Using Assignment Design as an Instrument to Collect Student Voice

Rita Garcia, Bradley Alexander
University of Adelaide
Adelaide, Australia
firstname.lastname@adelaide.edu.au

ABSTRACT

Students might have preconceptions about programming when enrolling in an Introductory Programming (CS1) course. These preconceptions might influence their expectations about programming assignments. Understanding these preconceptions could help give students a voice in their learning experience. This paper reports on a study for CS1 programming assignments. This study uses an assignment design activity as an instrument to collect student voice, asking students to design a programming assignment they expect to accomplish at the end of a CS1 course. A mixed-methods approach was used to analyse the subject matter and course learning outcomes of the students' assignment designs. The results show students applying prior knowledge, a process known as *transfer of training*, to design their assignments, predominately focused on math and gaming. The results also show that students with no prior programming experience had lower expectations from the programming assignments, which might have influenced their study effort in the course. We discuss integrating the results into CS1 assignments, helping students transition to new roles as programmers, and adjust their study expectations early to recognise when more effort is needed to successfully complete a course.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Student assessment**.

KEYWORDS

Assignment Design, Student Voice, Transfer of Training

ACM Reference Format:
Rita Garcia, Bradley Alexander. 2022. Using Assignment Design as an Instrument to Collect Student Voice. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.*ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499271

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA.

ACM ISBN 978-1-4503-9070-5/22/03…$15.00
https://doi.org/10.1145/3478431.3499271

1 INTRODUCTION

Students who enroll in Introductory Programming (CS1) courses might have preconceptions about programming. The preconceptions might differ based on prior experiences and interests in Computer Science (CS). Collecting students' diverse preconceptions on programming could give them a voice in their learning [7]. On a related theme, bringing student voice into the learning design can give students a "legitimate perspective and opinion, being present and taking part, and/or having an active role" [6, p. 362] in their learning. Students' input could aid in the design of programming assignments, potentially helping them better understand the problem and motivating them to successfully complete the assignment.

The goal of this study is to evaluate an assignment design activity as an instrument for collecting student voice. Our motivation is to use the voice to help to identify an instrument that could improve students' learning experience with CS1 programming assignments. Our work is guided by the following research questions:

- **RQ1:** *How can an assignment design activity facilitate student voice for CS1 courses?*
- **RQ2:** *How does student voice align with the CS1 learning outcomes?*

In designing this study, we wanted students to get involved in the assignment design process before being influenced by existing assignments, so we applied the instrument early in their learning and asked them to design a programming assignment they believe they could achieve by the end of the course. We examined the assignment designs for subject matter, to identify: topics the students found interesting and motivating; and learning outcomes, to identify the programming expectations for the course. We used a mixed-methods approach to evaluate the assignment designs. This paper reports on a study evaluating those designs.

The results show students applying prior knowledge to their designs, using maths, games, and daily living activities as the subject matter. Applying prior knowledge to a new task is a process known as *transfer of training*, which can help the learner prepare for a new role [21]. To support transfer of training, educators can integrate identified subject matter into CS1 programming assignments, to help students transition to new roles as programmers. The results from this study show that students with no prior programming experience have lower expectations from the CS1 course. These findings raise future research opportunities for using interventions to potentially adjust students' expectations, to help them successfully complete the course.

2 BACKGROUND ON STUDENT VOICE

Student voice gives students an active role in their educational development, enabling them to contribute in the learning and teaching practices used in the classroom [20, 30]. Through students'

involvement in processes, such as the development and redesign of educational materials [35], educators can gain insight into the students' educational experiences, making them "active agents in their own development who benefit from a nurturing and enabling learning environment" [36, p. 6].

Education governance, such as the UK's 2002 Education Act [39], includes statutory requirements focusing on student voice. The requirements enable students to contribute to the decision-making processes that affect their learning [38]. Education governance encourages student voice by providing guidelines that help educators apply the voice into designing educational materials [22]. Although policy exists, helping educators to facilitate student voice and students to understand how their voice is being heard can strengthen the impact that student voice has on instructional materials [19].

Student voice has been previously used in CS curriculum design [37], where students' contributions through interviews influenced change. However, student voice was not fully realised in the curriculum development due to students' reservations in participating, and educators perceiving difficulties in integrating student voice into the design process. In another student voice study [15], students were interviewed to collect their feedback on the presentation of programming assignments, identifying design treatments that helped them better understand the problem description. That study showed certain design treatments, such as listed subgoals, helping CS1 students better identify programming requirements and validate their solutions. Another project, Researching Equity, Access, and Learning in CS Education (REAL-CS), focused on student voice for underrepresented high school CS students [31]. The REAL-CS project was conducted in regions of the United States, using pre-post surveys and interviews with students to help shape educational materials that interest and motivate their learning.

3 RELATED WORK

In this section, we review work that provides students the opportunity to develop learning activities. The first is CrowdSorcerer [27], an open-source system for both educators and students that contains an authoring tool for creating programming assignments. CrowdSorcerer enables students to peer review assignments and help them "reflect on prior tasks and content, consider how they would themselves solve the problem they are creating and also learn to articulate what it means for a program to be correct (or incorrect)" [27, p. 326]. When CS1 students were asked as part of the study to design assignments, many of the resulting designs were suitable activities for the classroom, demonstrating self-reflection on concepts they learned in the course.

PeerWise [11] is an online platform that houses multiple choice (MC) questions created by students within various STEM disciplines, such as Biology, Physics, and Computer Science. The platform provides students the opportunity to discuss, answer, and rate peers' MC questions, allowing them to build a question repository for their course. When PeerWise was used by CS students, the results showed them gaining a deeper understanding of the learning concepts presented in the course. Another study [13] applied rubrics as a learning tool to support critical thinking skills. In this study, K-12 students were asked to develop rubrics for the assessing skills covered in a Music course. The activity was designed to help students better understand the requirements needed to achieve music skills taught in the course. The process of designing the rubric encouraged students to think of the music skills at a higher level and helped them better understand the stages involved to acquire mastery of the learning concepts. While the related work surveyed in this section promotes reflection on learning concepts, our study has the advantage of capturing students' reflections in the CS context early in their learning. These early reflections could shape the presentation of learning concepts.

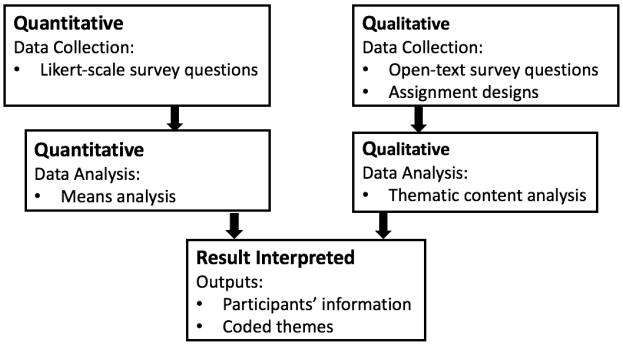

Figure 1: Diagram of Study Method (Adapted from [10])

4 STUDY METHOD

We adopted a mixed methods approach [9] using a concurrent triangulation design [10] to interpret the collected data. Figure 1 shows the equal weight given to the quantitative and qualitative data sources. The quantitative method interprets the students' programming background and potential interests for the course, while the qualitative method interprets their assignment designs.

4.1 Context

The study was conducted in the first week of a 12-week CS1 course (July 2020 Semester 2, 536 students) at the University of Adelaide. The course used MATLAB for the first seven weeks and C for the remaining five weeks to teach programming concepts. In addition to lectures, the course had workshops that gave students the opportunity to ask teaching assistants (TAs) for support on learning concepts while working on learning activities.

Students were given the assignment design activity in week 1; this was part of a series of six activities for the students to undertake during the two-hour workshop. The low-stakes activity contributed to 1% of the overall course grade. Students were given a week to complete the assignment design activity. Upon completion, students were instructed to upload their designs to Canvas, the Learning Management System (LMS) used at the university for course administration. For this study, students were also given a non-compulsory survey designed to collect their programming backgrounds and intended majors. Details about the survey are described in the next section.

4.2 Survey

A survey was designed to collect students' prior programming experiences and their declared majors, to better understand students' interests and motivations for taking the CS1 course. We wanted to determine how their prior experiences might influence their assignment designs. The survey did not collect students' programming expectations. Instead, we wanted the assignment design activity to promote self-reflection on their programming expectations. The survey was administered within Canvas LMS, giving students ten days to complete. After the ten days, the survey responses were exported from Canvas and imported into a spreadsheet for analysis.

The survey contained questions focusing students' prior programming experiences. The first question was a five-point Likert scale question, asking students to rank their prior programming experiences from *Very Experienced* (5 points) to *Very Inexperienced* (1 point). Two open-text questions were provided to get more details on students' programming experience and declared majors. Collecting declared majors might provide insight into students' motivations for taking the course. Students with intrinsic motivations in CS had higher perceived programming skills [42], which could also influence their assignment designs.

The survey responses were analysed using two methods. The first method was analysis of means (ANOM) performed on two survey questions: a Likert scale question rating students' programming experience and an open-text question asking students to provide previous years of programming experience, either self taught or through prior programming courses. To perform ANOM on the open-text responses, the data was numerically quantified, such as *"six months"* converted to 0.5 years. Responses that could not be accurately quantified, such as *"less than a year"*, were labelled *Unknown* and not included in the results. The second analysis approach was thematic content analysis [26] performed on the open-text question on declared majors using NVivo to code the responses. During coding, new nodes were created for the emerging majors. After the coding process, a matrix table was extracted from NVivo to present the coding frequencies for the declared majors.

4.3 Assignment Design Instrument

An assignment design activity, shown in Figure 2, was developed as an instrument to collect student voice. A multi-level assignment educational design pattern [23] was used to construct the instrument. The educational design pattern provides guidelines for the assignment structure, helping students to better understand the problem description. Guidelines include giving additional information on concepts at other knowledge levels for struggling students to understand the problem. Figure 2 shows the instrument constructed with scaffolded design treatments, such as bold-face text for emphasis [4] and lists of subgoals to identify the activity's requirements [40].

The instrument provides the activity's purpose, to design a problem description for a programming assignment that can be administered at the end of the CS1 course. By asking students to design an assignment for the end of the course, we wanted them to reflect on their programming expectations and what they would like to achieve in the course. The instrument also instructs the students to only develop the problem description, and suggests their designs can contain multimedia elements. In an effort to reduce task anxiety during the assignment design development, the instructions state there are no right or wrong answers, giving students the opportunity to take risks in their designs. Students were given a week to complete their designs. After the week, the students' designs were exported from Canvas and imported into NVivo for analysis.

Here's your opportunity to **create a programming assignment** that could be given to students at the end of this course.

This activity **only requires** you to create a problem description for a programming assignment given at the end of this course. You **do not** need to write a programming solution.

This activity is designed to be completed during the workshop. However, if you want more time, you can turn in by the end of the week. Please **upload your solution** as a file.

Additional Information:

- Think about what types of programs you would like to create at the end of this course and what you would like to program.
- **Do not** dwell on providing a **"correct"** solution. This activity does not have a right nor wrong answer.
- **Do not overthink** this activity. This activity should take no more than an hour to do.
- Solutions can be a page in length and contain multimedia elements, such as text, images, audio, and video.
- Remember: **Have Fun!**

Figure 2: Assignment Design Activity Instructions

Thematic content analysis [26] was used to interpret the students' assignment designs. The assignment designs were coded with multiple nodes representing different themes and information about the student creating the assignment design, such as gender. Students' survey responses were mapped to their designs to better understand their design motivations. An example coding process for a student's assignment design could contain five nodes to denote subject matter, programming background, declared major, the student's de-identified number, and gender. The de-identified number enabled us to anonymously report the results.

We analysed the assignment designs for learning outcomes. The analysis used the Australian CS1 course accreditation defined by the *Engineers Australia - Chartered Status Handbook* [1], which explained novices' competency through high-level goals, such as the ability to interpret and decompose problems. We also used the existing course assignments presented at the end of the semester, which supported the goals and learning concepts the course wanted students to achieve by the end of the semester. To code the learning outcomes, four nodes were added in the initial coding framework to denote incorrect designs and designs that were above, below, and at the course's desired learning outcomes.

The initial coding framework contained eight nodes: gender, de-identified student number, programming background, declared major, and four nodes denoting the learning outcomes. Any emerging themes that arose during the coding process were assigned a new node. To strengthen the coding validity, the authors used the

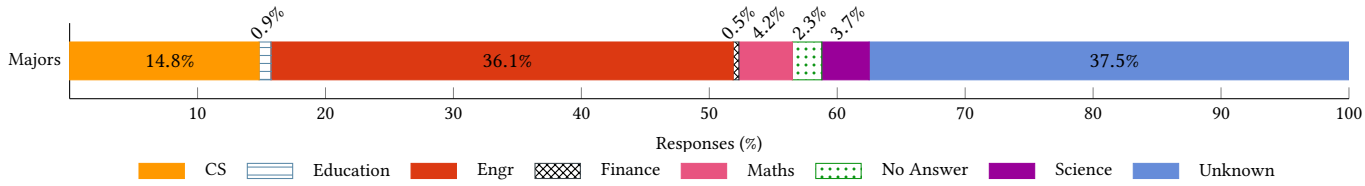

Figure 3: Participants' Declared Majors

inter-rater reliability process defined by Mackey and Gass [25]. The primary author was responsible for coding all assignment designs, while a co-author coded 10% of the students' responses. The validity process involved negotiation by the authors on 10% of the partially coded assignment designs. During the negotiation process, the authors discussed and agreed upon the coding protocol, to complete the coding of the remaining designs. When the coding of the assignment designs was completed, the coding was extracted from NVivo as a matrix table. The matrix table represented the coding framework with the initial and emerging nodes, identifying the common themes emerging from the students' assignment designs and presenting the thematic frequencies.

5 RESULTS AND DISCUSSION

5.1 Participants

From the 536 students enrolled in the course, 200 (32.13%) students participated in the survey. Fewer (30.04%, n=161) participants completed the assignment design activity, where 26.71% (n=43) were female and 73.29% (n=118) were male. We note the gender disparity of our participants with a female-to-male ratio of 1:2.74. However, we strive to find common ground in the student voice.

When examining the survey results, we found majority (63.00%, n=126) of the participants claimed to have no programming background, ranking themselves as *Very Inexperienced* (41.00%, n=82) and *Inexperienced* (30.50%, n=61). Few participants claimed prior experience, not many ranking themselves either as *Very Experienced* (1.00%, n=2) or *Experienced* (5.50%, n=11). From the 116 participants explaining prior programming experience, 37 were self taught, while 79 previously took a CS course. A smaller portion (32%, n=65) of those participants provided years of prior experience, averaging less than six months (0.47 years). Overall, the survey results show mostly novices with minimal programming experience participating in the study. These results align with the course design, which focuses on introducing novices to programming concepts.

Figure 3 shows the declared majors participants reported; these predominately fell into two categories: *Unknown* (37.50%, n=81) and *Engineering* (36.12%, n=78). The engineering category contains multiple engineering disciplines, such as Electrical, Chemical, Civil, and Mechanical Engineering. We isolated declared CS majors (14.8%, n=30) to determine participants that have higher motivations for taking the course. Three (1.39%) participants declared majors in non-STEM fields, so the majority of the declared majors were interested in STEM. Because of the distribution of unknown and declared majors, we cannot generalise the participants overall motivations for taking the course.

The rest of this section reports on the analysis of the participants' assignment designs. Section 5.2 presents which subject matter interested the participants, while Section 5.3 discusses how their assignment designs aligned with the CS1 course learning outcomes.

5.2 Subject Matter

Table 1 shows the subject matter found in the participants' assignment designs, listing eight categories in descending order and presenting the gender distribution for each theme. Two categories, *Programmed Solution* (3.11%, n=5) and *Pseudocode Solution* (3.11%, n=5), covered participants' designs that did not provide problem descriptions; instead they focused on a coded or pseudocode solution. These designs, focusing on the solution, are not included in the subject matter results.

Game Development (29.19%, n=47) was the most common theme used by both the male and female participants. For example, create *"a program that simulates a dice roll, i.e., will randomly display a value from 1-6"*. Though prior research [5] identified male students pursuing a CS degree because of their interests in gaming, our findings show that female students may also be influenced to study CS because of gaming, but more analysis is needed for confirmation. Assignment designs in this category assumed the reader had prior knowledge about the games referenced, because the designs either described the game at a high level or provided the game's name, such as Student08's design stating *"a program that acts as an automated tic-tac-toe player"*.

The next common theme was math related (24.84%, n=40). Similar to the gaming assignment designs, the math designs presented the problem at a high level with limited details. For example, Student87's design stated *"prompt the user for three numbers and print out the sum, product and quotients of the three numbers. These numbers should be irrational..."*. In this example, Student87 assumes the reader understands some mathematical terms. A potential reason students might have used math problems as their assignment designs is that they were applying previous knowledge to design their assignments. Drawing on prior knowledge to complete a new task is known as *transfer of training* [21] and has been shown [3] as a process used by students to transfer mathematical skills to real-world situations. In our results, we observe students applying their math skills to programming situations. Another example of transfer of training is with Student31's design: *"these problems on the scheme of maths are complicated and require program several steps that I am not aware of yet"*. Student31 is thinking about the math problem-solving process, but acknowledges the lack of skills to translate using programming concepts. Educators could incorporate math problems into assignments, building on students prior knowledge.

Theme	Responses	Gender Distribution	
		Female	Male
Game Development	29.19% (n=47)	27.66% (n=13)	72.34% (n=34)
Math Problem	24.84% (n=40)	25.00% (n=10)	75.00% (n=30)
Activities of Daily Living	21.74% (n=35)	22.86% (n=8)	77.14% (n=27)
Socially Relevant	11.18% (n=18)	50.00% (n=9)	50.00% (n=9)
Calculator	4.97% (n=8)	25.00% (n=2)	75.00% (n=6)
Programmed Solution	3.11% (n=5)	20.00% (n=1)	80.00% (n=4)
Pseudocode Solution	3.11% (n=5)	40.00% (n=2)	6.00% (n=3)
COVID	1.86% (n=3)	66.67% (n=2)	33.33% (n=1)

Table 1: Participants' Assignment Designs by Subject Matter

However, to make the math problem an effective learning tool for novel situations, the maths skills for these problems need to be well understood by the students [16]. Additionally, students do not always transfer their math skills to other disciplines [2]. When integrating math problems to future assignments, educators can scaffold the concepts to support recollection.

Another significant design theme in the results is *Activities of Daily Living* (21.74%, n=35) (ADLs), where participants applied common situations that occur in daily life. Participants applied a variety of ADLs, such as banking, school calendar, and address books. For example, Student2's design asked students to *"design code that can identify the amount of power that electrical appliances and machines require over periods of time. For example, the amount of power that will be needed to power a microwave for 2 hours"*. Related to ADLs are the designs building on socially-relevant (11.18%, n=18) themes that focus on social problems that benefit the community. For example, Student142 designed an assignment addressing obesity, which calculates *"the number of calories that need to be eaten for a person, given their amount of exercise and previous diet"*. The presence of socially relevant assignment designs continues to show current students seeking meaningful projects with societal impact, mirroring the interests identified by Layman et al. [24] for female, minority, and millennial students. Future programming assignments using socially relevant contexts could help current students recognise the contributions CS provides to society, potentially giving them additional motivation to continue their studies in the discipline.

5.3 Learning Outcomes

From the 161 submitted assignment designs, 143 were analysed for learning outcomes; see Table 2. There were 18 (11.18%) designs that could not be included these results because the participants did not correctly answer the activity or did not provide enough information to classify. The remaining designs were classified into three categories: *Below*, *At*, and *Above the Course Learning Outcomes*. Table 2 organises these categories in descending order according to the participants' designs. We discuss the assignment designs within the learning outcomes further in this section.

During the analysis process, we noted some (18.01%, n=29) participants reflecting on what they would like to achieve in the course, but felt they did not have the programming knowledge to express in an assignment. For example, Student124 stated *"I don't really know the scope of what programming can do as of yet, so I'm not sure what ... things you can program"*. Some participants' concerns related to their lack of programming knowledge. For example, Student32 stated *"I do not really know specifically how those steps should be done to lines of code, but I guess I would learn them in a future study"* and Student30 acknowledged they do not know enough programming concepts to provide game details, stating *"my coding knowledge is not deep enough to know how to code the actual part of the game."*. When designing their assignments, these students were concerned about language constructs above other design considerations, a software design approach known as *bottom-up design strategy* [12]. Though the activity did not ask for a coded solution, these exemplars demonstrate the students applying the software design strategy outside the scope of program production.

In addition to classifying the assignment designs by learning outcomes, Table 2 also aligns the learning outcome results with the participants' prior programming and declared major responses from the survey. We applied Pearson's chi-square (χ^2) tests to these results, and performed a statistical analysis using the R software environment [28]. The results showed the learning outcomes had a higher association with prior programming experiences, χ^2 (2, N=74)=3.160, p=0.697, than with the participants' declared major responses, χ^2 (2, N=74)=0.691, p=0.708. We report non-significant

Category	Response	Programming χ^2 (2, N=74)=3.160, p=0.697		Major χ^2 (2, N=74)=0.691, p=0.708	
		Experience	No Experience	Declared	Unknown
Below Course Learning Outcomes	37.27% (n=60)	26.47% (n=9)	73.53% (n=25)	41.18% (n=14)	58.82% (n=20)
At Course Learning Outcomes	33.54% (n=54)	37.04% (n=10)	62.96% (n=17)	51.85% (n=14)	48.15% (n=13)
Above Course Learning Outcomes	18.01% (n=29)	53.85% (n=7)	46.15% (n=6)	46.15% (n=6)	53.85% (n=7)
Did Not Answer Correctly	11.18% (n=18)				

Table 2: Assignment Designs Categorised by Course Learning Objectives

p values, which suggest the results are not unusual and consistent with the model assumptions [18].

5.3.1 Below Course Learning Outcomes. The results show 37.27% (n=60) of the assignment designs represent learning outcomes below these meant for the curriculum, unsuitable to present at the end of the CS1 course. When mapping the learning outcomes with survey responses, as shown in Table 2, we noticed participants with no prior programming experience (73.53%, n=25) contributed more to assignment designs in this category. For example, an assignment design asked and printed out a user's *"name and favourite country to travel"*. Because students begin to form mental models early their learning [34], this activity might be demonstrating the imprecise models formed early in students' learning process, which influence the simplistic assignment designs. A potential reason for these designs might be due to *defensive pessimism*, "unrealistically low expectations for ever succeeding or discount the importance of an assignment" [8, p. 182]. Because the activity was low stakes, participants might have perceived the effort to complete the activity outweighed its contribution to their overall grade for the course. Another potential reason for these simple designs might be a reaction to self-efficacy moments experienced while solving the activity, such as not understanding the problem or not knowing how to begin solving the activity [17]. The self-efficacy moments might have generated feelings of failure. To reduce these negative feelings, participants might have applied *failure-avoiding tactics* [8], potentially designing assignments they believed they could achieve. Though we provide plausible reasons for these designs, more research is required to draw conclusive explanations.

Assignment designs in this category could be perceived as unambitious, potentially designed by at-risk students needing help understanding how to solve the activity. We were interested in the voice of at-risk students, so we examined the designs from students that withdrew from the course. From the 20 students that withdrew, three participated in the activity. Because of the low participation rate from the at-risk students, we could not draw any conclusions about their voice. However, the low-participation results align with previous findings [14], showing poor performance in early assignments as an early indicator for identifying at-risk students. Any other assignment could also serve as a predictor, but the assignment design activity provided an additional layer that encouraged students to reflect on their learning goals. Because learning goals contribute to higher retention [33], the activity might help identify students without learning goals, giving educators the opportunity to intervene earlier in their learning.

5.3.2 At Course Learning Outcomes. Assignment designs (33.54%, n=54) in the *At Course Learning Outcomes* category required concepts and language constructs expected at the end of the CS1 course. The assignment designs were more involved, using familiar themes that required algorithms and planning for solving the problem. For example, Student154 developed an assignment averaging and sorting people's heights in descending order. Mapping the survey responses (See Table 2) to the results in this category does not show prior programming experience nor declared majors influencing the participants' assignment designs. More research is required to understand how students design appropriate assignments, especially those with no prior programming experience.

5.3.3 Above Course Learning Outcomes. Assignment designs (18.01%, n=29) classified in the *Above Course Learning Outcomes* category reveal participants applying problems they previously encountered in other STEM disciplines. For example, Student146 applied a robotic problem in their design, stating *"there are 5 objects in different color, shape and mass, student needs to design the structure and program that allow the simu-robotic arm to grasp the objects and put them in the circle that have corresponding color"*. Table 2 shows having a declared major or prior programming experience did not influence these assignment designs. Instead, these designs might be influenced by problems the participants encountered in other courses, which they then applied in the CS1 context; but additional research is required to support this theory.

6 THREATS TO VALIDITY

There are limitations to this study. First, 37.31% (n=200) of the cohort completed the survey, while fewer (30.04%, n=161) completed the assignment design activity. The low participation rate could be due to unmotivated students who do not value low-stakes assessments [41]. Volunteer bias is another threat to validity. Participants might have designed solutions because of their intrinsic motivations for taking the course, and wanting to share their interests in CS. To promote higher participation that better represents a CS1 cohort, future studies can make the study's instruments compulsory, giving the activities more weight in the final course grade.

7 CONCLUSION

This paper reports on a study that evaluates an assignment design activity as an instrument to collect student voice. The instrument gives students the opportunity to reflect on what they would like to achieve in CS1 courses through programming. The main contributions are identifying the subject matter that interests students in programming assignments and their programming expectations when starting the course. Participants drew on prior knowledge and experiences to create their designs, demonstrating transfer of training that can support the transition to the role of programmer. Educators can support transfer of training by blending subject matter, such as maths, games, and socially relevant activities, into CS1 programming assignments that are familiar to students.

Using participants' assignment designs to identify their range of learning expectations opens up future research opportunities. Though the range of expectations is not surprising, the results show students with no prior programming experience having lower learning expectations, which could impact their time and effort studying, and potentially influence their performance in the course. Helping students understand the faculty's expectations might raise students' awareness of what is required from them to successfully complete the course [32]. Future research can evaluate interventions to help students better understand the effort required to succeed in CS1 courses [29]. For example, educators could demonstrate early in the semester what students will accomplish by the end of the course. Future research can report on how students react to such demonstrations, to determine whether they are helpful in adjusting students' studying efforts.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Cheryl Pope for supporting the data collection process and helpful feedback on this work.

REFERENCES

[1] Engineers Australia. 2021. *Accrediatation Management System 2019.* Retrieved Feb 3, 2021 from https://www.engineersaustralia.org.au/sites/default/files/2019-11/Stage1_Competency_Standards.pdf

[2] Sandra Britton, Peter New, Andrew Robers, and Manjula Sharma. 2007. Investigating students' ability to transfer mathematics. In *Transforming a University. The Scholarship of Teaching and Learning Practice*, Angela Brew and Judyth Sachs (Eds.). Sydney University Press, Sydney, 127–140.

[3] Elizabeth Buckingham. 1997. Generic numeracy: Where does it live? Workers' views of problem solving at work. In *Mathematics, Creating the Future: Proceedings of the 16th Biennial Conference of the Australian Association of Mathematics Teachers*, N. Scott and H. Hollingsworth (Eds.). Australian Association of Mathematics Teachers, 100–103.

[4] Angela Carbone, John Hurst, Ian Mitchell, and Dick Gunstone. 2000. Principles for designing programming exercises to minimise poor learning behaviours in students. *ACSE '00 Proceedings of the 2016 ITiCSE* (2000), 26–33.

[5] Lori Carter. 2006. Why Students with an Apparent Aptitude for Computer Science Don't Choose to Major in Computer Science. *SIGCSE Bull.* 38, 1 (March 2006), 27–31.

[6] Alison Cook-Sather. 2006. Sound, presence, and power: "Student Voice" in educational research and reform. *Curriculum Inquiry* 36, 4 (2006).

[7] Alison Cook-Sather. 2014. Multiplying perspectives and improving practice: What can happen when undergraduate students collaborate with college faculty to explore teaching and learning. *Instructional Science* 42, 1 (2014), 31–46.

[8] Martin V. Covington. 2000. Goal theory, motivation, and school achievement: An integrative review. *Annual Review of Psychology* 51, 1 (2000), 171–200.

[9] John W. Creswell. 2012. *Educational research: Planning, conducting, and evaluating quantitative and qualitative research.* Pearson.

[10] John W. Creswell and Vicky L. Plano Clark. 2006. Choosing a mixed methods design. In *Designing and Conducting Mixed Methods Research.* Thousand Oaks, CA: Sage Publications, Chapter 4, 125–143.

[11] Paul Denny, John Hamer, Andrew Luxton-Reilly, and Helen Purchase. 2008. PeerWise: Students sharing their multiple choice questions. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08).* 51–58.

[12] Françoise Détienne. 2002. *Software Design—Cognitive Aspects.* Springer-Verlag, Berlin, Heidelberg. 21–41 pages.

[13] Joseph A. Eppink. 2002. Student-created rubrics: An idea that works. *Teaching Music* 9, 4 (Feb 2002), 28–32.

[14] Varick Erickson. 2019. Identifying at-risk computer science students early in semester utilizing data-driven models. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI).* 865–870.

[15] Rita Garcia. 2021. Student voice in the assignment design process. *EDIL '21 EdMedia + Innovate Learning Conference* (July 2021), 346–355.

[16] Peter Gill. 1999. The physics/maths problem again. *Physics Education* 34, 2 (1999), 83–87.

[17] Jamie Gorson and Eleanor O'Rourke. 2020. Why do CS1 students think they're bad at programming? Investigating self-efficacy and self-assessments at three universities. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20).* 170–181.

[18] Sander Greenland, Stephen J. Senn, Kenneth J. Rothman, John B. Carlin, Charles Poole, Steven N. Goodman, and Douglas G. Altman. 2016. Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations. *European Journal of Epidemiology* 31, 4 (2016), 337–350.

[19] Valerie Hall. 2007. A tale of two narratives: Student voice – What lies before us? *Oxford Review of Education* 43 (2007), 180–193. Issue 2.

[20] Raija Hämäläinen, Carita Kiili, and Blaine E. Smith. 2017. Orchestrating 21st century learning in higher education: A perspective on student voice. *British Journal of Educational Technology* 48, 5 (2017), 1106–1118.

[21] Beryl Hesketh. 1997. Dilemmas in training for transfer and retention. *Applied Psychology: An International Review* 46, 4 (1997), 317–386.

[22] Perpetua Kirby, Claire Lanyon, Kathleen Cronin, and Ruth Sinclair. 2003. Building a culture of participation: Involving children and young people in policy, service planning, development and evaluation. *A Research Report and Handbook* (2003).

[23] Christian Köppe and Leo Pruijt. 2014. Improving students' learning in software engineering education through multi-level assignments. *CSERC '14 Proceedings of the Computer Science Education Research Conference* (Nov. 2014), 57–62.

[24] Lucas Layman, Laurie Williams, and Kelli Slaten. 2007. Note to self: Make assignments meaningful. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (March 2007), 459–463.

[25] Alison Mackey and Sue M. Gass. 2005. *Second Language Research: Methodology and Design.* Lawrence Erlbaum Associates, Mahwah, NJ.

[26] Catherine Marshall and Gretchen B. Rossman. 1999. *Designing Qualitative Research* (3rd ed.). Sage Publications, London.

[27] Nea Pirttinen, Vilma Kangas, Irene Nikkarinen, Henrik Nygren, Juho Leinonen, and Arto Hellas. 2018. Crowdsourcing Programming Assignments with CrowdSorcerer. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (Larnaca, Cyprus) *(ITiCSE 2018).* Association for Computing Machinery, New York, NY, USA, 326–331.

[28] R Core Team. 2019. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.r-project.org

[29] Nathan Rountree, Janet Rountree, and Anthony Robins. 2002. Predictors of Success and Failure in a CS1 Course. *SIGCSE Bull.* 34 (Dec. 2002), 121–124.

[30] Jean Rudduck and Julia Flutter. 2000. Pupil participation and pupil perspective: 'Carving a new order of experience'. *Cambridge Journal of Education* 30, 1 (2000), 75–89.

[31] Jean J. Ryoo, Jane Margolis, Cynthia Estrada, Tiera Chante Tanksley, Dawn Guest-Johnson, and Sophia Mendoza. 2019. Student voices: Equity, identity, and agency in CS classrooms. In *2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT).* 1–2.

[32] Karen Maitland Schilling and Karl L. Schilling. 1999. Increasing expectations for student effort. *About Campus* 4, 2 (1999), 4–10.

[33] Duane F. Shell, Leen-Kiat Soh, Abraham E. Flanigan, and Markeya S. Peteranetz. 2016. Students' Initial Course Motivation and Their Achievement and Retention in College CS1 Courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16).* Association for Computing Machinery, New York, NY, USA, 639–644.

[34] Derek Sleeman, Ralph T. Putnam, Juliet Baxter, and Laiani Kuspa. 1986. Pascal and high school students: A study of errors. *Journal of Educational Computing Research* 2, 1 (1986), 5–23.

[35] Suzanne Soo Hoo. 1993. Students as partners in research and restructuring schools. *The Educational Forum* (1993), 386–393.

[36] Dennis Thiessen. 2007. Researching student experiences in elementary and secondary school: An evolving field of study. In *International handbook of student experience in elementary and secondary school*, D. Thiessen & A. Cook-Sather (Ed.). Springer, Dordrecht, The Netherlands, 1–76.

[37] Ari Tuhkala, Antti Ekonoja, and Raija Hämäläinen. 2020. Tensions of student voice in higher education: Involving students in degree programme curricula design. In *Innovations in Education and Teaching International.* 1–11.

[38] UK Department of Education. 2021. Listening to and involving young people. https://www.gov.uk/government/publications/listening-to-and-involving-children-and-young-people. [Online; accessed 20-May-2021].

[39] UK Legislation. 2002. Education Act. http://www.legislation.gov.uk/ukpga/2002/32/contents. [Online; accessed 20-May-2021].

[40] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. 2011. Extreme apprenticeship method in teaching programming for beginners. *SIGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (March 2011), 93–98.

[41] Lisa F. Wolf and Jeffrey K. Smith. 1995. The consequence of consequence: Motivation, anxiety, and test performance. *Applied Measurement in Education* 8, 3 (1995), 227–242.

[42] Noor Faridatul Ainun Zainal, Shahrina Shahrani, Noor Faezah Mohd Yatim, Rohizah Abd Rahmand, Masura Rahmat, and Rodziah Latih. 2011. Students' perception and motivation towards programming. *Procedia – Social and Behavioral Sciences* 59 (2011), 277–286.