It Seemed Like a Good Idea at the Time

(Special Session)

Jonas Boustedt
Univ. of Gävle, Sweden
jbt@hig.se

Robert McCartney
Univ. of Connecticut
robert@cse.uconn.edu

Josh Tenenberg
Univ. of Washington, Tacoma
jtenenbg@u.washington.edu

Titus Winters
Univ. of California, Riverside
titus@cs.ucr.edu

Stephen Edwards
Virginia Tech (VPI&SU)
edwards@cs.vt.edu

Briana B. Morrison
So. Polytechnic St. Univ.
bmorriso@spsu.edu

David R. Musicant
Carleton College
dmusican@carleton.edu

Ian Utting
Univ. of Kent, Canterbury UK
I.A.Utting@kent.ac.uk

Carol Zander
Univ. of Washington, Bothell
zander@u.washington.edu

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: self-assessment; K.4.2 [**Social Issues**]: employment

General Terms

Design, Experimentation, Human Factors, Measurement, Performance, Verification

Keywords

fiasco, disaster, breakdown, failure, humiliation, termination

SUMMARY

We often learn of successful pedagogical experiments, but we seldom hear of the the ones that failed. From an epistemological point of view, learning from failures can be at least as effecitive as learning from good examples. This special session has a structure similar to that of Parlante's Nifty Assignments*, i.e. we solicited submissions from the SIGCSE membership, selected the best from among these, and have presentations at the session by the selected authors. Our contributions describe pedagogical approaches that seemed to be good ideas but turned out as failures. Contributors will describe their pedagogical experiment, the rationale for the experiment, evidence of failure, and lessons learned.*

1. OVERVIEW

Learning in Computer Science courses sometimes appears random and accidental, but is often the result of deliberate acts taken by the instructor. An underlying assumption in the SIGCSE literature is that every decision we make in our class offerings is a small-scale local experiment to determine which acts can best lead to learning. In our community, as in most academic disciplines, there is an exclusive focus on the success of these experiments. Rarely, if ever, do we hear about the experiments that failed. Given that negative results can be as valid as positive results in the scientific endeavor, it seems natural that there should be a forum for the discussion of especially negative experiences in CS Education.

In the Comparative Method, Ragin [7] distinguishes between actual and artifactual causal determinants of phenomena. It follows that the absence of effect on the dependent variables is as important as its presence, especially for phenomena as complex as human learning. In the pedagogy of software development, there are precedents for learning from failures, including Anti-Patterns [3], Project Retrospectives [6], and Software Development Failures [4].

This Special Session, "It Seemed Like a Good Idea at the Time", is a forum for us to discuss the failures we have all had, with the hope that others can avoid the paths that led to these unanticipated results. This session is structured like Parlante's "Nifty Assignments" sessions: a set of presenters competitively selected from submissions solicited by a general CFP to the SIGCSE community. Our hope is that, if successful, this session will become a staple at SIGCSE Symposia for years to come.

2. WHAT MAKES A FAILURE GOOD?

Negative teaching experiences can span quite a range. Much like Tolstoy's description of happy families, successful experiments are all similar but failed experiments all fail in their own way. However, not all failures are interesting or would make for a good discussion. We selected for those teaching interventions adoptable by others for which there was reasonable expectation of success, often as a result of adopting previously published work within a local setting. We also selected for reports that provided evidence of failure – or at least significant deviations from expected outcomes, reasons for the failure, and implications for practice for other CS teachers.

SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA.
ACM 1-59593-361-1/07/0003.

By sharing these experiences, the authors provide cautionary tales (or at least some entertainment) to other instructors. Further, we feel that this will help map the space of design failures in CS Education. Comparing this space to the space of successes we hear so much about should provide valuable comparison for educators and researchers alike. By providing a forum for the failed experiment, we hope to encourage risk taking in the classroom, and by focusing on evidence we hope to foster a community with a greater eye for documenting our classroom experiments.

3. THE PERSONAL SOFTWARE PROCESS IN CS1 (BRIANA MORRISON)

In the late 1990's, the Personal Software Process (PSP) was introduced as a mechanism to improve the time estimation and quality of software, as well as improve programmer productivity [5]. In an attempt to reap these rewards in an academic setting, PSP was introduced to students in a CS1 course. It was hoped that students would learn to recognize their common errors and improve their estimation on how long it would take to complete an out-of-class assignment. Neither of these occurred. Instead, students were unable to accurately record data and no improvements were seen in reducing procrastination and turning in late assignments. Students were overwhelmed by trying to accomplish two new tasks at one time (learning to program, and recording errors) even when working as partners.

4. USING TABLET PCS IN A CS LAB CLASS (STEPHEN EDWARDS)

Three years ago, we began experimenting with the use of Tablet PCs to support active learning in our CS1 course. In Fall 2003, approximately 200 students enrolled in our CS1 course were split: one half met in a traditional computer lab, while the other half met in a non-traditional classroom with movable furniture, wireless access, and Tablet PCs. Along all of the performance dimensions we measured, there were no statistically significant differences between the two groups. In terms of the lab experience, while students liked the reconfigurable furniture and wireless access, they generally did not like using the tablets. Students brought in their own mice as pointing devices, at least one started bringing in a full-size external keyboard, and many complained that the small screen-size was inadequate for the pair-programming assignments used as lab exercises.

5. HUMAN EXECUTION STACK (CAROL ZANDER)

Learning methods that facilitate the active construction of knowledge have been correlated with learning gains across disciplines. One such active learning method, described in [1], is the *Human Tableau*, where members of the class act out a set of key ideas. This seemed a natural method for the computer science classroom, where a group of students can collectively simulate a computation. I had thought that applying this idea to the teaching of recursion would lead to an "ah ha" moment when the students acted out the pushing and popping of data onto their human-activation stack. But, as I found out, for students who don't *already* understand recursion, enacting out a non-trivial recursive problem, "uh oh" is as likely a response, with students little able to connect their simulation actions to the recursive code.

6. FORGETTING THAT ANONYMITY CAN BE ASYMMETRIC (IAN UTTING)

Facilities which allow students to ask question of staff anonymously can help to lower the social barriers to participation in learning among less confident students[2]. However, it must be remembered that the anonymity is not symmetric – although students make (proper) use of their own anonymity, "anonymous" responses are readily attributed to the member of staff making them, and the latitude in attitude (e.g. the use of direct language) claimed by students is not offered to staff in a way that it might be in less archival settings. I will discuss our experience, where that lack of diplomacy in answers led to a social breakdown between students and staff, resulting in formal complaints to the dean and extreme failure rates in the class.

7. USING THE CAMPUS NETWORK FOR CLASS PROJECTS (DAVID MUSICANT)

In my CS1 course, I did a term-long project involving the creation of a very simple email client. Two years later, I conducted a senior capstone project in creating a web search engine. Both of these projects seemed like wonderful examples that would catch students' interest. Both projects, despite the best of intentions, ended up in an accidental denial of service attack on campus email servers. How can it be so easy to bring down robust production technology? It's easier than it sounds, and class projects that use critical campus Internet services are considerably more dangerous than I had previously thought.

8. REFERENCES

[1] T. Angelo and P. Cross. *Classroom Assessment Techniques, 2e.* Jossey-Bass, San Francisco, 1993.

[2] D. J. Barnes. Public forum help seeking: the impact of providing anonymity on student help seeking behavior. In G. M. Chapman, editor, *Computer Based Learning in Science.* Univ. of Ostrava, Czech Republic, 1999.

[3] W. J. Brown, R. C. Malveau, H. W. McCormick(III), and T. J. Mowbray. *Anti Patterns: Refactoring Software, Architectures and Projects in Crisis.* John Wiley & Sons Inc, New York, 1998.

[4] K. Ewusi-Mensah. *Software Development Failures: Anatomy of Abandoned Projects.* MIT Press, Cambridge, Massachusetts, 2003.

[5] W. S. Humphrey. *A discipline foe software engineering.* Addison-Wesley Longman, 1995.

[6] N. L. Kerth. *Project Retrospectives: A Handbook for Team Reviews.* Dorset House Publishing, New York, 2001.

[7] C. Ragin. *The Comparative Method: Moving Beyond Qualitative and Quantitive Strategies.* University of California Press, Berkeley, 1989.