The Importance of Project-Scale Scaffolding for Retention and Experience in Computing Courses

Juliana Goncalves de Souza
Computer Science
Virginia Commonwealth University
Richmond, USA
juulianags@gmail.com

Mikaila Flavell
Electrical and Computer Engineering
Clarkson University
Potsdam, USA
flavelmm@clarkson.edu

Ahmad Daudu Suleiman
Electrical and Computer Engineering
Clarkson University
Potsdam, USA
suleimad@clarkson.edu

David Shepherd
Computer Science
Louisiana State University
Baton Rouge, USA
davidshepherd@gmail.com

Jan DeWaters
Institute For Stem Education
Clarkson University
Potsdam, USA
jdewater@clarkson.edu

Mary Margaret Small
Institute For Stem Education
Clarkson University
Potsdam, USA
mmsmall@clarkson.edu

Yu Liu
Electrical and Computer Engineering
Clarkson University
Potsdam, USA
yuliu@clarkson.edu

Daqing Hou
Electrical and Computer Engineering
Clarkson University
Potsdam, USA
dhou@clarkson.edu

***Abstract*— Teaching students complex problem-solving skills using large-scale, real-world problems is challenging for both students and teachers alike. As a result, most courses use small, well-specified, toy-like problems, which are not representative of what students will encounter in the workforce. One approach that allows teachers to use large-scale problems in class is by introducing scaffolding. Scaffolding breaks a larger problem into smaller steps, which students can solve independently, while deemphasizing tangential concepts such as the complex configuration files needed to compile open-source software systems. Strong scaffolding supports student learning, preventing them from getting bogged down with unnecessary tasks or overwhelmed by complexity. This work investigates a scaffolded problem-based-learning module for computing courses, using a realistically-sized project with characteristics representative of the industry. The project was implemented in a computer science course with roughly 100 students, and the results speak to the importance of scaffolding for student success. In fact, there were two student assignments that lacked sufficient scaffolding, compared with other tasks, and the reduction in student scoring and persistence shows that project scaffolding is necessary when implementing these types of assignments. Most students felt the project helped prepare them for a job in their chosen field.**

Keywords*— *scaffolding, project-based learning, problem-based learning, computer science pedagogy

I. Introduction

Teaching problem-solving skills is a well-known challenge. Students need to develop their own way of thinking to understand and analyze problems and develop solutions, rather than merely knowing how to regurgitate answers. Much like writing a novel, parts of the process can be taught, but it is ultimately up to the writer to piece the parts together into a cohesive whole.

To help with this process, scaffolding can help. Scaffolding is a teaching method that has been shown to be effective in teaching complex problem-solving [1, 2]. Scaffolding reduces a problem into simpler tasks that students can solve by themselves, while scaffolding - or sequencing - abstract concepts that students do not need to understand or master at that moment [3]. Students achieve the solution by following the teacher's instructions, completing smaller tasks using their previous knowledge, and developing skills they will need to complete future tasks [2, 4, 5, 6]. Applying this method in programming is possible and has shown positive results [7, 8]. Nonetheless, it has been used only for small examples.

In traditional Computer Science courses, students generally work with small projects that enable them to create programs from the beginning, implement each step, and complete all goals [9]. While students gain experience with software process development, the process does not reflect industry reality. When developers start a job, they often join teams or projects already in progress. Also, the amount of code and tools they need to manage is enormous compared to the academic environment they are trained in. Students need to adapt their knowledge and understand how the process works, generally doing this by themselves in a short time. What if we could use scaffolding to teach them with realistically sized projects that replicated the industry environment?

This paper describes a mixed-methods study that explores how students in a computing course respond to working with scaffolding at scale. We present background about the challenges of teaching programming, specifically at the early undergraduate level, and then describe the attributes and techniques of scaffolding. We describe the details of the course project, which enabled students to work on a large, complex project by progressing through a series of highly supported and scaffolded tasks that were equipped with instructions about navigating between files, changing or adding code, and testing results. Students experience a real-world situation in a safe environment, at the same time improving their problem-solving and code-reading skills, while seeing their work on individual tasks result in improvements to the overall application.

II. Background

A. Teaching Programming

Computer programming skills are generally taught to students early in their undergraduate curriculum, and their

experience forms an important foundation for the rest of their studies. At this point, students transform from being users of technology to becoming creators. The learning process is usually completely new and often quite difficult – yet this first impression can dictate the way a student will approach their relationship with computers and computer programming in the future [9, 10, 31]. A good relationship can be nurtured by an instructor who presents programming in a friendly way, for example by using examples from the students' lives and daily routines, creating a bridge between what they know and what they will learn, and showing that the programming logic is already used by them. Constructivism, a widely accepted learning theory, defends that knowledge is not independently incremented, it is an adaptive function. New information is related to something previously known and the student has an active role in the learning process [11, 12]. If the instructor helps with this connection, learning is faster and smoother. Similarly, while students can be inspired by exposure to the seemingly infinite potential of complex programming, too much complexity early on can be overwhelming [13]. It's better to demonstrate the power of complex programming, without getting mired in the details of how they work.

Still, programming – like nearly any type of problem-solving – is extremely difficult to teach because it is a creative and often individualized process. Students must be introduced to tools and how to use them, and then rather than the instructor explaining or modeling a solution, the students need to develop their own solutions. We can compare teaching programming with teaching how to paint, for instance. The teacher can explain painting methods, how to use colors, tools for painting and brushes, kinds of ink, and even how to reproduce some famous paintings, but in the end, the students need to create their own work. Like painting, it is impossible to teach everything in a few months. Students learn by doing – beginning with simple tasks and moving into more complex tasks as they practice and develop their abilities, eventually finding and organizing a solution. The teacher must choose appropriate tools and techniques for the student's level and goals.

The difficulties of teaching programming have long been recognized. Ismail [14] investigated the most common problems in the teaching of computer programming in Malaysia. They found the following four main issues, which they defended are the same in other countries: (i) lack of skills in analyzing problems; (ii) ineffective use of problem representation techniques for problem-solving; (iii) failure to understand and master programming syntax and constructs; and (iv) ineffective use of teaching strategies for problem-solving and coding.

The first issue can be addressed by introducing the concepts of logic, planning, discrete mathematics, and other skills at the appropriate level of the student. This will familiarize students with the idea of analyzing problems, and allow them to use their skills on problems at an appropriate level of difficulty. Giving students the opportunity to be successful at simple tasks will bolster their confidence to pursue more difficult problems. The second issue relates to the different paradigms that are used. Mayer [15] affirmed as early as 1981 that a concrete model is effective for providing a familiar context and goal that the student will try to enact. It is necessary to change the models of the solution representation while working in different paradigms. The pseudo-code, for instance, is not a good first representation when the teacher is explaining an object-oriented approach [14]. An earlier review by Dalbey and Linn [16] found that sharing many examples and exposing students to the possibility of seeing complete programs can help them write their own.

The third issue identified by Ismail et al. [14] relates to students' insufficient contact with programming syntax and constructs in the classes. While students spend a lot of time understanding concepts and representations of programming in the classroom, they do most of the exercises outside of the classroom, without supervision or real-time orientation. Students need to develop two skills simultaneously – rational logic, the more visible skill of understanding line-by-line text, as well as the less recognized skill of text interpretation. Students need to fully understand what they are being asked to do, and often the examples in the classroom are not enough to cover the needed knowledge.

The fourth issue identified by Ismail et al. [14] relates most closely to this work. Lin et al. [7], in 2021, emphasized that many students dropped computer science courses because of a lack of problem-solving skills. Students tend to give up when facing hard tasks. Ismail et al. [14] suggest that teaching students structured or procedural languages will better prepare them to learn more complex approaches, for example, object-oriented programming. Providing students the skills and knowledge they need to successfully accomplish simpler tasks, and working progressively toward more complex and difficult tasks, is a solution proposed by this work. Students have different levels of instructional needs to achieve the same goals – this was affirmed years earlier by Dalbey and Linn [16] who found that students at the University of California at Berkeley enrolled in self-paced programming courses earned higher grades compared with those enrolled in the lecture version. This finding demonstrates the importance not just of teaching strategies but of student engagement. If students are not motivated to complete the proposed activities, the learning outcomes will not be achieved. Improved teaching strategies will thus help achieve learning outcomes by improving student engagement and motivation [17]. One important way to do this is to provide scaffolding and support [14, 18], so students are able to approach complex, real-world problems in a stepwise fashion where they successfully accomplish simpler tasks and proceed to more difficult tasks as they gain the necessary knowledge and skills [2, 3, 30]. Active learning practices including project-based-learning, or PBL, also have been shown to help raise examination scores and reduce failure rates [13, 19, 32, 33].

B. Scaffolding Process

The idea of scaffolding was first discussed by Wood, Bruner, and Ross [20] almost fifty years ago. The authors defined scaffolding as a "process that enables a child or novice to solve a problem, carry out a task or achieve a goal which would be beyond his unassisted efforts". Parts of the work that are not necessary for the student to do, or are beyond the student's capacity, are done by the tutor so students can focus their efforts where they are most needed [3]. Wood et al. investigated problem-finding, not problem-solving. In the latter case, it is important to note that scaffolding only works when students are able to comprehend the solution.

Van de Pol et al. [1] analyzed 66 articles about scaffolding in teacher-student interactions and found two common characteristics. The first is adapting the task to the student's level, which requires the teacher to be aware of students' ability levels. The second is the idea of reducing the tutor's influence over time to transfer the responsibilities from teacher to student. The steps of the scaffolding process according to Wood et al. [20] are: (i) explain the goal and get the student's interest; (ii) simplify the task by reducing the problem into manageable parts; (iii) keep the student on track; (iv) mark relevant tasks; (v) provide help to minimize frustration; and (vi) show the solution in some way to the student so that they can imitate it. In this approach, the teacher/tutor and student are both active elements in the learning process, with the degree of influence by the tutor dependent on the situation [6].

A recent trend in scaffolding has been to use tools and technologies to help or, in some cases, replace the instructor [4, 34, 35, 36, 37]. In this case, the word `scaffold' can be used to represent the tools used to help a student complete a task. In other situations, particularly where student-teacher ratios are high, scaffolding techniques can be used to overcome limitations in the instructor's sensitivity to individual students' learning needs and capacities. Peer-to-peer learning, for example, allows one student to help another and complete the needed knowledge for each task. Even in an unconscious way, students and instructors are using a scaffolding process. Tutor interactions, scaffolds, and other strategies combine to create a distributed scaffolding [4].

Computer-based scaffolding is a well-known teaching strategy. Belland et al. [8] presented a meta-analysis of computer-based scaffolding in STEM education, analyzing 144 studies with 333 outcomes in total. They found that a major difference between scaffolding and traditional teaching methods is that, in computer-based scaffolding, the teacher needs to organize all of the detailed activities before applying them to students. There is no possibility of adjusting the messages once they are applied. This does not allow the instructor to adapt tasks depending on the student's response. Nevertheless, the results of using this strategy were positive. Students with computer-based scaffolding performed better on cognitive tests compared to students who did not use scaffolding. Another good result is that scaffolding is effective in all education levels, from primary school through adult. Although the idea began at the primary education level, the best results are shown at the graduate and adult levels.

III. Methods

A. *Scaffolded Project*

We created a course project using xFig, an "interactive drawing tool" written in C [5]. This choice was made based on three criteria:

1. Free and open source code: the students and instructors can access the code as needed.
2. Complete software application: we used software that was a complete program, as opposed to a library, including a user interface and a backend, so that we could implement a variety of tasks typical in software projects.
3. Visible results: because the software has a user interface, most changes are easy to verify if implemented properly, without extensive formal test development. Although testing is a necessary skill, it was outside the scope of this project and course.

Prior to this work, the xFig project, and the corresponding tasks, were used in Fall 2021 (described in detail in [21]). Tasks were prepared with increasing difficulty; the first task required only a few minutes to complete, while the last was relatively complex, requiring somewhat extensive work and building on knowledge from completing the previous tasks. Tasks were assigned during class and hints were released each following day until the task was due. The hints included descriptions of the functionality, explanations of code chunks, and files or specific code lines that should be changed. Most students completed the tasks but had difficulty following the project flow and instructions. The programming environment was also an issue, as students experienced problems compiling and debugging the code. During this first iteration extensive feedback was collected and major updates were made to the project scaffolding.

We used feedback from the first iteration to create HITCH (Hashtags, Information, Targets, Challenges, and Hints), our scaffolded approach for realistic tasks. This approach uses code comments to direct the students. These comments are semi-structured with elements thought to increase the scaffolding level. The hashtags are used for searching, identifying the places in the code that should be changed. The pieces of information describe the code and include necessary concepts. Targets define what the student should do in that part of the task. Challenges introduce optional activities. The hints guide students in the right way to complete what is needed. We created a GitHub repository, adding the original xFig code, creating a different branch for each task, and inserting comments - found by searching for the task hashtag - in each file that should be changed. A pre-prepared VM (Virtual Machine) was included on AWS (Amazon Web Service), further reducing the configuration problems. It included the xFig program ready to be compiled, a script that starts the compilation process, one IDE (Integrated Development Environment) to edit the code, and the GitHub Desktop program to get the tasks in their repository. The instructions included a detailed tutorial with videos about configuring this VM and executing the tasks.

Each task has a file "Assignment Information.txt" that has: (i) the name of the task; (ii) the task goal; (iii) the link for a GitHub page in the main repository with more explanation about the goal; (iv) orientation about what and how to do; (v) a written description of a test; and (vi) a link for a video with a general explanation about xFig and the project. Item (iii) is for students that did not understand the problem. On the cited page are helpful examples and figures. The orientation of item (iv) is almost the same for all tasks. It shows how to open the code, find the file where to work, and compile the code. We provide different types of information to accommodate different ways of learning. Some students can understand better by reading a tutorial, others by watching a video. Some students like to explore the GitHub repository, and others feel more confident having a direct link to access the information.

As before, in each task, we increase the difficulty in two aspects: knowledge about the xFig code and the complexity of programming concepts. Part of scaffolding is adapting the activities to students' level and observing the evolution to

continue adapting, which is complicated when working with a big group of students with previously-prepared tasks. Each task included challenges and optional instructions to maintain the students' engagement. Students with less programming ability can complete just the obligatory instructions to finish the task's main objective. Those with more knowledge or problem-solving skills can improve the solutions. In some tasks, the challenge is just to make the added function more user-friendly, others involve changing the code to make it more efficient, adding different ways to use the new function, or even creating a new optimization algorithm.

Students could locate the places they need to change by searching for hashtags. The IDE indicates the files that have the hashtags and all occurrences of them. The kind of information was separated using uppercase words as tags, like INFO and HINT, for example. The files to be changed have comments with: (i) the task hashtag; (ii) delimitations of the changing scope; (iii) the goal in that file and why; (iv) hints; (v) foreshadowing; and (vi) challenges. Item (iv) is created with open questions that make the students think about topics that will be used in future tasks.

To show an example of our approach, we present the entire second task scaffolding here. Users start with an informational file. The following code is the "Assignment Information.txt" file content for the second task.

```
#taskDefaultDepth

---------- Instructions Start Here -----------

TASK DEFAULT DEPTH

DETAILED INSTRUCTIONS:

https://github.com/VCU-ERB/xfig-3.2.8a/issues/2

GOAL: This project's goal is to increment the depth of new objects when added to the drawing. When an object is added, it does not have depth properties. As a result, all elements are on the same Z-axis. It makes the drawing confusing. It is not clear which object was added first and last. You will need to change the properties of the objects in xFig to increment their depth when added. You must add "w_indpanel.h" to every file you edit. That is the file that has the depth value properties for every object.

STEPS:

1. Open Visual Studio Code or your choice of IDE (open it from GitHub).

2. Check if you have opened the whole xFig folder.

3. Click on the search icon on the left toolbar and search "#taskDefaultDepth". You should see eight files to edit.

4. Use the directions provided as comments to work on the assignment. Try to work in the alphabetical order you see in VS Code.

5. Go to the desktop and compile your code by double-clicking the 'compile.sh' file to check if your solution works.

If you are confused at any point with xFig or Linux environment, check the youtube video at https://www.youtube.com/watch?v=Csxk165L358.

------------ Instructions End Here -----------
```

Fig 1 shows this file (right) and the result from the search for the hashtag #taskDefaultDepth (left). To view all the relevant files the user can click on the file names on the left. If the user opened the first file ("d_arc.c") they would see the following code, also shown in Fig 2. In this simple task, they would have to adapt the code on line 302 in Figure 2. Notice how the rest of the code, above and below this line, is composed of comments that explain the task inline, guiding the student through the two main subtasks, which in this case are incrementing depth and refreshing the user interface. Advanced users can add boundary-checking code to ensure the validity of the input.

```
// #taskDefaultDepth

//------------ Code Starts Here --------------

/* INFO: XFig works with objects (NOT object-oriented programming, but objects to draw, like arcs, ellipses, and boxes). This code is inside the method to create a new arc. What is the method's name? What is the file's name? Is there something similar between the names of the files to edit?

 * GOAL: The current code doesn't increment the value when a new object is added. How would you modify the code so that the default depth increases by 1 anytime a new object is added?

 * HINT: use post-increment.

* CHALLENGE: Valid the boundaries. The depth cannot be more than 999.
*/

arc->depth = cur_depth;

/* INFO: After incrementing the current depth, the line of code above only updates the depth of the object internally (i.e., the model).

* GOAL: Call the 'show_depth' method and pass in 'depth_button' as the argument to update the toolbar at the bottom.
*/


// GOAL: Save your work and continue to the second file.

//-------------- Code ends Here --------------
```

B. *Implementation*

The project was tested during the class CMSC 355 Introduction to Software Engineering at Virginia Commonwealth University in Fall 2022. The classes had approximately 100 students in different majors and years (1 to 5). The majority of students are taking second or third years of Computer Science major. It is a 3 credits class with one hour of lecture three times a week (Monday, Wednesday, and Friday).

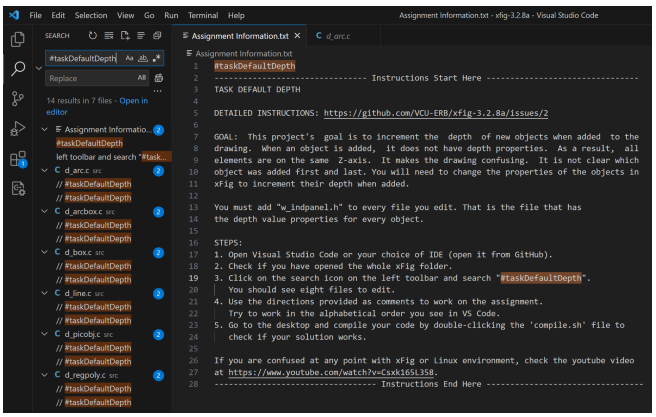

Fig. 1. Assignment Information file and hashtag searching.

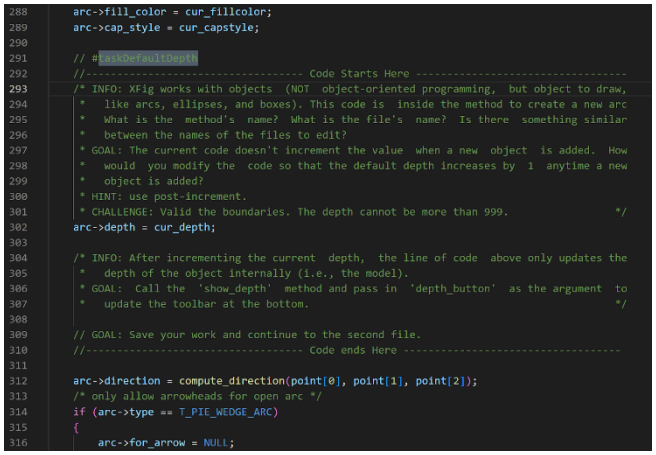

Fig. 2. Part of 'd_arc.c' file with instructions.

In Fall 2022, a detailed tutorial about configuring the VM and executing the tasks was presented to the students on the first day. The instructor released tasks on Fridays, using the one-hour class to answer questions about the previous task and to present the next one. The due date was the following Friday or Monday.

In the first four tasks, we slightly increased the difficulty but maintained the same level of scaffolding. In the first task (#taskEnhancedRotation) 2 files needed to be changed. In the second task (#taskDefaultDepth), students needed to change 6 files, changing the depth of an object when it was created. In the third task (#taskUndoShortcut), students needed to change 3 files, including adding a button to undo the last action. In the fourth task (#taskToggleUnit), students needed to change 5 files, including adding a button to change between imperial and metric units. In the fifth task (#taskSendBack), 9 files should be changed to create a button to change the depth of an object, sending it to the back or bringing it to the front. The students needed to create the button and the function as in the last task, but now they needed to create the functions in new files. In the sixth task (#taskQuickColoring), 9 files should be changed to create a button to change the objects' fill/border. In the seventh task (#taskFreeSelection), 6 files should be changed to create a button that selects many objects simultaneously.

C. Assessment

A combination of students' project completion data, pre/post Likert-type questionnaires, and open-response reflection questions were used to explore the project's impact on student competencies. For the first one, two key measures were considered for the evaluation: the percentage of students who successfully completed each task and the average grades they achieved. In addition to student performance data, we chose motivation as a key indicator of project success because of established links between academic success and student motivation, engagement, self-confidence, and self-efficacy [22, 23, 24]. 24 items from the Motivated Strategies for Learning Questionnaire (MSLQ) [25] were adapted for the pre/post questionnaire, supplemented with 9 self-efficacy items adapted from previous research by the authors [26] and 11 items added to the post-survey to gauge student satisfaction with the project. The reliability of the survey, as measured by Cronbach's alpha, ranged from 0.79 to 0.94 for each of the three subscales, all well above the generally accepted cutoff of 0.70 [27, 28]. Two free-response questions were also included in the post-survey.

Questionnaires were administered using the Qualtrics platform, before and after the project (beginning/end of semester). Data was downloaded into Excel for analysis. Likert-type responses were converted to a numerical rating scale according to the preferred direction of response, ranging from 1 (least preferred) to 5 (most preferred), and were used to calculate average mean values for each item and each subscale.

The open-ended questions aimed to provide a better perspective of how the students saw the project and its value to their futures. The questions are:

- QO1: From your perspective as a future member of the workforce, do you feel this project/assignment has helped prepare you for a job in your chosen field?
- QO2: Do you have any other comments or feedback to share about the course or the project?

To explore the impact of our scaffolded project (HITCH) on student outcomes, we analyzed the responses from the open-ended questions using Reflexive Thematic Analysis, as described by Braun & Clarke [29], in an inductive and semantic way, where the coding and theme development reflected and were directed by the content of the data.

Responses were first grouped by question in a spreadsheet. An open coding process was used, where every answer was analyzed, and one or more codes were attributed when applicable. New codes were created whenever necessary, as determined by the answer's content. Several of the most interesting themes are presented in the results section, with representative quotes.

IV. Results

Results are described from the class point of view, considering the project completion and grades, followed by quantitative and qualitative survey results.

A. Grades and Project Completion

Table 1 describes the number and percent of students that finalized each task, and the average grades, based on the initial enrollment of 102 students in the Fall of 2022.

TABLE I. Completion and Grades for Each Task

Task	*Students*	*Percent %*	*AVG Grades*
1	89	87.3	99
2	90	88.2	99
3	86	84.3	93
4	78	76.5	92
5	64	62.7	77
6	57	55.9	79
7	55	53.9	94

In the first three tasks, the results were as expected, with most students successfully using the scaffolding to complete the tasks on a realistic project. However, in the fourth task, the number of students who finalized the task decreased

significantly. We believe that several factors contributed to this, which we discuss below. We presumed that students had a working knowledge of C language before starting the class. During tasks 4–6 they needed to define and use functions, making these functions visible to other files, but some students did not have experience with functions. Nevertheless, the students that finished task 4 got good grades, indicating that if they were familiar with functions this task was not too difficult.

In the fifth and sixth tasks, we had a decrease in completion percentage and average grade. Upon reflection, we believe that the main difficulty students faced during these tasks that were not present in previous tasks was the required changes in build files (i.e., make files). The students had problems compiling the program since the errors in the make file are not as intuitive as other types of errors, and thus are harder to correct. The scaffolding we provided for these tasks was insufficient for all students to understand how to proceed. The functions to change and implement also were not simple. A possible solution for this problem is to divide the fifth task into two: include new files in the first week, then create the functions to solve the problem in the next. Including the recompilation in a more extensive task was a step too big for students to be confident doing it.

Most students who had given up by task 6 did not return to complete the seventh task; those who continued performed well.

B. Quantitative Analyses

In all we collected 77 pre and 73 post-surveys in fall 2022. Fig 3 presents average student responses to each of the three instrument subscales (self-efficacy, motivation, and post-only project satisfaction) for these three sample groups. Among the data set we were able to match pre and post-surveys for 42 students.

Student outcomes were mixed on the quantitative survey items. Average responses to the self-efficacy items rose slightly from an average mean of 4.07 (pre) to 4.12 (post), with the percentage of students who somewhat agree or strongly agree to questions in this subscale remaining high at 80%. Student responses to individual questions were mostly unchanged, but increased significantly to two questions about their ability to debug, compile, and run software programs (avg response 3.78 pre, 4.01 post, p=0.04). There was a significant decline in the average mean response to the motivation subscale, from 3.89 pre to 3.62 post (p=0.001), with the % who strongly or somewhat agree with items in this subscale decreasing from 71% to 65%. Despite a general decrease in intrinsic and extrinsic motivation and willingness to persist with complex tasks, there were slight increases in student responses to items related to students' belief that the class material challenged them to learn new things, their desire to get a good grade, and their confidence in understanding the most complex topics.

Overall, most students were satisfied with the project, with an average mean response of 3.31 and 56% strongly or somewhat agreeing with items in this subscale. However, the portion of students who disagreed or strongly disagreed with this group of questions was the highest among all three subscales, at 23%. This indicates that there were some issues with the project still to be addressed in the next iteration.

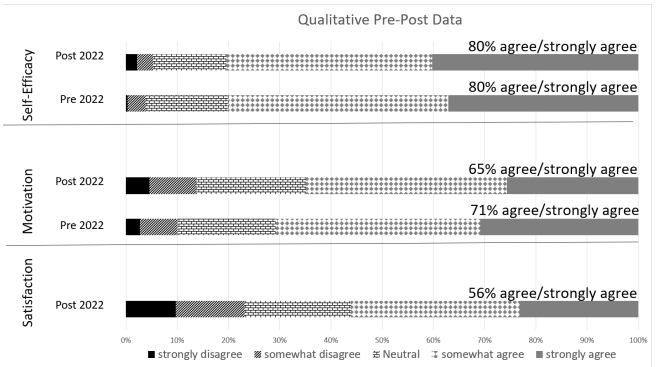

Fig. 3. Pre-post survey responses to three survey subscales

C. Qualitative Analyses

QO1 sought to identify whether students felt the project was of value to their futures and QO2 gave students the opportunity to share more about their general perception of the project. Considering students with a major or minor in Computer Science, 57 responded to the questions. For QO1, 36 responses were positive, 14 were negative, and 7 were inconclusive.

1) Theme: Working with legacy code and real-life experience

One theme that emerged was that the assignments helped students understand and work with legacy code. One student said:

"*I think the Xfig assignment did help prepare me for a job because I got to experience what it was like working on a complex project using legacy code. This assignment exposed me to important realizations that will come when I will enter the workforce as a software engineer. These include compiling and building complex projects, managing a lot of files, and reading documentation and other code in the project.*"

Others spoke of how novel it was to work on existing code, as exemplified by the following three responses:

"*Yes it was interesting working on long pre existing code. This is the first time I haven't had to do something from scratch.*"

"I feel that I have learned some useful skills, habits, and principles that are important when working on existing software."

"I think the XFig project really gave me a taste about how to actually work with code and integrate into pre-existing code."

Students seemed to understand how the project would be relevant to what they would be asked to do as programmers. A student said that *"It gave me real world experience"*. And others said:

"I feel it did. The assignment involved modifying existing programs by creating new code, something that happens often when programming."

"I was able to work on a real word application and see a large library of code and learn to navigate through it."

2) Theme: Challenges with Scaffolding

Not all students perceived scaffolding as helpful. For example, one student saw the cumulative tasks as repetitive, mentioning that *"They seemed kind of redundant."* Another student felt frustrated having to complete tasks without the full knowledge of the code and functionalities:

"I think it somewhat helped, as a lot of software engineering jobs require employees to work on the code of others with little explanation. However, it was a little frustrating to work on since I didn't really have a solid understanding of the code. Rather, I was just following the instructions without knowing what I was writing actually did."

One student was able to identify the changes in the scaffolding level, but he saw it in a positive way, as an opportunity to improve his problem-solving skills:

"I think the instructions were very straightforward during some of the first few assignments. The later assignments were more challenging and really taught me to spend time and understand the surrounding code before actually writing my own code."

It was also possible to identify that the scaffolding was helpful in guiding students to focus on important concepts and disregard concepts that were not needed. For example:

"I believe it was useful to learn how to learn a new language well enough to complete a task. Certain tasks may require small amounts of code in different languages so it is useful to be able to write this code without knowing the language fully."

3) Theme: The project increased confidence and motivation

The majority of students seem to have a good feeling from the project activities, reflecting the increase in confidence and motivation. One student mentioned that *"it was engaging"*. And a second student said:

"*I feel that this project has helped prepare me for a job in software engineering as it worked with different things I had not seen before. I feel more confident now if I were to encounter similar problems and issues in the real world.*"

4) Theme: The instructions were enough to complete the tasks

A frequent theme in QO2 was that the instructions were not enough to complete tasks. A student even mentioned that:

"*The instructions were terrible for the most part, but I understand that most of the time there will not be instructions in the real world.*"

In an opposite way, another one said that:

"I think one potential shortcoming was that in order to teach effectively, clear instructions were given. This is important for materials like this and should be kept, but engineers in the workforce may not always have access to such clear instructions or documentation, so I think there could be consideration for teaching students how to interpret what code is doing when documentation is poor or unclear."

Either way, the students seemed to understand the point that in the "real world", the instructions will not always be clearly organized and sufficient to complete the task without additional help or research.

5) Theme: The project was felt as out-of-scope

Another negative consideration was that the project was out-of-scope or not directly related to the class, being more complex than it was expected, especially after the third task. One student mentioned:

"I found that most of the material exceeding the first 3 Coding assignments was out of the scope of this course and overall went a bit too far past the amount of education in programming that most students who have never programmed before university contain."

6) Theme: The environment had many issues

The two main reasons for complaints were the issues with the environment and the compiling activities present in tasks 5 and 6. The students mentioned that AWS was slow and the GitHub instructions were not enough to manage the different versions of the code. A student commented:

"Just fix the compile file and instance for next semesters. The only reason I lost points was because of bad setup. It could be better if xfig run on students computers without having to create an instance on Amazon Web Services. Or without having specific compile file for it. A bit more organization would be much appreciated."

7) Theme: The project was a valuable experience

Overall, the students presented issues with the environment, but recognized the project's value. One student summarized the experience with:

"Xfig is old software that wasn't fun to navigate, but it was valuable experience."

Another said:

"I think the Xfig assignment did help prepare me for a job because I got to experience what it was like working on a complex project using legacy code. This assignment exposed me to important realizations that will come when I will enter the workforce as a software engineer. These include compiling and building complex projects, managing a lot of files, and reading documentation and other code in the project."

V. Discussion

From the grades and project completion rates we can see that the HITCH approach works well, but only when sufficient scaffolding is provided. Students who completed the first three tasks accomplished all goals with minimal mistakes. The decrease in completion rates and grades for tasks four to six suggest that the students did not have adequate support to succeed, and that the lack of scaffolding had a concrete impact on retention and grades. When reviewing these tasks, and possible reasons that students felt unsupported, it was easy to identify the gap. In these tasks, students were asked to update complex build configuration files, concepts that are often not taught as part of a formal computer science education but are necessary to learn to become a working software developer. While these skills would be good to learn prior to graduation, they are not necessary to achieve the goals of this particular project, and

thus adding additional scaffolding around build configuration would likely solve this issue.

For instance, on the seventh task, which was challenging and complex but did not require major build configuration changes, the students that completed this task performed well. However, many students had likely given up on the project tasks by this point, which led to a low completion rate, as they were frustrated by the challenges with the earlier tasks.

The quantitative results reflect these findings. The students completed the post-survey at the end of the semester, a moment when they are tired and stressed with final exams and projects. Nevertheless, the overall levels of motivation and self-efficacy were fairly positive, averaging between 3 and 4 on a 5-point Likert scale. The satisfaction with the project had less positive responses, but responses were still above neutral, confirming the good results from the scaffolding method and the issues.

Analysis of the open-ended questions shows that, although the students had issues compiling the program and using the AWS server, they appreciated the helpful project scaffolding, and importantly, they recognized the project's value in terms of preparing them for their future jobs. Students indicated that the experience improved their confidence to enter the workforce.

Alternatives for making the project more successful would be to apply the project in a more advanced class or define in the class specifications or pre-requisites that knowledge in C language is required. If the same configuration is used, it would be important for the professor to include more explanations about C language and complex programming concepts in the class.

VI. Conclusion

Completing a task inside a bigger program can be overwhelming for inexperienced students. Nonetheless, when the students have sufficient information to move inside and between codes and files, their confidence grows. This paper presents a mixed-methods study investigating how students in a computing course respond to working with scaffolding on a large scale. The course project allows students to work on a complex project through a series of well-supported and scaffolded tasks. These tasks provide instructions on navigating between files, making changes to code, and testing results. This approach offers students a real-world experience in a safe environment while enhancing their problem-solving and code-reading skills.

The project was tested with around 100 students, and the results showed positive outcomes. Most students completed the tasks successfully, however, some challenges were identified, such as difficulty with build configuration files and the AWS environment. Quantitative survey data indicated good levels of self-efficacy and motivation. Qualitative feedback from students highlighted the benefits of working with legacy code and gaining real-life experience.

The scaffolding approach HITCH was able to help students in many cases, showing that providing effective scaffolding can impact students' performance. Despite some issues, the project was perceived as valuable preparation for future careers in software engineering. In future applications of this project, we will ensure that all tasks provide the same level of scaffolding, ensuring a consistent experience across the semester. Future works will include studies with different projects and classes, and a refinement of the evaluation criteria to incorporate more robust statistical analyses.

Acknowledgment

This work is partially supported by the U.S. National Science Foundation Awards DUE-2111318 and DUE-2111294.

References

[1] J. Van de Pol, M. Volman, and J. Beishuizen, "Scaffolding in teacher–student interaction: A decade of research," Educational psychology review, vol. 22, pp. 271–296, 2010.

[2] K. E. Hogan and M. E. Pressley, Scaffolding student learning: Instructional approaches and issues. Brookline Books, 1997.

[3] B. Rosenshine and C. Meister, "Reciprocal teaching: A review of the research," Review of educational research, vol. 64, no. 4, pp. 479–530, 1994.

[4] S. Puntambekar, "Distributed scaffolding: scaffolding students in classroom environments," Educational Psychology Review, vol. 34, no. 1, pp. 451–472, 2022.

[5] XFig User Manual. "Introduction" SourceForge. [Online]. Available: https://mcj.sourceforge.net/. [Accessed: July 31, 2023].

[6] C. Stone, "What is missing in the metaphor of scaffolding. Contexts for learning: Sociocultural dynamics in children's development", pp.169-183, 1993.

[7] S. Lin, N. Meng, D. Kafura, and W. Li, "PDL: scaffolding problem solving in programming courses," in Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, 2021, pp. 185–191.

[8] B. R. Belland, A. E. Walker, N. J. Kim, and M. Lefler, "Synthesizing results from empirical research on computer-based scaffolding in STEM education: A meta-analysis," Review of Educational Research, vol. 87, no. 2, pp. 309–344, 2017.

[9] S. Reges and M. Stepp, Building Java Programs. Pearson, 2014.

[10] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," AcM SIGcSE Bulletin, vol. 39, no. 2, pp. 32–36, 2007.

[11] C. T. Fosnot, Constructivism: Theory, perspectives, and practice. Teachers College Press, 2013.

[12] L. P. Steffe and J. E. Gale, Constructivism in education. Psychology Press, 1995.

[13] P. C. Blumenfeld, E. Soloway, R.W. Marx, J.S. Krajcik, M. Guzdial, and A. Palincsar, "Motivating project-based learning: Sustaining the doing, supporting the learning," Educational Psychologist, vol. 26, no. 3-4, pp. 369-398, 1991.

[14] M.N. Ismail, N.A. Ngah, and I.N. Umar, "Instructional strategy in the teaching of computer programming: a need assessment analyses," TOJET: The Turkish Online Journal of Educational Technology, vol. 9, no. 2, 2010.

[15] R. E. Mayer, "The psychology of how novices learn computer programming," ACM Computing Surveys (CSUR), vol. 13, no. 1, pp. 121-141, 1981.

[16] J. Dalbey and M.C. Linn, "The demands and requirements of computer programming: A literature review," Journal of Educational Computing Research, vol. 1, no. 3, pp. 253-274, 1985.

[17] D.W. Shaffer and M. Resnick, ""Thick" authenticity: New media and authentic learning," Journal of Interactive Learning Research, vol. 10, no. 2, pp. 195-216, 1999.

[18] C.E. Hmelo and M. Guzdial, "Of black and glass boxes: Scaffolding for doing and learning," 1996.

[19] S. Freeman, S.L. Eddy, M. McDonough, M.K. Smith, N. Okoroafor, H. Jordt, and M.P. Wenderoth, "Active learning increases student performance in science, engineering, and mathematics," Proceedings of the National Academy of Sciences, vol. 111, no. 23, pp. 8410-8415, 2014.

[20] D. Wood, J.S. Bruner, and G. Ross, "The role of tutoring in problem-solving," Child Psychology & Psychiatry & Allied Disciplines, 1976.

[21] D. C. Shepherd, F. Fronchetti, Y. Liu, D. Hou, J. DeWaters, and M.M. Small, "Project-sized scaffolding for software engineering courses," in Proceedings of the First International Workshop on Designing and Running Project-Based Courses in Software Engineering Education, May 2022, pp. 27-31.

[22] E. A. Cudney and J.M. Ezzel, "Evaluating the Impact of Teach Methods on Student Motivations," Journal of STEM Education, vol. 18, no. 1, pp. 32-48, 2017.

[23] P. Pintrich and D. Schunk, "Motivation in Education: Theory, Research, and Application, 2nd Ed.," Englewood Cliffs, NY: Merrill, 2002.

[24] M.M. Chemers, H. Li-tze, and B.F. Garcia, "Academic self-efficacy and first-year college student performance and adjustment," Journal of Educational Psychology, vol. 93, pp. 55-64, 2001.

[25] P.R. Pintrich, "A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ)," 1991.

[26] S. Powers, J. DeWaters, M. Small, S. Grimberg, and D. Hou, "CLICS – Integrating Data from Campus Sustainability Projects across Disciplines," in Proceedings of the 122nd Annual ASEE Conference & Exposition, Seattle WA, 2015.

[27] J. Benson and F. Clark, "A guide for instrument development and validation," The American Journal of occupational therapy, vol. 36, no. 12, pp. 789–800, 1982.

[28] R. L. Linn and N.E. Gronlund, "Measurement and assessment in teaching (8th edition)," Englewood Cliffs, NJ: Prentice-Hall, 2000.

[29] V. Braun and V. Clarke, "Using thematic analysis in psychology," Qualitative Research in Psychology, vol. 3, no. 2, pp. 77–101, 2006.

[30] A. Garcia-Holgado, A. Vázquez-Ingelmo, F. J. García-Peñalvo, and M. J. Rodríguez Conde, "Improvement of learning outcomes in software engineering: active methodologies supported through the virtual campus," IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 16, no. 2, pp. 143-153, 2021.

[31] S. M. Souza and R. A. Bittencourt, "Sentiments and Performance in an Introductory Programming Course Based on PBL," in 2021 IEEE Global Engineering Education Conference (EDUCON), pp. 831-840, 2021.

[32] S. Arwatchananukul, P. Singpant, N. Chondamrongkul, and N. Aunsri, "Developing 21st century skills with project-based learning: an experience report in the introductory course of software engineering," in 2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON), pp. 451-455, 2022.

[33] C. Gupta, "The impact and measurement of today's learning technologies in teaching software engineering course using design-based learning and project-based learning," IEEE Transactions on Education, vol. 65, no. 4, pp. 703-712, 2022

[34] A. Ju, X. Fu, J. Zeitsoff, A. Hemani, Y. Dimitriadis, and A. Fox, "Scalable team-based software engineering education via automated systems," in 2018 Learning With MOOCS (LWMOOCS), Sept. 2018, pp. 144-146, IEEE.

[35] N. Piccinini and G. Scollo, "Cooperative project-based learning in a web-based software engineering course," Journal of Educational Technology & Society, vol. 9, no. 4, pp. 54-62, 2006.

[36] N. Wu, D. Hou, and Q. Liu, "Linking usage tutorials into API client code," in Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE '16), Association for Computing Machinery, New York, NY, USA, 2016, pp. 22-28.

[37] Y. Gao and D. Hou, "ArchFLoc: Locating and explaining architectural features in running web applications," in Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME) (ICSME '15), IEEE Computer Society, USA, 2015, pp. 333-335.