Reengineering an ”Introduction to Computing” course within a College-Wide Community of Practice

Dr. Wade Fagen, University of Illinois, Urbana-Champaign

Dr. Wade Fagen is a Lecturer in the Department of Computer Science in the College of Engineering at The University of Illinois at Urbana-Champaign (UIUC). He teaches one of UIUC's largest courses, Introduction to Computer Science, known as CS 105. His research aims to improve learning by using technologies that students already bring to the classroom.

Dr. Cinda Heeren, University of Illinois, Urbana-Champaign

Dr. Cinda Heeren is an award-winning Senior Lecturer at the University of Illinois, Urbana-Champaign. She teaches CS225, Data Structures and Programming Principles, to hundreds of enthusiastic and talented undergraduates every year. She is always game to try new pedagogical innovations, and she loves telling young women about her affection for computing.

Dr. Geoffrey L Herman, University of Illinois, Urbana-Champaign

Dr. Geoffrey L. Herman is a visiting assistant professor with the Illinois Foundry for Innovation in Engineering Education at the University of Illinois at Urbana-Champaign and a research assistant professor with the Department of Curriculum & Instruction. He earned his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign as a Mavis Future Faculty Fellow and conducted postdoctoral research with Ruth Streveler in the School of Engineering Education at Purdue University. His research interests include creating systems for sustainable improvement in engineering education, promoting intrinsic motivation in the classroom, conceptual change and development in engineering students, and change in faculty beliefs about teaching and learning. He serves as the webmaster for the ASEE Educational Research and Methods Division.

Prof. Matthew West, University of Illinois, Urbana-Champaign

Matthew West is an Associate Professor in the Department of Mechanical Science and Engineering at the University of Illinois at Urbana-Champaign. Prior to joining Illinois he was on the faculties of the Department of Aeronautics and Astronautics at Stanford University and the Department of Mathematics at the University of California, Davis. Prof. West holds a Ph.D. in Control and Dynamical Systems from the California Institute of Technology and a B.Sc. in Pure and Applied Mathematics from the University of Western Australia. His research is in the field of scientific computing and numerical analysis, where he works on computational algorithms for simulating complex stochastic systems such as atmospheric aerosols and feedback control. Prof. West is the recipient of the NSF CAREER award and is a University of Illinois Distinguished Teacher-Scholar and College of Engineering Education Innovation Fellow.

Reengineering an “Introduction to Computing” course within a College-Wide Community of Practice

0. Abstract

Widening Implementation and Demonstration of Evidence Based Reforms (WIDER) is a campus-wide, NSF-funded effort to create institutional change in the way we teach core gateway STEM courses. While instruction and course reform has historically been attempted by lone rangers to little lasting effect, WIDER’s paradigm is to organize instructors into Communities of Practice (CoPs) to provide mutual support and training, and to encourage and facilitate the organic dissemination of best practices across courses among the members of the community of practice. In particular, mentorship relationships within the community have provided ready avenues for the translation of best practices. In this paper, we describe and analyze the redesign of one such course in the WIDER community, highlighting how the redesign of this course was informed by its involvement within this larger community of practice.

1. Introduction

Since the 1980s the Computer Science (CS) department at The University of Illinois (UIUC) has offered a service course, “Introduction to Computing”, that was designed to serve non-CS and non-engineering majors from across campus. At UIUC, the largest populations of these non-engineering majors are either students within the College of Business or the College of Liberal Arts and Sciences. As part of this course in the 1980s and 1990s, students often had their first serious interaction with a computer and learned simple word processing and spreadsheet skills.

In 2011, a campus-wide effort was initiated to create institutional change in the way we teach core gateway STEM courses. This effort, Widening Implementation & Demonstration of Evidence Based Reforms (WIDER), had individual departments identify key STEM courses that were in need of reform. As part of WIDER, the course revisions to the courses identified as the most in need of reform would be completed within Community of Practices (CoPs) where instructors would provide mutual support and training that would encourage the dissemination of best practices across courses[6]. As part of the NSF-funded WIDER grant, Principle Investigators (PIs) aided each department in maximizing the utility of such communities based on an extensive literature that already exists on Communities of Practices[7,8,9,10].

Within the Department of Computer Science, we identified the “Introduction to Computing” course as a key course that was need for reform. Anecdotal feedback from both students and academic advisors found that the course no longer served significant value: nearly all students were already well versed with simple word processing and spreadsheets. Not surprisingly, other courses assumed students had knowledge of such tools and assignments were made within Microsoft Word and/or Excel for students to complete and students demonstrated they had those skills without the need for the existing course. The small amount of computing that was present in the course (a unit on VBA) was something that was not immediately useful to students and was not used as part of other courses at UIUC. Ultimately, students were unengaged and bored by a course that felt like a course that was literally from the 1980s.

Beyond student feedback, faculty in all disciplines had rapidly increasing expectations for students' competencies in computing that went beyond simply word processing and spreadsheets. In response, our "Introduction to Computing" course was reengineered during the Spring 2014 semester with a four-pronged vision: (1) modernizing the curriculum by moving the course from a tools-based course to a computing-based course, (2) elevating student engagement, (3) scaling the course for growth, and (4) making the course relevant and accessible to any student, regardless of background or technology.

Toward modernizing the curriculum, the course met with relevant stakeholders across campus, surveyed top courses from other universities, and reflected on best practices from within the community of practice on campus. Borrowing the computational model of thinking from courses like Harvard's CS 50[1], Berkeley's Beauty and Joy of Computing[2], and understanding the needs for non-majors to understand how to process data when tools are not available for them to do so, the course focuses on a data-centric model of computing. Beginning with Scratch[3], the course teaches students basic logic through a graphical programming language. After learning Scratch, the course moves into JavaScript, advanced Excel for data processing, and finishes off with d3.js to connect Excel with JavaScript through data visualization.

To elevate student engagement, the course iterates through nine instructional "cycles" consisting of four elements, borrowing from the "flipped" course model: (a) pre-lecture content presentation and practice problems, (b) active learning exercise during in-class lectures, (c) collaborative, context-rich problem solving lab sections, and (d) a programming assignment to complete on their own. These elements were initially developed in other courses redesigned by other members of the WIDER community, providing inspiration and guidance from within the community of practice.

To scale the course, the course makes use technological innovations that allow for high-quality, automated feedback on assignments. Along with the technology, the course is staffed with a large group of former students who serve as undergraduate course assistance. While every interaction is still lead by an instructor or teaching assistant, course assistants are abundantly available for students to get help when they get stuck working through any of the course content.

To improve accessibility, the course was designed with a "mobile-first" approach. With the goal that every interaction with course content can be complete on a cell phone or tablet, students are able to view lecture slides, complete labs, and finish homework assignments all within a web browser. Each week, 10-40% of the traffic to the course website was received from mobile devices. Further, the course content was modernized to include JavaScript, data visualization, and advanced Excel skills to ensure that students are able to immediately utilize the skills learned as part of the course.

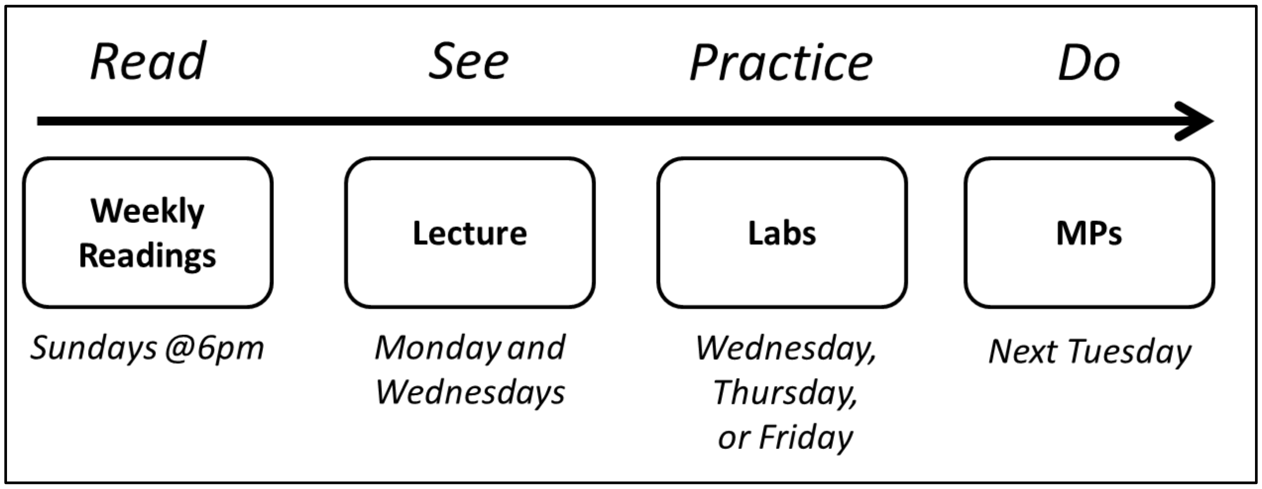


Figure 1: Pattern for each of the nine instructional "cycles" that make up the redesigned "Introduction to Computing" course. Each cycle follows a predictable pattern, engaging students multiple times each week with course material.

Finally, with the success of a CS1-style "Introduction to Computing" course, many students are excited about applying computing to their discipline. The demands for a portfolio of computing skills motivated a follow-on CS2-style course, called "Data Driven Discovery", using the same course design and teaching techniques within the WIDER community of practice.

2. Towards Elevating Student Engagement

Through the community of practice, we learned a key indication of the success of a course is the engagement that students have with the course. This engagement largely comes from one of two sources: a student's intrinsic interest in the content in the course and/or the structure that the content is presented in. As a service course, many students see computing as a course they are required to take as part of their degree and offers little interest to them personally. Instead, we rely on a strong, predictable model of interaction with the course where the students are engaging with the course multiple times every week.

Figure 1 outlines the interaction model that each student has with the course. The model itself is a total of nine instructional "cycles", each spanning a little over one week, where students see an introduction of the material through a pre-lecture activity ("Weekly Reading"), the in-person lecture, the group-based lab assignment, and the solo programming assignment as homework (called a "Machine Problem" or "MP").

The first element instructional activity in the "cycle" is the "Weekly Reading". While they are called readings, these activates are often interactive modules or activates where students engage with the material that they will be learning about in lecture during the upcoming week. With all course material being delivered online, the course supplements course material with high-quality content created by other content creators including examples from codecademy.com, videos form youtube.com, interactive modules from code.org, and other interactive modules. We found that even though these weekly readings were tailored correctly in

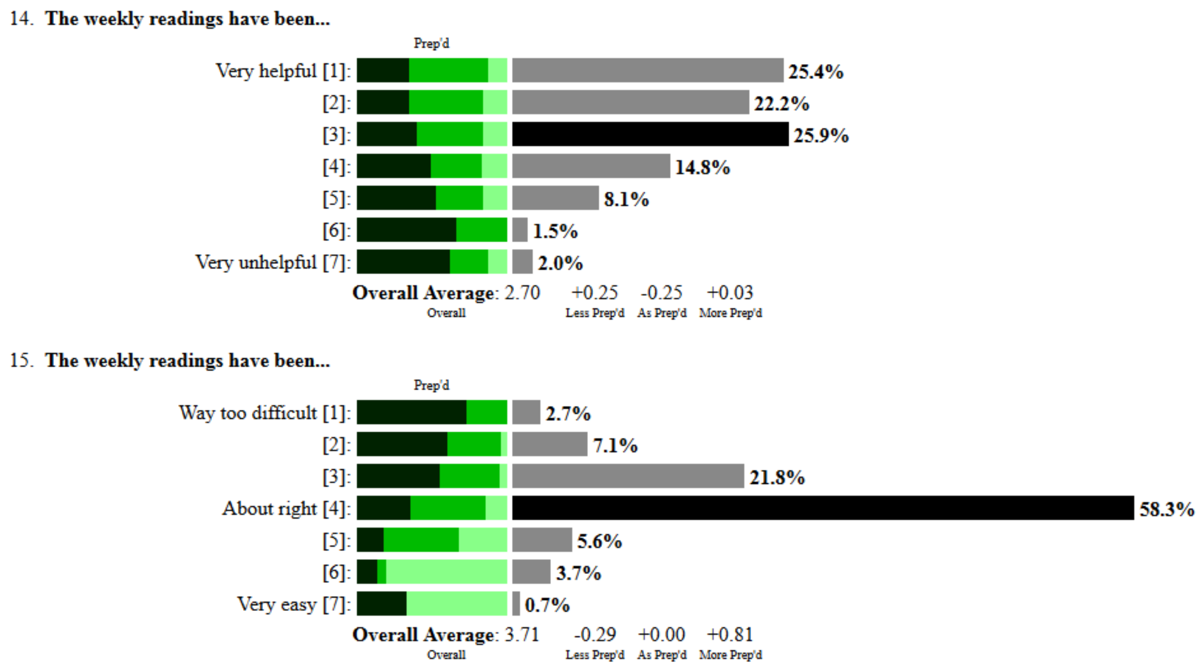


Figure 2: Length and helpfulness of weekly readings, as reported by students during Spring 2014 (anonymously survey, participation of 89% of enrolled students). On each answer, the stacked bar chart (labeled "Prep'd") represents students' self-reported preparedness for the course. The darker green/black portion consists of students who considered themselves "less prepared for the course than their peers" (40.8% of course), the middle green consists of students who consider themselves "as prepared" (44.6% of course), and the light green consists of the students who consider themselves "more prepared" (14.6% of course).

---

terms of difficulty, students found the readings quite helpful (2.70 / 7.00, where 1.00 is "Very Helpful") but found them the least helpful among the four aspects of the instructional activity "cycle" (Figure 2).

Following an introduction to the material through "pre-lecture" activities, an in-person lecture makes up the second part of each of the nine cycles. Following best practices from the WIDER community, lectures include significant active learning activities in each lecture: interactive examples, role playing exercises where student execute code by moving around the room and interacting with other students, clicker questions using i>clickers[4], and all of the slides and lecture activities available natively through a web browser. For this, the lecture slides are created using HTML5 and reveal.js[5] instead of using PowerPoint or PDF.

After seeing and experiencing the material two different times across two different days, students move to the third instructional activity: 50 minute long lab sessions. In lab, students are randomly placed into groups of 3-4 students. Labs contain no "lecture" component; students come in lab, get paired up, and begin working on an assignment designed to be challenging but achievable with the help of their group of peers. If a group is ever stuck and unable to progress, several members of the course staff are available to help students understand the concept so they are able to continue on with the lab. We have found that a single graduate student (Teaching Assistant or "TA") or undergraduate students who have recently taken the course (Course

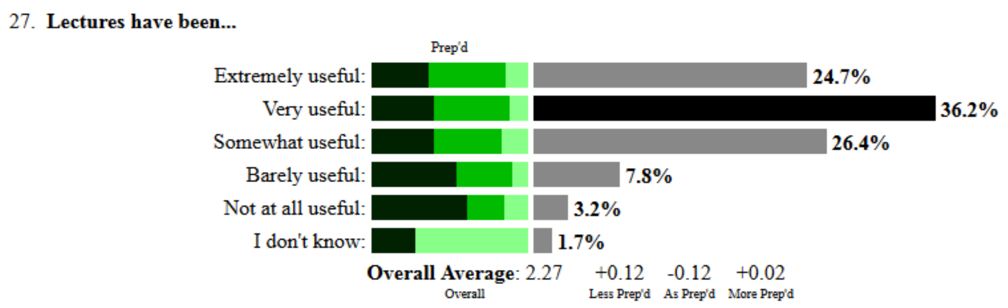


Figure 3: Usefulness of in-person lectures, as reported by students during Spring 2014 (anonymously survey, participation of 89% of enrolled students). See Figure 2 for additional details on the visualization of the data.

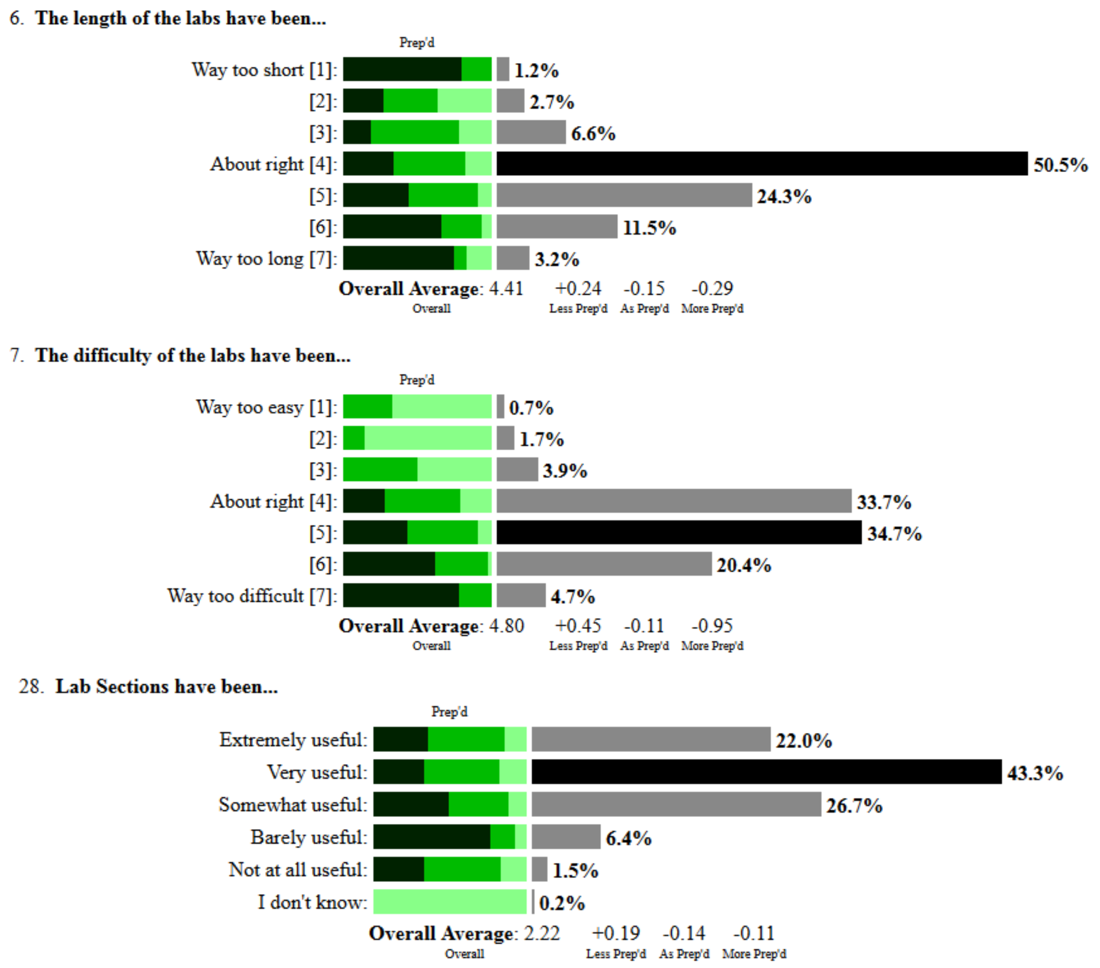
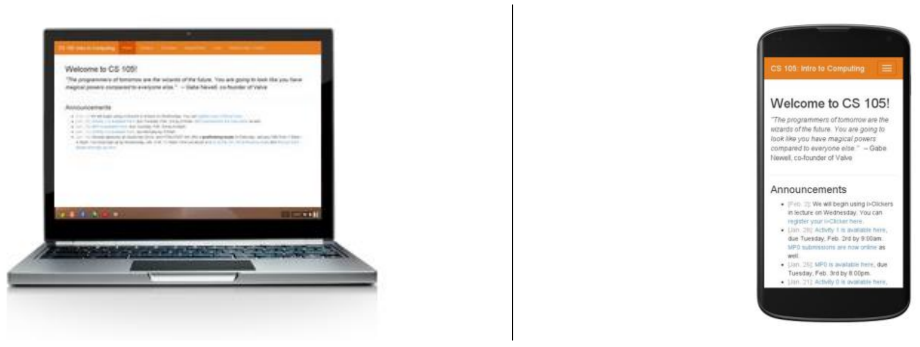


Figure 4: Length and difficultly of lab assignments, as reported by students during Spring 2014 (anonymously survey, participation of 89% of enrolled students). See Figure 2 for an explanation of the stacked green bars for each result. In both Q6 and Q7, a clear trend exists between the students' self-reported preparedness for the course and their opinion about the difficulty and length of the lab. However, Q28 shows that students' self-reported preparedness did not have a major impact on how useful they found the lab. We are encouraged that both strong students and weaker students find the labs nearly equally valuable.


CS 105: Intro to Computing

Welcome to CS 105!

"The programmers of tomorrow are the wizards of the future. You are going to look like you have magical powers compared to everyone else." – Gabe Newell, co-founder of Valve

Announcements

- [Feb. 3]: We will begin using i>Clickers in lecture on Wednesday. You can register your i>Clicker here.
- [Jan. 28]: Activity 1 is available here, due Tuesday, Feb. 2rd by 9:00am. MP0 submissions are now online as well.
- [Jan. 28]: MP0 is available here, due Tuesday, Feb. 3rd by 8:00pm.
- [Jan. 21]: Activity 0 is available here,

Figure 5: Rendering of the course website on a desktop/laptop platform (left) and on a mobile platform (right). Pages use responsive JavaScript and CSS to adapt to the screen the user uses to view course content. The entire course, including weekly readings, lecture slides, lab assignments, and MPs are hosted on this platform.

Assistant or "CA") are able to help 3-4 groups of students a piece. With each lab containing up to 42 students across 12 groups, an ideal staffing of the lab has been 1 TA being supported with 2-3 CAs.

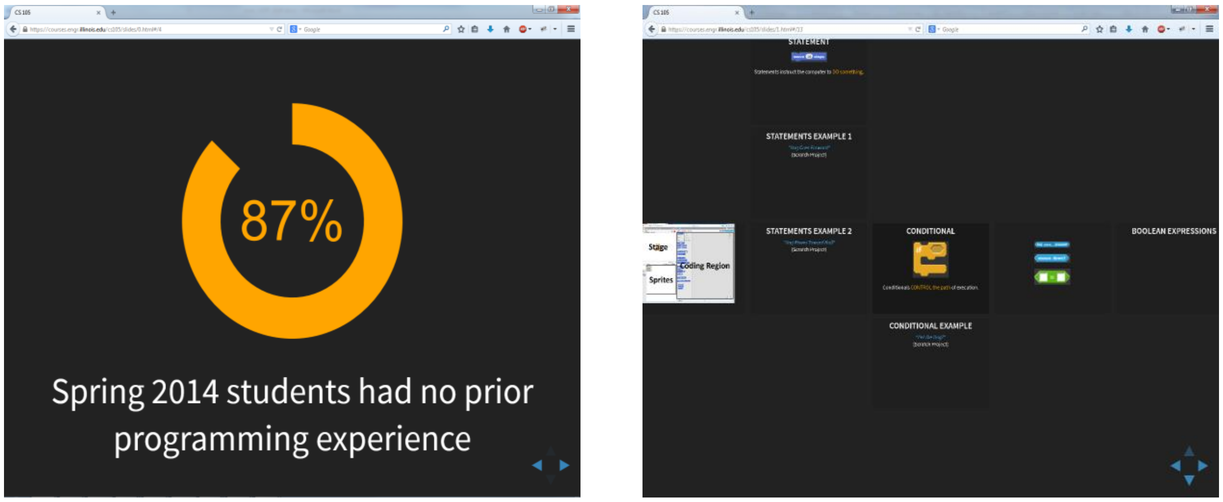
Finally, having experienced the material at least four times over the past week (pre-lecture, two lectures, and a lab), the students are required to complete a solo assignment. This assignment, called a "Machine Problem" or "MP", is usually the longest activity for most students in the "cycle". We aim for the MP to be of roughly equal difficulty as the lab assignments. Without the assistance of their peers, students report to take three to four times as long on the MPs than the labs.

As part of the implement-evaluate culture engendered by the WIDER community, we have collected a variety of data demonstrating the effectiveness of this "cycle"-based approach that was developed as part of the community of practice. The single most significant instrument for measuring this was an anonymous survey completed during the lab session. As the students had lab time dedicated to completing the survey, the survey participation was high (89% of all enrolled students). Various data from this survey is reported in Figures 2, 3, and 4.

3. Towards Creating an Extremely Accessible Course

As of 2014, the majority of courses at UIUC are offered though closed Learning Management Systems (LMSs) like Blackboard and Moodle. These LMSs are full of functionality, such as offering quiz and test modules, calendars, peer grading, and other useful tools for running a course. However, these large systems present many challenges to making the course accessible:

- Legacy Blackboard and Moodle applications have no mobile-friendly views, making navigation on small screens tedious. This discourages students from interacting with the course on mobile devices while walking, on a bus, or in lecture. Some features utilize browser plugins and are completely unavailable on mobile.


87%

Spring 2014 students had no prior programming experience

Figure 6: Lecture slides, rendered dynamically within a web browser using reveal.js. The use of reveal.js allows for non-linear slide sets that allow examples to "live under" to main slide. Renders identically on any size screen.

- All course content in traditional LMSs is required to be inside the secure area of the course, requiring students to log in every time to access course materials. This login process discourages frequent interaction with the course due to the overhead of getting to the course content and prevents students from using search engines to find the course.

- LMSs restrict access to previous semesters' content, decreasing the availability of materials to students to view, study, or review.

Given that these challenges are undesirable, we focused on replacing the benefits the LMSs provide with tools that were designed with a "mobile-first" approach. Figure 5 shows renderings of the course website on both a mobile and traditional desktop platform. This was done using largely supported, industry-standard tools such as Twitter's Bootstrap and jQuery. The basic framework for the course's web presence has already been reused for other courses within the community of practice.

As noted, the goal we set out for ourselves was not simply to have a course website that contains information about assignments that had to be done on a full-scale computer. Instead, every aspect of the course was placed within the web framework. Lecture slides (Figure 6) were created using reveal.js, JavaScript programming within a web browser was made possible using the ACE Code Editor by c9.io[11], and assignment submissions were done via simple HTML upload forms.

4. Continuing the Cycle of Improvement and Innovation within the Community of Practice

The success of the reengineering has seen the course enrollment jump to over 900 students in Fall 2014. In the spirit of WIDER, the community is now translating these practices to a follow-on "Data Driven Discovery" course at the sophomore level. This new course, serving as a practical approach to a CS2-style curriculum, enables students to understand the data structures necessary to do analytics on increasingly complicated data sources: structured text, images, social graphs, tabular data, and more.

In the same way that “Introduction to Computing” sought out best practices from other courses, “Data Driven Discovery” is doing the same. As a course that no longer focuses on basic computing skills, “Data Driven Discovery” enables students to create an individual portfolio of their computing achievements allowing their peers and employers alike to view the work they are capable of doing with a computer. This portfolio of work has been particularly appealing to students who are not technical majors but desire to show off their technical skills.

References

[1]: “This is CS 50”, Harvard University, https://cs50.harvard.edu/
[2]: “Beauty and Joy of Computing: BJC”, University of California at Berkeley, http://bjc.berkeley.edu/
[3]: “Scratch - Imagine, Program, Share”, Lifelong Kindergarten Group at the MIT Media Lab, http://scratch.mit.edu/
[4]: “i>Clicker: Clicker & Audience Response Systems”, iClicker, https://www1.iclicker.com/
[5]: “reveal.js - The HTML Presentation Framework” Hakim El Hattab, http://lab.hakim.se/reveal-js
[6]: Lave, J. and Wenger, E. Situated Learning: Legitimate Peripheral Participation. *Cambridge University Press*. 1991.
[7]: Wenger, E. and McDermott, R. and Snyder, W. M. Cultivating Communities of Practice. *Harvard Business Press.* 2002.
[8]: T. H. Davenport and L. Prusak. Working Knowledge: How Organizations Manage What They Know, second edition. *Harvard Business School Press*. 2000.
[9]: Wenger, E. Communities of Practice: Learning, Meaning, and Identity. *Cambridge University Press*. 1998.
[10]: L. E. Lesser and J. Storck. Communities of Practice and organizational performance. *IBM Systems Journal*. Volume 40. 2001.
[11]: “Ace – The High Performance Code Editor for the Web”, Cloud9 IDE, Inc, http://ace.c9.io/