JavaTea: An on-line tool for teaching beginners to develop unit tests

Ian Barland
Radford University
ibarland@radford.edu

Abstract:

JavaTea is an on-line system being developed to help teach beginning programmers to write good unit test suites. JavaTea presents a problem specification, and then asks the student to enter a set of test-cases (using, for now, Java syntax for expressions). Then, JavaTea runs a battery of incorrect implementations against the student's test-cases. If any incorrect implementation passes all of the student's tests, JavaTea reports that their test cases are insufficient. (This can be viewed as a complement to sites like codingbat.org, where students write code and are then told whether it passes all tests. Indeed, in a classroom setting, JavaTea assignments would presumably be followed up with implementing the functions they just designed tests for.)

The goal is to help students view test cases as helpful tools in (a) understanding a problem's specification, (b) thinking of corner cases in advance, and (c) giving confidence that an implementation really solves the entire problem.

All submissions are recorded in a database, allowing for the possibility of identifying common shortcomings of test suite design.

For now, JavaTea uses simple functions without mutation. Ideas for how to best relax that restriction, as well as other desired features, are welcome. Planned enhancements include allowing the students to export their test cases in various languages/formats (including JUnit and bare-bones printing), letting instructors create their own assignments and view performance reports, and letting instructors add their own new problems (including a file with incorrect implementations).

Sampling the Error: Research into Underrepresented Groups in Computing

Tony Clear
Auckland University of Technology
Tony.clear@aut.ac.nz

Abstract:

There has been considerable handwringing over time about the unbalanced proportions of women and 'minority' students in computing courses. Unfortunately CS Education researchers do not find themselves in a strong position to investigate these issues or effectively address them. Research designs that investigate the performance and retention of underrepresented students in Computing classes are already working with an unrepresentative group – those who have chosen to be there! Not only do such studies typically suffer from small numbers and difficulties with drawing soundly based statistical conclusions, but even qualitative studies can make few claims to generalisability. Speaking statistically the sampling strategy needs to identify the absent population of underrepresented students and investigate them, rather than merely "sampling the error" of those present in computing classes.

In a recent analysis aimed at determining the equity impacts on Maori and Pasifika students of changing the computing degree entry criteria at Auckland University of Technology (AUT), a disturbing set of figures were identified. In many respects the patterns echoed those found by Margolis and colleagues (2008) investigating the experiences at high school in Los Angeles of Black and Hispanic students. Determining effective counter measures to such entrenched patterns of discrimination is difficult for a university at the end of such a chain. Currently the Design and Creative Technologies Faculty at AUT, (which has an espoused commitment to becoming the University of choice for Maori and Pasifika students) is beginning to design an intervention strategy. This talk will touch on some of the issues identified and seek input to an effective, research-based design for change.

Margolis, J., Estrella, R., Goode, J., Holme, J., & Nao, K. (2008). Stuck in the Shallow End - Education, Race,and Computing. Cambridge, Massachusetts: MIT Press.

Self Study—A Viable Research Methodology for Computing Instruction?

J. Philip East
University of Northern Iowa
east@cs.uni.edu

Abstract:

Self-study is used by teacher educators to enhance "understanding of teacher education in general and the immediate improvement of our practice." [1, p. 818] It has been conceptualized as "'a methodology for studying professional practice settings' that has the following characteristics: it is self-initiated and focused; it is improvement-aimed; it includes multiple, mainly qualitative methods; and it defines validity as a validation process based in trustworthiness." [1, p. 817] Data can be qualitative (researcher notes, student products, discussion transcripts, etc.) or quantitative. "Critical friends" are often involved in data analysis and/or other study activity.

My interest in self-study arises from: its explicit focus on improvement of teaching practice, dissatisfaction with the personal utility of much educational research, and my perception that more influential studies are often "stories" that connect personally with the reader (and self-study offers an excellent opportunity to tell rich stories).

I wish to examine a course that I have taught in the past and will be teaching in the future. With critical friend(s) I plan to: gather past course plans, records of actual activity, and perceived successes and failures; develop a detailed plan of activities for the next course instance (with rationales); meet and discuss while teaching; and develop a report indicating lessons learned for publication.

I think work such as this "can both inform the practices of [those] who conduct it and contribute to the knowledge and understanding of … the larger community of scholars and educators." [2, p. 11] Let's talk about it.

[1] LaBoskey, V.K. 2004. The methodology of self-study and its theoretical underpinnings. In *International Handbook of Self-Study of Teaching and Teacher Education Practices*. Kluwer, The Netherlands.

[2] Zeichner, K. 1999. The new scholarship of teacher education. *Educational Researcher*, 28, 9, (Dec. 1999), 4-15.

Longitudinal Evaluation of a Computing Degree Program

Allison Elliott Tew
University of British Columbia
aetew@cs.ubc.ca

Abstract:

While recent assessment efforts have focused on meaningful evaluation of individual computer science courses, there is little research about how to gather useful information about how a curriculum, or series of courses, affects student knowledge development and retention. We propose a method for investigation using a longitudinal study of a focused stream of courses, where students' progress through this area is indicative of their progress overall. In this talk, using the CS program at University of British Columbia as a case study, we describe a process for identifying a set of courses to target and present the design of the longitudinal study.

Results of Student Involvement in HFOSS

Heidi Ellis
Western New England University
ellis@wne.edu

Gregory Hislop
Drexel University
Hislop@drexel.edu

Abstract:

Student participation in Humanitarian Free and Open Source Software (HFOSS) has the potential to attract students to computing. Between summer 2008 and fall 2010, students in nine different courses and summer internships across four different institutions were surveyed as to their opinions on their experiences in participating in an HFOSS project. Approximately 130 data points were gathered. Overall results indicate that 89% of students agreed or strongly agreed that participation in HFOSS increased their motivation and interest in computing. 92% of students agreed or strongly agreed that participation in an HFOSS project increased their understanding of professional behavior and 94% agree or strongly agree that they can actively participate in an H-FOSS community to develop a software project. One interesting result was that when comparing the pre and post survey results, there was a decline in confidence in software engineering skills. This may be due to students gaining a better understanding of the magnitude and complexity of software development.

Early-stage (Informal) acquaintance with computing with the goal to disseminate seeds of computational thinking

Bruria Haberman
Holon Institute of Technology and
Davidson Institute of Science Education
Bruria.Haberman@weizmann.ac.il

Valentina Dagiene
Vilnius University
valentina.dagiene@mii.vu.lt

Abstract:

Computational thinking is a fundamental skill for everyone. It includes a range of mental tools that reflect the breadth of the field of computer science. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability [1]. Many studies were conducted regarding how to develop computational thinking in the framework of formal K-12 computing programs. However, attracting students to computing studies has always been a challenge; we believe that early-stage informal acquaintance with computing and computational thinking in an attractive way may be beneficial for youngsters.

In our talk we wish to spark discussion among conference participants regarding the importance of informal learning activities aimed at: (a) disseminating seeds of computational thinking, and (b) prompting students' curiosity and positive attitudes towards computing. One important question relates to how to evaluate the influence of such activities on students' skills and attitudes as well as on the school instructional milieu.

We present the Beaver International Contest on computer science and computer fluency that was established with the goal to informally convey computing concepts to as many youngsters as possible from the very beginning at school, in a way that can motivate them to be more interested in computing (http://www.bebras.org). The key idea is to pose to students interesting problems that do not require specific pre-knowledge in a playful way that leads to explorative learning. Criteria for good tasks have been developed. Attraction, invention, tricks, and surprise should be desirable features of each problem; using a proper "narrative cover story" problem statement enables many aspects of computing to be an underlying topic of a Beaver problem [2]. Recently, in Israel, Beaver has been recommended as an outreach program with the goal to disseminate seeds of computational thinking, and to prompt students' curiosity and positive attitudes towards computing.

[1] Dagiene, V., Futschek, G. (2008). Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. In: R.T. Mittermeir, M.M. Syslo (Eds.), *Lect. Notes in Computer Science*, Informatics Education – Supporting Computational Thinking, Springer, Heidelberg, 5090, 19–30.

[2] Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

"Studioize" your courses with OSBLE

Christopher Hundhausen
Washington State University
hundhaus@wsu.edu

Abstract:

The Online Studio-Based Learning Environment (OSBLE) is a new learning management system specifically tailored to support the iterative construction and critical review of a variety of disciplinary artifacts, including computer code, software design documents, and videos of user interface designs. Besides being an excellent platform for setting up an online course presence, OSBLE enables instructors to define iterative "studio" assignments involving artifact submission, expert-moderated peer review (based on structured inline comments and/or rubrics), collaborative issue voting, author rebuttal, and artifact resubmission. In addition, OSBLE enables communities of instructors to share resources and discuss teaching practices within the context of their live courses. For computing education researchers, OSBLE provides a convenient testbed for exploring research questions regarding the design space of studio-based learning, as well as for empirically comparing face-to-face and online studio-based learning.
Hosted at osble.org, OSBLE can be used for any course free of charge.

Peer Instruction in Theory of Computation

Cynthia Bailey Lee
University of California, San Diego
clbailey@cs.ucsd.edu

Abstract:

This talk will report three successful quarters of teaching upper-division Theory of Computation using Peer Instruction (PI), and solicit collaborators for adoption of the materials and development of empirical methods for evaluating PI's impact on students' proof writing skills. Peer Instruction, popularized by Harvard Physics Professor Eric Mazur [1], consists of using lecture time to have students solve multiple-choice problems. Students first respond to a question individually, using a wireless "clicker" device, then they discuss the problem in small groups and submit a second response. Theory of Computation is well suited to Peer Instruction because it allows students to practice making proof arguments in class as they justify their answers to their classmates. That PI can work in a course like Theory of Computation may be surprising, given that PI is often associated with introductory level courses, and it may not be obvious how abstract theoretical concepts and proof writing can be taught using multiple choice questions and clickers. Student response to the use of PI in this course has been positive, but evaluation of the effectiveness of this approach compared to traditional lecture remains a challenge. In particular, this talk will invite suggestions for designing pre- and post- testing questions for Theory of Computation, and for ways to empirically compare student proof writing skills.

[1] Mazur, E. Peer Instruction: A User's Manual. Prentice Hall, 1997.

Pair programming is not always better!

Colleen M. Lewis
University of California, Berkeley
ColleenL@berkeley.edu

Abstract:

A lot of times people say "Pair programming works!", but most of the comparisons of pair programming have compared pair programming to educational environments with obvious deficiencies. Of course pair programming is better than a classroom where students sit idle for 30 minutes waiting for their question to be answered. In a similar way solo programming seems obviously better than sitting idle next to someone programming without understanding or asking questions. In many ways we're still early in understanding the educational implications of pair programming. When does it work? And why? And for whom? It is naive to assume that pair programming or any educational intervention produces equivalent benefits for all students. We don't know for which students, or when, we should be recommending individual or pair programming.

See Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students? Computer Science Education. 21 (2), 105-134.

Studying CS education for scientists

Elizabeth Patitsas
University of Toronto
patitsas@cs.utoronto.ca

Steve Easterbrook
University of Toronto

Abstract:

Within the CS education community, we already know that a lack of computational skills can present a serious detriment in a wide variety of professions [1] -- and that research in the natural sciences is one such case [2]. We would like feedback on our planned research in improving CS education for natural scientists. This is relevant to the ICER community in terms of expanding the audience affected by CS education research -- in line with work such as by Dorn [3] and Ko [4]. Our planned starting point to our project is to perform a cross-sectional study of scientists, using interviews to gather "war stories" [5] of specific coding issues they have faced recently. An open question we face is how to effectively use these war stories to build a needs-based CS curriculum for scientists -- an idea we have is to ask CS/SE experts to evaluate what concepts were missing, or misconceived in the war stories -- and seek input on how this could best be done, such as in which experts to consult, and a method for gathering their feedback. Another question we will raise in the lightening talk is how broadly we should scope the initial gathering of war stories, given that large variation is expected between different scientific fields. Finally, a third question we face is in how to effectively evaluate any CS curriculum for scientists -- and with that, our own work.

[1] Guzdial, Stephenson, Thompson. 2011. Computing Education Blog: A Joint Call for Research on Why Computer Science Education is Important for K-12. http://computinged.wordpress.com/2011/01/12/a-joint-call-for-research-on-why-computer-science-education-is-important-for-k-12/

[2] Wilson. 2006. American Scientist: Where's the Real Bottleneck in Scientific Computing? http://www.americanscientist.org/issues/pub/wheres-the-real-bottleneck-in-scientific-computing

[3] Dorn. 2010. A Case-Based Approach for Supporting the Informal Computing Education of End-User Programmers. http://www-static.cc.gatech.edu/~dorn/papers/dorn_brian_j_201012_phd.pdf

[4] Ko et al. 2011. The state of the art in end-user software engineering. http://portal.acm.org/citation.cfm?doid=1922649.1922658

[5] Lutters, Seaman. 2007. Revealing actual documentation usage in software maintenance through war stories. http://userpages.umbc.edu/~cseaman/papers/ISTJ07.pdf

Concept Inventories for Computer Architecture

Leo Porter
Skidmore College
leo.porter@skidmore.edu

Abstract:

I am in early stages of development of pre/post questions for an upper division computer science course -- computer architecture. These questions relate, in many ways, to concept inventories (like the Force Concept Inventory in physics) but upper division classes introduces one core challenge:

Given the depth of terminology in upper division classes, should a pre/post test attempt to separate class concepts from class terminology? For example, can such questions use course specific terms (e.g. “caches”, “pipeline depth”) or should it attempt to explain those terms (through analogies, etc.) as part of the question?

In this lightning talk, I introduce this challenge and the trade-offs inherent with potential solutions.

Promoting engagement, retention and persistence in computer science through a sense of belonging

Nanette Veilleux
Simmons College
veilleux@simmons.edu

Rebecca Bates
Minnesota State University

Abstract:

Research has identified a wide variety of factors affecting student engagement, which in turn affects key outcomes including academic achievement, retention and persistence. Feeling connected to a community and sense of belonging can substantially influence student engagement. Education studies in Computer Science [e.g.1] have called attention to the problem of feelings of isolation and their impact on interest, achievement, and retention, particularly for underrepresented groups.

Our five year, multi-campus study focuses on the key concept of "belonging" to a community [2] which has proven to be especially relevant to engagement, persistence, and drop-out at the K–12 level. Our study does not presume that the academic community alone must be welcoming. We also probe the effect of outside communities (e.g., extra-curricular groups, internships, family) as well as the ability of our students to create the communities they need to be effective scholars. Moving from academia to the work place, a sense of belonging can result in increased feelings of security, stronger self-concept, self-respect and coping abilities [3]. Thus, from the perspective of the 21st century workforce, improved understanding of and ability to build community in the undergraduate experience links to essential needs in the technological workforce.

[1] Biggers, M., Brauer, A., & Yilmaz, T. (2008). Student perceptions of computer science. In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education. Association for Computing Machinery.

[2] Plett, M. Carlson-Jones, D., Crawford, Floyd-Smith, T., Peter, D., Scott, E., Wilson, D., Bates, R., and Veilleux, N., "STEM Seniors: Strong Connections to Community Are Associated with Identity and Positive Affect in the Classroom," ASEE Conference 2011

[3] DeNeui, Daniel L.C. (June 2003). An Investigation of First-Year College Students' Psychological Sense of Community on Campus. College Student Journal, 37, 224-234.