Quantifying the Programming Process to Help Teach Incremental Development

Ayaan M. Kazerouni

Department of Computer Science, Virginia Tech

VIRGINIA TECH

TEACHING INCREMENTAL DEVELOPMENT TO EXPERIENCED PROGRAMMERS

HYPOTHESES
- Good programming process is important for successful project outcomes
- Continuous, adaptive feedback about the process can help developers adhere to good process

PROBLEM
- Research on student programmers tends to be aimed at novices
- Assessment practices tend to focus on post-hoc semantic and static checks

GOAL
- A process for **continuous assessment** and **self-correction** while programming

THE PROGRAMMING PROCESS AS CLICKSTREAM DATA

DEV-EVENT-TRACKER [1] – Observing a 30-hour programming process carried out at home.

- We instrumented the Eclipse IDE to capture developer activity
- Collects **Edit**, **Launch**, and **Debug** events, along with **program snapshots** using Git
- Now produces approximately **6M events each semester**

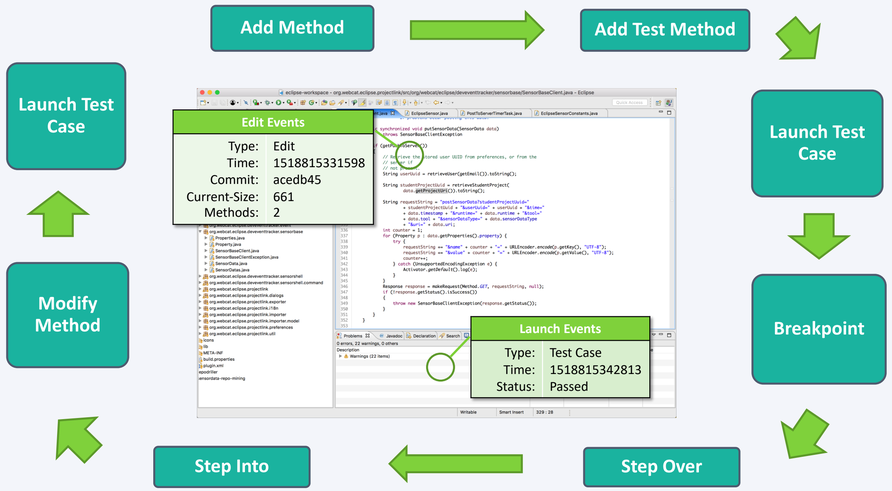

Fig. 1: High-resolution event-stream captured from Eclipse.

QUANTIFYING INCREMENTAL DEVELOPMENT

Reducing a large event-stream into actionable feedback. [2]

Quantifying Procrastination

Early/Often Index: How many days before the deadline do you write code, on average?

If E is the vector of all edits on the project, then

$$earlyOften(E) = \frac{\sum_{e \in E} size(e) * daysToDeadline(e)}{\sum_{e \in E} size(e)}$$

Positively related to **project correctness** ($p < 0.001$) and **earlier completion times** ($p < 0.001$).

Quantifying Incremental Test Writing

Solution/Test Coevolution: How much test code do you write each time you sit down to code?

If SE and TE are edits to solution code and test code:

$$coevolution = Avg(\frac{|TE|}{|SE| + |TE|})$$

across all work sessions.

Positively related to **project correctness** ($p = 0.007$).

The metrics described above could be applied in educational, industrial, and open-source contexts.

CLOSING THE LOOP

Once we have modelled aspects of incremental development, the next step is to give programmers feedback about their process.

What is the most effective method? We are set up to experiment with a variety of approaches.
- Regular, adaptive emails – known to help reduce procrastination [4]
- Learning dashboard

DevEventTracker data facilitates **visual feedback and analysis.**

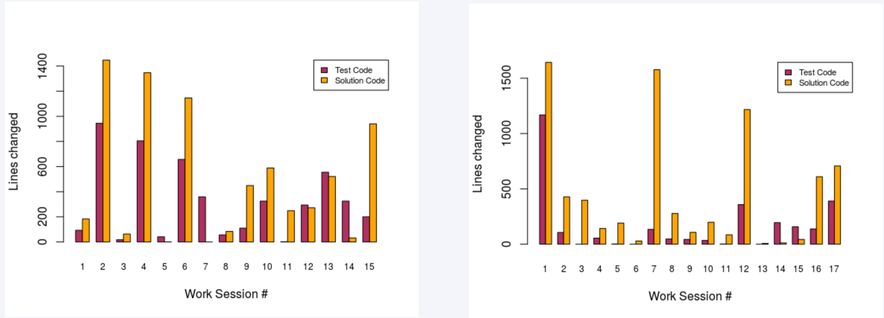

Fig. 2: The work of two students on the same project.

KEY CONTRIBUTIONS

- A suite of vetted programming-process measurements
- Measures are applicable in a variety of programming contexts
- An end-to-end pipeline that takes in a developer's event-stream and responds with intelligent and time feedback about their programming process

REFERENCES AND ACKNOWLEDGMENT

[1] Ayaan M. Kazerouni, Stephen H. Edwards, T. Simin Hall, and Clifford A. Shaffer.. DevEventTracker: Tracking Development Events to Assess Incremental Development and Procrastination.
[2] Ayaan M. Kazerouni, Stephen H. Edwards, and Clifford A. Shaffer. 2017. Quantifying Incremental Development Practices and Their Relationship to Procrastination.
[3] Stephen H. Edwards and Manuel A. Perez-Quinones. Web-CAT: automatically grading programming assignments.
[4] Joshua Martin, Stephen H. Edwards, and Clifford A. Shaffer. The effects of procrastination interventions on programming project success.

This work was funded in part by NSF grants DUE-1245334 and DUE-1625425.