-	-
Zookeeper	69	**0.67**	-	-	**0.77**	-	-	**0.42**	-	-
Android	94	0.60	**0.81**	0.71	0.68	**0.87**	0.79	0.52	**0.70**	0.60
Openstack	82	0.66	**0.73**	0.66	0.78	**0.83**	0.77	0.44	**0.60**	0.55
QT	202	**0.63**	0.45	0.34	**0.71**	0.52	0.41	**0.51**	0.41	0.31
Libreoffice	64	0.36	**0.91**	0.47	0.44	**0.93**	0.59	0.31	**0.84**	0.40

Tie [Xia et al., ICSME'15]

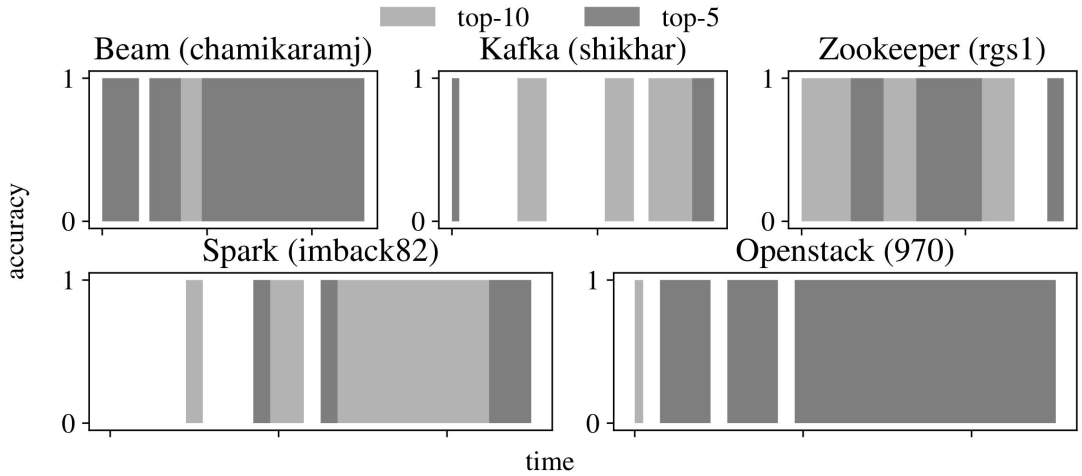
Rev. - RevFinder [Thongtanunam et al., SANER'15]

Summary:

REx scores lower than *Tie* but close to *RevFinder* on the provided data set. However, *REx* has more functionality, as it recommends both previous and new possible reviewers, and exceeds the numbers for QT.

Results – RQ2

Tracking plain developers who over time become reviewers:



Summary:

For the selected projects, *REx* has found 466 plain developers who eventually become developer-reviewers.

Conclusion

- A novel approach, ***REx*** for OSS projects
  - to recommend previous reviewers
  - to expand the number of reviewers from among the appropriate developers
- A rich collection of data within five ASF projects and four Gerrit communities
- An exploratory study on the current state of reviewing in OSS projects
- Source code and data available on GitHub

Emails:

- Aleksandr Chueshev <aleksandr.chueshev@lip6.fr>
- Julia Lawall <julia.lawall@inria.fr>
- Reda Bendraou <reda.bendraou@lip6.fr>
- Tewfik Ziadi <tewfik.ziadi@lip6.fr>