Quantifying the Programming Process to Help Teach Incremental Development

Ayaan M. Kazerouni, SIGCSE Student Research Competition
Computer Science, Virginia Tech
February 24, 2018
The Problem

The programming process is complex and is not thoroughly assessed.

Post-mortem Measures

- Correctness
- Code style
- Code coverage

Web-CAT [1], AutoLab [2], CI
The Problem

The programming process is complex and is not thoroughly assessed.

Incremental Development

- Time management
- Effective software testing

Hackystat [3], Marmoset [4], NPSM [5], Error Quotient [6], Watwin [7]

Post-mortem Measures

- Correctness
- Code style
- Code coverage

Web-CAT [1], AutoLab [2], CI
DevEventTracker
Modelling Incremental Development

Writing, testing, and debugging small chunks of code at a time.

- Working Early and Often
- Software Testing Practices
Early/Often Index

A quantification of procrastination.

- Early/Often Index: The average number of days until the deadline, across all edits.

If $E$ is the set of all edits made, then

$$
earlyOften(E) = \frac{\sum_{e \in E} size(e) \times daysToDeadline(e)}{\sum_{e \in E} size(e)}
$$
Early/Often Index

Better Early/Often scores were related to more semantically correct programs and earlier project completion times.

<table>
<thead>
<tr>
<th>Project Outcome</th>
<th>F</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>16.2</td>
<td>&lt; 0.0001 *</td>
</tr>
<tr>
<td>Time of completion</td>
<td>55.9</td>
<td>&lt; 0.0001 *</td>
</tr>
</tbody>
</table>

Mixed Model: John Doe did better on projects when he had a higher Early/Often score, than when he had a lower one.
Incremental Test Writing *

Quantifying Solution-Test Coevolution.

- For a given work session:
  - $TE$ is the set of test edits
  - $SE$ is the set of solution edits

\[
STC = \text{Avg}(\frac{TE}{SE + TE}) \text{ across all work sessions}
\]

- Data suggests a relationship with project correctness ($F = 7.2$, $p = 0.007^*$)
Visual Feedback and Analysis

Fig. 1: Good Test Writing

Fig. 2: Poor Test Writing
Future Work

- Improve assessments of software testing
- Design and implement interventions
  - Regular, adaptive emails
  - Learning dashboard
- Assess their impact
- Iterate
Contributions

- Process-based assessments should benefit students working on large and complex programming projects
- Scope for adoption in the software engineering community at large
References


Thank you

Questions?