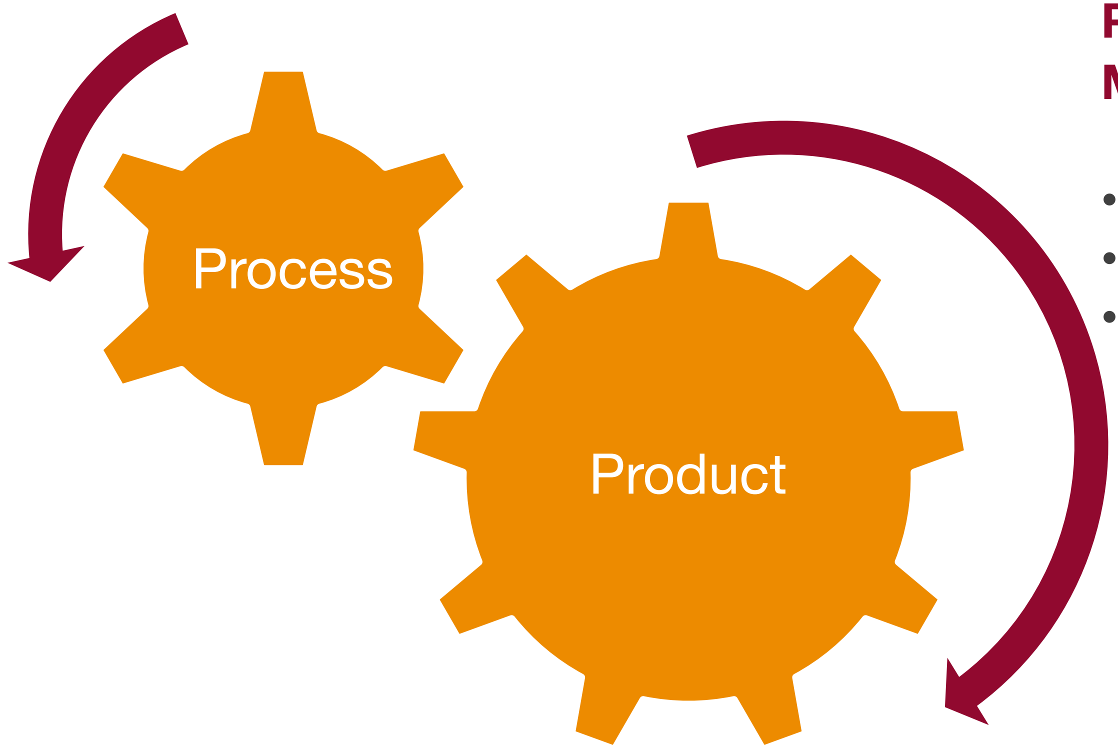
Quantifying the Programming Process to Help Teach Incremental Development

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

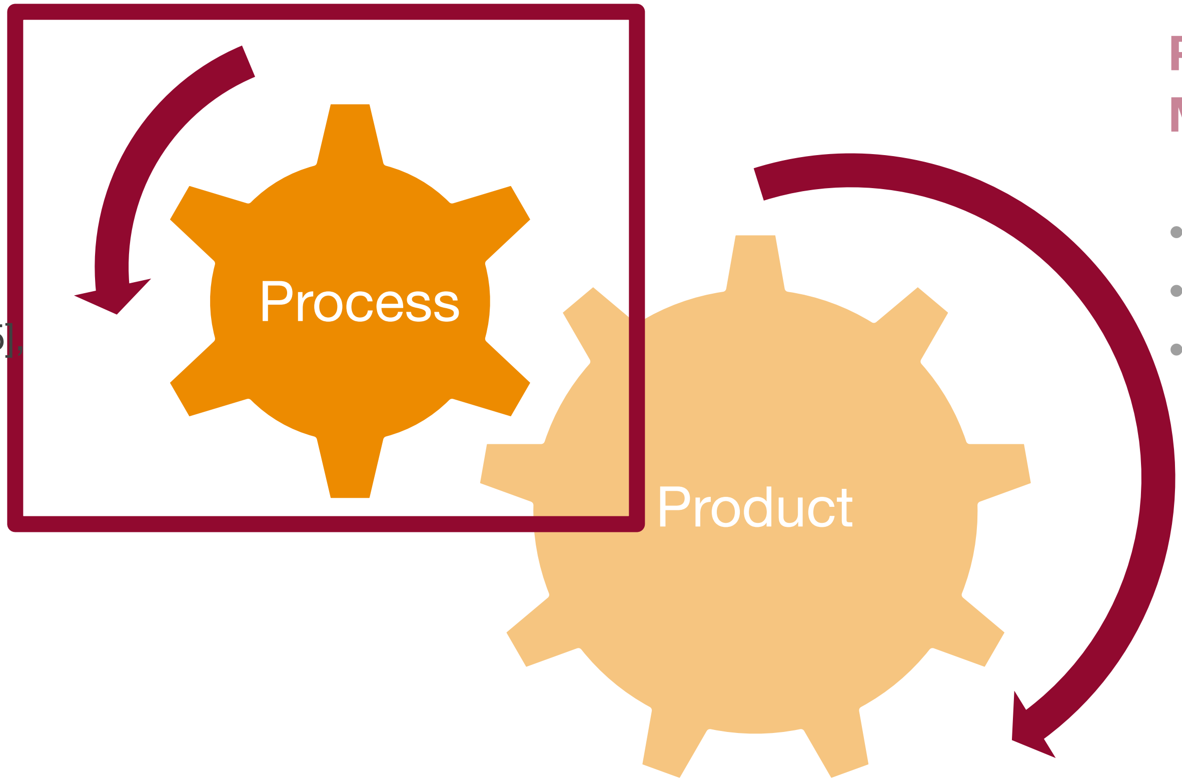
The Problem

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]

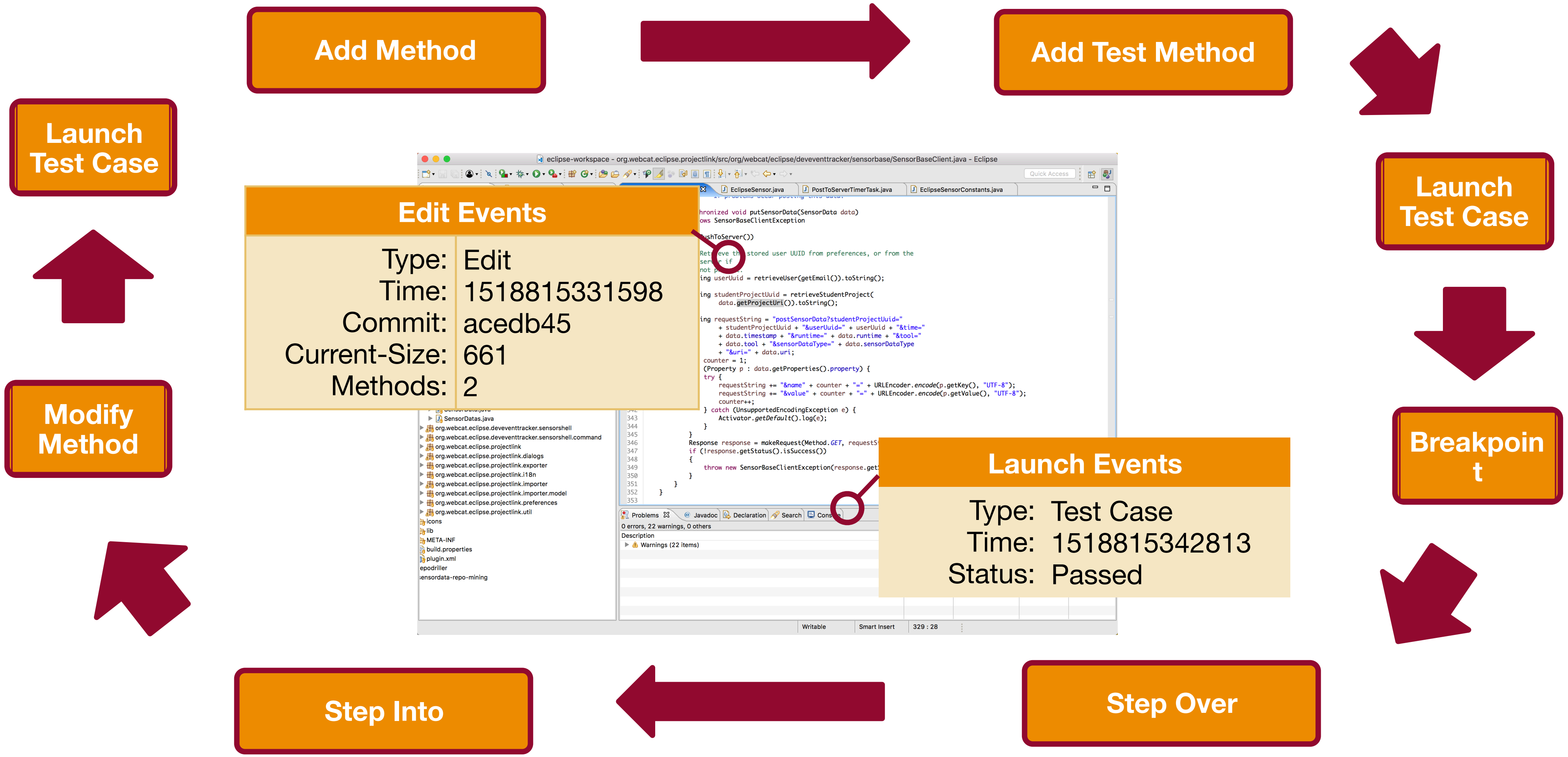


Post-mortem Measures

- Correctness
- Code style
- Code coverage

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

DevEventTracker



eclipse-workspace - org.webcat.eclipse.projectlink/src/org/webcat/eclipse/deveventtracker/sensorbase/SensorBaseClient.java - Eclipse

EclipseSensor.java | PostToServerTimerTask.java | EclipseSensorConstants.java

```
hronized void putSensorData(SensorData data)
ows SensorBaseClientException

ushToServer())

Ret eve th stored user UUID from preferences, or from the
ser r if
not p
ing userUuid = retrieveUser(getEmail()).toString();

ing studentProjectUuid = retrieveStudentProject(
      data.getProjectUri()).toString();

ing requestString = "postSensorData?studentProjectUuid="
      + studentProjectUuid + "&userUuid=" + userUuid + "&time="
      + data.timestamp + "&runtime=" + data.runtime + "&tool="
      + data.tool + "&sensorDataType=" + data.sensorDataType
      + "&uri=" + data.uri;
 counter = 1;
 (Property p : data.getProperties().property) {
 try {
      requestString += "&name" + counter + "=" + URLEncoder.encode(p.getKey(), "UTF-8");
      requestString += "&value" + counter + "=" + URLEncoder.encode(p.getValue(), "UTF-8");
      counter++;
 } catch (UnsupportedEncodingException e) {
      Activator.getDefault().log(e);
 }
 }
 Response response = makeRequest(Method.GET, requestS
 if (!response.getStatus().isSuccess())
 {
     throw new SensorBaseClientException(response.get
 }
 }
}
```

SensorDatas.java
org.webcat.eclipse.deveventtracker.sensorshell
org.webcat.eclipse.deveventtracker.sensorshell.command
org.webcat.eclipse.projectlink
org.webcat.eclipse.projectlink.dialogs
org.webcat.eclipse.projectlink.exporter
org.webcat.eclipse.projectlink.i18n
org.webcat.eclipse.projectlink.importer
org.webcat.eclipse.projectlink.importer.model
org.webcat.eclipse.projectlink.preferences
org.webcat.eclipse.projectlink.util
icons
lib
META-INF
build.properties
plugin.xml
epodriller
ensordata-repo-mining

Problems | Javadoc | Declaration | Search | Cons e

0 errors, 22 warnings, 0 others

Description

Warnings (22 items)

Writable | Smart Insert | 329 : 28

Edit Events

Type:	Edit
Time:	1518815331598
Commit:	acedb45
Current-Size:	661
Methods:	2

Launch Events

Type:	Test Case
Time:	1518815342813
Status:	Passed

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

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

Early/Often Index

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

Project Outcome	F	p-value
Correctness	16.2	**< 0.0001 ***
Time of completion	55.9	**< 0.0001 ***

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*

$$STC = Avg\left(\frac{TE}{SE + TE}\right) \text{ across all work sessions}$$

- Data suggests a relationship with project correctness (F = 7.2, **p = 0.007***)

Visual Feedback and Analysis

Fig. 1: Good Test Writing

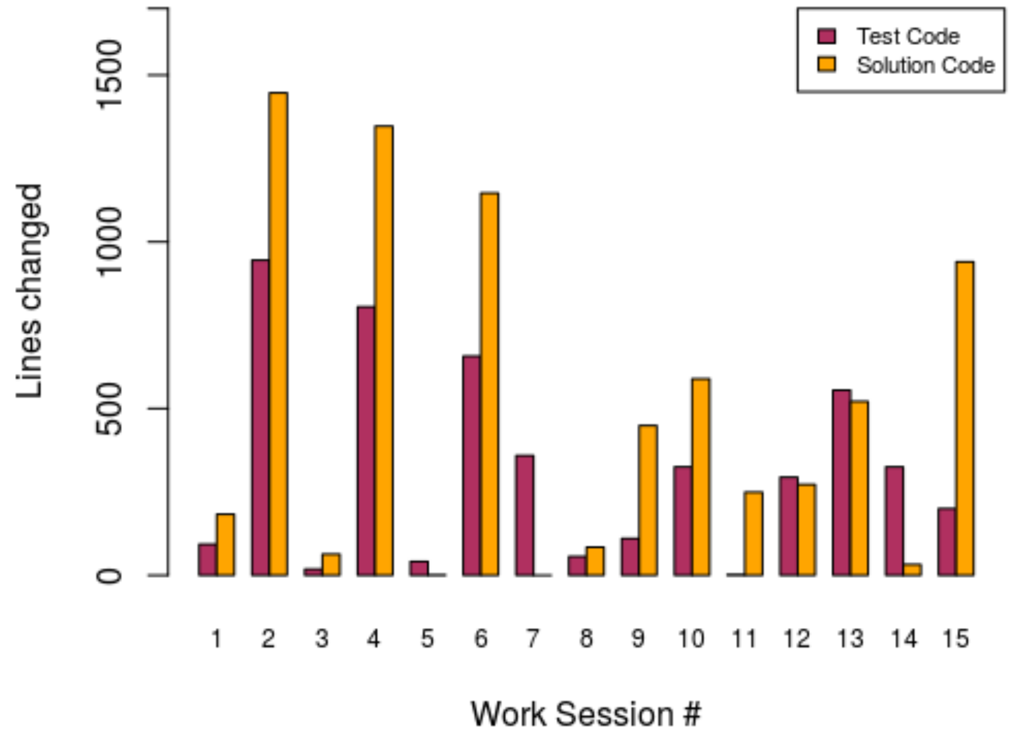
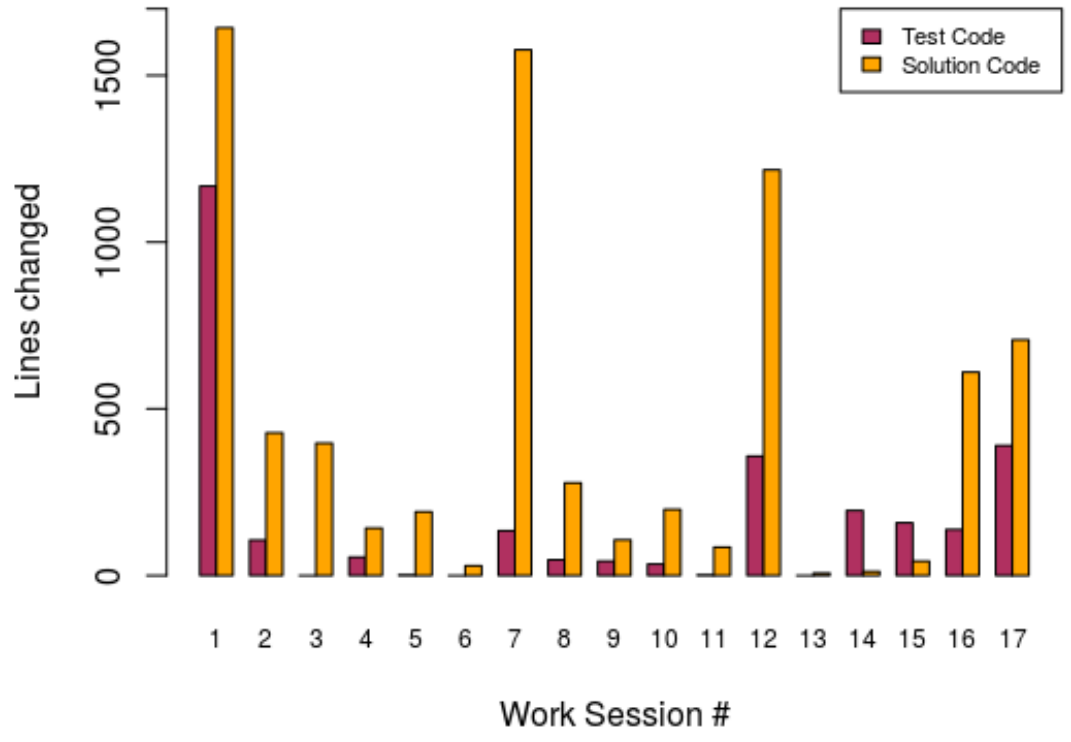


Fig. 2: Poor Test Writing



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

[1] Stephen H. Edwards and Manuel A. Perez-Quinones. 2008. Web-CAT: automatically grading programming assignments. SIGCSE Bull. 40, 3 (June 2008), 328-328. DOI=http://dx.doi.org/10.1145/1597849.1384371

[2] AutoLab: http://autolab.github.io/2015/03/autolab-autograding-for-all/

[3] Philip M. Johnson, Hongbing Kou, Joy Agustin, Christopher Chan, Carleton Moore, Jitender Miglani, Shenyan Zhen, and William E. J. Doane. 2003. Beyond the Personal Software Process: metrics collection and analysis for the differently disciplined. In Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, USA, 641-646.

[4] Jaime Spacco, William Pugh, Nat Ayewah, and David Hovemeyer. 2006. The Marmoset project: an automated snapshot, submission, and testing system. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA '06). ACM, New York, NY, USA, 669-670. DOI: https://doi.org/10.1145/1176617.1176665

[5] Adam S. Carter, Christopher D. Hundhausen, and Olusola Adesope. 2015. The Normalized Programming State Model: Predicting Student Performance in Computing Courses Based on Programming Behavior. In Proceedings of the eleventh annual International Conference on International Computing Education Research (ICER '15). ACM, New York, NY, USA, 141-150. DOI: http://dx.doi.org/10.1145/2787622.2787710

[6] Matthew C. Jadud. 2006. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the second international workshop on Computing education research* (ICER '06). ACM, New York, NY, USA, 73-84. DOI=http://dx.doi.org/10.1145/1151588.1151600

[7] Christopher Watson, Frederick W. B. Li, and Jamie L. Godwin. 2013. Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior. In *Proceedings of the 2013 IEEE 13th International Conference on Advanced Learning Technologies* (ICALT '13). IEEE Computer Society, Washington, DC, USA, 319-323. DOI=http://dx.doi.org/10.1109/ICALT.2013.99

Thank you

Questions?