2020

A Course Plan for Principles of IS Programming to Withstand COVID-19

Amy J. Connolly
James Madison University, conno3aj@jmu.edu

Leigh A. Mutchler
James Madison University, mutchlla@jmu.edu

Communications of the Association for Information Systems

Accepted Manuscript

A Course Plan for Principles of IS Programming to Withstand COVID-19

Amy J. Connolly
Computer Information Systems & Business Analytics
James Madison University
conno3aj@jmu.edu

Leigh A. Mutchler
Computer Information Systems & Business Analytics
James Madison University

Please cite this article as: Connolly, Amy; Mutchler, Leigh: A Course Plan for Principles of IS Programming to Withstand COVID-19, *Communications of the Association for Information Systems* (forthcoming), In Press.

Communications of the Association for Information Systems

Research Paper ISSN: 1529-3181

A Course Plan for Principles of IS Programming to Withstand COVID-19

Amy J. Connolly
Computer Information Systems & Business Analytics
James Madison University
conno3aj@jmu.edu

Leigh A. Mutchler
Computer Information Systems & Business Analytics
James Madison University

Abstract:

The COVID-19 pandemic has created a no-win situation for Fall semester classes: bring students back to campus and risk spreading the virus or teach online and risk insolvency. As faculty work through summer to plan even as situations change, flexible course plans must be built to weather COVID-19 while still meeting students' needs and expectations for teaching at their institutions. This paper discusses how Principles of IS Programming was quickly and successfully transitioned from a face-to-face course to fully online in Spring and how lessons learned from that transition will be applied to Fall semester. In addition, the paper describes how to organize the course and course schedule to maximize engagement and active learning while remaining agile enough to shift from face-to-face to online (or even vice-versa if the opportunity occurs). This paper is intended to help other IS instructors build a course plan to meet student needs for programming courses.

Keywords: Principles of programming, Active learning, Student engagement, COVID-19 pandemic

[Department statements, if appropriate, will be added by the editors. Teaching cases and panel reports will have a statement, which is also added by the editors.]

[Note: this page has no footnotes.]

This manuscript underwent [editorial/peer] review. It was received xx/xx/20xx and was with the authors for XX months for XX revisions. [firstname lastname] served as Associate Editor.] **or** The Associate Editor chose to remain anonymous.]

1 Introduction

In March 2020, in response to the COVID-19 global pandemic, faculty teaching face-to-face courses were abruptly mandated to transition to online. This mid-semester shift was (to put it mildly) overwhelming. In addition to rethinking assignments and schedules, everyone learned new ways to live and new vocabulary such as social distancing, mask wearing, and “flattening the curve.” If they were lucky, faculty had a week or two to prepare, prior training, and IT support to transition, and if they were agile enough and very lucky, everything worked.

Luckily, students in the spring semester said that out of all their courses, Principles of IS Programming felt like the smoothest transition to fully online. Now, universities have announced plans to reopen physical campuses in Fall 2020, despite the COVID-19 pandemic growing worse and all the science against it. Out of 1,150 universities in the U.S. being tracked by The Chronicle of Higher Education, very few have declared a completely online plan and many are doggedly determined to hold face-to-face instruction despite the costs and risks involved (Chronicle Staff 2020). At our institution, fall classes shall engage students in face-to-face, in-person classes at 50% occupancy in classrooms.

This paper describes how a face-to-face Principles of IS Programming was successfully transitioned to completely online in Spring 2020, and how to use lessons learned from Spring to meet requirements for Fall 2020, while still remaining flexible enough to pivot to fully online at a moment’s notice and without sacrificing student engagement, active learning, or rigor. This work is intended to advise instructors in how to build an active learning experience, particularly in (but not limited to) Principles of IS Programming.

2 Designing a Course Schedule with COVID Constraints in Mind

To design the course schedule, students’ expectations and abilities were weighed against their safety. At this university, students expect face-to-face instruction, personal communication, significant one-on-one interaction, and engagement with fellow students. Students entering the course are not expected to have any coding experience. Programming can be frustrating even for experienced programmers. These feelings can be worse for beginners who lack confidence and exacerbated further for women and minorities (Hamrick 2019; Rubio et al. 2015; Stiller 2009). Student success and hands-on assistance from the instructor positively correlate, especially at the beginning of programming courses, as well as other skill-building courses (Beaubouef and Mason 2005; Sparks 2019; Sriram 2015). Students who don’t get help are more likely to withdraw or fail, and even mild encouragement early-on can improve students’ commitment. It also helps students to dispel harmful stereotypes and beliefs about their abilities versus their peers (Alford et al. 2017). Because new material builds on prior material throughout the semester, students who misunderstand concepts at the beginning will struggle with later concepts. Therefore, it is important to closely monitor student success, in addition to safeguarding their health.

In Fall, this course is advised to include as much face-to-face class time as possible, despite COVID risks. To limit exposure to COVID, the university has limited classroom occupancy to 50% capacity. The first author teaches two sections of Principles of IS Programming, typically 75-minute sessions two days a week with 30 students per section. Students generally prefer live lectures rather than pre-recorded videos, but, due to the 50% room occupancy constraint, if given in-person, then lectures would have to be repeated in each section, effectively doubling the workload without added benefit.

Based on lessons learned from Spring and current constraints, the course plan is to lecture online to the entire class one day of the week and on the other day of the week, work problems with half the class (more details below). Lectures will be given synchronously during class time. Rather than 75 minutes of simply talking over slides, lectures will include hands-on problems and live demonstrations using a digital whiteboard to mimic the traditional use of white or chalk boards in class. Live coding demonstrations in the software development environment will be shown via screensharing. In programming courses, students are more successful in developing skills and tend to understand computation-intensive concepts better when they can observe problems being solved step-by-step, as opposed to reading a static slide. These sessions will be recorded for later viewing to accommodate students in case they experience issues that prevent attendance, such as loss of power, slow or unstable Internet, or even issues related to their current living space, such as multiple people competing for bandwidth. Additionally, programming demonstration videos can serve as a vetted reference for students as they work problems on their own throughout the semester.

In essence, this plan applies the traditional flipped classroom model in a new way, by starting from an online-only mindset and adding an optional face-to-face component, rather than asking how to move face-to-face instruction to online. Similar to a flipped model, students will read material and watch videos outside of class and then physically come to class to work problems. Unlike the typical flipped model, students will have an engaging and synchronous lecture period. Problems worked in-class will include homework assignments that would normally be done outside of class, thereby providing students with desired in-person instruction. In Spring, online class time was partly used for working programming problems rather than tests, thereby using in-class time more efficiently to improve students' coding skills. The in-person problem sessions can be moved online at a moment's notice without substantively affecting student learning; any student who wants to attend completely online is encouraged to do so and will not lose out on instruction in this model. For students who cannot attend in-person due to health issues, this model maximizes their ability to participate.

For the Fall plan to succeed, two things are vital: duplication of effort must be avoided and the class schedule must be preserved. First, material will be divided in such a way that students in the second section will not need to have watched the first section's lecture to understand theirs (although the two lectures will cover different material in the module and be given the same day, back-to-back). Secondly, it will help if students read or watch provided videos (more detail below) before coming to the lecture but this is not guaranteed. This schedule is presented in Table 1.

Table 1. Weekly Schedule for "Reverse-Flipped" Classroom

		First Class Day	**Second Class Day**
Week 1	*Section 01*	Lecture Part 1 online with entire class	Work problems in-person with Group B
	Section 02	Lecture Part 2 online with entire class	Work problems in-person with Group B
Week 2	*Section 01*	Work problems in-person with Group A	Lecture Part 1 online with entire class
	Section 02	Work problems in-person with Group A	Lecture Part 2 online with entire class
Students will be assigned to Group A or Group B in the LMS before classes begin.			

Students not physically in-class to work problems (Group A in Week 1 or Group B in Week 2) will not attend class in-person, but will be expected to virtually meet with their partners (more details below) to work homework assignments independently off-site. Students can choose to attend both online lectures if their schedules permit and all students can watch the recordings afterward. We designed the schedule in Table 1 in order to increase full-class engagement in online lectures, limit physical exposure to too many people at once, and fulfill the need for physical face-to-face class time, while still remaining flexible to switch to online at a moment's notice. Also note that students will only physically interact once every 2 weeks (the recommended COVID quarantine time) and the instructor doesn't need to use the "HyFlex" model, a model that requires instruction of in-person and online students while simultaneously managing the technology for online instruction. The HyFlex model was briefly considered, but based on our experience we determined it might be too distracting, complex, or prone to fail, particularly given the short notice to learn the new method.

This course plan relies on the following pillars of instruction: (1) a well-organized learning management site (LMS), (2) a mix of synchronous and asynchronous materials that persist, (3) flexible faculty availability (online office hours by appointment), and (4) clear instructions and rubrics for grading. Additionally, to build engagement and an open culture, particularly with the ever-shifting landscape of COVID restrictions, the instructor will increase communication and accountability. While these recommendations in general apply to any course with an online component, they are especially important for a course such as Principles of Programming, due to the breadth of material required for students to gain mastery of the subject matter.

3 How to Design the Learning Management Site (LMS)

Students in any course need to know what to do and when to do it. Reduce confusion by using the same LMS site as the institution (e.g., Canvas) and if other sites are used, embed links in the LMS whenever possible. Students appreciate an organized LMS that's easy to navigate with lots of resources and multiple ways to find help. There is a vast amount of reference content available on the Internet, especially for learning programming, so it is particularly important that students know exactly where to find support resources that are recommended and valid. Therefore, the LMS should serve as a "single source of truth" for students. When the move to online was announced in Spring, the first author's LMS was already well-organized with all assignments posted and ready for students' electronic submissions. Although this was an advantage over other instructors who did not rely as heavily on the LMS, it was still necessary to rethink how students would use the LMS resources and how additional support would best be provided before transitioning from face-to-face to completely online. The course content was already organized into modules (a.k.a. units), but each module needed to have a central page that listed learning objectives, readings (required, optional and additional), assignments (with due dates and direct links), and links to other resources. Lectures were moved online, and paper exams were transitioned to an online format. Upon entry to the LMS course site, students see a single centralized page to acclimate, starting with a syllabus "at a glance" page with links to course resources. Multiple sections of the same course are cross-listed (merged) into one course to avoid duplication of effort and enforce the "single source."

Key additional resources include videos which were curated from preexisting, online sources rather than created from scratch (to save time). These additional materials expanded opportunities for learning material, explained concepts in more detail, and exposed students to additional coding styles and techniques beyond those of the instructor. The short videos (less than 5 minutes each) explain specific programming content, concepts, and coding techniques. The length and description for each video is listed with a link to their location. Videos were checked for closed captioning, difficult to understand narrators, or inconsistent information.

4 Student Support and Engagement

It is crucial to monitor student participation and engagement in Principles of Programming, because student success depends on consistent practice to build coding skills. In this course plan, the instructor will closely monitor student participation in the course, particularly before major deadlines. What is different, however, is a greater reliance on the LMS to perform monitoring. The instructor can review the Canvas Gradebook and send reminder emails before due dates to anyone who has not yet submitted the assignment or to those who receive a certain grade to ask if they need additional help. Most LMS have indicators of student participation, such as a roster that shows when each student last logged in, how much time they spent online, and even which pages they viewed. Regularly checking for stragglers and emailing them to check-in (in some ways) replaces interactions that would normally be done physically in-class. If a disengaged student doesn't respond within a reasonable time, the Dean of Students can contact the student as a wellness check. The Spring course had two such stragglers. One of them fell out of touch due to depression and the first author was their only professor to take time to follow-up (via the Dean). This additional level of diligence encouraged the student to restart school in Fall rather than dropping out.

In Spring, students reported that interactions in the classroom with peers were an important part of their learning. To replace and formalize this support and engagement in Fall, a buddy system is being implemented. First, students will record a 30-second video to introduce themselves and show their workspace. Each student will then be assigned a partner to work with during the course. Pairs will be expected to meet virtually to complete homework problems that we will then discuss in class. These problems will be turned in for a small homework grade. This buddy system should help to not only increase student engagement and support, but also increase accountability throughout the course.

5 Give Clear Instructions and Rubrics

Building effective rubrics for assignments is time consuming, difficult, and absolutely necessary in courses like Principles of Programming. Designing the rubric to grade an assignment before handing it out helps identify problems with or confusing instructions in the assignment, similar to building test cases before building software. Use of online rubrics within the LMS makes them available to students before the due date. It is possible to word a rubric without giving away all of the answers, but if students understand the assignment, they will understand the rubric. (And if they don't, it creates a good learning opportunity.) Rubrics can be designed to enforce programming logic skills and as such are also learning tools. Diligent students use the rubric to improve assignments before submission, making grading easier. An added benefit is that someone else familiar with the course can take over at a moment's notice in case of emergency. With this in mind, the LMS course has an additional instructor who serves as an "understudy."

Exams in the past were given on paper in a face-to-face setting. The design of the exams demanded students memorize keywords and know how to program from memory. After the forced move to online in

Spring, exams had to be conducted online but it was clear that enforcing a closed-book policy was impractical. Students were allowed to use Internet resources and their notes, and the development environment could be used to run code. The exam questions were rewritten to test each student's understanding of the code's inner workings, and to ensure just running the code to get the "right" answer would not suffice.

Some universities, including ours, have engaged the services of an online exam proctoring service for Fall, but even these do not prevent cheating. To reduce the likelihood of cheating on online exams, we recommend the following steps. First, the course is designed with the goal of building a supportive class culture by encouraging students to share their ideas, to ask questions, and to help each other learn. This growth-mindset coupled with flexibility in scheduling makes goals more achievable. Online tests are setup to be available one to three days ahead of the due date but timed to class-time (50 or 75 minutes). This allows students the flexibility to work around connectivity and scheduling issues but still demands that they study for the test to pass. Questions on the exam can be built with numeric problems and answers that change for each student. Students were challenged most often by questions like "select all that apply" (versus which of these is not), being asked to show intermediate calculations (versus only deriving the final answer) and requiring explanations of causality. Challenging, thought-provoking questions on exams should help reduce the likelihood of cheating.

Providing lots of feedback to students is critical in a programming course; therefore, to ensure there is time to grade and provide meaningful feedback, schedule due dates on specific days of the week that meet the instructor's grading schedule. There is no good day of the week to schedule due dates; every student wants a different day. As part of a well-organized course, all assignments of a specific type are (as much as possible and working around holidays) scheduled on the same day of the week. For example, tests on Mondays, projects on Wednesdays, reading quizzes on Fridays, etc. This regular pattern provides consistency and stability. For additional suggestions to teach principles of programming, see the recently published teaching tips by (Sharma et al. 2020; Zhang et al. 2020), from which we adapted concepts in this course plan.

6 Conclusion

The experiences of the Spring semester made it clear to us that now more than ever we need to be flexible and reasonable as well as transparent with our students. They will miss deadlines or class due to physical illness, mental stress, loss of power or Internet, or family emergencies. We are all grappling with similar issues in trying to live through COVID. In Spring when students missed a due date, the no-late work policy was relaxed. Students appreciated a reasonable second chance and didn't press for more. We tried to be open about why some assignments might be turned in late but others couldn't, for whatever reason. Unfortunately, the future is uncertain and the constraints placed on us are many. But by building a course plan using an atypical, reverse-flipped model and by designing activities that are not wholly dependent on a face-to-face or online modality, we hope to remain flexible and meet the challenges posed this Fall. We have developed a course plan for teaching principles of IS programming that is nimble and should easily withstand pivoting back to online, if needed. By sharing that plan here, we hope we have provided others with useful advice to successfully do the same for their courses. Good luck out there.

References

Alford, L. K., Dorf, M. L., and Bertacco, V. 2017. "Student Perceptions of Their Abilities and Learning Environment in Large Introductory Computer Programming Courses," in ASEE Annual Conference and Exposition, Conference Proceedings.

Beaubouef, T., and Mason, J. 2005. "Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations," ACM SIGCSE Bulletin (37:2), pp. 103–106.

Chronicle Staff. 2020. "Here's a List of Colleges' Plans for Reopening in the Fall," The Chronicle of Higher Education. (https://www.chronicle.com/article/Here-s-a-List-of-Colleges-/248626).

Hamrick, K. 2019. "Women, Minorities, and Persons with Disabilities in Science and Engineering: 2019 | NSF - National Science Foundation," No. Special Report NSF 19-304, Alexandria, VA: National Science Foundation. (https://ncses.nsf.gov/pubs/nsf19304/digest).

Rubio, M. A., Romero-Zaliz, R., Mañoso, C., and de Madrid, A. P. 2015. "Closing the Gender Gap in an Introductory Programming Course," Computers & Education (82), pp. 409–420.

Sharma, M., Biros, D., Ayyalasomayajula, S., and Dalal, N. 2020. Teaching Programming to the Post-Millennial Generation: Pedagogic Considerations for an IS Course, (31), p. 12.

Sparks, S. D. 2019. "Why Teacher-Student Relationships Matter," Education Week. (https://www.edweek.org/ew/articles/2019/03/13/why-teacher-student-relationships-matter.html).

Sriram. 2015. "5 Tips to Prevent Student Attrition and Dropouts in Colleges | Creatrix Campus," , May 20. (https://www.creatrixcampus.com/blog/5-tips-prevent-student-attrition-and-dropouts-colleges, accessed July 15, 2020).

Stiller, E. 2009. "Teaching Programming Using Bricolage," Journal of Computing Sciences in Colleges (24:6), pp. 35–42.

Zhang, X., Crabtree, J. D., Terwilliger, M. G., and Jenkins, J. T. 2020. Teaching Introductory Programming from A to Z: Twenty-Six Tips from the Trenches, (31), p. 15.

About the Authors

Amy J. Connolly is an Assistant Professor of Computer Information Systems and Business Analytics in the College of Business at James Madison University. Her doctorate is in Management Information Systems from the University of South Florida. Her research interests include the role of social media in volunteer organizations and active learning and inclusion in information systems pedagogy. She has published in journals including *European Journal of Information Systems, Journal of IS Education, Information Systems Education Journal,* and *Informing Faculty*. In 2019, she was awarded the prestigious Stafford Beer Medal from The Operational Research Society and named Reviewer of the Year by The Data Base for Advances in Information Systems.

Leigh A. Mutchler is an Assistant Professor of Computer Information Systems and Business Analytics in the College of Business at James Madison University. She received her MSIS and Ph.D. from the Department of Management and Information Systems at Mississippi State University. Her research interests are primarily in the areas of information security end user behaviors and Awareness instruction. She has published in the *Communications of the Association for Information Systems,* the *Journal of Computer Information Systems, Information and Computer Security,* and the *Journal of Database Management*. Leigh has presented her works at local and national conferences, including the IFIP Dewald Roode Workshop on Information Systems Security Research, the Americas Conference on Information Systems, the National Decision Sciences Institute Annual Conference, the Annual Meeting of Southeast Decision Sciences Institute, and the Conference on Information Systems Applied Research. Dr. Mutchler serves as a reviewer for multiple national and international journals and IS conferences.