Assessing Developer Contribution with Repository Mining-Based Metrics

Jalerson Lima, Christoph Treude, Fernando Figueira and Uirá Kulesza

Introduction and Motivation

- ✓ Assessing developer's contribution is a challenging task
  - Many potential sources of contribution have to be considered
  - Managers usually have to make decisions based on the performance of their developers
  - There is no consensus on how to measure developer contribution
- ✓ Several metrics have been proposed
  - Few have been evaluated by project and team leaders
  - Project and team leaders are those who will base their decision on them

Our goal is to design a suite of developer contribution metrics based on empirical evidence obtained from project and team leaders

Repository Mining-based Metrics

Code Contribution (CC)

$$CC_{dev} = added\_loc_{dev} + changed\_loc_{dev}$$

Average Complexity per Method (ACM)

✓ Average complexity per added methods (ACAM)

$$ACAM_{dev} = \frac{\sum(complex\_added\_methods_{dev})}{added\_methods_{dev}}$$

✓ Average complexity per changed methods (ACCM)

$$ACCM_{dev} = \frac{\sum(\Delta complex\_changed\_methods_{dev})}{changed\_methods_{dev}}$$

Bug Fixing Contribution (BFC)

✓ Commits in bug fixing tasks
✓ Percentage value

$$BFC_{dev} = \left(\frac{commits_{dev}}{commits_{all}}\right) \times 100$$

Introduced Bugs (IB)

✓ Based on Eyolfson et al.'s approach
✓ Number of introduced bugs per developer

Dev	CC	ACM		IB	BFC
		ACAM	ACCM		
A	62	5.00	2.33	2	17%
B	459	1.59	0.40	9	76%
C	313	1.62	0.93	0	7%

Research Method

Location: SINFO, a Software Factory of the Federal University of Rio Grande do Norte, Brazil

1 Metrics Extraction
- ✓ 12 weeks
- ✓ 4 development teams
- ✓ Code repository
- ✓ Issue tracking system

2 Interviews
- ✓ Semi-structured
- ✓ 20 main and 10 supporting questions
- ✓ 7 team leaders

3 Data Analysis
- ✓ Transcription
- ✓ Grounded theory
- ✓ Data coding
- ✓ Memos

Preliminary Results
Code Contribution

- ✅ Useful information
- ⚠ *"May be useful with the complexity metric"* (PL1)
- ⊗ May penalize developers for using modern technologies or techniques

Preliminary Results
Average Complexity per Method

- ✅ *"Allows to perform a technical analysis"* (PL3)
- ✅ *"Helps to identify a developer who needs training"* (PL2)
- ⚠ Should be followed with task list

Preliminary Results
Introduced Bugs

- ✅ Useful information
- ⚠ Can't be used in isolation
- ⊗ May penalize developers who have been on the project for longer

Preliminary Results
Bug Fixing Contribution

- ⊗ The metric only quantifies commits, however, some tasks don't require coding
- ⊗ The number of commits is not a reliable attribute to measure effort

Preliminary Results
Overall Benefits and Limitations

- ✅ Useful to perform a quantitative contribution assessment
- ✅ May reduce the amount of time to evaluate developers
- ✅ Technical and objective criteria to evaluate developers
- ⚠ Can't replace the subjective evaluation

Discussion and Future Work

Best evaluated metrics
- Code contribution
- Average complexity per method

Worst evaluated metrics
- Introduced bugs
- Bug fixing contribution

Evaluate further metrics
- Communication (e-mail, Slack, HipChat and Gitter)
- Collaboration (GitHub, GitLab and BitBucket)
- Task distribution

Interview developers about being evaluated by these metrics
- Compare their answers with team leaders' answers

Investigate the impact of measuring contribution on developers' behavior
- Hawthorne effect

Metrics-based reward mechanism
- Gamification

Jalerson Lima, Christoph Treude, Fernando Figueira Filho and Uirá Kulesza
Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte, Brazil
jalerson.lima@ifrn.edu.br, {ctreude,fernando,uira}@dimap.ufrn.br