ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #36918

Cloud technologies for scalable engagement and learning in flipped classrooms

Sotiria Koloutsou-vakakis (Dr.)

Sotiria Koloutsou-Vakakis (Ph.D.) is a Senior Lecturer and Research Scientist in Civil and Environmental Engineering, at the University of Illinois at Urbana-Champaign. She holds degrees in Civil-Surveying Engineering, Geography and Environmental Engineering. Her most recent research is about gaseous emissions of reactive nitrogen from fertilized fields into the atmosphere and impacts on air quality and climate change. She teaches undergraduate and graduate courses on Air Quality, Science and Environmental Policy, and Engineering Risk and Uncertainty and is active with K-12 outreach. She has strong interest in engineering education. She develops materials and researches best practical classroom approaches for integrating computation and computational thinking in introductory CEE courses; and for promoting teamwork, communication and problem-solving in context, throughout the CEE curriculum.

Christopher Tessum

Eleftheria Kontou

Hadi Meidani

Hadi Meidani is an Associate Professor in the Department of Civil and Environmental Engineering at the University of Illinois at Urbana-Champaign. He obtained his Ph.D. in Civil Engineering and his M.S. in Electrical Engineering from the University of Southern California (USC) in 2012 and also a M.S. in Structural Engineering from USC. After his Ph.D. he was a postdoctoral research associate in the Department of Aerospace and Mechanical Engineering at USC in (2012-2013) and in the Scientific Computing and Imaging Institute at the University of Utah (2013-2014). He is the recipient of the NSF CAREER Award to study fast computational models for energy-transportation systems. His research interests are uncertainty quantification, scientific machine learning, computational modeling of civil infrastructure systems, and resilient infrastructures.

Lei Zhao

Dr. Lei Zhao is an Assistant Professor in the Department of Civil and Environmental Engineering at the University of Illinois at Urbana-Champaign (UIUC). His research concerns the physical and engineering processes in the Atmospheric Boundary Layer where most human activities and environmental systems are concentrated, with a particular focus on built surfaces and urban environments. He combines theory, numerical modeling, remote sensing and in situ observations, and cutting-edge machine learning methods to study environmental fluid mechanics and land-atmosphere dynamics that relate to urban environments, climatology and hydrology, climate change, climate impacts and adaptation. He received his Ph.D. (2015) in Atmospheric and Environmental Science from School of the Environment at Yale University. Before joining at UIUC, Dr. Zhao was a postdoctoral research fellow in the Program in Science, Technology and Environmental Policy (STEP) at Princeton University. Dr. Zhao obtained his B.S. degree (2009) in Physics and Atmospheric Physics from Nanjing University in China.

Cloud technologies for scalable engagement and learning in flipped classrooms

Sotiria Koloutsou-Vakakis, Hadi Meidani, Eleftheria Kontou, Lei Zhao, Christopher W. Tessum
Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 N. Mathews, Urbana, IL 61801, sotiriak@illinois.edu

1. Introduction

Engineers of the future need be equipped with the culture and skills for a fast-changing and uncertain professional, socioeconomic and natural global environment. This need is fueled by rapid changes in science and technology and by pressures on societies to respond to emerging environmental crises and situations such as the recent pandemic. In the engineering classrooms, this translates into very high expectations on students and educators alike. Technology in the classroom has been used to help attain learning objectives, equip students with practical skills valued by employers, and link communities of learners. However, this diversity of learning and communication tools and information-rich learning environments can lead to cognitive overload. Cognitive load refers to the bandwidth of our working memory. Cognitive overload occurs when working memory demands exceed working memory capacity, causing learning and performance to suffer [1], [2]. Two types of cognitive load are mentioned in the literature *intrinsic* and *extraneous* [3]. Intrinsic cognitive load mostly affects students who are new to a subject and they have not yet constructed networks to connect ideas, theories, facts and figures. Extraneous cognitive load refers to factors not necessary for learning that can be altered by instructional interventions [3]. For example, extraneous cognitive overload can be caused by factors that make processing of information difficult, such as unclear course expectations, confusing LMS (learning management system) interface, poorly explained assignments, continuous switching of ICT (information and communication technology) tools, long pre-recorded lectures, inadequate online teaching methods, unnecessary distractions (e.g., provision of superfluous information or diverting attention to check something online) [4].

Combating extraneous cognitive overload in courses that use computational tools (on physical or cloud platforms) is connected to *usability* of these tools. With reference to software systems, the term 'usability' has been broadly used to include execution time, performance, user satisfaction and ease of learning [5]. While older literature focused on software attributes, recent literature includes student and teacher user interaction, perceived usefulness and satisfaction. For example, in [6] six factors are identified that affect e-learning usability: information quality, system navigation, system learnability, visual design, instructional assessment, and system interactivity.

In this presentation, we demonstrate the online environment we use in two second year civil and environmental engineering (CEE) courses that have enabled teaching computational thinking in the CEE context. We focus on an example approach for reducing extraneous cognitive overload: switching from the default in-class code developing and testing environment to one enabled recently in the learning platform that allows easy access to Jupyter workspaces [7] (JW). This presentation provides an example of technologies that enable teaching and learning of computation thinking regardless of class size. A demonstration class has been created on the ICT

learning platform for readers who wish to have a hands-on experience with the tools we describe (see Appendix).

2. Background and overview of tools and pedagogies we use

Already before the pandemic, our departmental curriculum committee had been assembled as the primary community of practice (COP) to launch a department-wide curriculum innovation effort. Two second year CEE courses became the foundation for this effort. Curriculum innovation was launched with three major focal points: computational thinking, communication, and experiential learning. To make the changes scalable and sustainable, we created course-specific COPs among course instructors. Materials are developed as a result of coordinated efforts among these course specific communities of practice and the courses are taught the same way, no matter who teaches the course each semester, with continuously evolving materials and pedagogies across semesters. We also adopted a student-centered teaching model [8].

We have previously presented our approach for integrating computational thinking with the fundamental content matter of our courses and for shifting to student-centered learning, along with cognitive and affective learning outcomes, after first implementation [9]. The two courses we initially redesigned cover introductions to (1) systems engineering and economics and (2) engineering risk and uncertainty. Both classes are offered every Fall and Spring semester. Python 3 is used in the first course and R in the second. We chose these two coding languages, because they are currently prevalent but also considering coding language use in subsequent courses, as curriculum innovation efforts propagate to upper-level courses. We also intended to communicate to students that 1) computing literacy is more than the specific language used; 2) different tools are better suited for different applications; 3) different tools can work together, taking advantage of each tool's strengths for a given application; 4) computer languages keep evolving, requiring everyone to adopt new tools using essential common background knowledge.

Developing the learning online environments for the two courses is an evolving iterative process. A major guiding factor was the negligible coding experience of the vast majority of our incoming students. Therefore, user interface friendliness and simplicity (broadly included in the tool usability concept) are very important. The generalized steps we follow in choosing tools and teaching approaches are: 1) identification and articulation of learning objectives to prepare students for a rapidly changing physical, social and technological environment; 2) creation of an instructor community of practice for the specification of a strategic sustainable frame for action; 3) consideration of the diversity of student backgrounds in our early (first and second year) courses; 4) research of pedagogies to enhance student engagement; 5) research of campus available learning tools to enable our teaching goals and learning objectives; 6) gradual development of educational materials; 7) adoption of a continuous learning, iteration, improvement process, as we keep learning from earlier implementations, using student survey feedback, student assessment results and our own classroom observations.

Choice of LMS and ICT tools is important to support the sustainability of changes and to provide an appropriate environment for computational thinking practice and learning. Specifically, our choice criteria for LMS and ICT included: 1) ability to support teaching of computational thinking and practice; 2) cost for use; 3) use by other courses our students take in earlier

semesters or simultaneously. The latter is important for reducing student stress and confusion induced by use of multiple learning environments for different courses.

Currently, we use Canvas [10] as a repository for course resource materials. Modules are organized by week, consistent with the syllabus, and contain study materials and links to assignments with reminders of due dates/times. We use Prairie Learn (PL) [11], an open source learning environment, to host pre-lecture videos, in-class worksheets, homework assignments and not-for grade practice homework problems. PL supports automatic and manual grading for a variety of problem types: multiple-choice, numeric, coding and symbolic. PL has been a major enabling factor of our efforts to introduce and create an accepting culture for computational thinking, among CEE students. It was also the essential enabling factor for the online shift of operations due to the COVID-19 pandemic.

In our student-centered learning approach, students watch short pre-lecture videos and answer checkpoint questions. Then, in class, after an overview of the key concepts of the day, students work on problem solving, for understanding the new material at a deeper level. In class-worksheets are made available on PL and as printable documents. Answers are submitted on PL for grading. In-class worksheets are formative assessments. Unlimited multiple attempts are allowed, and students get full credit as long as, they complete a certain percentage of the problems correctly. Solved worksheet problems become available on Canvas as study material, soon after class. Summative assignments comprise of weekly homeworks, exams and an assigned team project. Homework and exam problems are randomized to minimize cheating [12].

Here, we focus on the in-class student problem solving period. Our in-class worksheet approach is inspired by that described in [13] about self-regulated learning, even though we refer to a single class session with short problems rather than bigger projects. English and Kitsantas [13] distinguish three phases in the classroom environment: problem launch, guided inquiry/solution creation and problem conclusion that correspond to three student processing phases: forethought, performance and reflection, respectively.

In our class implementation, in phase 1 (overview), students initiate an action plan for solving the problems using the tools made available to them. This is after the instructor presents a brief overview of the concepts students first learned about by watching the pre-lecture videos. The overview is a reminder and a lead to help students use that early knowledge to solve a problem. In phase 2 (problem solving), students arrive to a computationally assisted solution through careful reading of the problem, discussion with their teammates and experimentation that involves learning iteratively by erring, reasoning about where they erred and why, and finally correcting and arriving to the solution. At this stage, the instructor and TAs walk around the room or visit online breakout rooms, to provide prompts and answer questions but not to solve the problems for the students. In phase 3 (reflection – communication of ‘muddiest point’), the goal is for students to reflect on the materials and activities of the day about what they learned and about what remaining questions they have about the key concepts, the solution process or the tools they have available to help them solve a problem. Answers to muddiest point questions are posted on Canvas and the most common ones are discussed in the following class.

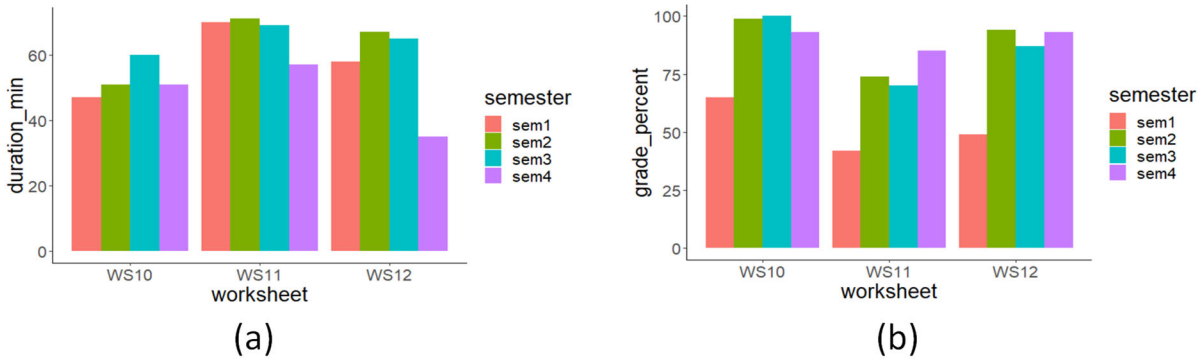
In our initial implementation, using the default PL tools, we observed that students were struggling with organizing their laptop workspaces, despite our guidance about how to organize and manage their laptop screens. The workflow required that students needed to switch windows multiple times during class between the software on their laptops and the PL web browser window. A few students were vocal about these difficulties, while others were silently adjusting. It was apparent to us that even for the students who were accepting the original workflow as a fact of life, the need to manage multiple software and windows on a single class session was creating distraction and slowed down the learning process, becoming a cause of extraneous cognitive overload.

3. Implementation of JWs and preliminary observations

The addition of access to JW from inside PL enabled improved streamlining of in-class workflow for students. Use of JWs allows students to access their worksheets from a single browser window using two tabs. Students can complete, test and submit code for autograding just using these two tabs. In addition, the cloud environment offers independence from the laptop environment. This is another important consideration because student struggles with laptop specific issues that arise during class time detract from learning by shifting student attention to troubleshooting their machines. In the Appendix, we present instructions for interested readers to access an example and gain a sense of the student experience, while working on an in-class worksheet using PL. We provide two versions of the example, the 'original' one that requires availability of R/RStudio on one's laptop to be able to submit and grade answers, and the 'improved' workflow using PL JWs, where one needs only a browser to access PL.

Next, we present early observations about student interactivity with the new approach using two metrics: *duration* for completing a worksheet and *grade* upon submission. We use duration as an indicator of workflow ease and grade as an indicator of learning. In Figure 1, we present these metrics for three worksheets across four semesters. The new approach was used in semester 4. Our data do not allow statistical evaluation, yet, but results are in the desired direction. Even though differences in duration and grade depend on the requirements and difficulty of questions in each worksheet, observationally, we note a general trend indicating reduced duration and higher grade, when JWs are used. We also surveyed the students in semester 4 asking the question "Please, indicate your preference about how you like the Jupyter Workspace questions we currently add to PL worksheets." On a scale from 1 (I do not like) to 10 (I like best), 77% of students gave a rating of 6 or above, with 36% rating with 9 and above.

Figure 1. Comparison of mean duration (a) from beginning of class until worksheet (WS) submission and grade (b), over four semesters. Worksheets using Jupyter Workspaces were used in semester 4.



4. Discussion

Introduction of computational thinking and practice in non-Computer Science majors presents challenges for instructors which include the development of discipline relevant materials and overcoming student resistance under the false perception that they do not need coding in their chosen discipline. This is particularly true for the classes that are the first to transition to a new teaching and learning environment. What we have witnessed is that these initial reactions fade after a couple of semesters of implementation [9], as student culture and expectations change. However, it is important to continue staying tuned and responsive to student perceptions and responses. When online learning platforms are used, ease of access and use are important for reducing student anxiety and extraneous cognitive load. It takes time to conclusively state the effectiveness of a given educational improvement effort, especially one where research, development and implementation occur simultaneously. At this stage, our example of an intervention aimed to reduce cognitive load appears to have a positive effect on optimizing student workflow and learning, during class, for CEE students.

From the instructors' perspective, staying responsive to student perceptions and responses, requires significant time commitment and increased workload. For example, our transition to JWs was hardly straightforward, as it first, required re-configuration of the PL R autograder, in addition to re-creating the PL questions. This seems to be a persistent issue, for teaching with technology because rapid technological advances reduce the lifetime of materials developed on learning platforms. This is beyond the scope of this presentation, but it is a challenge for the user-instructor communities, as well as for the platform developer communities. In addition to *usability*, as described in the beginning of the presentation, *sustainability* of learning platforms needs become an important consideration, with consideration to the instructor user of the platforms. Nevertheless, what we have learned from this effort is that use of cloud learning environments during class offers advantages such as, ease of access, student team collaboration, and simplification of student workflows. This works well for in-person, online, or hybrid types of courses and is scalable for all class sizes.

Acknowledgments

The online course PL environments for the two courses have been made possible thanks to the contributions of many graduate teaching assistants who have coded questions. Specifically, for this presentation, we acknowledge the contributions of *Priyam Mazumbar*, who helped create the Jupyter workspace example and *Advai Podduturi* who enabled autograding of R Jupyter workspace cells, building upon the R autograder developed by *Dirk Eddelbuettel* for PL: https://github.com/eddelbuettel/pl-r-demos

References

[1] J. Sweller, "Cognitive load during problem solving: Effects on learning" *Cognitive Science,* vol. 12, no. 2, p. 29, 1988. https://doi.org/10.1016/0364-0213(88)90023-7.

[2] J. L. Sewell, L. Santhosh, and P. S. O'Sullivan, "How do attending physicians describe cognitive overload among their workplace learners?" *Med Educ*. 54: 1129– 1136, 2020. https://doi.org/10.1111/medu.14289.

[3] J. Eyler, "How Humans Learn: The Science and Stories Behind Effective College Teaching". United States, West Virginia University Press, 2018.

[4] V. Iturbe-LaGrave, "Teaching Through a Pandemic: Cognitive Load, Mental Health and Learning Under Stress" Apr. 2020. [Online]. Available: https://otl.du.edu/teaching-through-a-pandemic-cognitive-load-mental-health-and-learning-under-stress/ . [Accessed Dec. 29, 2021].

[5] A. Abran, A. Khelifi, W. Suryn, and A. Seffah, "Usability meanings and interpretations in ISO standards". *Software Quality Journal*, vol. 11, pp. 325–338, 2003.

[6] A. Alshehri, M. Rutter, and S. Smith, "Assessing the Relative Importance of an E-learning system's Usability Design Characteristics Based on Students' Preferences", *European Journal of Educational Research*, vol. 8, no. 3, pp. 839-855, 2019. https://doi.org/10.12973/eu-jer.8.3.839

[7] T. Kluyver T, B. Ragan-Kelley, P. Fernando, B. Granger, M. Bussonnier, J. Frederic, K. Kyle , J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila , S. Abdalla, and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides, B. Schmidt, Ed. IOS Press, 2016, pp. 87–90.

[8] A. Karabulut-Ilgu, N. Jaramillo Cherrez, and C. T. Jahren, "A systematic review of research on the flipped learning method in engineering education," *British Journal of Educational Technology,* vol. 49, no. 3, p. 13, 2018.

[9] S. Koloutsou-Vakakis, E. Kontou, C. W. Tessum, L. Zhao, and H. Meidani, "Educational Technology Platforms and Shift in Pedagogical Approach to Support Computing Integration Into Two Sophomore Civil and Environmental Engineering Courses", 2021. Paper presented at 2021 ASEE Virtual Annual Conference Content Access, Virtual Conference. https://peer.asee.org/37005.

[10] Instructure, "Teaching and Learning. To the Power of Canvas LMS", 2022. https://www.instructure.com/canvas. [Accessed Mar. 16, 2022].

[11] M. West, G. L. Herman, and C. B. Zilles, “PL: Mastery-based Online Problem Solving with Adaptive Scoring and Recommendations Driven by Machine Learning”, *122nd ASEE Annual Conference and Exposition, Conference Proceedings*: *Making Value for Society, 14-17 June 2015*. 10.18260/p.24575. 2015.

[12] B. Chen, M. West, and C. Zilles, "How much randomization is needed to deter collaborative cheating on asynchronous exams?", presented at the *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, 2018.

[13] M. C. English and A. Kitsantas, "Supporting Student Self-Regulated Learning in Problem- and Project-Based Learning," *Interdisciplinary Journal of Problem-Based Learning*, vol. 7, no. 2, 2013.

Appendix. Instructions for accessing the demo course

1. Go to https://www.prairielearn.org/ .
2. Click ‘Log in’.
3. Sign in with a Google or Microsoft account.
4. Once logged in, click on ‘Add or remove courses’.
5. In the course list, find ‘CEE 201/202DM: CEE201/202 demo course, CEE 201-202 Demo’ and click ‘Add course’. In the popup window confirm the addition.
6. Hit the browser back button to go to the first PL page you landed, after login. You will see the course listed.
7. Hit the course link and you will then be able to access and explore the demo assignments we have created.
8. Click Assignments in the top menu to see the example assignments listed and just follow the instructions.
9. For your convenience with testing, the complete code for the example problems is available on PL to copy/paste and submit for grading, so that you gain the full experience of using the platform.

Notes:
We have created two versions of the same assignment, one using the default PL environment and the other using the Jupyter Workspace (JW) option.

In the default version, users not familiar with R will not be able to produce the graph files. The graphs can be produced in the R or RStudio [14] environment, be saved in an appropriately named .pdf file and uploaded to PL, where then they need to be checked by the instructor manually, after class. This is an inconvenience that is eliminated, when we use JWs. In the JW version, coping/pasting the code we provide and running each cell sequentially will provide both numerical and graph output. In JW the graphs are also autogradable through testing for the values of the variables that are plotted.