University of Pittsburgh

dB-SERC Mentor-Mentee Evidence-Based Teaching Award

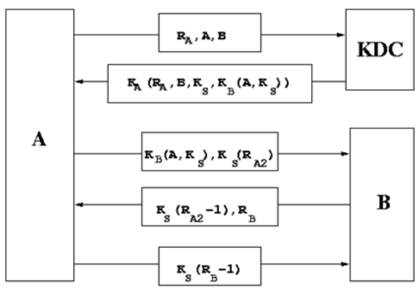
Adam J. Lee, Associate Professor
William C. Garrison III, PhD Candidate

Department of Computer Science

Designing and building secure systems is hard!

"The black art of programming Satan's computer" [1]

Longstanding designs and implementations have been proven insecure:



<u>Needham-Schroeder</u>
Man-in-the-middle discovered
after 17 years in use

<u>OpenSSL</u>
Heartbleed vulnerability discovered
2 years after introduced

Formal verification is very difficult, even for experienced software engineers!

CS 1653 teaches security engineering with a focus on a semester-long group project

CS 1653: Applied Cryptography and Network Security

Lectures present algorithms and protocols, students apply these in an interleaved semester project

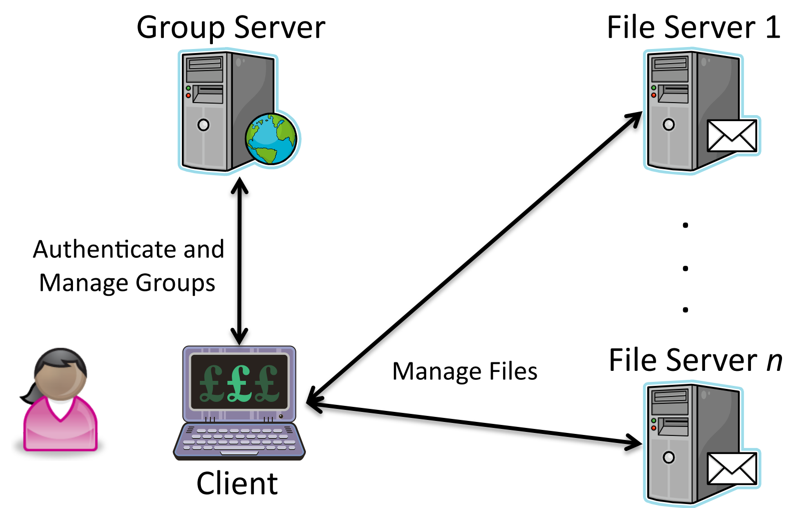
In this project, students must:

- Work in groups for the full semester
- Propose their own solutions to adversarial tasks
- Develop, maintain, and extend a non-trivial code base (~5k lines)

Requires both design and coding!

A summary of the CS 1653 semester project

Students develop a secure distributed file-sharing system



Five phases, each considering additional security threats

Students meet with instructor to propose solutions, demo with TA after submission

Even the best students run into problems with this project…

The most common problems:

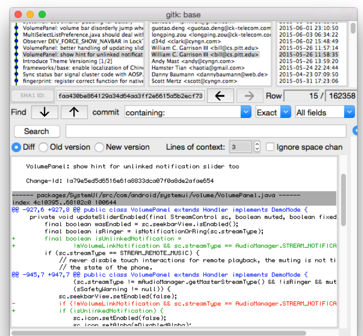
- Uneven distribution of work
- Lack of communication among group members
- Procrastination, submitting last-minute
- Juggling design and code
- Rushing through code
- Combining code written by multiple group members
- Design and code not matching, evolving out-of-sync

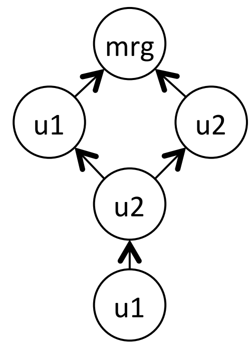
Can using a version control system help mitigate these issues?

Why develop code using a version control system?

In a VCS, any change to a code base is called a commit

The VCS maintains a history of previous commits with descriptions





A commit is relative, to ease the merging of work from multiple users

Commit logs are *time series* describing development at a fine granularity, and have been used for a variety of experiments:

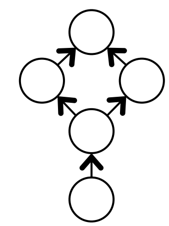
- Adoption of new APIs does not keep pace with their development [2]
- Programming language design has a modest effect on code quality [3]
- Gender and tenure diversity are positive predictors of productivity [4]
- Functions with asserts have significantly fewer defects [5]
- Asking questions on Q&A sites catalyzes development (and vice versa) [6]

How can using a VCS improve the CS 1653 project for our students?

Stay organized: students review their changes when committing

Commit logs improve communication: see what your groupmates have completed

Much simpler merging when working simultaneously: no more emailing code and manually combining!



Continuous submission: work until the deadline, committing as you go

What about using analytics?

VCS analytics to improve the course project

High-level goal: improve group performance… how?

During the semester

- Use analytics to detect problems in groups
- Allow the instructor to intervene as needed

Between semesters

- Use analytics to discover what makes some groups more successful
- Adjust course to encourage behavior seen in strong groups

We have collected and begun to analyze the first round of statistics from the Spring 2015 offering of CS 1653

- 33 students, 14 groups, 4 project phases
- This summer: identify important analytics
- Next offering: changes based on these results

Basic analytics that may correlate with group performance

Group analytics

- Number of commits
- Number of lines changed (can include multiple changes to a line)
- Number of surviving changes (excludes multiple changes, reverts)

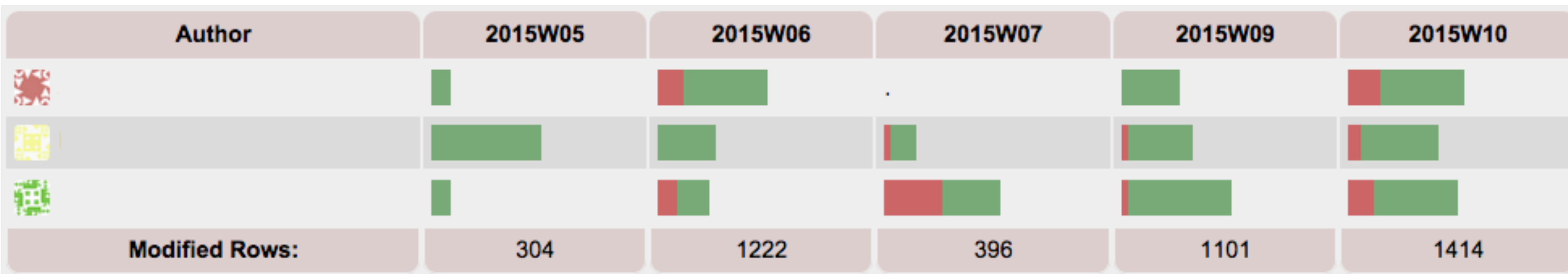
Why?

- Larger commits may indicate tackling too much at once
- Low stability (surviving ÷ total changes) may indicate struggles

Per-student variants on above analytics: distribution of labor

Author ∨	Rows	Stability	Age	% in comments
	887	81.4	2.3	4.06
	864	71.7	2.6	13.31
	702	58.7	1.4	11.82

Temporal analytics show changes in workflow



Break each phase of the project into weeks, measure changes per week (overall and per student)

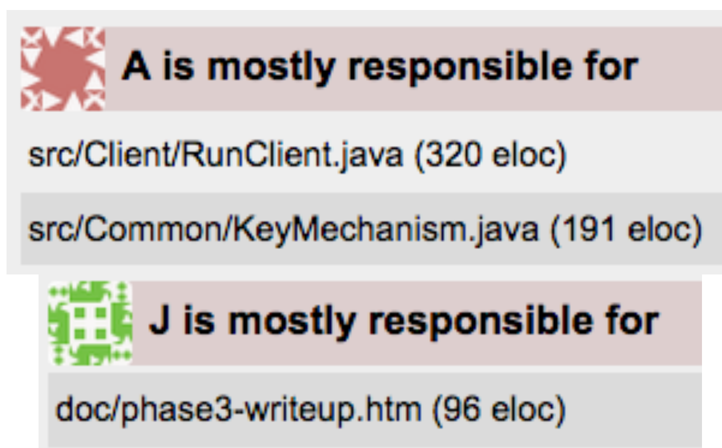
Why?

- Sharp increases may indicate procrastination
- Working around the same time may be better than “trading off”

Relationships between code and documentation changes

Are all group members working on both code and documentation, or are there “writers” and “coders”?

Do the same group members code and document the same assignment tasks?



Comments and commit messages: more than good practice?

Students are encouraged to thoroughly comment their code

Comments and good commit messages may also be indicative of how well a group communicates

Author ∨	Rows	Stability	Age	% in comments
	887	81.4	2.3	4.06
	864	71.7	2.6	13.31
	702	58.7	1.4	11.82

Analytics, overview

	In code		In Documentation	
Commits	*Overall*	*Per student*	*Overall*	*Per Student*
Commits, temporal	*Overall*	*Per student*	*Overall*	*Per Student*
Changes	*Overall*	*Per student*	*Overall*	*Per Student*
Changes, temporal	*Overall*	*Per student*	*Overall*	*Per Student*
Surviving changes	*Overall*	*Per student*	*Overall*	*Per Student*

Commit length	*Overall*	*Per student*
Code comments	*Overall*	*Per student*

How can we evaluate whether a group works well together?

Most obvious is grade. But is it the best?

- Coarse granularity: one grade for full project
- Struggling groups can succeed in the end
- Poor group dynamics may not cause poor grade
- Why not just ask?

Project P3 Group Assessment

For these questions, consider only how your group worked toward Proj

Indicate the extent to which you agree with each of the following stater

My groupmate(s) and I worked well together. *

1 2 3 4 5

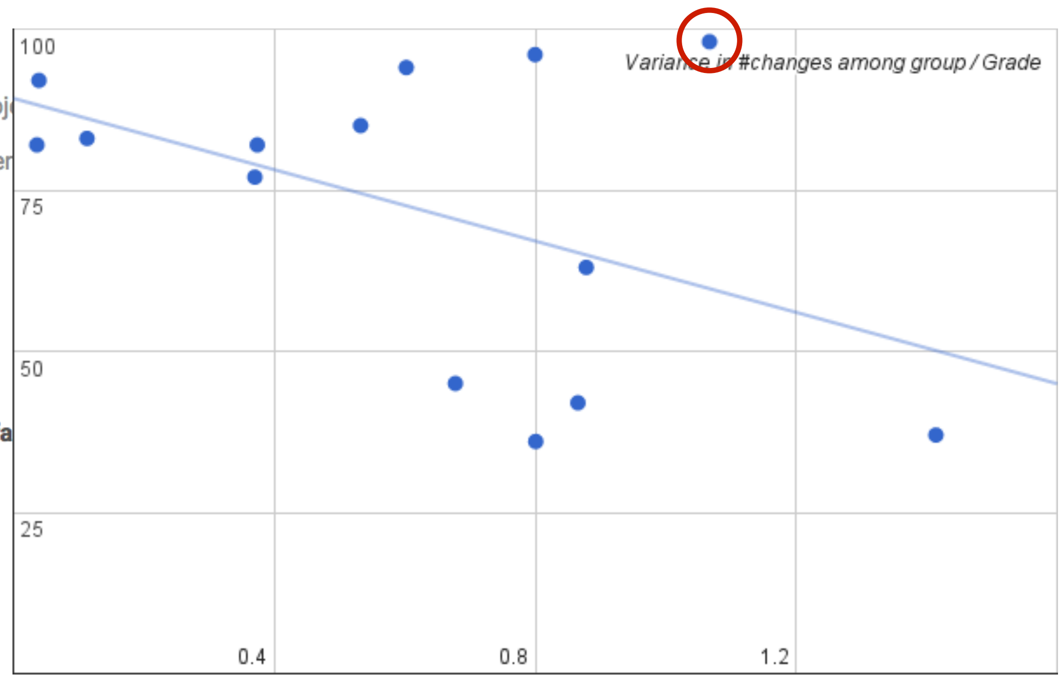
Strongly disagree ○ ○ ○ ○ ○ Strongly agree

My groupmate(s) and I divided the work of this phase of the project fa

1 2 3 4 5

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

My groupmate(s) and I communicated effectively. *

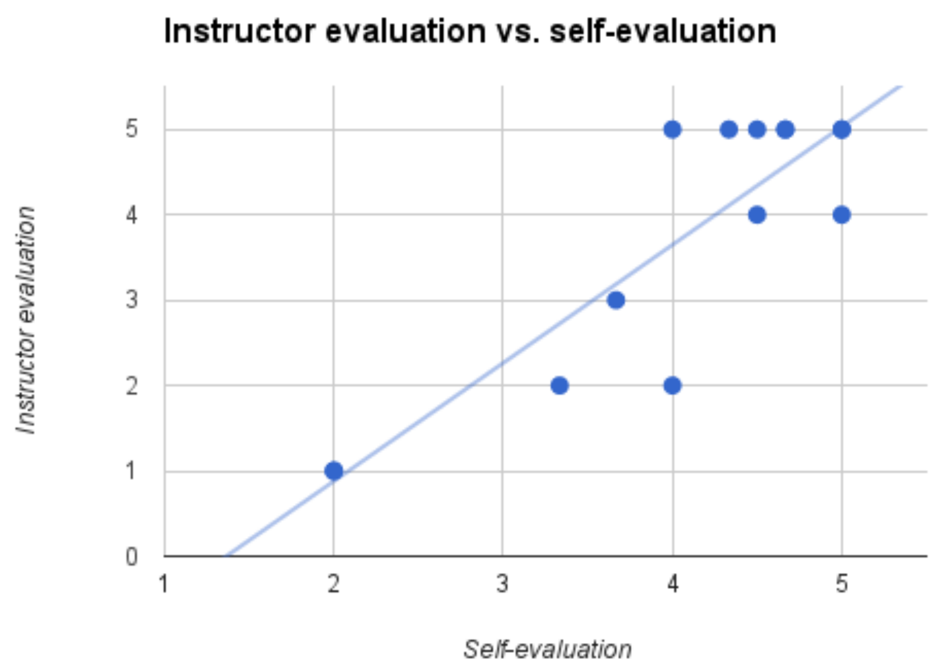


Meetings and demos provide opportunity for subjective evaluation

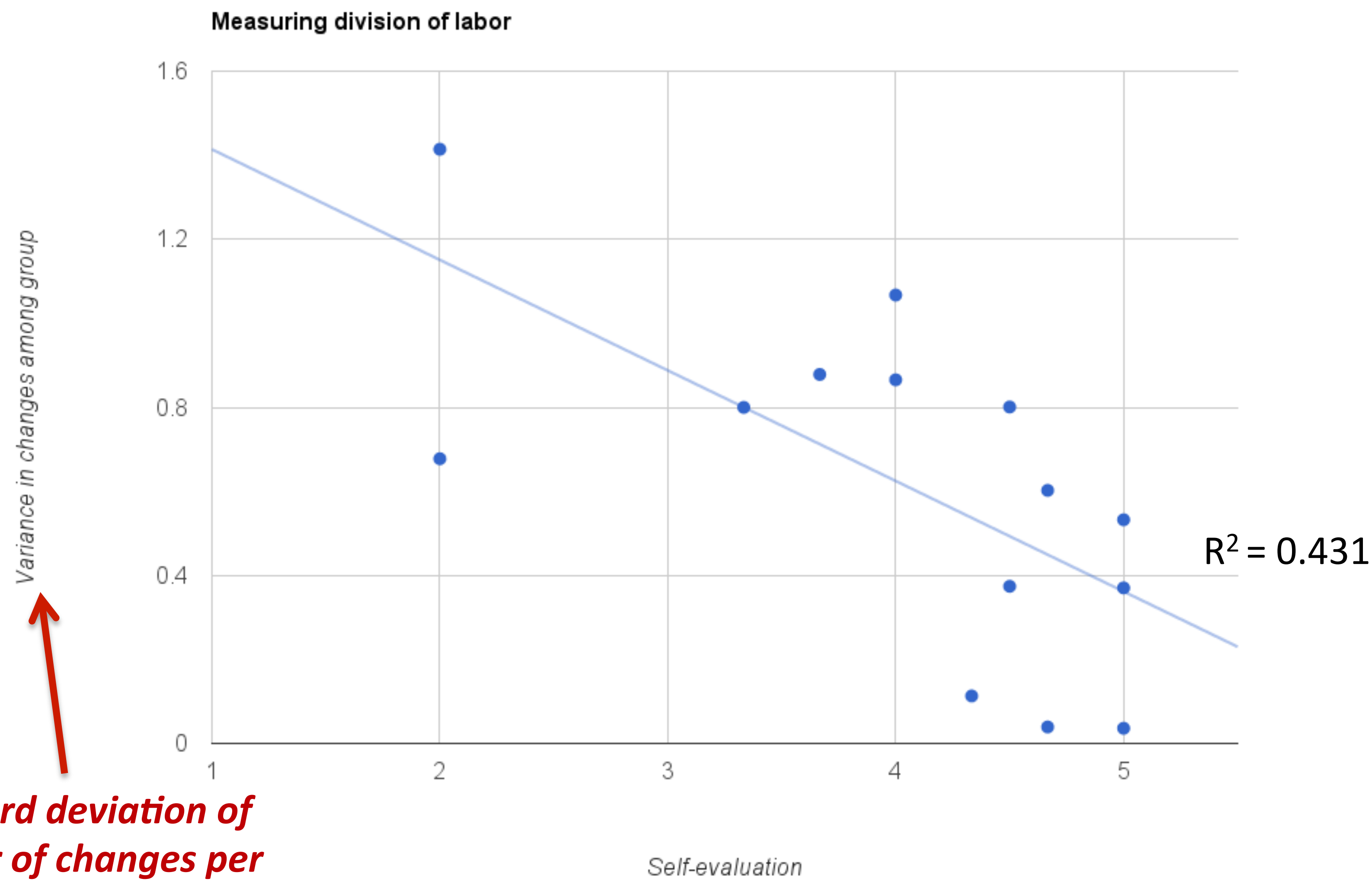
As mentioned, course staff meets with groups at each phase:

- Designs must be approved in meeting with instructor
- Code must be demoed in meeting with TA

Instructor evaluation correlates strongly with self-evaluation



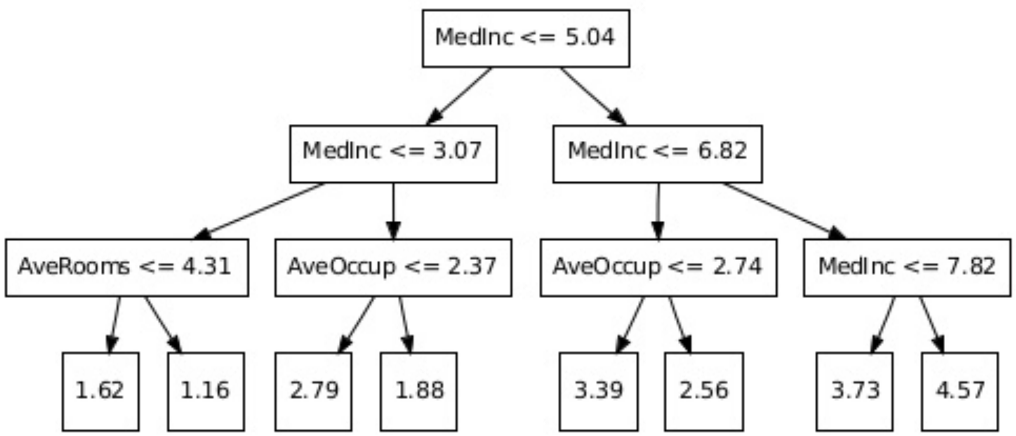
Analysis techniques: manual combination of features?



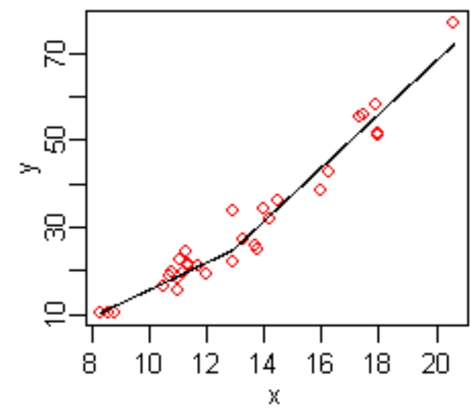
Standard deviation of number of changes per group member

More advanced analysis techniques

Since we see at least moderate predictive value in a variety of features, we will also use multivariable regression techniques



Regression trees



Regression splines

Possible outcomes of our study

As mentioned, early warnings can allow instructor to intervene

Changes can be made to the course depending on the variables that indicate high degree of group success

- Each student working on many assignment tasks → encourage collaboration across tasks by targeting demo questions
- High activity early → add additional checkpoints between deadlines
- More comments → require turned-in Javadoc generated from comments
- Same student writing code and documenting the same feature → increase the detail of the write-up
- Even split of changes → discuss statistics in demo, ask for justification if uneven

Questions? Thank you!

References:

1. Ross J. Anderson and Roger M. Needham, "Programming Satan's Computer," In *Computer Science Today: Recent Trends and Developments*, 1995.
2. Tyler McDonnell, Baishakhi Ray, Miryung Kim: An Empirical Study of API Stability and Adoption in the Android Ecosystem. ICSM 2013:70-79
3. Baishakhi Ray, Daryl Posnett, Vladimir Filkov, Premkumar T. Devanbu: A large scale study of programming languages and code quality in github. SIGSOFT FSE 2014:155-165
4. Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G. J. van den Brand, Alexander Serebrenik, Premkumar T. Devanbu, Vladimir Filkov: Gender and Tenure Diversity in GitHub Teams. CHI 2015:3789-3798
5. Casalnuovo Casey, Devanbu Prem, Oliveira Abilio, Filkov Vladimir, and Baishakhi Ray: Assert Use in GitHub Projects. ICSE 2015
6. Bogdan Vasilescu, Vladimir Filkov, Alexander Serebrenik: StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge. SocialCom 2013:188-195