3-2020

It Seemed Like a Good Idea at the Time (Hindsight is 2020)

James Huggins

Dan Garcia

Kevin Lin

Raja Sooriamurthi

Leo Ureel II

See next page for additional authors

Authors

James Huggins, Dan Garcia, Kevin Lin, Raja Sooriamurthi, Leo Ureel II, and Ursula Wolz

It Seemed Like a Good Idea at the Time

(Hindsight is 2020)

Dan Garcia (co-Moderator)
EECS
UC Berkeley
Berkeley CA USA
ddgarcia@berkeley.edu

Jim Huggins (co-Moderator)
Computer Science
Kettering University
Flint MI USA
jhuggins@kettering.edu

Kevin Lin
Computer Science & Engineering
University of Washington
Seattle WA USA
kevinl@cs.uw.edu

Raja Sooriamurthi
Information Systems
Carnegie Mellon University
Pittsburgh PA USA
raja@cmu.edu

Leo Ureel II
Computer Science
Michigan Tech. University
Houghton MI USA
ureel@mtu.edu

Ursula Wolz
RiverSound Solutions
Lang, The New School
Montclair, NJ/NYC NY, USA
ursula.wolz@gmail.com

ABSTRACT

Conference presentations usually focus on successful innovations: new ideas that yield significant improvements to current practice. Yet educators know that we often learn more from failure than from success. In this panel, we present four case studies of "good ideas" for improving CS education that resulted in failures. Each contributor will describe their "good idea", the failure that resulted, and wider lessons for the CS community.

CCS CONCEPTS

• Social and professional topics→Computing education

KEYWORDS

Experience report; learning from failure; comp.risks

1 SUMMARY

Charles Kettering reportedly quipped: "99% of success is built on failure". Yet, those failures rarely see the light of day, as publications naturally focus on successful innovations rather than the many failures that preceded them. The academic community is poorer as a result, as we are all left to re-create the same failures independently, rather than learning from one another.

In this panel, we offer an opportunity to "celebrate failure", by presenting four separate case studies of computing education initiatives that "seemed like a good idea at the time", but ended up being spectacular failures. The presenters will discuss their "good ideas", the disappointing results, and (most importantly) the lessons learned! Our goal is to foster a supportive community where failure is celebrated rather than criticized. We hope to laugh and learn together from these experience reports.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03.
https://doi.org/10.1145/3328778.3366974

This session is modeled after several similar sessions (e.g. [1,2,3]) offered at SIGCSE symposia between 2007-2011. We gratefully acknowledge the contributions of Jonas Boustedt, Robert McCartney, and Josh Tenenberg for creating the format of this session, and allowing us to continue in their footsteps.

2 SESSION STRUCTURE

We will open with five minutes to describe this unique format, and its history. Each panelist will have ten minutes to share their respective positions. Following each presentation, the moderators will facilitate audience questions for the remaining thirty minutes.

3 KEVIN LIN

Kevin Lin is a Lecturer in the Paul G. Allen School of Computer Science & Engineering at the University of Washington and previously coordinated autograders for CS courses at UC Berkeley.

Autograders provide instant feedback on student work, but they can also harm learning if students grow dependent on autograder feedback to solve problems. The resulting autograder-driven development cycle occurs when students make minor adjustments to their code seemingly at random, submit code to the autograder, and repeat until their program passes all of the given tests. Anecdotal evidence from other instructors suggested that rate-limiting student submissions on the server-side autograder to 3 or 4 submissions per hour was an effective intervention. We hypothesized that introducing a 3-5 minute "cooldown timer" on the client-side autograder could mitigate student over-reliance on autograder feedback by requiring students to spend more time independently debugging, planning, and evaluating their changes before receiving autograder feedback. However, a lack of user-testing, expectations management, and course integration led to students and course staff alike perceiving the cooldown timer as an inconvenience more than a learning opportunity.

4 RAJA SOORIAMURTHI

Raja Sooriamurthi is a Teaching Professor at Carnegie Mellon University and regularly teaches their system development course.

In an academic environment, we often develop code from scratch, whereas in industry, we most often make changes to an existing code-base (greenfield vs. brownfield development). The latter requires a person to understand existing work and then extend it.

To model this environment in the classroom, I created a term project in my system development course with the following perspective. Students would implement one user-story all the way from the UI to the database. Then they would swap their entire github repos (code-base + documentation) with another assigned team. The second team would then build upon the earlier work and implement a second user story.

Unfortunately, this experience went down in flames. The quality of the initial work varied considerably from team to team. Some teams were especially upset when they did good work in phase-1 but got subpar work in phase-2 from another team. Overall, the class was quite upset resulting in one of my worst course evaluations ever.

5 LEO UREEL II

Leo Ureel II is a Lecturer at Michigan Technological University and regularly teaches introductory programming courses.

We offer a two-semester introductory programming sequence in Java that assumes no prior programming experience. Alongside this "standard" sequence, we offer an "accelerated" option for students with programming experience (in any language). The one-semester accelerated course covers the same material as the two-semester sequence. The intent is to provide an introductory course where novice programmers need not compete with experienced programmers, while students with programming experience learn at a pace both manageable and challenging.

Students were given an opportunity to take a placement exam to determine their preparedness for the accelerated course, either during the summer or during orientation week prior to the beginning of fall classes. During Fall 2015 all incoming students were expected to attend a session during orientation which discussed the different courses and then provided the students with the opportunity to take the exam. Yet, contrary to our expectations, many with programming experience enrolled in the standard introductory sequence [4]. A follow-up survey in spring 2016 revealed that roughly 30% of the computer science majors who responded had a year or more of formal programming coursework in high school [5]. A survey was conducted at the end of the Fall 2015 semester. Of the students with no previous programming experience, 63% indicated that the standard introductory programming sequence was too difficult for novice programmers. Simultaneously, 66% of the students with programming experience indicated the course was too easy. None of the students in the accelerated course indicated the course was too difficult. We also had not recognized that the enrollment in the accelerated introductory course was quite low. In 2013, the course had decreased in enrollment to only 14 students, when prior enrollments had trended toward approximately 35 students.

The conclusions were discouraging: our standard sequence was populated with novice programmers and a large number of experienced programmers, both unsatisfied for different reasons, and our advanced course was disappointingly underpopulated.

6 URSULA WOLZ

Ursula Wolz is a veteran educator who regularly taught advanced algorithms at several Universities.

In an advanced algorithms class, I used a collaborative assessment system where I did not grade homework. I subtly publicly shamed them into attempting the homework on time. There were 20 people in the class and homework was assigned once a week. The day that homework was due, we did a recursive class review:

- Pairs reviewed the homework and noted whether they agreed or disagreed on the answer.
- Then each pair met with another pair and compared.
- Then each group of four met with another group of four.

Issues, concerns, questions bubbled to the top, and we addressed them as a class. Sometimes I deferred discussion and prepped a lesson to really get at the problem. The first time I did this I was new to the institution and there were at least four truly outstanding, highly competitive students who did the homework passionately for its own sake. This worked really (REALLY) well. It helped that 2 of them were also consummate tutors. This class was so internally competitive that they were shamed into being prepared for class.

The next three times I did this (at the same and another institution) it failed in its original conception. Some of it was the caliber of the students, some of it was their prior preparation, some of it was the change in culture I witnessed over a recent four-year period. Students play the system, and they are overworked. However, I will keep trying, because when it works, it really works.

REFERENCES

[1] Jonas Boustedt, Robert McCartney, Josh Tenenberg, Titus Winters, Stephen Edwards, Briana B. Morrison, David R. Musicant, Ian Utting, and Carol Zander. 2007. It seemed like a good idea at the time. In Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07). ACM, New York, NY, USA, 346-347. DOI: https://doi.org/10.1145/1227310.1227432

[2] Jonas Boustedt, Robert McCartney, Josh Tenenberg, Scott D. Anderson, Caroline M. Eastman, Daniel D. Garcia, Paul V. Gestwicki, and Margaret S. Menzin. 2008. It seemed like a good idea at the time. In Proceedings of the 39th SIGCSE technical symposium on Computer science education (SIGCSE '08). ACM, New York, NY, USA, 528-529. DOI: https://doi.org/10.1145/1352135.1352311

[3] Jonas Boustedt, Robert McCartney, Josh Tenenberg, Stephen Cooper, Daniel D. Garcia, Michelle Friend Hutton, Nick Parlante, and Brad Richards. 2011. It seemed like a good idea at the time. In Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11). ACM, New York, NY, USA, 163-164. DOI: https://doi.org/10.1145/1953163.1953215

[4] Archer, G., Bohmann, L., Carter, A., Cischke, C., Ott, L. M., & Ureel, L. (2016, October). Understanding similarities and differences in students across first-year computing majors. In 2016 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.

[5] Archer, G., Bettin, B., Bohmann, L., Carter, A., Cischke, C., Ott, L. M., & Ureel, L. (2017, October). The impact of placement strategies on the success of students in introductory computer science. In 2017 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). IEEE.