Solution Diversity in Engineering Computing Final Projects

Ms. Sara Willner-Giwerc , Tufts University

Sara Willner-Giwerc is a Ph.D. candidate in mechanical engineering at Tufts University. She graduated from Tufts University with a B.S. in mechanical engineering and a double minor in engineering education and engineering management in 2018. She is a National Science Foundation Graduate Research Fellow, which supports her research at the Tufts Center for Engineering Education and Outreach (CEEO) on technological tools, learning experiences, and environments for teaching engineering in classrooms pre-k through college.

Dr. Kristen B Wendell, Tufts University

Kristen Wendell is Associate Professor of Mechanical Engineering and Adjunct Associate Professor of Education at Tufts University. Her research efforts at at the Center for Engineering Education and Outreach focus on supporting discourse and design practices during K-12, teacher education, and college-level engineering learning experiences, and increasing access to engineering in the elementary school experience, especially in under-resourced schools. In 2016 she was a recipient of the U.S. Presidential Early Career Award for Scientists and Engineers (PECASE). https://engineering.tufts.edu/me/people/faculty/kristen-bethke-wendell

Prof. Chris Buergin Rogers, Tufts University

Chris is a professor of Mechanical Engineering at Tufts University with research in engineering education, robotics, musical instrument design, IoT, and anything else that sounds fun.

Dr. Ethan E Danahy, Tufts University

Ethan Danahy is a Research Assistant Professor in the Department Computer Science at Tufts University outside of Boston MA, having received the B.S. and M.S. degrees in Computer Science in 2000 and 2002, respectively, and a Ph.D. degree in Electrical Engineering in 2007, all from Tufts. Additionally, he acts as the Engineering Research Program Director at the Center for Engineering Education and Outreach (CEEO), where he manages educational technology development projects while researching innovative and interactive techniques for assisting teachers with performing engineering education and communicating robotics concepts to students spanning the K-12 through university age range.

Isabella Stuopis, Tufts University

PhD Candidate in Mechanical Engineering at Tufts University. Interests: undergraduate learning, learning outside of the classroom setting, collaboration in engineering, learning assistants

Solution Diversity in Engineering Computing Final Projects

Abstract

This research paper describes a solution diversity analysis of the final projects completed in an undergraduate engineering computing course. The course was taught by four different professors using three distinct instructional approaches. One professor used a distributed-expertise model, and the other three used traditional content-delivery methods. A distributed expertise model is a pedagogical approach where students specialize in different topic areas within the broader course subject. In the traditional sections, students were free to choose any application of computational thinking in which they were interested for their final project. Based on an analysis of the final projects submitted by students, these projects had a high diversity of applications but the computational tools that students selected for their analyses were very similar across all projects. In the distributed-expertise section, students were given a very scaffolded design problem as their final project. Each group of four students completed the same challenge; however, the ways they pieced together their distributed knowledge to solve the problem varied widely between groups. In other words, there was low solution diversity in application and a high diversity of processes used to solve the problem. This case study explores how the solution diversity of students' final project submissions varied across both the distributed-expertise section and the lecture-based and problem-based sections, and the ways in which the instructional models and final project prompts did or did not afford solution diversity in the final projects.

Introduction

In undergraduate engineering curricula, design projects provide opportunities for students to demonstrate understanding of their technical knowledge through solving a complex problem [1]. Additionally, project-based learning allows students to acquire and apply valuable non-technical skills such as teamwork, systems thinking, communication, ethics, and creativity [2, 3]. Industry demands that students be able to engage effectively in the practice of engineering, which includes not only technical knowledge but also the ability to apply that knowledge to new and complex situations in the real world [3, 4]. Therefore, the teaching of engineering should focus on getting students to think independently, rather than simply asking students to replicate the existing knowledge of others [5].

All accredited universities in the United States are required to provide students with the opportunity to complete "a culminating major engineering design experience that (1) incorporates appropriate engineering standards and multiple constraints, and (2) is based on the knowledge and skills acquired in earlier course work." The ABET standards state that

“engineering design involves identifying opportunities, developing requirements, performing analysis and synthesis, generating multiple solutions, evaluating solutions against requirements, considering risks, and making trade-offs, for the purpose of obtaining a high-quality solution under the given circumstances” [6]. Many institutions have transitioned from one large capstone design experience to several cornerstone design experiences which provide students with the opportunity to do several larger design projects throughout their undergraduate careers [7,8]. An increasing amount of technical courses have also begun leveraging design projects in addition to, or instead of, traditional end of semester evaluations in an attempt to give students different kinds of opportunities to apply and demonstrate their engineering knowledge [9,10].

While the value of design projects in engineering curricula is accepted by most institutions, the actual design of these experiences is not necessarily straightforward [11]. Designing final projects in a way that authentically requires students to apply their technical knowledge while simultaneously fostering the 21st century skills that are needed for the workplace can be difficult for instructors. One of the many challenges instructors face is how to know whether or not students are meaningfully engaged in the practice of engineering through a final engineering design project [12]. Looking at the diversity of solutions produced by students in regards to a final project prompt is one potential way of knowing if students are thinking independently and engaging in engineering practices [13]. If all of the students in a class are creating exactly the same solution to an engineering problem (low/no solution diversity), that may be an indicator that there are too many constraints on the problem or that students aren’t truly thinking for themselves. A high solution diversity, where students are generating many different kinds of solutions to the same problem, indicates an authentic prompt and that students are thinking independently.

In this qualitative case study, we analyze the final project submissions from all students in a first year computing in engineering course. The sections of this course were taught with different instructional approaches and students were given different final projects in each section. Through the analysis of student survey responses, student submissions for the final project, and classroom observations, our goal is to understand the different ways in which the instructional approaches afforded solution diversity in the final project submissions. In this analysis, we divide solution diversity into three categories: *diversity of problem*, *diversity of process*, and *diversity of product* and use them as a way to offer insight into what aspects of a final project did or did not facilitate student engagement in engineering practices that may otherwise have gone unnoticed. *Diversity of problem* is determined by how many different specific problems students in a class define within a dictated topic or problem space. *Diversity of process* is based on the different design decisions and progressions that students or groups of students make as they solve a problem.

Diversity of product is determined by how many different types of final solutions students in a class produce to solve a problem.

Study Context

Course Setup

Introduction to Computing in Engineering is a course required for all 200 engineering students at a research university in Massachusetts. In the last few years, the course underwent a transition from a large, lecture-based course taught by one professor to several smaller sections taught by different professors, each using their own instructional technique. In the spring of 2019, four professors taught the Introduction to Computing Course using three different instructional methods. All courses had the same syllabus goals, outlined in Table 1 below.

Table 1. Course Goals (as defined in the 2019 syllabi)

Overall Goal	**Key Components**			
Fluency in a computer language	Master basic coding concepts	Know common commands and data types	Use good code style	Plan both small and medium-scale projects
Understand tools for engineering computing	Quantify numerical error in solutions	Know how to use symbolic math tools	Understand matrix/vector calculations	Know how to leverage built in help resources
Apply these tools to data analysis	Fit curves/models to noisy data	Apply descriptive statistics to datasets	Work with a variety of data formats	Have exposure to modeling physical systems in code

Two of the professors used a lecture and lab-based instructional model. There were approximately 45 students in each lecture-based section (one section per professor). Students in these sections attended two 75-minute lectures each week and learned about computational concepts using MATLAB, a numerical computing environment commonly used by scientists and engineers [14]. Students then applied these concepts during a weekly 75-minute lab section run by a teaching assistant (TA). Students also completed weekly homework assignments using MATLAB.

The third professor used a problem-based instructional approach which consisted of a mixture of lecture and projects. The 31 students in this section attended three, 75-minute classes each week. In each class, the professor first offered a mini-lecture on computational concepts with live coding demonstrations. The majority of the class period was devoted to students collaborating on projects and homework while the professor and TAs circulated to answer questions. Python and

the BBC micro:bit were used as the main computational tool in this section. Python is a popular interpreted, general-purpose programming language [15]. The BBC micro:bit is a low-cost, hand-held microprocessor that can be programmed in Python [16].

The fourth professor used a distributed-expertise approach. A distributed-expertise model is a pedagogical approach where students specialize in a topic area that falls within the broader content goals of the course [17]. This is similar to an instructional technique known as *Jigsaw Learning*, with the key distinction being that the distributed expertise model embraces the idea that different students will develop different knowledge sets whereas the jigsaw learning model culminates with all students having the same knowledge set [18]. The 69 students enrolled in this section attended three 75-minute class sessions per week. The professor occasionally conducted short demonstrations in class, but the majority of class time was dedicated to students working in small groups on computational thinking assignments with the support of the professor and TAs. Homework and projects were presented to students via the Jupyter notebooks platform, which is an open-source web application that allows teachers to create and share workbooks that contain live code, narrative text, images, and videos [19]. Students were able to write and test Python code for their Raspberry Pis, and submit assignments through their Jupyter notebooks. Raspberry Pi is a small, low cost programmable computer that was used as the main computational tool in this section [20]. Students spent the beginning of the semester learning as one large group (where every student worked from the same Jupyter notebook) and then broke into specialty groups for the middle portion of the semester (where each specialty group worked from a different Jupyter notebook). There were four specialty groups: system integration and communication, computer vision and image processing, robotic sensing and actuation, and cloud-based exploration. While students learned different concepts in each specialty group, the key underlying computational thinking principles learned by each specialty (e.g. problem decomposition, coding structures, numerical methods, etc.) were the same. Students spent the final portion of the semester working in final project groups that consisted of one student from each of the specialty groups.

Final Projects

In the lecture-based and problem-based sections, the final projects given to students were very similar. In these sections, students were free to choose any application of computational thinking in which they were interested and design a computational tool for an engineer or scientist. Students in both sections had approximately three weeks total to work on the final project.

In the lecture-based section, students worked individually whereas in the problem-based section students worked in pairs. Table 2 below outlines the exact project requirements and deliverables for each section. The major topic areas that students had to pick from to satisfy the final project requirements were: matrix/vector products, descriptive statistics, fitting lines to data, numerical differentiation, importing data from a file, classification, image processing, numerical

integration, clustering, extrapolation, signal processing, and interpolation. In the problem-based section students were also allowed to incorporate microcontroller based data collection and/or microcontroller based user interface as a major topic for their final project.

In the lecture-based section, students were given access to ten sample projects (including both the project documentations and sample code) spanning various engineering computing applications. These sample projects included projects using convolution to enhance low quality images, using the Euler method to solve mathematical equations, using classification to analyze medical data, analyzing sound files to decode messages, etc. Students in the problem-based section were given a shorter list of similar example topic ideas without accompanying documentation or code.

Table 2. Final Project Requirements and Deliverables for Lecture-Based and Problem-Based Sections

Project Requirements		**Major Deliverables**	
Lecture-Based Section	*Problem-Based Section*	*Lecture-Based Section*	*Problem-Based Section*
Written in MATLAB	Written in Python	Project proposal	Project proposal
STEM focused application	STEM focused application	Project Code with Documentation Files	Project Code with Documentation Files
Must have some kind of guided user interface (GUI)	Must use at least one function or module that they did not use in class	Project Powerpoint (not given in person, slides submitted electronically only)	3 minute in class presentation
Must use at least one of the major topics learned in class	Must use at least three of the major topics learned in class		Written Project Report

In the distributed-expertise section, the final project involved constructing a warehouse robot that could autonomously navigate a model warehouse and find and identify a specific product. Each group had to produce a robot that could: (1) receive remote instructions and broadcast findings wirelessly, (2) visually identify a location, (3) plan an efficient path, (4) navigate an unknown warehouse layout, (5) identify a product, and (6) coordinate all of these actions between multiple devices. Students constructed their robots using the Raspberry Pi 3B+ [19] , Dexter Industries' GoPiGo (a kit including motors and a chassis designed to be used with the Raspberry Pi) [21], Grove Sensors [22], and a Sony PlayStation Eye camera [23] (Figure 1). The robots were coded in Python using the Jupyter notebooks interface and several cloud services to complete the warehouse challenge.

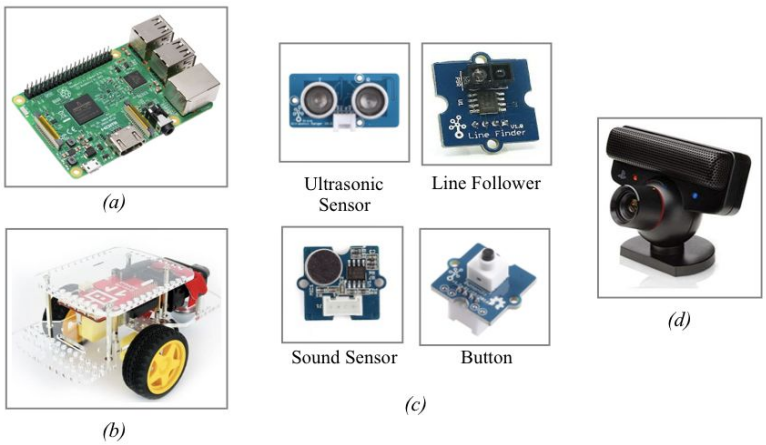

Figure 1: (a) Raspberry Pi 3B+, (b) GoPiGo Robot, (c) Grove Sensors, and (c) Sony Playstation Eye Camera

Students worked in groups of four (one student from each specialty group) and were required to use at least two Raspberry Pis communicating wirelessly to solve the warehouse challenge. Robots were placed in a modeled warehouse environment (shown in Figure 2) in a random room with a random orientation. Students used the Twitter API to send a tweet when their robots began navigating through the warehouse [24]. The robots then used image processing to decode a barcode (posted on the wall of the warehouse) that would tell the robot in which room of the warehouse it was located and in what direction it was facing. Warehouse layout and product definitions were stored in a cloud service called Airtable [25]. The system then had to dynamically load the warehouse map from the cloud and plan a path to navigate through the warehouse in search of a specific product at an unknown location. At each location, the robot took pictures and leveraged Microsoft Azure, a cloud-based image processing and computer vision module for object detection, to determine what product was in each room [26]. Once the correct product was found, the robot had to Tweet the product name and price (pulled from Airtable).

Figure 2: Robot in Modeled Warehouse for Distributed Expertise Final Project

This project was designed to require knowledge developed in each of the specialty groups, thus requiring the expertise distributed throughout the group. In the weeks leading up to the final project showcase, the homework assignments given to students in each of the specialty groups scaffolded skill building that was directly applicable to the final project requirements. When students then reassembled into final project groups they also worked from Jupyter notebooks that helped scaffold their problem solving process. While students were guided by the Jupyter notebooks, the way in which they integrated and deployed all of the different components of the challenge was largely up to them.

Students spent approximately 4 weeks working on their final projects to produce a robot for the final project showcase, when all of the robots were tested in a modeled warehouse setting. The deliverables for this final project were: (1) a working robot, (2) the completed final project Jupyter notebook for each group (which contained images, code, and other documentation of the robot), and (3) a video of the robot working.

Data Collection

The different methods being used to teach the same course created a fruitful context for a comparative case study across different instructional approaches. As part of this larger case study, a variety of different data types were collected by the research team including surveys, interviews, student-generated artifacts, and in-class observations. These data were collected with the intent to conduct a mixed methods comparative case study analyzing the different learning outcomes of each course. As the data analysis for this larger case study unfolded, the analysis of the artifacts produced by students for their final projects became of particular interest to the research team. Only surveys, student final project submissions, and in-class observations were used to support the qualitative findings presented in this paper. In a future paper we plan to dive more into this case study as a whole to explore metrics such as student attitudes, confidence levels, and technical proficiencies across all sections.

Data Analysis

The goal of this qualitative case study analysis is to identify affordances in each different pedagogy for diversity of problem, diversity of process, and diversity of solution. The course instructional models, final project prompts and requirements, and many other factors varied widely between sections making it not possible in this naturalistic setting to determine the direct causal links between specific course attributes and project outcomes. Rather, the goal of this analysis is to better understand the final project outcomes through the lens of solution diversity and student attitudes within each of the instructional approaches.

Analysis of Student Final Project Submissions

An analysis of the final project code, reports, and presentations submitted by the students in the lecture and problem-based sections revealed that while the engineering computing applications selected by students were varied, the computational tools that students leveraged were very similar across projects. Figure 3 below shows how many projects (out of the 100 projects submitted) leveraged each core concept.

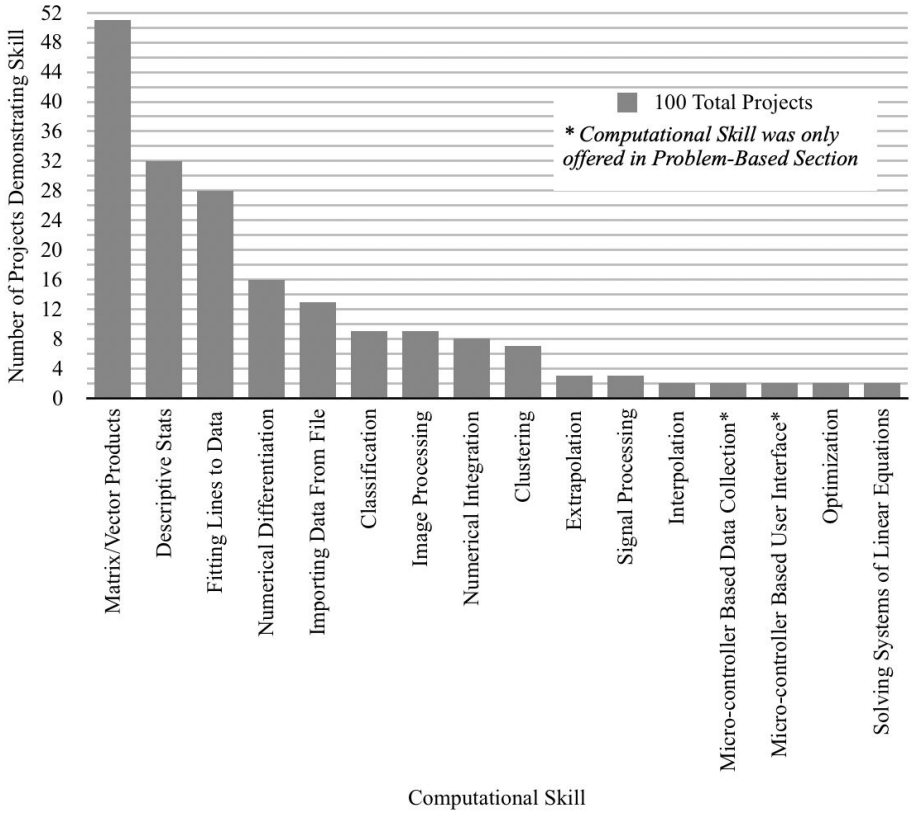

Figure 3: Lecture and Problem-Based Final Projects: Computational Skills Used

Despite the open-ended nature of the final project prompt, the students used only a small fraction of the 16 core concepts available for implementation. The majority of students used one of the provided example computational thinking applications for their project and changed some aspect of the example code for their project submission (i.e. used a different data set with the same algorithm or edited the user interface). Many of the projects followed a very similar structure, where users were prompted to upload a data set through a guided user interface. That data set was then fed through an algorithm to give a useful final result in the form of descriptive statistics. In the problem-based section many of the projects followed the same general structure

of importing data from a file and using the same clustering and classification process to draw insights from these data. The popularity of the clustering and classification problem was most likely a result of the fact that the last assignment before the final project was a clustering and classification assignment.

In the distributed-expertise section, every group of four students created the same robot with the same materials to solve the same problem. There was no example robot or code supplied, but students were working off of scaffolded Jupyter notebooks prompting them through the required aspects of the projects. However, the analysis of the final project submissions from the distributed-expertise groups showed that the ways in which students pieced together their distributed knowledge to solve the problem varied widely between groups. Based on the analysis of the fourteen final project submissions, there were seven distinct ways that students structured their final project code, and no one structure was identified in more than three project submissions. Figure 4 shows three example system diagrams illustrating how three different groups configure their solutions to the warehouse robotics challenge.

As illustrated by these system diagrams, the approach students took when solving the warehouse robot challenge was different across project groups. The solutions varied in many ways, including the number of Raspberry Pis used, the grouping and order of execution of major tasks, and the algorithms used for path planning and image detection. The students who created the system diagram shown in Figure 4a used three Raspberry Pis to solve the problem. One Raspberry Pi was responsible for reading sensors, taking images, and doing the image processing (attached to the physical robot). The second Raspberry Pi did the path planning (not physically attached to the robot), and the third Raspberry Pi (also located on the actual robot) interpreted the path and converted it to motor commands to make the robot move forward. The system depicted in Figures 4b and 4c both show solutions that used two Raspberry Pis. The group that created the diagram in Figure 4b had one Raspberry Pi that imported the warehouse layout from Airtable, determined the best path, and sent the path directions to the second Raspberry Pi. The second Raspberry Pi (located on the robot) did the remainder of the tasks to complete the challenge. The solution in 4c shows a system where responsibility was shared more equally between the two Raspberry Pis. Raspberry Pi #1 (located on the robot) read the room information, actuated the motors, and collected and processed images of the warehouse. Raspberry Pi #2, which handled all of the path planning and tweeting to indicate the start and end of the challenge. These are just three examples of the seven different ways students in the distributed-expertise section stitched together their knowledge to solve the warehouse robot challenge.

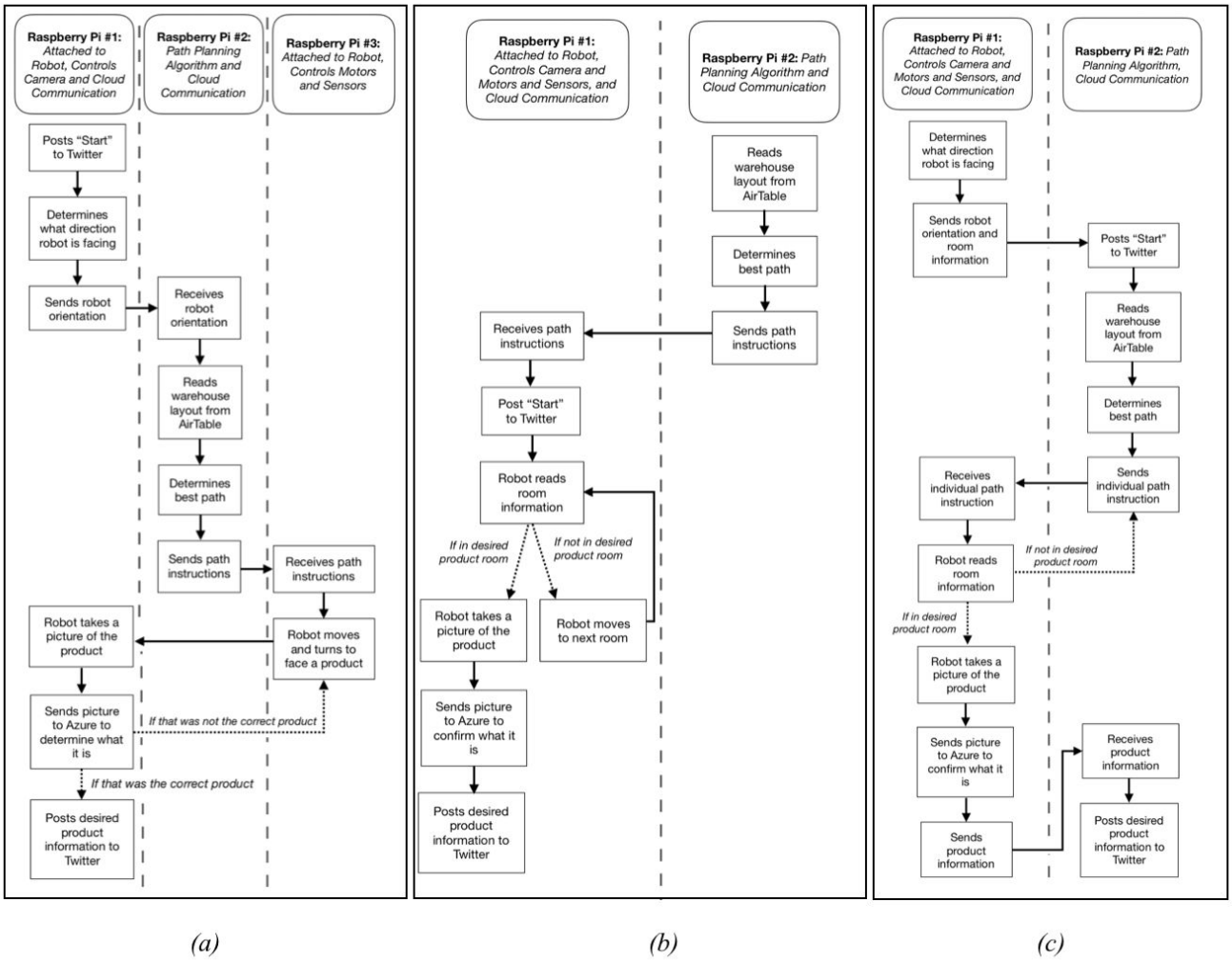

Figure 4: Three Example System Diagrams for Warehouse Robot Project

Table 3 below summarizes the solution diversity data based on our qualitative analysis of the final project submissions and the availability of examples for each section. These data are described using a *none-few-some-many* scale to qualitatively represent diversity of problem, process, and product. The qualitative labels were assigned based on: (1) the number of submitted final project problems, processes, and products that were distinctly different, (2) the distribution of these unique submissions throughout each section, and (3) the originality of the submissions (were most of the ideas taken from provided examples or new ideas that the students came up with). Components of the numerical data set used to generate this table can be found in the Appendix A.

Table 3. Summary of Final Project Solution Diversity

	Number of Examples Provided	Diverse Problems	Diverse Processes	Diverse Final Products
Lecture-Based	Many	Some	Few	Some
Problem-Based	Some	Many	Few	Many
Distributed-Expertise	None	None	Many	None

In the lecture-based section, despite being given the opportunity to choose their own project idea, many students leveraged existing example ideas and/or followed the same structure as provided examples to complete their final projects, therefore producing many final project products that were very similar to each other. In the problem-based section, students identified many different project problems and therefore produced many different final products, but many students used the same code structure. The distributed-expertise section had no diversity of problem or product but a high diversity of process.

Analysis of Post-semester Survey Responses

In the post-semester survey given to students, they were asked several qualitative questions about their experience in the course. These questions included: (1) What aspect of the course do you think helped you learn the most (i.e. lab time, lecture, homework assignments, etc.)?, (2) What was the most challenging part of the course and why?, and (3) How do you feel the final project did or did not supplement and capture your knowledge of computational thinking?

No students in the lecture-based or problem-based sections made any reference to the final project in their responses to the first two questions. In the distributed-expertise section, 3 students cited the final project as being an aspect of the course that helped them learn the most, and 16 students said it was the most challenging part of the course. An example response that captures one student's attitude towards the final project is:

> *"The most challenging part of this course was the final project. As someone who had never coded before I took this class, I had never attempted to create that much code to accomplish that complicated of a problem before. However, working with my group was a wonderful experience and I enjoyed the challenge. It was extremely rewarding to see our robot move around the warehouse according to the directions that our code sent it, successfully identify the product, and post to twitter all at once."*

Table 4 details the types of responses students gave to the question about how the final project did or did not supplement and capture the learning that happened through the semester.

Table 4. Themes in Final Project Survey Responses

		Lecture-Based (89 students enrolled, 29 survey responses)	**Problem-Based** (31 students enrolled, 11 survey responses)	**Distributed-Expertise** (69 students enrolled, 50 survey responses)*
Theme	Representative Quote	*I thought that based on the requirements, this project was a good representation of all the skills we learned in class (with more freedom)*	*It gave me the opportunity to come up with my own type of project that let me express what I learned over the year.*	*The final project did a really good job at bringing together our individual knowledge of computational thinking. The final project almost forced us to use what we knew and understood to then apply it to finish the project.*
Final project was a good representation of knowledge gained in class	Percentage of Responses	48%	64%	82%
Final project was too broad and/or too difficult	Representative Quote	*Did not-too vague of an assignment structure*	*Would have liked more guidance on a direction, didn't really advance anything as we couldn't think of anything cool/challenging.*	*The final project is a bit too challenging and has caused me more stress than it has captured my knowledge of computational thinking.*
	Percentage of Responses	27%	36%	12%
Final project did not represent the knowledge gained in class	Representative Quote	*I think it did not capture many aspects of what we learned in this class.*	*NA*	*I don't think the final project captured the knowledge of computational thinking. My specific part of the final project wasn't learned well throughout the course so it's kinda hard to know what to do.*
	Percentage of Responses	25%	0%	6%

* Students in the Distributed-Expertise section were given in class time to complete the survey, which contributed significantly to the higher response rate compared to the other two sections.

Discussion

As summarized in Table 3, the analysis of the final project submissions across all of the sections showed varying levels of diversity of problem, diversity of process, and diversity of product. What we found particularly interesting was the ratio between diversity of problem and diversity of solution. We expected that a highly open-ended challenge would yield a higher solution diversity, and a more constrained/defined problem would yield a lower solution diversity. In the problem-based section, this expectation held true. Students identified many different problems within the broader topic of computational thinking, and therefore came up with many different solutions to solve those problems.

In the lecture-based section, this was not the case. There are many possible reasons why the lecture-based submissions had such low diversity of problem and process. The examples that were provided to students in the lecture-based sections appear to have played a large role in influencing students' design decisions [27]. In the problem-based section, the timing of the last assignment may have led to the low diversity of process, because students used the tools that were the freshest in their minds. Additionally, students in the lecture-based section worked independently, so the opportunities for teamwork were nonexistent. Many students in the lecture-based section may have felt overwhelmed and therefore defaulted to using one of the example problems and or structures rather than coming up with an idea on their own. This "analysis paralysis" is a common challenge that students face when conducting open-ended design problems [28].

Students in the distributed-expertise section were given a predefined problem to solve for their final project, resulting in zero diversity of problem. They all created a robot that performed the same function yielding zero diversity of product. However, there was a large diversity of processes used across the different groups to complete the warehouse robot challenge. We found the high diversity of process despite zero diversity of product or solution particularly interesting.

We hypothesize that one of the factors that afforded this diversity of process in the distributed-expertise section was the instructional model itself. Each final project group had one student from each specialty area, therefore each group contained quadruple the expertise of a four-student group in a traditional course setup, where every member would have the same knowledge. This distributed expertise and the way that the final project was scaffolded necessitated a unique kind of collaboration, where no one person could do the final project without collaborating with others. Throughout the entire final project, students were arguing, troubleshooting, and constantly iterating in an attempt to combine their knowledge to solve the

warehouse challenge. This was evident when observing students working and by seeing the diversity of ways in which the student groups structured the code for their robots.

The distributed-expertise model also allowed for students to complete a project that combined many different concepts and ideas. The ability to complete a project as complex as the warehouse robot challenge was a direct result of students not having to know all of the content needed to complete the project. Since each student needed only a quarter of the skills required, a project as complex as the warehouse challenge was feasible for first-year engineering students.

The high ratio of process diversity to problem/product diversity seems to be indicative of the kinds of engineering practices students were engaged in when completing the final project. In order to solve the warehouse challenge, students were required to leverage not only their technical knowledge, but also engineering practices. This is supported by the survey results, which illustrated that students were excited by the complexity of the final project. The opportunity to do some "real engineering" was both rewarding and pushed students to engage in a multitude of engineering practices, such as teamwork, communication, and systems thinking, as well utilize their computing in engineering knowledge.

Implications and Future Work

As design-based projects continue to become integrated into the technical parts of undergraduate engineering curricula, it is important for instructors to know how to structure these projects and integrate them into other aspects of their course to effectively facilitate the learning of engineering practices. The case study presented in this paper illustrates how thinking about solution diversity, and the ways in which project design and instructional techniques afford solution diversity, is one possible metric to consider when designing an engineering project. This study also illustrates how a less constrained problem doesn't always yield a higher solution diversity, and how in some cases, the structure of the course itself can be used to motivate students' independent thinking in a design-based project. In future work we hope to analyze ways that the different pedagogical models influenced learning outcomes beyond solution diversity such as group dynamics.

Acknowledgments

This material is based upon work supported by the National Science Foundation grant number A451001 SF9018. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We would also like to thank the students, teaching assistants, professors, and administrators who made this research possible.

References

[1] L. Schachterle and O. Vinther, “Introduction: The role of projects in engineering education,” *Eur. J. Eng. Educ.*, vol. 21, no. 2, pp. 115–120, 1996.

[2] T. A. Litzinger *et al.*, “Engineering education and the development of expertise,” *J. Eng. Educ.*, vol. 100, no. 1, pp. 123–150, 2011.

[3] “Attributes of Engineers in 2020,” in *The Engineer of 2020 : visions of engineering in the new century*, Washington, D.C.: National Academies Press, 2004, pp. 53–57.

[4] E. W. Banios, “Teaching Engineering,” in *Proceedings Frontiers in Education Twenty-First Annual Conference. Engineering Education in a New World Order*, 1991, pp. 161–168.

[5] C. Rogers, “Learning STEM in the Classroom,” *LEGO Engineering*, 2014. [Online]. Available: http://www.legoengineering.com/learning-stem-in-the-classroom/.

[6] “Criteria for Accrediting Engineering Programs, 2020 – 2021,” *abet.org*, 2020. [Online]. Available: https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2020-2021/. [Accessed: 21-Jan-2020].

[7] B. I. Hyman, “From Capstone to Cornerstone: A New Paradigm for Design Education,” *Int. J. Eng. Educ.*, vol. 17, no. 4–5, pp. 416–420, 2001.

[8] R. N. Savage, K. C. Chen, and L. Vanasupa, “Integrating Project-based Learning throughout the Undergraduate Engineering Curriculum,” vol. 8, no. 3, pp. 15–27, 2007.

[9] C. M. Kellett, “A project-based learning approach to programmable logic design and computer architecture,” *IEEE Trans. Educ.*, vol. 55, no. 3, pp. 378–383, 2012.

[10] R. N. Savage, K. C. Chen, and L. Vanasupa, “Integrating Project-based Learning throughout the Undergraduate Engineering Curriculum,” vol. 8, no. 3, pp. 15–27, 2007.

[11] A. J. Dutson, R. H. Todd, S. P. Magleby, and C. D. Sorensen, “A Review of Literature on Teaching Engineering Design Through Project-Oriented Capstone Courses,” *J. Eng. Educ.*, vol. 86, no. 1, pp. 17–28, 1997.

[12] L. Vanasupa, J. Stolk, and R. J. Herter, “The Four-Domain Development Diagram: A guide for holistic design of effective learning experiences for the twenty-first century engineer,” *J. Eng. Educ.*, vol. 98, no. 1, pp. 67–81, 2009.

[13] C. Rogers, *Solution Diversity in Engineering Education*. USA: CADREK12, 2016.

[14] "MATLAB," *mathworks.com*, 2020. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 21-Jan-2020].

[15] "Python," *python.org*, 2020. [Online]. Available: https://www.python.org. [Accessed: 21-Jan-2020].

[16] "Meet micro:bit," *microbit.org*. [Online]. Available: https://microbit.org/guide/. [Accessed: 21-Jan-2020].

[17] A. L. Brown, D. Ash, M. Rutherford, A. Gordon, and J. C. Campione, "Distributed expertise in the classroom," *Distrib. Cogn. Psychol. Educ. considerations*, pp. 188–288, 1991.

[18] E. Aronson, The jigsaw classroom. Sage, 1978.

[19] "Jupyter," *jupyter.org*, 2019. [Online]. Available: https://jupyter.org/index.html. [Accessed: 21-Jan-2020].

[20] "Raspberry Pi 3 Model B+," *raspberrypi.org*, 2019. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed: 21-Jan-2020].

[21] "GoPiGo: Build and Program Your Own Robot," *dexterindustries.com*, 2019. [Online]. Available: https://www.dexterindustries.com/gopigo3/. [Accessed: 21-Jan-2020].

[22] "Grove Sensor Introduction," *wiki.seeedstudios.com*, 2018. [Online]. Available: http://wiki.seeedstudio.com/Sensor/. [Accessed: 21-Jan-2020].

[23] "PlayStation® Eye," *playstation.com*, 2010. [Online]. Available: https://www.playstation.com/en-ae/content/dam/support/manuals/scee/web-manuals/peripherals/ps3/ps3-eye/SCEH-00448_PS Eye Web_GB.pdf/. [Accessed: 21-Jan-2020].

[24] "Twitter API Developer Docs," *developer.twitter.com*, 2020. [Online]. Available: https://developer.twitter.com/en/docs. [Accessed: 21-Jan-2020].

[25] "Airtable," *airtable.com*. [Online]. Available: https://airtable.com. [Accessed: 21-Jan-2020].

[26] "Microsoft Azure," *microsoft.com*, 2020. [Online]. Available: https://azure.microsoft.com/en-us/. [Accessed: 21-Jan-2020].

[27] M. Agogué, A. Kazakçi, B. Weil, and M. Cassotti, "The impact of examples on creative design: Explaining fixation and stimulation effects," *ICED 11 - 18th Int. Conf. Eng. Des. - Impacting Soc. Through Eng. Des.*, vol. 2, no. August, pp. 266–274, 2011.

[28] N. J. Mourtos, "Challenges students face in solving open-ended problems," *Int. J. Eng. Educ.*, vol. 26, no. 4, pp. 846–859, 2010.

Appendix

Appendix A: Quantitative Final Project Solution Diversity Data

	Number of Final Projects Submitted	**Number of Examples Provided**	**Number of Different Problems Solved**		**Number of Different Code Structures**	**Number of Unique Final Products**
			Counting ideas taken from examples	*Not counting ideas taken from examples*		
Lecture-Based	84	Many example problems with code	32	8	36	44
Problem-Based	16	Example problem ideas, no code	16	12	6	16
Distributed-Expertise	14	None	1	1	7	1