How Does Contributors Involvement Influence Open Source Systems

Reem Alfayez, Pooyan Behnamghader, Kamonphop Srisopha, Barry Boehm
Open Source Systems

• A company open sourcing a system.
• A single developer or a group of developer.
Core vs. Peripheral Contributors

<table>
<thead>
<tr>
<th>Core</th>
<th>Peripheral</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Contribute more to the system.</td>
<td>• Contribute less to the system.</td>
</tr>
<tr>
<td>• Make major decisions regrading the system.</td>
<td>• Have less power when making decisions.</td>
</tr>
</tbody>
</table>
Motivation

• Peripheral contributors represent a large part of the OSS community (48.98%)

• They:
  – Fix bugs (30.20%).
  – Fix typos and grammar issues (28.64%).
  – Add new features (18.75%).
  – Refactor code (8.85%).
Research Questions

• **RQ1:** Do core developers change LOC more than peripheral developers?

• **RQ2:** Do core developers increase LOC more than peripheral developers?

• **RQ3:** Do core developers change TD more than peripheral developers?

• **RQ4:** Do peripheral developers introduce TD more than core developers?
What is Technical Debt

• In 1992, Ward Cunningham described technical debt as writing immature or “not quite right” code in order to ship a new product to market faster.

• Technical Debt consists of:
  – Principle: measures the cost or effort for eliminating technical debt.
  – Interest: measures the extra cost or effort over some period of time incurred for NOT eliminating the technical debt.
Why Do We Take Technical Debt

- Release faster.
- Decrease current release cost.
- Gather more information.
- Delay decisions.
Technical Debt Consequences

- Increased time to delivery.
- Increased number of defects.
- Raising maintainability cost.
- Decreased customer satisfaction.
Methodology

Projects Selection

Static Analysis

Statistical Test

Results

Commits and Committer Data
Data Selection

• 38 Apache Software Foundation Systems.

• Selection Criteria:
  – There are more than one official releases.
  – The programming language is Java.
  – The Git repository is active and has an update in 2017.
  – The system has less than 3,000 commits.
  – The system has one module that contains most of the source code.
Technical Debt Calculation

- **Vulnerabilities**: 13,083
  - **New Vulnerabilities**: 1
  - **Security Rating**: E
  - **Security Remediation Effort**: 354d
  - **Security Remediation Effort on New Code**: 10min

- **Code Smells**: 327,178
  - **New Code Smells**: 232
  - **Maintainability Rating**: B
  - **Technical Debt**: 10342d
  - **Added Technical Debt**: 6d
  - **Technical Debt Ratio**: 6.6%
  - **Technical Debt Ratio on New Code**: 3.1%
  - **Effort to Reach Maintainability Rating A**: 2520d

- **Coverage**
  - Uncovered Lines on New Code: 0
  - Uncovered Conditions on New Code: 0
  - Lines to Cover on New Code: 0

[https://sonarqube.com](https://sonarqube.com)
Technical Debt Calculation

Debt (in man days) =

\[\text{cost\_to\_fix\_duplications} + \text{cost\_to\_fix\_violations} + \text{cost\_to\_comment\_public\_API} + \text{cost\_to\_fix\_uncovered\_complexity} + \text{cost\_to\_bring\_complexity\_below\_threshold}\]
Contributors Categorization Perspectives

- Social perspective: study developers communication and collaboration.
- Technical perspective: study system artifacts.
Contributors Categorization Approaches

- Count-based techniques.
- The most used metrics are:
  - Commit count.
  - LOC count.
  - Bug tracking messages count.
  - Mail count.
Statistical Test

- K-means clustering.
- We clustered the developers in each system based on the total number of commits they authored.
Statistical Test

• Fisher’s Exact Test.
  – Determines whether the differences in the values between the two categories are statically significant or random.
  – Significance level is 0.05.
  – Confidence level is 95%.
**Results**

<table>
<thead>
<tr>
<th>Category</th>
<th>Significant</th>
<th>Not Significant</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC Change Difference</td>
<td>34.21%</td>
<td>65.79%</td>
</tr>
<tr>
<td>LOC Increase Difference</td>
<td>26.32%</td>
<td>73.68%</td>
</tr>
<tr>
<td>TD Change Difference</td>
<td>34.21%</td>
<td>65.79%</td>
</tr>
<tr>
<td>TD Increase Difference</td>
<td>21.05%</td>
<td>78.95%</td>
</tr>
</tbody>
</table>
Results

• System age.
• System size.
• System number of releases.
Systems Break Down based on Age

- Old: 65.79%
- Young: 34.21%
Systems Break Down based on Size

- Big: 84.21%
- Small: 15.79%
Systems Break Down based on Number of Releases

- Big: 73.68%
- Small: 26.32%
## Results

<table>
<thead>
<tr>
<th></th>
<th>AGE Old vs. Young</th>
<th>Releases Large vs. Small</th>
<th>Size Big vs. Small</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC change</td>
<td>.148</td>
<td>.594</td>
<td>.076</td>
</tr>
<tr>
<td>LOC increase</td>
<td>.262</td>
<td>.278</td>
<td>.642</td>
</tr>
<tr>
<td>TD change</td>
<td>.473</td>
<td>.594</td>
<td>.644</td>
</tr>
<tr>
<td>TD increase</td>
<td>.221</td>
<td>.023</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Implications

• Peripheral contributors:
  – Overcome their fear of joining new OSS systems.
  – Contribute regardless of their inability for long time commitment.
  – Increase the number of new volunteers and insure a constant influx of new volunteers.
Implications

• CS educators:
  – The learners of today are the practitioners of tomorrow.
  – Win-win for the OSS community and the students.
  – Encourage educators and students.
Implications

• Project owners:
  – Encourage project owners to accept and welcome new-comer
Implications

• Researchers:
  – What other factors relate to the differences.
  – Pair programming matching.