Re-designing a Computer Science Course in Higher Education

Nathan Fox

Design & Development Project
Instructional Psychology & Technology, Brigham Young University
Project Description

- Standardize CS 235
- Use two existing versions as a framework
- Create new learning materials as needed
Goals

- Promote deeper learning
- Help students transition to proactive student-owned learning
- Scalability to meet increased demand
- Help students to help each other
Deliverables
New Reading Assignments

- Keep students accountable for doing the reading
- Integrate assessment items into content for spaced learning
- Immediate feedback
- Multiple attempts to deepen understanding
- 25 new reading assignments for 14-week semester
New Exams

- Deliberate balancing of content
- Target more higher-level learning outcomes (application-level)
- Mostly objectively-scored for scalability
New Live Coding Activities

- Literature encourages live coding over static code review
- Models expert performance for both coding and debugging
- Use coding principles that apply to coding labs
New Coding Lab

- Uses student solution to solve a real-world problem
- Introduces heapsort into coding labs
Design Process
Unique Factors

- Evaluate two versions of course to build off of
- My roles were evaluator, designer, subject matter expert, and instructor
- Unavoidable overlap in design, development, and implementation
Analysis

- Analyzed grade data
- Interviewed students and instructors
- Content and task analysis
- Lit review
Initial Evaluation

- Observed both versions of CS 235
- Interviewed students and instructors
- Evaluated teaching techniques, assessments, and technology
- Decided to build off of the flipped version of CS 235
Planning

- Primary deliverables
  - New reading assignments
  - New exams
  - New live coding activities
  - Think-pair-share web application
- Course schedule
- Completed existing assessments (labs and exams)
Prototypes

- Developed reading assignments
- Developed coding exercises
- Developed think-pair-share web application
- Started designing the midterm
- Tested the prototypes as part of teaching the class
Design in Flight

- Designed and developed later materials as I taught
- Received regular feedback from students
- Developed reading assignments three weeks out, live coding activities one week out
- Met weekly with other CS 235 instructors and TAs to report
Design Evolution
Before Implementation

- Live coding
- Think-pair-share
- New reading assignments with spaced learning
- New exams
During Implementation

- Live coding - make smaller, slow down
- Cloud9 - tutorial on Linux/Bash, debugging
- Think-pair-share - took too much time for too little benefit
- New lab 9
After Implementation

- Modified schedule to account for Linux/Bash training
- Lab 8 - extra credit
- Lab 9 - more time
- Modified/rewrote some reading assignments to improve clarity
Product Implementation
Required Resources

- Students - laptop, internet connection, basic C++
- Instructor - computer with internet connection, classroom, projector
Learners

- 34% not comfortable with C++ syntax
- 70% not comfortable with debugging tools (7% can’t define)
- 28% not comfortable with string data type
- 39% not comfortable with functions
- 39% not comfortable with arrays and vectors
- 55% not comfortable with object-oriented programming (7% can’t define)
- 66% not comfortable with polymorphism (7% can’t define)
Evaluation
Criteria

- Promote deeper learning
- Help students transition to proactive student-owned learning
- Scalability to meet increased demand
- Help students to help each other
Criterion 1 - Promote Deeper Learning

- Exams test higher-level learning outcomes
- Midterm mean of 92%, final mean of 83%
- “It has been a really good class and I actually feel like I am starting to understand Computer Science for the first time ever.”
- “Nathan was the best at teaching difficult concepts simply. The reading quizzes were awesome.”
Criterion 2 - Promote Student-owned Learning

- Flipped/blended course design
- “Nathan gives all the help he can give, but still places the responsibility to learn on the students.”
- “I personally learned to be better about managing my time and putting the requisite effort from this class.”
Criterion 3 - Scalability

- Autograded reading assignments, coding labs, and exams
- Cloud9 allows for efficient TA lab verification
- Class attendance optional
Criterion 4 - Students help each other

- Course design and technology allowed students to help each other
- Relatively few took advantage
Outcomes

- Need to make some changes to encourage students to help each other
  - Organize students into communities of learning
- CS department should consider reworking coding labs to improve student motivation and applicability
Reflection and Critique
Reflection

- Importance of developing empathy for learners through story
- Importance of relying on existing literature
- Learner-centric design
Critique

- Polish new reading assignments to synthesize into a textbook
- Experiment more with communities of learning
- Redesign coding labs