SPLICE: Self-Paced Learning in an Inverted Classroom Environment

Curt Clifton and Matt Boutell
Department of Computer Science and Software Engineering
Rose-Hulman Institute of Technology
5500 Wabash Ave.
Terre Haute, IN 47803
{clifton, boutell}@rose-hulman.edu

Abstract

Learning to program is hard for many students. Practice with an expert coach is key to overcoming this challenge. We adopted the 3×2 format for our introductory courses to give students such mentored practice. In a single class session, students learn a concept, experiment with it, and apply it to a real problem, all with expert coaching at hand.

While this format has been effective, we still find two significant problems: time and pace. Time is an issue because presenting concepts, showing examples, and modeling problem solving decreases the time available for mentored practice. Pace is an issue because some students arrive with confidence and prior experience and are thus bored, while other students struggle and become overwhelmed.

To address these problems, we propose creating on-line videos for introductory programming courses to present concepts, show examples, and model the problem solving process. As a result, our students will spend every class session entirely in active learning activities with expert coaching, receive more individual attention, and set their own pace.

This grant will enable the investigators to produce video modules for the C programming portion of CSSE 120/221, develop a plan for assessing the effectiveness of this approach, and begin seeking additional funding to extend the approach to the rest of our introductory curriculum.

This proposal aligns well with the selection criteria for the Rose-Hulman Summer Professional Development Grants. The project has pedagogical merit, will lead to enhanced scholarly activity for the investigators, will have some interdisciplinary benefits, and is sustainable.

Background and Rationale

Problem Statement

Anyone who has taught introductory computer programming courses understands that *learning to program is hard for many students*. Most students readily grasp the basic pieces, but they fall down in their logical thinking. They struggle to compose the basic pieces to create software that solves a real problem. We often hear some students say, "I understand all the concepts in class, I just can't program." Their exam performance supports that claim: these students can answer conceptual questions, but have difficulty actually solving problems by programming.

But solving problems by programming is the essential skill. Even in our introductory courses, we're focused on helping students achieve competence at the *application* level in Bloom's taxonomy [1]. Reading through the course outcomes for CSSE 120 or 221 we see words like *design*, *implement*, *test*, *debug*, *demonstrate*, and *solve*. To achieve this level of learning, students must practice.

Before students can practice, though, they need to learn basic concepts (like syntax and control structures); get to know their tools (like integrated development environments); and watch an expert model the problem solving process. After that, they are ready to move into structured practice, where they fill in pieces of the puzzle under the guidance of an instructor and with appropriate scaffolding. At this stage, students are developing comfort with the basic concepts and tools, while the instructor coaches them. Although students could do this on their own, many spend a significant part of their work time on incidental challenges, like syntax or obscure error messages, rather than the core concepts. A coach can help the students move past the incidental challenges and focus their energy on the essential challenge of logical thinking.

As students build confidence, we gradually pull away the initial scaffolding and ask them to solve more realistic problems. At this stage, expert coaching and feedback are even more important. An attentive instructor helps students continue to develop their skills and confidence. Ultimately, students are able to work in teams, independent of the instructor, on larger projects.

This progression of activities, from understanding basic concepts through independent problem solving, is why we adopted the 3×2 studio format for our introductory courses. In a single class session, students learn a new concept, experiment with it, and apply it to a real problem, all with expert coaching at hand. While this format has been effective, in practice, we still find two significant problems: time and pace.

Time: We've found that we spend a majority of our class time, even in the 3×2 format, on the earliest stages in our progression of activities. Although textbooks can describe the basic concepts, they are abstract, so we spend some class time reiterating the textbook content. A textbook cannot provide examples of the dynamic problem solving process, so we spend significant class time modeling that. Students say that they really appreciate these *live coding* examples, where an instructor programs in front of the class and thinks aloud about his or her problem solving process. Unfortunately, live coding is not very interactive and consumes substantial class time—time that would be better spent providing expert coaching.

Pace: The other problem we face is finding the right pace for our introductory courses. It seems like regardless of the pace chosen, midterm surveys and course evaluations show that some students are bored while others are overwhelmed. While programming takes time for most novices to learn, some students need even more time. At the opposite end of the spectrum, other students, either due to prior experience or great aptitude, quickly become bored because the pace is too slow for them. This compounds the problems of the slower students, who become intimidated not just by the material, but also by their peers. This "erosion in confidence" is particularly pronounced for women [2, ch. 5]. Furthermore, because the distribution of students tends to be fairly uniform between the two extremes, using any lock-step approach, regardless of pace, fails to meet the needs of many of our students.

Proposed Solution

To address the complementary problems of time and pace, we propose replacing lectures, examples, and live coding for our introductory programming courses with on-line videos. This will maximize the amount of class time that students spend in mentored practice and allow students to control the pace at which they digest the material. The videos enable self-paced learning in an inverted classroom environment, or *SPLICE*.

The SPLICE videos will fall into three different categories. Some videos will introduce new concepts with short screencasts combining slides and examples. These videos will supplement and reinforce the concepts presented in the course textbooks. A second category of videos will consist of live coding. An expert will model logical thinking by solving sample programming problems while thinking aloud (as Doering and Mu did for circuit analysis in their videos [3]). These live coding videos present a thought process that cannot be readily conveyed by a traditional textbook.

Students will watch videos in these first two categories to prepare for class. While watching these videos, students will complete active learning exercises, such as answering quiz questions and writing small code snippets. This will help them engage with the material. To give students immediate feedback, a later portion of each video will give solutions to the coding questions.

The third category of videos will be short supplementary ones that cover basic technical details, like configuring tools and troubleshooting problems. While these might not necessarily be required viewing, they will be available for the students to reference so that they don't "spin their wheels" on a problem that is easy to solve for anyone who knows the right incantation.

By using these videos, students will arrive in class prepared to practice the ideas to which they've already been exposed. We will give them an assignment over the material and they will get right to work. The instructor will circulate, observing their work and offering appropriate assistance. In class, students' attention will be completely focused on active learning, while the instructor's attention will be completely focused on coaching his or her students.

If the students arrive in class ready to work, and immediately engage with a new assignment, what's to keep students from working ahead? In a word, *nothing*. By using videos to deliver the basic concepts, model expert solution strategies, and present examples, we will decouple the pace of the class from the pace of lecture. Students who would have found the pace too slow will be able to work quickly through material that they already know and delve into more interesting problems. Students who would have struggled with concepts will be able to rewind and watch tricky segments a second time. These students will also benefit from a greater share of the instructor's time—time that can be spent identifying the particular and individual sources of a student's confusion.

One interesting possibility that we hope to investigate is whether the faster students might serve as peer mentors for the other students in the class. Suppose some student covers a three-week module on C programming in only one week. We're interested in learning whether a student like that might benefit from acting as an in-class peer mentor. Like the instructor, the peer mentor would circulate around the classroom, observing classmates, offering constructive suggestions, and helping with points of confusion. This approach would mean that the slower students have even more help available to learn the concepts. And the faster students might achieve the deeper understanding that comes from explaining a concept to someone else. This would also mitigate the risk in self-paced learning where a student quickly crams through material, but isn't engaged with it for a long enough time for long-term retention.

Outcomes and Significance

Once we've implemented SPLICE, students will receive more individual attention and class time will be devoted entirely to active learning activities with expert coaching. Each student will set his or her own pace. Advanced students will stay engaged by moving quickly on to more challenging problems. Slower students will get more of the help they need with less of the intimidation and confusion that

can come from a more traditional classroom environment, while continuing to meet our rigorous expectations for progress.

Beyond just helping students who are enrolled in our introductory courses, the SPLICE videos will also be available to them for reference later. This will be particularly helpful when they take later courses where a familiarity with C is expected, for example CSSE 332: Operating Systems, CSSE 432: Computer Networks, ECE 483: DSP System Design, and ECE 580: Digital Signal Processing.

The SPLICE videos will also be available to people outside the Rose-Hulman community. This will raise our profile with the wider world. It is easy to envision the videos as a recruiting tool if high school students use them.

Background

The inverted classroom environment [4], in which lecture is moved before class to make room for other in-class activities, is not a new idea. Kaner and Fiedler [5-6] and Day and Foley [7] used video lectures to invert their upper-level software courses. In contrast to our proposal, these videos were merely a "talking head" in the corner of PowerPoint slides. There was no active learning component for students while watching the videos. Furthermore, class time wasn't used for hands-on application of the ideas, but for further discussion of the concepts. Gannod, *et al.* [8] used video podcasts to deliver full lectures successfully to students in their introductory computer science course, but their videos are not freely available.

In general, audio and video instruction to teach computer programming is not new, either. Georgia State offers an Intro to Programming in C# video course (as part of Apple's *iTunes U* program), but it is incomplete [9]. Trinity College (Dublin) offers Intro to Computer Programming in C++ on iTunes U, but only some sessions are available. The course is also specific to Mac OS X. UC Berkley's Computer Science 61C contains five lectures on introductory C programming, but this course is the third in their CS sequence, so presumes two full terms of programming experience. The Berkley course is on iTunes U but is audio-only. MIT's Open Courseware includes EECS 6.00 Introduction to Computer Science and Programming in Python [10]. This is a very different curriculum than ours, taking a breadth-first approach. It is also not a screencast, but a professor at a blackboard teaching concepts. Stanford offers full video lectures for a three-course introductory software development sequence as part of the Stanford Engineering Everywhere initiative [11]. This sequence begins with Java. C is not covered until the third course. At the Naval Academy, Carlisle assigns 5-minute YouTube clips to his programming students, but the course is in Java [12]. A variety of commercial training videos on C programming are also available on-line, but they are all incomplete, inaccessible, or inappropriate for our course sequence and tools [13-16].

Short-term Merit

SPLICE aligns well with the selection criteria for the Rose-Hulman Summer Professional Development Grants.

The *overall merit* of our proposal is primarily pedagogical. The self-paced, inverted classroom approach will solve the dual problems of time and pace in our introductory courses. Furthermore, this cutting-edge approach will explore a new way of delivering content, while maintaining the hands-on approach that distinguishes a Rose-Hulman education. The pedagogical benefits accrue to students currently enrolled in the courses, students reviewing for subsequent courses, and prospective students interested in learning to program.

The project will *lead to enhanced scholarly activity* for Clifton and Boutell, as they assess and report on the benefits of the SPLICE approach. We also envision eventually extending the approach to other courses, and other disciplines. This would lead to additional scholarly activity.

Initially SPLICE is proposed for our introductory CSSE courses. Nevertheless, it would have some *interdisciplinary benefits*. Students in these courses come from Computer Science, Software

Engineering, Electrical Engineering, Computer Engineering, Mechanical Engineering (through the Robotics program), and Mathematics. Students in upper division ECE courses would also benefit from the videos. We have letters of support from faculty in ECE and ME.

The SPLICE project is *sustainable*. It represents a new pedagogical approach that aligns with the goals of multiple external funding organizations. Our planned deliverables include submitting funding proposals to these organizations. The results we obtain from producing and deploying these initial videos will provide the supporting evidence we need to attract additional resources.

Long-term Implications

Our proposal is well aligned with Rose-Hulman's traditional strengths of highly interactive classrooms and individual attention. At the same time, the SPLICE approach represents a possible segment in President Branam's vision for offering the "leading edge of techniques for teaching Math, Science, and Engineering." Our proposal provides an opportunity to investigate what many see as a likely future for higher education – a future where a student's education is guided by an academic mentor [17] and measured not by the time spent in a seat, but by the competence he or she demonstrates [18-20].

We are not advocating for wholesale change in the delivery of all courses. Certainly there are many courses where the discussion of ideas in class is essential; video lectures would not be appropriate for such courses. However, we are convinced that an institution that aspires to greatness has to be open to a wide variety of pedagogical approaches.

Plan of Work

Figure 1 (at right) lists the proposed deliverables to be produced by the activities described below. Following that is a description of our assessment and dissemination plans.

Activities

A basic outline of the materials to present in the videos already exists from our current CSSE 120: Introduction to Software Development. Clifton was involved in the original design of this course and Boutell has been involved in several successive refinements of it. The course includes seven weeks of introduction to programming in Python, followed by a three-week module introducing programming in C. This C module is also used in CSSE 221: Fundamentals of Software Development (Honors).

Summer 2010

- Videos produced
- Videos available for download
- Assessment plan written
- Course materials revised
- Production process documented

2010–2011 Academic Year

- Videos deployed in CSSE 120 and CSSE 221
- Videos available to students in upper division courses
- Assessment data collected
- Conference paper written
- Grant proposal drafted

Figure 1. Deliverables

Because the basic materials are in place, our primary activities for **Summer 2010** will be:

- Learn to use the tools for producing the video modules.[1] Clifton has limited experience already; Dr. David Fisher has experience with screencasts and inverted classrooms and has offered his advice as needed (see letter of support).
- Produce video modules for the C programming portion of CSSE 120/221. Assuming that production time is approximately eight times the running time, which is in line with Clifton's and Fisher's experience, there is four person-weeks of labor just in video production.
- Research the most effective mechanisms for distributing the video modules.

---

[1] The department used money from the Dean's equipment fund to procure screencasting software and hardware during 2009-2010. Therefore, the only new expense is the instructors' time.

- Develop a plan for assessing the effectiveness of the SPLICE approach when we deploy it during the 2010-2011 academic year. Shannon Sexton and the IRPA staff have agreed to assist with developing and executing this plan (see letters of support).
- Make necessary revisions to CSSE 120 and CSSE 221 course materials to support introduction of the SPLICE approach, including schedules, slides, assignments, and communicating with colleagues who will be teaching these courses.
- Document the process of creating the video modules.

During the **2010–2011 academic year** we will:

- Deploy the video modules for C programming in CSSE 221 in the fall, and CSSE 120 at some point in the year. Dr. Boutell is scheduled to teach CSSE 221 in the fall. At least one of Dr. Clifton, Dr. Boutell, or Dr. Claude Anderson will deploy the videos in CSSE 120 (see letter of support from Dr. Anderson).[2]
- Make the videos available to students in upper division courses. Dr. J.P. Mellor has committed to using the videos for CSSE 332: Operating Systems and CSSE 351: Computer Graphics. Dr. Wayne Padgett has committed to using the videos for ECE 580: Digital Signal Processing or ECE 483: DSP System Design. (See letters of support.) We're also hopeful that the videos will be used for CSSE 432: Computer Networks, although the instructor is unknown at this time.
- Collect data per the assessment plan.
- Prepare a conference paper documenting the results.
- Draft a grant proposal for continuing the work, pending collection of sufficient assessment data.

Long term, we plan to:

- Submit grant proposal(s) for continuing the work. Assuming the SPLICE approach proves successful, we would extend it to the first seven weeks of CSSE120 in Python. We have identified other funding sources to sustain this program including the National Science Foundation through its TUES program (previously CCLI), and the Hewlett and Lumina Foundations. To support additional funding proposals, we will work with IRPA to gather assessment data and will present our preliminary findings at an appropriate conference. This publication record will strengthen our ability to secure additional funding.
- Offer a faculty workshop on the SPLICE technique.

Assessment Plan

The proposed work for summer 2010 will be deemed successful if the proposed videos are created and posted; and if a plan is in place for assessing the effectiveness of the videos when they are deployed in classes during the 2010–2011 academic year. As noted above, we will work with IRPA to create this assessment plan, including identifying the qualitative and quantitative data needed. We envision collecting data from students in CSSE 120 and CSSE 221. We also envision collecting data on the perceived value of the videos from students in upper division CSSE and ECE courses to whom the videos are made available.

Dissemination Plan

The videos will be disseminated to our students and the broader community by posting them to the web, either via a Rose-Hulman server or an external video sharing site like Connexions or YouTube. We will submit preliminary results to an education conference such as the ACM International Symposium on Computer Science Education (SIGCSE) once assessment data are available and analyzed. Longer term, we envision offering faculty workshop on the SPLICE technique, either as an Assessment and Learning Forum, or perhaps a half- or full-day session if there is sufficient interest.

---

[2] Because staffing in the CSSE department for 2010-2011 is not known at this writing, we cannot commit to a specific term for deploying the videos in CSSE 120. However, at least one of Dr. Anderson, Dr. Boutell, or Dr. Clifton has taught CSSE 120 every year for at least the last five.

Appendices

Biographies

Curt Clifton received his PhD in Computer Science from Iowa State University in 2005. He is Assistant Professor of Computer Science and Software Engineering at Rose-Hulman. In the last five years at Rose-Hulman, he has taught four sections of CSSE 120. He has taught courses introducing at least six different programming languages, and has used video-based instruction to cover for classes when traveling to attend a conference. He has developed a video-based project specification for the final team project in CSSE 220: Object-Oriented Software Development, which has been reused in at least four offerings of the course.

Clifton is interested in programming language design and implementation and the cognitive barriers to learning to program. He is a co-PI on a seven-institution National Science Foundation grant and has worked with two undergraduate students who were funded by this grant. A third student will join that project beginning in June 2010.

Matt Boutell received his PhD in Computer Science from the University of Rochester in 2005. He is Assistant Professor of Computer Science and Software Engineering at Rose-Hulman. In the last five years at Rose-Hulman, he has taught 13 sections of the courses that will use the C Programming unit (ten sections of CSSE 120 and three sections of CSSE 221).

Boutell is interested in researching pedagogical techniques that challenge students to their potential. He published work about challenging CSSE 221 students through peer teaching [21].

He has worked with IRPA to develop assessment plans for this work and for the Lilly Foundation-funded Faculty Success Grant: *MERI: Multidisciplinary Educational Robotics Initiative*.

Boutell has eleven years of full-time teaching experience at the undergraduate and secondary levels, including teaching introductory programming in five languages at three schools.

References

[1] *Taxonomy of Educational Objectives: The Classification of Educational Goals*, B. S. Bloom (Ed.) Susan Fauer Company, Inc. 1957, pp. 201–207.

[2] Margolis, J. and Fisher, A, *Unlocking the Clubhouse: Women in Computing*. MIT Press, 2002.

[3] Doering, E. and Mu, X., "CLEO: Circuits learned by example online," http://www.rose-hulman.edu/cleo/ (accessed 4/12/2010).

[4] Lage, M., Platt, G., and Treglia, M., "Inverting the classroom: A gateway to creating an inclusive learning environment," *Journal of Economic Education*, 31:1, pp. 30-43, Winter 2000.

[5] Kaner, C. and Fiedler R., "Inside out: A computer science course gets a makeover," *Association for Educational Communication and Technology International Conference*, Orlando, FL, Oct. 2005.

[6] Kaner, C. and Fiedler, R., "Blended learning: A software testing course makeover," *11th Sloan-C International Conference on Asynchronous Learning Networks*, Orlando, FL, Nov. 2005.

[7] Day, J. and Foley, J., "Evaluating a web lecture intervention in a human-computer interaction course," *IEEE Transactions on Education*, 49:4, pp. 420-431, Nov. 2006.

[8] Gannod, G. C., Burge, J. E., and Helmick, M. T. ,"Using the inverted classroom to teach software engineering," *30th international Conference on Software Engineering*, Leipzig, Germany, pp. 777-786, May 2008.

[9] http://www.apple.com/education/itunes-u/ (accessed 4/7/2010).

[10] http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/index.htm (accessed 4/7/2010).

[11] http://see.stanford.edu (accessed 4/8/2010).

[12] Carlisle, M., "Using YouTube to enhance student class preparation in an introductory Java course," *ACM Technical Symposium on Computer Science Education (SIGCSE)*, Milwaukee, WI, Mar. 2010.

[13] http://www.computer-training-software.com/c-programming.htm (accessed 4/5/2010).

[14] http://apex.vtc.com/c-programming.php (accessed 4/5/2010).

[15] http://www.softwaretrainingtutorials.com/c-programming.php (accessed 4/5/2010).

[16] http://showmedo.com/videotutorials/series?name=MjNtBGUsy (accessed 4/5/2010).

[17] Lake, P.F., "Will your college be sued for educational malpractice?" *Chronicle of Higher Education*, Aug. 11, 2009.

[18] Wellman, J.V. and Ehrlich, T., "Re-examining the sacrosanct credit hour," *The Chronicle Review*, Sep. 26, 2003.

[19] Zemsky, R., "Will higher education ever change as it should?" *Chronicle of Higher Education*, Aug. 3, 2009.

[20] Blumenstyk , G., "Beyond the credit hour: Old standards don't fit new models," *Chronicle of Higher Education*, Jan. 3, 2010.

[21] Kaczmarczyk, L., Boutell, M., and Last, M., "Challenging the advanced first-year student's learning process through student presentations," *The Third International Computing Education Research Workshop*, Atlanta, GA, Sep. 2007.

Letters of support

Dr. Claude Anderson, Dr. David Fisher, Dr. J.P. Mellor, Dr. Wayne Padgett, Dr. Julia Williams, and Ms. Shannon Sexton have agreed to participate, as discussed in the Plan of Work above. Their letters of support follow.