Project-Based Software Engineering Curriculum for Secondary Students

Isabella Gransbury
North Carolina State University
Raleigh, NC, USA
igransb@ncsu.edu

Janet Brock
North Carolina State University
Raleigh, NC, USA
jdbrock@ncsu.edu

Emily Root
North Carolina State University
Raleigh, NC, USA
eeroot@ncsu.edu

Veronica Catete
North Carolina State University
Raleigh, NC, USA
vmcatete@ncsu.edu

Tiffany Barnes
North Carolina State University
Raleigh, NC, USA
tmbarnes@mcsu.edu

Shuchi Grover
Looking Glass Ventures
Palo Alto, CA, USA
shuchi.stanford@gmail.com

Akos Ledeczi
Vanderbilt University
Nashville, TN, USA
akos.ledeczi@vanderbilt.edu

ABSTRACT

Background. Software Engineering (SE) is a new and emerging topic in secondary computer science classrooms. However, a review of the recent literature has identified an overall lack of reporting on the development of SE secondary curriculum. Previous studies also report low student engagement when teaching these concepts. **Objectives**. In this experience report, we discuss the development of a 9-week, project-based learning (PBL) SE curriculum for secondary students. During this curriculum, students create a socially relevant project in groups of two to three. We discuss displays of participant engagement with CS concepts through the PBL pedagogy and the SE curriculum. **Method**. We examine participant engagement through group artifact interviews about student experiences during a week-long, virtual summer camp that piloted activities from our curriculum. During this camp, students followed a modified SE life cycle created by the authors of the paper. **Findings**. Participants showed engagement with the curriculum through various aspects of PBL, such as autonomy, creativity, and personal interest in their project topic. **Implications**. The lessons learned from this experience report suggest that PBL pedagogy can increase student engagement when teaching CS concepts, and this pedagogy provides detail and structure for future secondary SE curriculum implementations to support educators in the classroom.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; **Computer science education**; **Software engineering education**.

KEYWORDS

software engineering education, K-12, K-12 computer science education, project-based learning pegadogy

ACM Reference Format:
Isabella Gransbury, Janet Brock, Emily Root, Veronica Catete, Tiffany Barnes, Shuchi Grover, and Akos Ledeczi. 2023. Project-Based Software Engineering Curriculum for Secondary Students. In *The 18th WiPSCE Conference on Primary and Secondary Computing Education Research (WiPSCE '23), September 27–29, 2023, Cambridge, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3605468.3605501

1 INTRODUCTION

There has been growing interest in integrating software engineering (SE) concepts with K-12 computer science (CS) curriculum. Most of these integrations have taken place in secondary schools and include concepts such as testing and software development cycles. However, in a recent systematic mapping study, it was identified that the K-12 SE literature lacks details of the course materials [27]. It was also identified that there has been a lack of student engagement when teaching SE concepts [27].

The goal of this experience report is to *share our findings of student engagement with CS from exposure to our modular project-based learning (PBL) SE curriculum for secondary students* [19, 30]. In our SE module, students work within groups to develop a final project over a duration of 9 weeks. These projects are meant to be socially relevant and connect to students' interests to make the topic of CS more engaging. We use a block-based learning environment that connects to Application Programming Interfaces (APIs), which, in turn, provide students with a wide range of online datasets to use in their projects. Students also learn about various SE topics such as Human-Computer Interaction, Prototyping, and the various software development life cycles.

To pilot this curriculum, we ran a virtual week-long summer camp that implemented activities from our SE module. We conducted group artifact interviews with participants to understand their engagement with camp activities and materials. We chose to analyze group project interviews using thematic analysis to find patterns in participant experiences [23].

Three themes emerged from the thematic analysis of group interviews: *Challenges With Design*, *Attitudes Towards Project*, and *Attitudes Towards Collaboration*. We found students showed engagement in CS with the curriculum materials and with various aspects of PBL pedagogy, such as student autonomy and creativity.

1.1 SE in Primary and Secondary Education

There has been a growing interest in software engineering (SE) education to prepare K-12 students for the workforce. Despite the relevance of SE education for K-12 students, most of the available instructional units lack detail and are limited for large-scale use [27]. Previous research has also indicated that educators fear that SE concepts are too complicated for secondary students [14]. Hermans and Aivaloglou address this in their findings of a massive open online course (MOOC) that introduces K-12 students to software engineering concepts [14]. One main finding was that students thought that software engineering concepts were no more difficult to understand than foundational programming concepts [14]. In another study investigating students' ability to learn SE best practices, Gutierrez et al. confirmed that primary and secondary students are more than capable of comprehending SE concepts [13].

1.2 Project-Based Learning

Our curriculum aims to provide the K-12 CER community with a detailed SE curriculum for secondary students using project-based learning (PBL) [19, 30], which has been found to have significant benefits in increasing student performance, communication and engagement when used in CS classes [11]. PBL is a student-centered pedagogy in which students decide what actions they wish to take to solve a problem [19, 30]. Project-based learning allows instructors to teach SE concepts and best practices while helping students develop new collaboration skills. This pedagogy is especially relevant to teaching SE due to the importance of teamwork and agile development in CS. In software development teams, better teamwork quality has been associated with better team performance and improvements in personal success and learning of the team member [15, 22].

Our SE module is part of a larger course, *Computer Science Frontiers* (CSF) [20] , designed for students to take after completing the US College Board's Advanced Placement Computer Science Principles (AP CSP) course [8]. A purpose of this course is to trigger secondary student interest in CS early in their academic careers by introducing them to advanced topics, such as Artificial Intelligence (AI), Internet of Things (IoT), and SE, that they would typically not be exposed to until undergraduate studies. We target secondary students in this curriculum because previous research has shown that female interest in CS develops in high school, when many other career choices occur [24].

This introduces another goal of this course: broadening participation in CS by increasing young girls' engagement in the CS curriculum. Previous research has shown that young girls are mostly interested in topics of social relevance in interdisciplinary and people-oriented fields [18, 29]. Socially relevant projects have also been shown to increase the participation of underrepresented and young female students [7, 10, 26]. Therefore, we centered our curriculum around socially relevant and people-oriented projects, similar to how the other modules in the CSF course are structured [1, 12, 31].

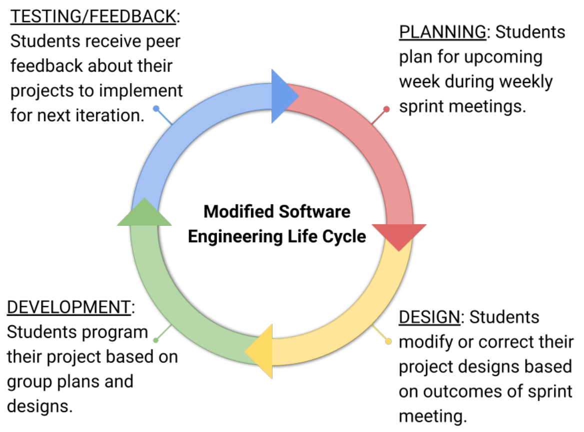

TESTING/FEEDBACK: Students receive peer feedback about their projects to implement for next iteration.
PLANNING: Students plan for upcoming week during weekly sprint meetings.
Modified Software Engineering Life Cycle
DEVELOPMENT: Students program their project based on group plans and designs.
DESIGN: Students modify or correct their project designs based on outcomes of sprint meeting.

Figure 1: Diagram of the modified software engineering life cycle used in the SE module.

2 MODULE CURRICULUM

In this section, we give details about our SE module curriculum, a modified SE life cycle, previous work that uses similar PBL pedagogy and the block-based learning environment used in our curriculum.

2.1 Software Engineering Module

Our SE module curriculum enables students to learn about the software engineering process while creating programs that are accessible and socially relevant. In its full implementation, the SE module takes place over nine weeks and is designed to fit a typical secondary classroom schedule. Module materials have been adapted from a college-level SE course to be appropriate for secondary students. Examples of SE topics taught in the module curriculum include the following: documentation, product specifications, prototyping, project management, human-computer interaction (HCI), user experience, affinity diagramming, accessibility, and ethics. Each week, students will complete lessons related to a modified version of the agile SE life cycle that has been developed for this module (Figure 1). This version was developed by the authors of this paper who have experience in curriculum development for secondary students and have experience teaching an undergraduate SE course at a large research institution.

In total, there are four phases of our modified life cycle: *Planning*, *Design*, *Development*, and *Testing/Feedback*. In the *Planning* phase, students would answer questions such as, "What specific features will your program have to meet Sprint '#' requirements?", and "Who will be working on which tasks?", to develop a plan for their next iteration. This ensures that all members of the group know which tasks to complete and that projects are completed in a timely manner. In the next phase, *Design*, students are able to modify their design based on concepts from their previous lesson or on tasks delegated in the planning meeting. During the *Development* phase students would program the tasks they were deemed responsible for per the planning and design meetings. Finally, in the *Testing/Feedback* phase, students would test their programs and give other groups feedback on how to improve their programs.

Every three days, each group would complete an iteration of this cycle with the goal of completing their group project by the end of the module. The topics mentioned previously (e.g. human-computer interaction, documentation) will also be integrated into their classroom time. Preferably before the next planning phase, so students can apply concepts from the lesson to their next cycle iteration. Contingent on publication, online access to our curriculum will be provided.

2.2 NetsBlox

To facilitate the introduction of advanced topics to students, the curriculum uses a block-based programming environment (BBP), NetsBlox. Similar studies that have introduced K-12 students to SE concepts also use BBP environments, such as Scratch [13, 14]. NetsBlox supports advanced programming techniques and connects to several Application Programming Interfaces (APIs) to enable interdisciplinary projects and access to online datasets [5, 6]. Examples of these datasets include, but are not limited to, climate and weather data, New York Times articles, and song lyrics. By forming this connection, students can create a wide range of projects on topics they are interested in. This can spark interest in CS in students who are typically not drawn to the subject [4]. This also opens up the possibility of device programming for students interested in app development.

Using a BBP language, reduces the possibility of syntax errors, while ensuring successive development of code [34]. This allows students to focus more on the computing concept they are learning than on the semantics of a programming language. NetsBlox also contains a collaboration feature, similar to Google Docs, that allows students to work on a project simultaneously on different machines [3]. The concurrent editing was crucial in the design of the SE module in order for students to program their project at the same time, on their own computers. During the virtual summer camp, this was vital as students were not physically together in a classroom environment.

3 RESEARCH METHODS

In this section, we give details about the study and research methods, including camp implementation, facilitators, and participants. We also describe the restructured camp curriculum, interview protocol, and analysis of interview transcripts.

3.1 Camp Implementation

We condensed the SE module into a virtual, week-long summer camp curriculum to test the material with secondary students. Students participated in the camp for seven hours each day. As a requirement to attend the SE camp, the students had to attend at least one of the camps that we facilitated the two weeks prior. These camps corresponded to other modules of the CSF curriculum (AI and IoT) and were also taught using NetsBlox mentioned in Section 2.2.

The camp was facilitated using the Zoom online platform [32]. Zoom allowed remote communication between groups with the breakout room feature, screen sharing, and audio/video recording. Group interviews for this study were conducted on the fourth day of camp. How the students were grouped is discussed in Section 3.2.

3.1.1 Camp Facilitators. The SE camp was co-taught by five teachers, one male and four female. All teachers have experience teaching the AP CSP course to secondary students. All camp facilitators also participated in a week-long professional development (PD) several weeks before the camp. The purpose of this PD was for teachers to become familiar with the materials of the SE camp and ask questions about unfamiliar concepts. Four out of five of the teachers also have experience facilitating camps for the other modules, AI and IoT, in the full CSF curriculum. Camp facilitators were also joined by one female graduate student and one female undergraduate student.

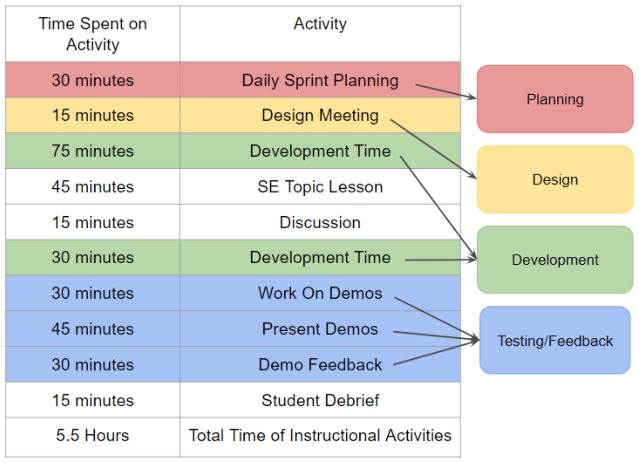
3.1.2 Participants. A total of eight students, two female and six male, consented to be part of this study. Of the six male participants, 5 self-identified as Southeast Asian/Indian, and one self-identified as East Asian /Pacific Islander/Asian Other. Of the two female participants, one identified as Southeast Asian/Indian, and one self-identified as East Asian /Pacific Islander /Asian Other. All participants were recruited from the institutions of the teacher facilitators mentioned above.

3.2 Restructured Camp Curriculum

On the first day of camp, students were introduced to SE and the criteria for the projects they would be creating throughout the week. Following the introduction, students created proposals for their project and presented them. Then, students chose their three favorite proposals and ranked them in order of most to least preferred. After the rankings were submitted, students were put into groups based on their pitch rankings; This resulted in four different groups: three groups of two students and one group of three students (Note: not all students participated in interviews). The next two days followed a schedule similar to that shown in Figure 2. This figure also connects the activities from the camp to the modified software development life cycle discussed in section 2.1

During these days, the camp began with group sprint planning, then a project design meeting. After the design meeting, the groups had 75 minutes of development time. The topic of the day was presented after the development time and was followed by an hour-long break, then 30 minutes more development time. The SE topics that were taught throughout the week were Documentation, Project Management, HCI, and Play Testing. At the end of the day, the groups presented short demos of their projects to show other students and facilitators for feedback.

On the fourth day of camp, the students completed play testing of each group's project. They received peer feedback on how to improve their project and make it more accessible. After play testing, the groups were given development time for one hour and 45 minutes. During this time, two researchers (female) conducted



Time Spent on Activity	Activity
30 minutes	Daily Sprint Planning
15 minutes	Design Meeting
75 minutes	Development Time
45 minutes	SE Topic Lesson
15 minutes	Discussion
30 minutes	Development Time
30 minutes	Work On Demos
45 minutes	Present Demos
30 minutes	Demo Feedback
15 minutes	Student Debrief
5.5 Hours	Total Time of Instructional Activities

Figure 2: Example schedule of a day from the summer camp. Activities from the camp are mapped to difference phases of the curriculum's modified software development life cycle.

group project interviews. The students had interacted with these researchers every day at the camp, so they had gained a level of familiarity with them. On the last day of camp, the students received more development time before their final presentations. Finally, the groups presented their projects to the camp and were able to interact with other projects using a feature in NetsBlox.

3.3 Interview Protocol

To gain insight into student's experiences during the camp we conducted group project interviews with each of the four groups. Questions asked in these interviews related to student engagement with activities and attitudes towards their artifacts. All interviews took place on the fourth day of camp and were approximately 30 minutes long. Two female graduate students facilitated the interviews through Zoom. One researcher asked questions, while the other took notes on participant responses. The following questions asked in these interviews are shown in Table 1.

Table 1: Questions asked during group interviews.

Group Questions
1. What is your project about?
2. How did you decide on your project?
3. Now can one of you share your screen and walk me through the code and show me what the different parts do?
4. What were some bugs you had to deal with or are dealing with?
5. What was the most challenging part of going through the SE process for your project?
Individual Questions
1. How is this project similar to or different from previous projects you've worked on?
2. What are you most proud of about creating your project?
3. Overall how would you describe this project experience?

3.4 Interview Analysis

After completing the group project interviews, three researchers used thematic analysis to analyze the transcript data [23]. Thematic analysis involves a team of researchers coding qualitative data, then condensing that data into common themes. This methodology is often used to analyze dialog in study participant interviews [23]. Due to the low population size of the study, a qualitative approach gives us greater insight into student engagement with SE concepts, compared to quantitative data. During our thematic analysis we completed the following 4 phases: (1) Review data and rectify transcripts; (2) Code first interview together and discuss codes; (3) Code other three interviews; (4) Group codes into themes. We completed thematic analyses for all group project interviews and coded the first interview together to establish a norm across all four interviews.

4 RESULTS: THEMATIC ANALYSIS

This section describes the results of the thematic analysis of group interviews. Groups were formed based on the interest the participants had in the project proposals on the first day of camp. Common themes we found across all four groups consist of: *Challenges With Planning / Design*, *Attitudes Towards Project*, and *Attitudes Towards Collaboration*. Table 2 displays participants responses to the group interview questions that display engagement with our curriculum and activities.

5 DISCUSSION

5.1 Student Engagement

As mentioned in the literature review, the pedagogical method of PBL has been shown to increase student engagement when used in CS classes. An initial objective of this study was to determine the effects of using PBL on student engagement in our curriculum. A surprising result was that the student in Group 1 showed engagement with her project, despite not having a partner who also contributed code to the project. Previous research has shown that young women enjoy collaborating when actively learning CS, therefore increasing their engagement [21]. The student explained that her strong feelings were due to her interest in the focus of her project. It may be the case that if it is not possible for a woman to collaborate while learning CS, the use of PBL may counteract the engagement lost from not being able to learn collaboratively.

Another important finding was that the students in Group 2 and Group 3 specifically commented on their engagement of the camp due to the use of the PBL pedagogy. The students in Group 2 said that they enjoyed being able to implement a concept they had previously learned in "their own way". This comment directly relates to autonomy, which conveys that one's behavior is an expression of their self [28]. Student autonomy is one of three psychological needs that serve as the main driver of human behavior, as stated in Self-Determination Theory (SDT) [17]. STD is a theoretical lens within developmental and educational psychology to understand how humans are self-motivated and engaged [9]. This finding suggests that the use of PBL pedagogy can cause students to engage with concepts being taught through student autonomy.

Table 2: Student responses from the group interviews organized into the themes found through thematic analysis.

Groups	Challenges With Planning / Design	Attitudes Towards Project		Attitudes Towards Collaboration
Flood Prevention (1F)	"I think it was during the beginning stages on like Monday, Tuesday, when I was still trying to figure out like what I wanted to do with the app."	She commented on the "positive outlook" she had throughout the project because of her interest in the project topic.	One student reported it was "nice" to be able to use that code as a starting point and make the code their "own" through the SE project	"I enjoyed it when the instructors came into the breakout rooms. I also liked an overview of the day. The instructors were extremely kind and easy to reach out for help."
Inspirational Quotes (1F, 1M)	They decided to begin testing different sentiment analysis techniques by "[testing] different codes to see what is going to work better".	The young woman said that they were proud of their project because of its possibility of helping people and "motivat[ing] them to do what they want".	Both students commented on being able to use this concept "in their own way" by implementing it in their program.	"It was fun thinking of ways to solve our problems together"
Drawing Game (3M)	Early in the week, the members of group 3 realized that they may not be able to implement more than 2 users at the same time.	".... [the user] can sort of customize it to whatever they feel like and not just have to ... deal with [the program] because they ... [can] make it fit their needs".	"better experience than what I had before [and] had more creativity to add what we wanted"	"... working alone can get a little bit tedious but ... working with two or three people ... it's a lot more fun"
Remote Control App (2M)	One student explained that it was hard for the pair to organize their thoughts such as "... what should we do on this day, what should we do tomorrow, what should we do day after?"	"[They] were able to create something that could actually help people"	Another member of the group liked that you could also have fun with their project and that it also has "a serious point"	They commented they enjoyed being able to work with a partner because they were able to overcome their "roadblocks" easier than if they were working alone.

We will now consider comments made by a student in Group 3 who had previously taken CS courses: "[The camp was a] better experience than what I had before [and] [w]e had more creative [freedom]. We didn't really have projects [in previous classes] because there was a set curriculum". This student expressed that he was able to engage with the curriculum more than his previous CS classes because of the creative aspect of the PBL pedagogy. Previous literature has shown that creative thinking can increase self-efficacy [25, 33], leading to increased enjoyment and engagement in CS introductory courses [16]. Therefore, it can be assumed that the creative aspects of PBL can lead to student engagement of CS concepts. These findings raise intriguing questions regarding the nature and extent of student autonomy and creativity in CS classes and their important roles in student engagement with CS.

5.2 Lessons Learned: For Practitioners

The initial objective of this study was to determine student engagement with our project-based learning curriculum in a virtual summer camp. Below we discuss the lessons learned from a practitioners stand point for future implementation of our curriculum:

- When implementing this module and SE in general, it would be beneficial for students to have completed coding activities they can draw inspiration from for their projects. All groups used concepts from activities in the camp they completed the week prior to the SE camp in their projects. We believe this also had a large impact on their engagement of the SE camp since they had knowledge of how to implement other CS concepts besides SE.
- Students also seemed to be more engaged due to the collaborative aspects of the camp. One student said it made him feel more creative, while another student said it was easier to overcome the obstacles they had with more than one person working on an program. Another student also said their group was able to create more ideas for their project together versus them brainstorming ideas on their own.
- When pairing secondary students it is important to consider the factors of familiarity (students who are friends with each other), previous programming experience, and project interest. Some factors may produce more productive and collaborative groups than others, and greatly depends on the students [2, 35] .

5.3 Limitations

A limitation of this study is that it was conducted virtually, thus we were not aware of the specific distractions in each student's environment. These distractions may have negatively impacted students engagement with the materials. Another limitation is low camp attendance; which makes these findings less generalizable for female engagement in our curriculum. Although only two female students interviewed, their experiences give us insight into possible ways to increase female engagement in CS using a SE curriculum.

Despite the relatively limited sample, this work also offers valuable information on overall student engagement in CS using SE.

6 CONCLUSIONS & FUTURE WORK

The purpose of the current study was to examine student engagement with CS using our SE curriculum after participating in a virtual week-long camp. The first major finding was students expressed their engagement with CS through statements about student autonomy and creativity. Both of these concepts are directly related to the use of the PBL pedagogy. The findings reported here shed new light on how to use advanced CS topics, such as SE, in CS classrooms to increase student engagement. More research using a traditional classroom environment and a larger sample size could provide insight into how the curriculum specifically directly affects the engagement of women and underrepresented students.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1949472, 1949488, and 1949492. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Lauren Alvarez, Isabella Gransbury, Veronica Cateté, Tiffany Barnes, Akos Lédeczi, and Shuchi Grover. 2022. A Socially Relevant Focused AI Curriculum Designed for Female High School Students. *Thirty-sixth AAAI Conference on Artificial Intelligence* (2022).
[2] Margarita Azmitia and Ryan Montgomery. 1993. Friendship, transactive dialogues, and the development of scientific reasoning. *Social development* 2, 3 (1993), 202–221.
[3] Corey Brady, Brian Broll, Gordon Stein, Devin Jean, Shuchi Grover, Veronica Cateté, Tiffany Barnes, and Ákos Lédeczi. 2022. Block-based abstractions and expansive services to make advanced computing concepts accessible to novices. *Journal of Computer Languages* (2022), 101156.
[4] Brian Broll, Akos Lédeczi, Gordon Stein, Devin Jean, Corey Brady, Shuchi Grover, Veronica Catete, and Tiffany Barnes. 2021. Removing the Walls Around Visual Educational Programming Environments. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–9.
[5] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L Weeden-Wright, Chris Vanags, Joshua D Swartz, and Melvin Lu. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 81–86.
[6] Brian Broll, Péter Völgyesi, János Sallai, and Akos Lédeczi. 2016. NetsBlox: A visual language and web-based environment for teaching distributed programming.
[7] Stephen J Ceci, Donna K Ginther, Shulamit Kahn, and Wendy M Williams. 2014. Women in academic science: A changing landscape. *Psychological science in the public interest* 15, 3 (2014), 75–141.
[8] Jan Cuny. 2015. Transforming K-12 computing education: AP® computer science principles. *ACM Inroads* 6, 4 (2015), 58–59.
[9] Edward L Deci and Richard M Ryan. 1985. The general causality orientations scale: Self-determination in personality. *Journal of research in personality* 19, 2 (1985), 109–134.
[10] Allan Fisher and Jane Margolis. 2003. Unlocking the clubhouse: Women in computing. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*. 23.
[11] Alexandre Grotta and Edmir Parada Vasques Prado. 2019. Benefits of the Project-Based Learning to Cope with Computer Programming Education: A Systematic Literature Review. http://pbl2019.panpbl.org/wp-content/uploads/2019/09/AlexandreGrottaBenefitsoftheproject-basedlearning.pdf
[12] Shuchi Grover, Veronica Cateté, Tiffany Barnes, Marnie Hill, Akos Ledeczi, and Brian Broll. 2020. FIRST principles to design for online, synchronous high school CS teacher training and curriculum co-design. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. 1–5.
[13] Francisco J Gutierrez, Jocelyn Simmonds, Nancy Hitschfeld, Cecilia Casanova, Cecilia Sotomayor, and Vanessa Peña-Araya. 2018. Assessing software development skills among K-6 learners in a project-based workshop with scratch. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 98–107.
[14] Felienne Hermans and Efthimia Aivaloglou. 2017. Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. 13–22. https://doi.org/10.1109/ICSE-SEET.2017.13
[15] Martin Hoegl and Hans Georg Gemuenden. 2001. Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence. *Organization Science* 12, 4 (2001). https://pubsonline.informs.org/doi/abs/10.1287/orsc.12.4.435.10635
[16] Geetha Kanaparan, Rowena Cullen, David Mason, et al. 2019. Effect of self-efficacy and emotional engagement on introductory programming students. *Australasian Journal of Information Systems* 23 (2019).
[17] Virginia Grow Kasser and Richard M Ryan. 1999. The relation of psychological needs for autonomy and relatedness to vitality, well-being, and mortality in a nursing home 1. *Journal of Applied Social Psychology* 29, 5 (1999), 935–954.
[18] Chris Michael Kirk, Rhonda K Lewis, Kyrah Brown, Corinne Nilsen, and Deltha Q Colvin. 2012. The gender gap in educational expectations among youth in the foster care system. *Children and Youth Services Review* 34, 9 (2012), 1683–1688.
[19] Joseph S Krajcik and Namsoo Shin. 2014. Project-based learning. In *The Cambridge Handbook of the Learning Sciences*, R Keith Sawyer (Ed.). Cambridge University Press, Cambridge, 275–297.
[20] Ákos Lédeczi, Shuchi Grover, Veronica Catete, and Brian Broll. 2021. Beyond CS Principles: Bringing the Frontiers of Computing to K12. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 1379–1379.
[21] Janet Liebenberg, Elsa Mentz, and Betty Breed. 2012. Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education* 22, 3 (2012), 219–236.
[22] Yngve Lindsjørn, Dag I.K. Sjøberg, Torgeir Dingsøyr, Gunnar R. Bergersen, and Tore Dybå. 2016. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software* 122 (2016), 274–286. https://doi.org/10.1016/j.jss.2016.09.028
[23] Moira Maguire and Brid Delahunt. 2017. Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars. *All Ireland Journal of Higher Education* 9, 3 (2017).
[24] Microsoft Corporation. 2017. Why Europe's Girls Aren't Studying STEM. https://news.microsoft.com/europe/features/dont-european-girls-like-science-technology/. Accessed: 2021-12-01.
[25] L Dee Miller, Leen-Kiat Soh, Vlad Chiriacescu, Elizabeth Ingraham, Duane F Shell, Stephen Ramsay, and Melissa Patterson Hazley. 2013. Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses. In *2013 ieee frontiers in education conference (fie)*. IEEE, 1426–1432.
[26] Marina Papastergiou. 2008. Are computer science and information technology still masculine fields? High school students' perceptions and career choices. *Computers & education* 51, 2 (2008), 594–608.
[27] Fernando D. C. Pinheiro, Von W. Christiane Gresse, and Raul M. Filho. 2018. Teaching Software Engineering in K-12 Education: A Systematic Mapping Study. *Informatics in Education* 17, 2 (2018), 167–206. Copyright - Copyright Institute of Mathematics and Informatics 2018; Last updated - 2019-04-15.
[28] Richard M Ryan, Edward L Deci, et al. 2002. Overview of self-determination theory: An organismic dialectical perspective. *Handbook of self-determination research* 2 (2002), 3–33.
[29] Oshani Seneviratne. 2017. Making computer science attractive to high school girls with computational thinking approaches: A case study. In *Emerging research, practice, and policy on computational thinking*. Springer, 21–32.
[30] Gwen Solomon. 2003. Project-based learning: A primer. *Technology and learning-dayton-* 23, 6 (2003), 20–20.
[31] Gordon Stein, Isabella Gransbury, Devin Jean, Lauren Alvarez, Marnie Hill, Veronica Catete, Shuchi Grover, Tiffany Barnes, Brian Broll, and Akos Ledeczi. 2022. Engaging Female High School Students in the Frontiers of Computing. In *2022 ASEE Annual Conference & Exposition*.
[32] Zoom Video. 2020. Video conferencing, web conferencing, webinars, screen sharing.
[33] Shiyuan Wang, Duane F Shell, Abraham Flanigan, Markeya Peteranetz, and Leen-Kiat Soh. 2017. Impact of Creative Competency Exercises on College Computer Science Students' Learning, Achievement, Self-Efficacy, and Creativity. *AERA Online Paper Repository* (2017).
[34] David Weintrop. 2019. Block-based programming in computer science education. *Commun. ACM* 62, 8 (2019), 22–25.
[35] Laurie Williams, Lucas Layman, Jason Osborne, and Neha Katira. 2006. Examining the compatibility of student pair programmers. In *AGILE 2006 (AGILE'06)*. IEEE, 10–pp.