

ABSTRACT

MILLIKEN, ALEXANDRA ANN. Redesigning How Teachers Learn, Teach, and Assess Computing with Block-Based Languages in their Classroom. (Under the direction of Tiffany Barnes.)

This work applies to Computer Science (CS) K12 Education through a user experience and human-centered design and research approach. Due to the increase in use of technology, CS courses are important opportunities for K12 students to explore. In order to support the courses added to K12, in-service teachers lacking CS background are stepping up and tackling the CS field. To prepare in-service teachers to teach CS K12 courses, such as Advanced Placement Computer Science Principles (APCSP), curriculum providers host professional developments (PDs) for teachers to learn the content of the curriculum, pedagogies for teaching computing, and methods to support their students into technology. As novice CS teachers begin teaching their new field, many want continued support and community as they learn the new material. In this proposal, I present five studies exploring how to assist in-service teachers with teaching CS and computing in their classrooms when using block-based languages. For my work, I first learned about teacher needs by redesigning teacher professional developments. This motivated me to develop new ways to support teachers integrating computing and teaching computing in K12.

In the Teacher Professional Development experience report, I start by assisting teachers attending an Advanced Placement CS Principles (APCSP) Beauty and Joy of Computing (BJC), a curricula for the AP CSP course, Professional Development (PD) by redesigning the PD to more align with typical PD standards. In the Initial Design of Gradesnap user experience sprint study, I interviewed block-based language (BBL) graders to develop an understanding of what elements to have in a BBL grading tool as well as the flow of the tool. In the Initial Development Prototype and usability study, I discuss the phases of development and what elements have been tested thus far to ensure the usability and usefulness of Gradesnap. In the Grading BJC Programming Labs study, I investigate teachers individually, comparing their created rubrics and their grading habits when grading with without a rubric and with a rubric they have created. Additionally I discover common pain points for teachers when grading BBL projects in their current grading environment. In the Grading with Gradesnap study, I compare how the same teachers' grading habits are impacted by Gradesnap, as well as how Gradesnap addresses the teachers' pain points or what further needs to be done to ease the tedious task of grading BBLs. The contributions of this work furthers research into effective K12 computing teacher professional development designs and pedagogical models, understanding of K12 computing and non-computing teachers' grading methods of BBLs, and a new tool for teachers to grade BBLs. Additionally, this research will contribute a new user experience research method for designing human-centered applications with a small research team.

© Copyright 2021 by Alexandra Ann Milliken

All Rights Reserved

Redesigning How Teachers Learn, Teach, and Assess Computing
with Block-Based Languages in their Classroom

by
Alexandra Ann Milliken

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2021

APPROVED BY:

Thomas Price

Sarah Heckman

Tamecia Jones

Tiffany Barnes
Chair of Advisory Committee

DEDICATION

I dedicate my dissertation work to my supporting friends and family. I give special thanks to my parents, Bobbie and Rick Milliken, for always believing in me and being interested in my current endeavor ever since I was young.

I would also like to dedicate my work to Tyler Orr, my friend and partner, and Dr. Veronica Cateté, my friend and colleague, for always being by my side throughout my entire doctoral program, supporting me through my struggles and celebrating my triumphs. To our achievements in the future!

Finally, as my research specifically focused on helping teachers learn, teach and grade computing, I dedicate my dissertation to the future of computer science education; to the teachers who teach computing and to our learners who will someday impact our society further with technology.

BIOGRAPHY

I grew up in South Bend, Indiana and was an avid soccer player through high school. I went to college at Xavier University for my undergraduate career, from 2011-2015, where my advisor, Dr. Gary Lewandowski, introduced me to Computer Science. After learning about Computer Science, I wanted to teach everyone else about programming; I loved being able to make something and receive instant gratification - and I wanted to show others they could, too! I started with showing my family - during holidays, I would show my younger brother and little cousins programming games. On the weekends (and some weekdays) throughout my undergraduate career, I participated in outreach events where I guided a range of ages in learning to code; from elementary to high school. My desire for teaching others continued, I decided that I wanted to create a tool to teach students. Gary helped me find a Research Experience for Undergrads (REU) opportunity, which is where I met Dr. Tiffany Barnes.

While I knew I fit in the computing field, I was not sure what my next steps were until I participated in the REU with Dr. Barnes and her Game2Learn Lab. The summer of 2014 is when I fell in love with research focused on improving the user's experience in a serious educational game. I was lucky to have a wonderful experience. Early on, Dr. Barnes asked me to come to graduate school and join her lab; she was always straight to the point, especially if she was recruiting. I was surprised and excited. Throughout my entire doctoral program, she continued to have excitement for working with as many people as possible - because people are inspiring.

Dr. Barnes had a large lab, from which I loved and benefited. The more people in your working environment, the more collaboration can occur and the more projects we can work on, making the day-to-day change, and stay interesting and new. I continued my outreach interests by leading STARS events at the university for high school students and by leading summer camps for middle and high school students. I continued to find inspiration in teaching novices (or nearly novices) about the field of computing and all the ways to intertwine it with their other interests, such as sports or arts or novels. Beyond helping students, I also found great pleasure in helping teachers, which turned into being my focus-point for my dissertation. During the doctoral program, I started rock climbing with many of the others who the program during the same year: Tyler, Brian, Christa, and Zach. Our tight friend group also played board games on the weekends with another close friend, Justin. I did not have very many other hobbies besides rock climbing and playing board games throughout the past six years - to be honest, I did not have much time, which I'm sure others in the program would smile, nod, and understand.

In my last year of the program, the pandemic drastically changed my day-to-day. No more in-class observations or in-person professional developments or camps. I still met frequently with others online (thank you Zoom), however, I was alone at home most days, so Tyler and I rescued Dakota, our dog. She has been by my side through the last 9 months of my doctorate! Dakota and Tyler kept me sane during my last months of my doctoral career.

ACKNOWLEDGEMENTS

I would like to first acknowledge my committee members, Dr. Sarah Heckman, Dr. Tamecia Jones, Dr. Thomas Price, and Dr. Tiffany Barnes, who provided insightful assistance when refining my proposed studies to find more meaningful and interesting impact, especially during a pandemic. Their advising and helpful feedback has been useful for research directions. I would like to especially thank Dr. Tiffany Barnes, my advisor, who has always believed in and supported me ever since I participated in her Research Experience for Undergraduates (REU) program in the summer of 2014.

I would like to acknowledge the Game2Learn lab and all of those who I have worked alongside throughout my doctoral program. I would like to specifically thank some of my past lab members. Thank you Dr. Drew Hicks for mentoring me during my REU and inspiring me to go to graduate school. Thank you Dr. Veronica Cateté for being my first friend in Raleigh, mentoring me in graduate school, continuing to mentor me as a research scientist, and being an excellent peer as we progress computer science education research. Thank you, to the recently titled, Dr. Yihuan Dong for always helping me debug and program features in GradeSnap (or other projects) I led throughout my doctoral program; sincerely, thank you. Thank you Dr. Christa Cody for happily providing statistics assistance and for being a great co-author and friend; you were an inspiration throughout our doctoral program. Thank you Dr. Nick Lytle for always giving me positive energy and pressure to continue writing; your passion and presence fueled me to work hard all the way to the end. And a last special thank you to Ally Limke, Bella Gransbury, and Hannah Chipman who were the three researchers who assisted me in facilitating long interviews and analyzing data for my last two studies, enabling me to make my tight Spring deadline. Finally, I would also like to thank the many wonderful undergraduate teams I have mentored who have worked on GradeSnap or other projects, including: Josh, Qua, Malik, Trevor, Crys, Emma, Cindy, Leo, Alicia, Charlie, Jeel, Savan, Smit, Soham, Kat, Justin, Neeloy, Nico, Sarah, and Kiran. This does not cover everyone who mentored me nor everyone I mentored, however, I am touched and thankful for all those I've been lucky enough to work with. Thank you Dr. Barnes for advising a lab with great people, including yourself.

I would also like to acknowledge all of the teacher participants who made it possible for me to learn how to address teachers' learning, teaching and grading of block-based computing programming projects. Without your patience and time, computer science education research could not effectively progress with meaningful results.

Lastly, I would like to thank and acknowledge Dr. Gary Lewandowski, my advisor from my undergrad, Xavier University. Gary was the first to show me Computer Science. He not only believed in me but he convinced me to believe in myself and that I am smart and capable in what I set my mind to. He did not find my learning difference to be challenging to address in class, but instead he found my questions interesting, giving me a safe space to explore computing and have thorough, intellectual conversations. He briefly showed me how to lead a computing outreach event and left me to my own devices, sparking my interest in helping those without access to computing receive the opportunities they deserve - just as Gary did for me. I look forward to our continued friendship.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	x
Chapter 1 Introduction	1
Chapter 2 Teacher Professional Development	4
2.1 Introduction	4
2.2 The BJC Professional Development program	5
2.3 Professional Development Design Rationale	5
2.4 Evolving Design of BJC Professional Development	6
2.4.1 2012 - 2015 BJC PD Design	6
2.4.2 Recent BJC PD Commonalities	7
2.4.3 2018 BJC PD	8
2.5 Methods	10
2.6 Results and Discussion	13
2.6.1 Teachers' Assessment of the PD	13
2.6.2 Teacher Perception on Teaching BJC	14
2.6.3 CSP Adoption Rate	15
2.7 Conclusion	16
Chapter 3 Literature Review	18
3.1 Research Context	18
3.2 Grading Programming Projects	19
3.2.1 General Program Grading Methods	19
3.2.2 Block-Based Language Programming Methods	22
3.3 Rubrics	24
3.4 User Experience Research Methods	25
3.4.1 Personas	25
3.4.2 Participatory Design	26
3.4.3 Sprint	26
3.4.4 Other UXR Methods	27
3.5 General Good Design Principles	27
3.6 Teacher Dashboard	28
3.6.1 Dashboard Designs	29
3.6.2 Dashboard Impacting Teacher Actions	30
Chapter 4 Initial GradeSnap Design	32
4.1 Initial user design - Phase 1	33
4.1.1 High-Level Overview of Adapted UX Stages	33
4.1.2 In-Depth UX Sprint Discussion	34
4.2 Final Prototype Design, Flow and Decisions based on Sprint	39
4.3 Conclusion	40
Chapter 5 Initial GradeSnap Prototype	43
5.1 StatSnap	44

5.2	Interactive Tutorial	45
5.3	First Usability Study	47
5.3.1	User Satisfaction	48
5.3.2	Perceived Difficulty	49
5.3.3	Likelihood of Adoption	49
5.3.4	Limitations	49
5.4	Conclusion	50
Chapter 6 Grading BJC Programming Labs		51
6.1	Related Literature	51
6.2	Participants and Pre-Study Actions	52
6.3	Study Design	52
6.3.1	Assignment Teachers are Grading	53
6.4	Data Collection and Analysis	53
6.4.1	Screen Recording and Audio Recording	54
6.4.2	Rubrics	54
6.4.3	Thematic Analysis Process	54
6.5	Results	56
6.5.1	Themes	56
6.5.2	Teacher Cases	56
6.6	Discussion	67
6.6.1	Comparing Teachers Grading	67
6.6.2	Rubric Creation	72
6.6.3	Themes	75
6.7	Conclusions and Contributions	79
Chapter 7 Grading with GradeSnap		81
7.1	Related Literature	81
7.2	Updated GradeSnap Features	82
7.3	Participants and Pre-Study Actions	83
7.4	Study Design	84
7.4.1	Assignment Teachers are Grading	85
7.5	Data Collection and Analysis	86
7.5.1	Screen Recording and Audio Recording	86
7.5.2	Rubrics	86
7.5.3	Thematic Analysis Process	87
7.6	Results	88
7.6.1	Themes	88
7.6.2	Teacher Cases	88
7.7	Discussion	93
7.7.1	Grades	93
7.7.2	Rubrics Changes	95
7.7.3	Comparing Teachers Grading Goals	95
7.7.4	Solving Teachers Pain Points	96
7.7.5	Themes	99
7.8	Conclusions and Contributions	103
Chapter 8 Conclusion		105

8.1 Contributions	105
8.2 Summary of Contributions	107
8.3 Concluding Thoughts	107
BIBLIOGRAPHY	109

LIST OF TABLES

Table 2.1	Time spent on varying activities in the BJC PD schedule for 2018.	11
Table 2.2	This table shows the teacher demographic for the 2016, 2017, and 2018 PD years.	11
Table 2.3	This table shows the average survey responses to questions intended to assess the PD.	12
Table 2.4	These teacher perception question were in the Post-PD survey comparing responses from 2016, 2017 and 2018.	14
Table 5.1	Satisfaction results comparing Tutorial Users and Non-Tutorial Users	49
Table 6.1	The study outline consisted of these six parts.	53
Table 6.2	Each researcher coded 2 different participant’s think aloud interview. They coded their first assigned interview (indicated with a 1), then the research met as a group, then they coded their second assigned interview (indicated with a 2).	55
Table 6.3	Alice’s grades for the submissions only had a change of over 2 percentage points for 3 submissions; Student 2, 3, and 4 between the two grading sessions. The submission level refers to the achievement of the submission.	58
Table 6.4	Beth’s rubric has three categories, each with 4 category items (or level of correctness for each category). Two of her categories (Abstraction and Functionality) had double the weight as the other category (Appearance & Naming), signified by a (x2).	60
Table 6.5	Beth’s grades for the submissions only had a change of over 5 percentage points for 2 submissions; Student 1and 4 between the two grading sessions. The submission level refers to the achievement of the submission.	61
Table 6.6	Christina’s rubric has 4 categories and 3 category items (levels of correctness). All of her categories have the same distribution of weight.	63
Table 6.7	Christina’s grades for the submissions only had a change of over 5 percentage points for 2 submissions; Student 1and 4 between the two grading sessions. The submission level refers to the achievement of the submission.	64
Table 6.8	Deborah’s translation of Letter Grades to Grade Percentages.	65
Table 6.9	Deborah’s rubric has 6 categories, one of which is a Bonus/Extra Credit category. Her categories have different distributions of weight and a different number of category items (levels of correctness).	66
Table 6.10	Deborah’s grades for the submissions changed for each submission. The submission level refers to the achievement of the submission.	66
Table 6.11	The participants’ grades for each submission for each grading session. Each participant row is denoted with the first letter of their name (Alice - A) and the grading sessions are denoted with the accompanying number (1 - first grading session, no rubric; 2 - second grading session, with rubric). The level row denotes the the achieving level of the submission. The avg column is the calculated average of grades given during that grading session for that teacher.	68
Table 6.12	Deborah’s grading in the first grading session D(1), the second grading session with her rubric (2a), and the second grading session removing one category that was not expected by the assignment (2b).	70

Table 6.13	The participants' prior grading and teaching experience with block based programming languages, their grading goals for both sessions, their rubric type and categories with the number of corresponding category items.	73
Table 6.14	A descriptive table comparing the rubrics of the four participants. For the Type of Rubric: S - Single Point, T - Task Specific, and A - Analytical.	74
Table 7.1	The study outline consisted of 5 parts.	85
Table 7.2	Each researcher coded 2 different participant's think aloud interview. They coded their first assigned interview (indicated with a 1), then the research met as a group, then they coded their second assigned interview (indicated with a 2).	87
Table 7.3	The participants' grades for each submission for the third grading session. Each participant row is denoted with their name. The levels row denotes the achieving level of the submission. The Avg. column is the calculated average of grades given during that grading session for that teacher. The Avg row (along the bottom) is the calculated average for grades for that student.	95
Table 7.4	Comparing grading goals from the first grading session (without a rubric), the second grading session (with a rubric), and the third grading session (with GradeSnap).	96
Table 7.5	Beth's feedback for the submissions from the first grading session and the third grading session.	98
Table 7.6	Christina's feedback for the submissions from the first grading session and the third grading session.	99
Table 7.7	Deborah's feedback for the submissions from the first grading session and the third grading session.	99

LIST OF FIGURES

Figure 2.1	BJC Curriculum Adoption Rate Comparison 2016, 2017 and 2018	16
Figure 4.1	Decided dashboard (landing page) elements	39
Figure 4.2	Gradebook where teachers can review students' grades	40
Figure 4.3	Decided flow based on interviewees and participants preferences	41
Figure 4.4	Grading interface based on Gradescope's grading interface	42
Figure 5.1	Grading interface where teachers can see their rubric and student's code simultaneously	44
Figure 5.2	Rubric creation interface.	45
Figure 5.3	Example of the pop-up interactive tutorial	46
Figure 5.4	Help dashboard where teachers can find assistance with common tasks within GradeSnap	46
Figure 5.5	Likelihood Adoption of GradeSnap	50
Figure 6.1	The final output of the Brick Wall assignment and an example of a defined custom block, provided in the instructions for the students.	54
Figure 7.1	The first page a teacher arrives at upon logging into GradeSnap.	83
Figure 7.2	The rubric page within GradeSnap.	84
Figure 7.3	The grading page within GradeSnap.	85
Figure 7.4	The final output of the Brick Wall assignment and an example of a defined custom block, provided in the instructions for the students.	86

CHAPTER

1

INTRODUCTION

- **RQ2a-GS design:** What features do teachers want and need in an online grading tool for programming projects?
- **RQ2b-GS prototype:** How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change with tutorial usage?
- **RQ2c-Grading:** How do teachers grade programming projects with and without rubrics, but without tool support?
- **RQ2d-GS evaluation:** How do teachers grade with and perceive GradeSnap grade support? How does teacher-use of the grading tools differ based on their expertise/experience or goals for student feedback?

Due to the increase in use of technology in every day lives, computer science (CS) courses are becoming a critical element of students' education. To accommodate, computing courses are being added or integrated into high school curricula creating a need for more in-service teachers, most of whom lack Computer Science backgrounds, to teach these courses [Coo17]. Many of the new high school courses, such as Advanced Placement Computer Science Principles (APCSP), use block-based languages, such as Snap! [Gar15]. The Beauty and Joy of Computing (BJC), a curriculum for the APCSP course, uses Snap! because it allows for the students and teachers to learn programming concepts without having to worry about complicated, traditional syntax [Gar15].

The BJC team has hosted teacherProfessional Development workshops (PD) since 2012 [Pri16], attended primarily by in-service teachers with no background in computer science and little programming experience [Pri16; Mil19]. BJC teachers have expressed that grading programming assign-

ments is challenging and tedious and would benefit from rubrics and other grading resources after leaving the PD [Cat16]. Based on a Fall 2019 Facebook poll, most AP CSP teachers use self-made rubrics that only give summative feedback to students and do not explicitly express the learning outcomes or misconceptions displayed in students' work. This may be due to the fact that these new curricula may not have clearly defined how learning goals for each programming activity will be observed in student code [Eri14; Cat16]. To assist high school teachers with grading block-based programming projects in the varying introductory computer science courses, a few research teams, such as Dr. Scratch, ITCH, and Lambda, have developed block-based autograders to try to solve this challenge [ML15b; Bal18; Joh16]. While this method of grading has identified coding patterns in students' projects, teachers may not understand the reasoning for the scoring and may continue to struggle giving formative and useful feedback to students. Autograders can also incorrectly mark correct solutions as incorrect if the students did not follow the same design as the autograder. This can be very frustrating for both teachers and students. For example, typically in these systems teachers are unable to override the grade the autograders assigns and therefore students may not receive the score they deserve. More importantly, many block-based programming activities have the goal of promoting student creativity but autograders with a single target solution may lead both students and teachers to incorrectly assume that there is a single correct answer to a programming exercise.

The goal of this dissertation is to investigate how to help in-service teachers, especially those with little prior computer science or programming experience, prepare to teach computer science. As previously stated, these in-service teachers are unfamiliar with computing [Eri14] and they need Professional Development (PD) to not only teach them pedagogy for teaching the class but to develop an understanding about CS and programming as well [Pri16]. As if learning CS concepts and programming is not difficult enough for in-service teachers from other disciplines, these teachers must also understand CS and programming well enough to teach it and monitor their students' progress. While teachers do learn some pedagogical techniques for teaching computing at their PD, more support is needed to build their understanding and to help them address students' learning outcomes to provide formative feedback on projects or exams [Eri14]. While block-based languages allow novice programmers to develop projects without the overwhelming, traditional syntax, administering these courses is not as simple. For example, in the AP CSP course using BJC, teachers must determine how they want to distribute assignments and collect finished projects to grade. Often the assignment instructions, the programming tool, and the teacher LMS operate independently, requiring teachers to complete tedious tasks to establish their grading environment. Supporting teachers with PDs and tools as they learn about computing enables them to support their students to enter into technology-related fields. I am researching solutions for these problems by developing a teacher tool, allowing them to manage their class and assignments, as well as grade their students' submitted work and give valuable feedback. The research questions to investigate how to help in-service teachers learn to teach computer science and programming are:

Guiding research question: How do we help in-service teachers teach programming?

RQ1: How do we help in-service teachers learn to teach programming with summer professional development?

RQ2: How do we design an effective tool to support teachers in grading programming projects?

Within RQ2, I leveraged user-experience research methods to investigate the following, more specific research questions:

- **RQ2a-GS design:** What features do teachers want and need in an online grading tool for programming projects?
- **RQ2b-GS prototype:** How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change if we offer a tutorial on the tool?
- **RQ2c-Grading:** How do teachers grade programming projects with and without rubrics, but without tool support?
- **RQ2d-GS evaluation:** How do teachers grade with and perceive GradeSnap grade support? How does teacher-use of the grading tools differ based on their expertise/experience or goals for student feedback?

In this dissertation, I explore several dimensions of supporting teachers using block-based programming environments in their classes. The dissertation is organized as follows: Chapter 2 describes how I designed teacher professional development workshops to help in-service teachers learn to teach the Beauty and Joy of Computing course. Based on this experience, I learned that teachers faced barriers in the amount of time and effort it took to collect student work and provide useful feedback to students. The remaining chapters focus on teacher grading and the design and study of GradeSnap, a software platform I designed to support teachers to grade block-based programming assignments using easily-created rubrics. Chapter 3 discusses the literature around existing grading strategies and tools for programming and block-based programs, teacher dashboard design, and teacher grading practices. Chapter 4 presents the design of GradeSnap, and Chapter 5 presents a pilot study of the system. Chapter 6 presents a think-aloud study of 4 teachers grading Snap! programs without the use of GradeSnap, first without rubrics, and then using rubrics, to investigate teacher needs and a baseline understanding of BJC teacher grading.

In Chapter 7, I present a study to evaluate (1) a teacher's effectiveness when using the tool to grade (measured in time and consistency) and (2) the usability of the tool with our target users, and (3) how the tool impacts the teacher's ability to synthesize the performance of a small group of students. Throughout this work, I have iteratively updated the system design to better align with teacher needs. My goal is that in-service teachers, and especially those who are new to computer science, will find GradeSnap both usable and useful for their grading and classroom needs for teaching BJC.

CHAPTER

2

TEACHER PROFESSIONAL DEVELOPMENT

2.1 Introduction

In the United States, Computer Science education is not widespread in K-12 schools, meaning that most teachers and schools are unequipped to handle computer science coursework and exams. In data collected on the “State of Computer Science Education” in 2018, only 35% percent of secondary schools in the US taught computer science and only 22 states had K-12 computer science standards[Cod18]. As of 2020, these numbers have increased with 47% of high schools offering computer science and 37 states adopting CS standards [Cod], demonstrating that states recognize the imperative need for students to learn CS. As of 2020, 40 states have computer science teacher certification (up from 34 in 2018) and most concerning, only 20 states have state-approved pre-service teacher preparation at institutions of higher education[Cod]. This means that the majority of in-service teachers stepping up to the role have little to no experience in programming or computer science but are willing and passionate to learn how to teach these new topics so we can address the need for Computer Science in K-12. With computing occupations making up 58% of all projected new jobs in Science, Technology, Engineering and Mathematics fields[LS18], preparing in-service teachers to not only handle teaching a topic very new to them, but also giving them tools and resources to learn with their students is critical.

The National Science Foundation and the College Board’s Advanced Placement (AP) program worked together with educators to develop an AP Computer Science Principles (CSP) course [Ast12]. This rigorous, college-credit, course familiarizes high school students with the main concepts of

computer science and explores how computing affects our world. By exploring computing innovations and impacts on society, students learn about how expansive computer science is as a field of study and work. The course requires students to create and write computational artifacts (e.g. programs) as a part of the exam. Since this course does not have a designated programming language, teachers have the flexibility to choose the programming language for their classroom [Boa17].

The Beauty and Joy of Computing (BJC) is a curriculum endorsed by the College Board for the AP CSP course and introduces teachers and students to 7 Big Ideas of computer science: creativity, abstraction, algorithms, programming, data, the internet, and global impacts, using 6 Computational Thinking Practices: connecting computing, programming, abstracting, analyzing problems and artifacts, communicating, and collaborating [Gar15]. BJC was developed to appeal to a wide range of students, with the goal of providing computing education opportunities to groups that have been underrepresented in computer science. In order to help prepare teachers to teach Computer Science in K-12 schools, the BJC team hosts professional development (PD) programs [Pri16; Mil19], as do other CSP providers [Mor14a; Cun14].

This chapter illustrates how I have applied design-based implementation research and “infra-structuring” to continuously refine the BJC professional development program to identify and meet teacher needs and to align with Penuel’s effective PD characteristics [Pen07]. The effectiveness of this approach is supported with the results of our teacher PD surveys for summers 2016-2018.

2.2 The BJC Professional Development program

From 2016-2020, as part of the national CS10K effort, I have been working with Dr. Tiffany Barnes to coordinate regional AP CSP PD workshops to help teachers prepare to teach the rigorous BJC high school computer science course. Since most in-service high school teachers have little programming experience or computer science knowledge, the PD typically has a primary focus on helping teachers learn the programming content and related CS concepts, along with the pedagogical content knowledge about how to teach it. The BJC PD workshops are designed to be led by current “Master” BJC teachers who (1) use the BJC curriculum, (2) have attended both a BJC PD, (3) completed the BJC facilitator PD and served as a BJC PD teaching assistant (TA), and (4) attended a facilitator PD and served as a BJC Master Teacher to lead a regional workshop. Each year, new (TA) and experienced (Master) facilitators attend the facilitator PD to prepare for the upcoming BJC PDs, and learn about new updates to the BJC curriculum and national AP requirements. Typically, one or two experienced BJC Master Teacher facilitators, accompanied with one or two new facilitator TAs, lead a BJC PD.

2.3 Professional Development Design Rationale

The BJC PD is developed through a design-based implementation research (DBIR) approach of iteratively designing, testing, and refining our model with frequent feedback loops and participatory

design [Pen11]. Each year, the BJC PD is reevaluated and updated to provide the most effective PD experience for teachers. The program is also iteratively revised to align with emerging effective PD practices. For example, as described below, from 2016-2018, we adapted the PD to specifically help teachers practice some of their professional responsibilities by integrating the Teacher Learner Observer model [Cod19; Goo14] and the Lead Learner mindset [Lau11] to our PD schedule. I also describe how the latest BJC schedule aligns with effective PD characteristics based on research [Pen07; Mor14a].

According to Penuel et al., effective professional development should be designed to (1) focus on content, (2) promote active learning, and (3) foster coherence [Pen07]. Two of five of Mobile CSP's recommended core features for CS PDs include focus on content knowledge and active learning [Ros17]. Focusing on content is critical, especially during the initial PD [Pen07], when teachers focus on developing their knowledge and comfort with the material. Promoting active learning involves hands-on activities during the PD, promoting teacher inquiry and discussion, which can lead to higher student achievement outcomes [Pen07]. Providing teachers time to explore and understand the curriculum by planning, enacting, and revising lessons during the PD, is a concrete example of a relevant hands-on activity. A core design feature of the "Exploring CS" teacher PD is the Teacher Learner Observer (TLO) practice, where teachers role play three different roles, practicing their teacher responsibilities, acting as students (learners), and reflecting on teaching and learning as observers [Goo14]. Fostering coherence during PD allows time for teachers to compare their responsibilities and their district's goals to the curriculum's goals; if the goals and responsibilities align, teachers are more likely to adopt or adapt the curriculum [Pen07].

2.4 Evolving Design of BJC Professional Development

In this section, I describe the original 2012-2015 CSP PD model and the changes made to the model from 2016-2018.

2.4.1 2012 - 2015 BJC PD Design

From 2012-2015, we held two, 6-week PD workshops each year preparing a total of 133 high school teachers. During the 6-week PDs, teachers spent one week face-to-face, four weeks completing the BJC course online, and a final face-to-face week to prepare to teach. Price, et al. found that the 2012-2015 BJC PDs improved participants' confidence and listed several lessons learned over the years [Pri16]. They determined that one of the important aspects of high-quality PD is to address participants' needs and to adjust the PD for the participants [Pri16]. For example, in 2012 and 2013, the six-week long BJC PD was taught by university faculty, but feedback quickly suggested that teachers would learn better from other teachers. Therefore, in 2014 and 2015, the PD was taught by experienced teachers who had taken the BJC PD and taught the course. Another important research finding on BJC PD was the need to model teaching and explicitly discuss pedagogy in the PD [Pri16]. Since few participants had a computer science background, these in-service teachers needed to learn

both content and pedagogy during the PD. The final lesson learned was that teachers reported that experience with pair programming and team assignments helped build a community for support throughout the academic year [Pri16], and this finding led the team to keep pair programming as an important method for learning during the PD workshops.

2.4.2 Recent BJC PD Commonalities

In 2016, the PD was redesigned to allow more teachers to participate, transformed into a 3-week model, with just one 5-day week in person. In the next two sections I discuss the 2016, 2017, and 2018 BJC PDs focusing on the major design changes and the reasons behind the changes. The common elements in the face-to-face BJC PD schedule since 2016 include introductions to: (1) the BJC curriculum, (2) programming labs, (3) the programming language/environment, (4) growth mindset, (5) pair programming, (6) computing in the news, (7) course readings, and (8) the AP CSP Explore and Create Performance Tasks. Added in 2016, teachers also complete a Create Task, where teachers create their own programs, demonstrate them to others, and answer the AP Create Task prompts based on their programs.

During the PD, teacher participants take a pre-PD survey, revealing their expectations and needs, and a post-PD survey, revealing their satisfaction with the PD and their expected use of the curriculum. We used online forums as a discussion platform to enable the teachers to connect before, during, and after the PD.

A central design component aligned with design-based implementation research is the use of Daily Feedback surveys at the end of each day, to which facilitators (Master Teachers and TAs) respond the following morning. Each day of the 5-day PD week, teachers completed short Daily Feedback surveys asking 3 questions: (1) "What went well today," (2) "What can be improved," and (3) "Any other comments," which the facilitators and central BJC team review in an afternoon debriefing and planning session. The next morning, the PD facilitators celebrate accomplishments, answer questions, and explain how the program has been revised based on teacher participant Daily Feedback. These revisions help explain some of the changes in the PD schedule over the years. Table 2.1 compares the amount of time spent on pedagogy, programming, and reflection during the BJC PD in-person weeks from 2016 to 2018.

2.4.2.1 2016 PD

During the 2016 PD, 104 teachers participated in 12 regional BJC PD workshops. In 2016, we expanded to regional PDs to make it more convenient for teachers to attend. We also shortened the length of our PD experience from the 6-week model to a 3-week model because many teachers previously expressed that they could not commit to a 6-week PD, with many struggling to complete the online course on their own without pay. In the 2016 three-week model, the first week comprised preparatory online assignments, the second week was face-to-face, and the third week was online, comprising two assignments and office hours.

2.4.2.2 2017 PD

During the 2017 PD, 141 teachers participated in 7 regional BJC PD workshops. We used a 2-week model, continuing to shorten the PD to make it more feasible for more teachers. In the 2-week model, the first week comprised a smaller set of online preparatory assignments and the second week was the face-to-face week. We eliminated the third week by incorporating the 3rd-week assignments into the face-to-face week, and instead focused on engaging teachers in academic year online mentoring. We also lessened the amount of preparatory work, as noted in Table 2.1 since many teachers were unable to complete the preparatory work on their own due to time constraints and/or lack of prior knowledge.

We adjusted the 2017 BJC PD components which correlated with three of Penuel's recommendations to focus on **content**, to incorporate **active learning** into the PD, and to encourage **coherence** [Pen07], which is a complex concept related to alignment between the curriculum, the PD, and what and how teachers implement in their classrooms. With regard to content, the BJC PD was improved by implementing the Lead Learner mindset [Lau11] to help teachers reflect on how students learn the BJC material. In this mindset, teachers are encouraged to think of themselves as lead learners, helping students by showing them what an exemplary learner of CS content will do when facing problems [Tho18]. In 2016, we integrated more active learning into the PD in multiple ways, having teachers review student work and practice teaching labs and leading discussions. To review student work, the teachers spend an hour reviewing and discussing past student submissions for the AP CSP Explore Performance Task. Teachers, assisted by BJC leaders, also lead short discussions about "Computing in the News". During the Computing in the News discussions, one teacher presents a current news article about new or interesting topics in computing and facilitates a group discussion on its social implications or how the topic affects them.

In 2016, the BJC PD coherence was improved by adding more time for networking during a teacher panel with experienced BJC teachers and from recent AP readers for the AP CSP course. These discussions helped teachers better understand the AP requirements and how the BJC curriculum supports those requirements.

After the PD, teachers were encouraged to join online small groups facilitated by PD facilitators, and are added to our online support forum monitored by the BJC team.

2.4.3 2018 BJC PD

When designing the 2018 PD, we kept our 2017 PD additions and incorporated more of Penuel's recommendations to continue increasing the efficacy of the BJC PD [Pen07]. We kept the the Lead Learner mindset to help teachers think of themselves as models for their students, enabling them to learn alongside the students and demonstrate effective learning strategies [Lau11]. We increased the number of active learning activities by incorporating an adapted version of the Teacher Learner Observer (TLO) practice [Goo14] [Cod19], giving teachers the chance to practice teaching, added

more reflection and discussion time, having teachers complete the written components of the Create Task earlier, and adding a grading component to the PD.

To implement the TLO practice, we divided the teachers into three groups who would prepare and lead lessons (teachers) on three BJC programming labs, while other teachers would act as students (learners), and the facilitators would lead a discussion, prompting all participants to reflect on their observations (observers) of what worked from the teacher, student, and classroom perspectives [Cod19; Goo14]. This practice integrates many of Penuel's suggestions for incorporating more active learning into the PD and helps align our PD content to the teacher's professional responsibilities, which Penuel suggests increases adoption rates [Pen07]. We also increased coherence by adding 15 minutes of reflection and discussion time to the end of each teaching session of 60-90 minutes, where teachers take a few minutes to reflect on the activity, and then discuss what went well, what could have been improved, and how they would implement the content in their own classrooms. The remainder of this section will break down each day of the 2018 BJC PD.

Before attending the PD face-to-face week, participants were asked to complete a few short, well-specified lab exercises from the first unit of the BJC curriculum as preparatory work, designed to take 60-90 minutes total. We dramatically reduced the amount of preparatory work in hopes that more participants would complete the work. Participants in prior years reported being dissatisfied with having to repeat preparatory content during the face-to-face week since other teachers did not complete it beforehand.

Day One. On the first morning, we introduce the programming environment, the BJC curriculum, and resources we will use during the remainder of the week. To set the tone for the week, we explain and discuss growth mindset. People with a "growth mindset believe that they can develop their intelligence over time" suggesting people's minds are not fixed [Hoc15]. This is an important framing for the BJC PD, as many teachers in past years have reported feeling unprepared and unable to learn computer science and/or programming. We briefly review the pre-PD work and then complete a "lab", consisting of a coding activity followed by a reflection. After lunch, we complete another lab, discuss some AP materials for the course, and review the expectations and schedule for the remainder of the week.

During each lab, the PD leader introduces the goal of the coding activity and any new blocks or computer science concepts. Then teachers complete the lab in pairs to introduce them to pair programming. After completing the coding activity, the teachers participate in a reflection, discussing what they learned and sharing how they would teach the lab in their own classrooms.

Day Two. On the second morning, one teacher presents a recent article for Computing in the News. Then we complete a lab in pairs. After the reflection, we divide participants into 3 groups to learn and plan to teach 3 labs for the TLO activity. In the afternoon, one of the three groups teaches their lab and the PD facilitators lead a TLO reflection discussion. We close the day by exploring classroom environments and social impacts, discuss the challenges with teaching and recruiting a diverse classroom, and introduce the AP Explore Task with past students' projects as demonstrations.

A core goal of BJC is to be appealing and achievable for students from diverse groups, includ-

ing race/ethnicity, socioeconomic backgrounds, and gender, and prior CS/programming experience. Since computer science is not a required course, but voluntary, students from groups that are traditionally underrepresented in computing must be invited and encouraged to participate. We encourage teachers to recruit students from all kinds of groups to join the AP CSP class by introducing growth mindset on day 1 and by discussing recruitment and diversity in the PD on day 2.

Day Three. On the third morning, the other two TLO groups teach their programming labs for the TLO activity, each concluding with a facilitated discussion. As a whole class, we discuss the benefits of the TLO practice and teachers point out any pedagogical techniques they plan to bring back to their classroom. In the afternoon, we complete a lab focused on the CS concept of *lists* and reflect on the activity. Then, teachers complete another lab focused on the CS concepts of *algorithms* and *abstractions*, afterwards reflecting on the activity and how it completes all of the requirements for the AP exam's Create Task. To end the day, we introduce the teachers to the Create Task, which will be their project for next day.

The CSP Create Task is a required component for the AP CSP exam. Teachers choose a partner and decide on a project which would complete all of the requirements for the AP Create Task. The teachers work on their Create Task all of day four and part of day five. After teachers have finished programming their projects, they create a short video demo of their program, and write a report to answer the Create Task questions. This means that, by the end of the PD, all teachers will have completed the required components for the open-ended Create Task that their students must complete for the AP exam.

Day Four. On the fourth morning, we practice a Computing in the News discussion. Then, we present the AP Create Task requirements and demonstrate student work samples. The teachers spend the majority of the day developing and programming their own Create Task with their partner. At the end of the day, a panel of experienced BJC teachers answer participant questions about BJC, teaching BJC, the AP exams, or other topics.

Day Five. The last day of the 5-day synchronous PD week contains overviews of the remaining BJC units we did not cover in depth during the PD week. In the morning, we introduce teacher-created BJC materials, and share AP Reader teacher insights and tips. Teachers finish their Create Task and practice grading another team's work using the AP Create Task rubric. After lunch, we reflect on the PD week, answer any remaining questions, discuss how teachers sign up to teach an AP course, and the teachers take the Post-PD survey, telling us how our BJC PD prepared them to teach the AP CSP course.

2.5 Methods

In order to evaluate and compare the effectiveness of the PD from 2016 to 2018, we collected data on (1) participant satisfaction with the PD, (2) participant's perception on their preparedness to teach the AP CSP course, and (3) the number of teachers who plan to adopt the BJC curriculum. We also

Table 2.1 Time spent on varying activities in the BJC PD schedule for 2018.

Time in hours spent on..	2016	2017	2018
<i>Pedagogy</i>	1.75	7.00	9.75
<i>Programming</i>	14.50	12.75	11.75
<i>Reflection</i>	3.50	3.75	4.50

collected supporting demographic information in the PD application to understand their contexts and prior CS/programming experience. The number of teachers who participated in research for the 2016, 2017 and 2018 PDs were 36, 72 and 67, respectively.

Demographic information collected on the PD application includes questions about teacher gender, school type (public/private/ charter), school demographics by gender, race, and free/reduced lunch, and teacher experience level in programming (None, A Little, Moderate, and A Lot) as noted in Table 2.2.

Table 2.2 This table shows the teacher demographic for the 2016, 2017, and 2018 PD years.

Year	N	Coding Experience	Gender	School Type
2016	34	None 12%	56% F 43% M	91% Public 2% Private 5% Other
		A Little 53%		
		Moderate 32%		
		A Lot 6%		
2017	54	None 15%	65% F 35% M	83% Public 14% Private 1% Other
		A Little 44%		
		Moderate 35%		
		A Lot 6%		
2018	67	None 7%	60% F 40% M	73% Public 17% Private 8% Other
		A Little 43%		
		Moderate 36%		
		A Lot 15%		

On the post-PD survey, participants answered 18 five-point Likert scale questions to rate their assessment of the utility, efficiency, and organization of the PD, as well as the appropriateness of the teaching environment provided. The questions, shown in Table 2.3, are asked to assess how well the PD met participant needs. Teachers read the questions, such as “I can use this training to positively impact the achievement of my students” and respond with one of the five following: Strongly Disagree, Disagree, Neither Disagree or Agree, Agree, or Strongly Agree.

Furthermore, the post-PD survey asked participants about their planned BJC curriculum adoption, and 37 questions rating their self-perceived preparedness in the following areas: **Content** (level of fluency on CS concepts), **Inquiry/Engagement** (level of fluency when fostering varying

Table 2.3 This table shows the average survey responses to questions intended to assess the PD.

Assessing the PD Questions	2016	2017	2018
	Avg	Avg	Avg
N=	34	54	67
1. I can use this training to positively impact the achievement of my students	4.35	4.64	4.49
2. The content of the pd is relevant to my professional responsibilities	4.35	4.45	4.51
3. The facilitators helped me understand how to implement my learning	4.38	4.48	4.49
4. This pd will extend my knowledge, skills, and performances	4.44	4.44	4.58
5. This pd was tailored to meet my needs as a learner	4.23	4.18	4.21
6. The pd was supported by effective and appropriate use of technology	4.47	4.46	4.55
7. New practices were modeled and thoroughly explained	4.26	4.21	4.37
8. Sufficient time was provided for guided practice and tasks	4.15	3.83	4.13
9. The instructional techniques used facilitated my learning	4.35	4.19	4.40
10. The material used were accessible and enhanced my learning	4.50	4.25	4.51
11. The pd activities were carefully planned and well organized	4.38	4.14	4.36
12. The pd goals and objectives were clearly specified	4.38	4.24	4.37
13. The pd included a variety of learning activities relevant to the topic	4.38	4.41	4.51
14. Time was used efficiently and effectively.	4.15	3.97	4.31

CS teaching techniques in the classroom), **Equity** (level of comfort with engaging their students to consider equity-based practices and **Preparation** (level of comfort with managing different types of situations that arise in the classroom). Teachers rated themselves on questions like “What is your level of fluency in fostering self-directed learning with your students” on a 5-point Likert scale: Don’t know, No experience, Somewhat fluent, Fluent, and Expert.

To determine if the changes to the PD from 2016 to 2018 had an effect on teacher responses, we used the Kruskal-Wallis test. Then, we further analyzed the significant differences using the Mann-Whitney U test to determine which years have differences between them.

To determine if the participants’ planned adoption rates correlated with their actual adoption rates, each year we asked participants to share which labs from the BJC curriculum they used in

their classroom. Nineteen teachers from the 2016 PD reported their actual BJC adoption. Due to logging errors, we do not have this information for the 2017 PD. And since the 2018-2019 school year was still in session at the time of this writing, we also do not have this information for the 2018 BJC PD.

2.6 Results and Discussion

In the following sections, we share results about how our participants rated the BJC PD, their self-perceived ability to teach BJC, and their planned adoption rates from 2016 to 2018. When reading this section, keep in mind that a limitation of this paper is that there are a number of differences between years (e.g. shortening from 3 weeks to 1 week), and diverse teacher populations, so we do not claim that one year's PD was better than another. Rather, we use this data to explore whether the design changes seemed to improve teacher perceptions of preparedness for teaching BJC.

2.6.1 Teachers' Assessment of the PD

To evaluate the significance of the differences between the teacher perceptions of the PD from 2016 to 2018, we used the Kruskal-Wallis test and Mann-Whitney U test as appropriate. Though we did not discover many significant differences between the years for the questions, the average rating for nearly all of the teacher survey questions was above a 4.0, meaning that, on average, teachers agreed with the questions, as shown on Table 2.3. That table also suggests that no harm was introduced by the design changes made between the years. Considering the reduction of time in the 2018 schedule, this can be viewed as a positive indication of the changes made.

2.6.1.1 Time Spent during PD

Two of the statements found on Table 2.3 asked about time spent during the PD: *Sufficient time was provided for guided practice and tasks* and *Time was used efficiently and effectively*. For these statements, we expected the average to show a downward trend from 2016, a 3-week PD, to 2018, a 1-week PD because we reduced the length of the PD; however, this is not the case. Although there were no significant differences in the teachers' perceptions of time usage during the PD, we eliminated 2 weeks from the PD and our participants felt there was high-quality content and instruction, and that their time was used effectively. It may be that teachers were only responding about the face-to-face time spent in each year, not including the work outside of this time in their survey responses.

2.6.1.2 Professional Responsibility and Teaching Techniques

Statements 2, 7, 9 and 13, found on Table 2.3, are related to teachers' responsibilities and instructional teaching techniques. From 2016 to 2018, the average response to these statements increased, which corresponds with the increase in the PD time spent on pedagogy from 3.75 hours to 9.75 hours as shown in Table 2.3. We also believe that our explicit focus on the Lead Learner mindset and

TLO practice helped participants connect the PD to their professional responsibilities as a teacher. Teachers also got a chance to grade programming projects, which is another task these teachers will have to do regularly.

2.6.2 Teacher Perception on Teaching BJC

To evaluate the significance of the differences between the teachers' perceptions on teaching BJC on post-PD survey responses for each area (Content, Inquiry/Engagement, Equity, and Preparation as described in the Methods section) from 2016 to 2018, we used the Kruskal-Wallis test and Mann-Whitney U test as appropriate. Of the 37 questions, 9 have significant p-values.

Table 2.4 shows the 9 questions with significant differences. Each of the questions, except the last question, have a significant difference between years 2016 and 2017, and between years 2016 and 2018. The last question on the table only has a significant difference between years 2016 and 2017. This reflects the emphasis in PD on teaching for diversity, although social justice is a topic we need to continue to improve and reiterate upon.

Table 2.4 These teacher perception question were in the Post-PD survey comparing responses from 2016, 2017 and 2018.

Teacher's Perception on Teaching CSP	2016 (avg)	2017 (avg)	2018 (avg)	p
<i>What is your level of fluency in fostering ... with your students</i>				
self-directed learning	3.53	3.90	3.94	0.020
interest in cs activities	3.23	3.76	3.72	0.016
concentration of cs activities	3.26	3.68	3.64	0.031
enjoyment of cs activities	3.26	3.70	3.73	0.008
<i>How well prepared do you feel to teach...</i>				
cs to female students	3.35	3.96	3.97	0.000
cs to racial or ethnic students	3.32	3.80	3.81	0.008
cs to students of low socioeconomic backgrounds	3.26	3.83	3.78	0.002
students the relevance of cs in their daily lives	3.50	3.93	3.91	0.012
<i>What is your level of fluency with integrating...into your instruction</i>				
principles of social justice	3.29	3.72	3.58	0.033

2.6.2.1 Interest and Enjoyment

During 2017 and 2018, teachers self-reported a higher comfort level of fostering enjoyment and interest with CS activities in their classroom. One possible reason is teachers at the 2017 and 2018 PD received more insight from a student's viewpoint with the Lead Learner mindset and TLO practice than the 2016 teachers. In addition, our group of facilitators expanded over these three years from 8

to 36 leaders, bringing varied viewpoints and growing a larger community of practice rooted by the BJC Master Teachers.

Another design change that may have improved these perceptions is the purposeful inclusion of more reflection time for teachers to discuss each activity and determine how they would integrate them into their classrooms. In Penuel's paper, he concluded that teachers who have more time to think about the PD content, and ask questions to experienced teachers, would feel more prepared to teach their students [Pen07].

2.6.2.2 Prepared for a diverse classroom

During 2017 and 2018, teachers self-reported a significantly higher preparedness level to teach Computer Science to students from groups that are traditionally underrepresented in computing, including females, black/African American and Hispanic/Latinx and lower socioeconomic status groups (e.g. those eligible for free/reduced lunch). The design changes that may have improved these perceptions include the larger number of PD facilitators, and our improved facilitator training that helped facilitators present a more consistent and research-based focus on helping diverse students succeed.

2.6.3 CSP Adoption Rate

Figure 2.1 demonstrates that many participating teachers planned to adopt or integrate some percentage of the BJC curriculum into their classroom. The 2016 data is more skewed towards the right, which may be explained by a difference in the project funding between 2016 and other years; in 2016 all participants were required to apply for crowd funding online, and this was a long and complex process that could only be completed by the most persistent and determined of teachers. Crowd-funded teachers had also publicly posted that they would teach the course, perhaps a higher bar than simply submitting an administrator support letter with their online application as required for participant teachers in 2017 and 2018. A higher percentage of teachers from the 2018 PD (73%) planned to adopt a larger amount of the BJC curriculum (81-100%) than teachers in the 2017 PD teachers (60%). We believe this is because of our design changes to spend more time during the PD on the professional preparation needed to lead, facilitate, and grade programming assignments.

Nineteen teachers from the 2016 PD reported their actual BJC adoption during the 2016-2017 academic year. Of these nineteen teachers, three taught more than they originally planned, three taught within the range they originally planned, and 13 taught less than they originally planned, including only AP relevant curriculum material. Having two-thirds of teachers using less than initially planned for their CS courses is obviously not ideal, however, this may be due to the fact that BJC PD does not assume teachers have previously taught CS courses or experience programming prior to this PD. New non-CS teachers, who will be teaching a CS course, may not be able to accurately estimate how much of the curriculum they would be able to get through in their classrooms. In addition, it is expected that in the first year, new teachers will adopt the minimal units needed to

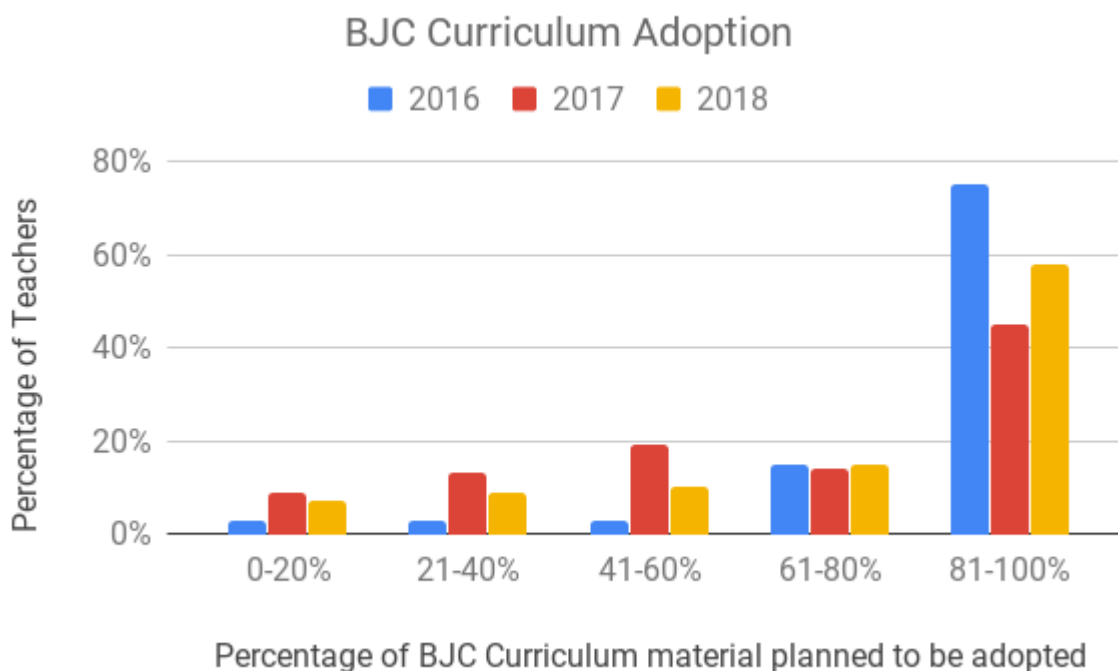


Figure 2.1 BJC Curriculum Adoption Rate Comparison 2016, 2017 and 2018

meet the AP CSP requirements, and that additional units would be adopted as the teacher develops comfort and expertise in the curriculum.

2.7 Conclusion

The primary contribution of this chapter is to show that design-based implementation research methods can be used to refine the BJC teacher professional development to better discover and meet teacher needs. Specifically, the design changes implemented from 2016 to 2018 reduced the time teachers spent in summer workshops (and on related pre- and post-work) while maintaining teachers' perceived levels of quality and preparedness to teach the BJC curriculum. Based on these results, I recommend that teacher professional development programs for AP CSP should (1) limit preparatory work to small, well-specified tasks that do not require prior CS or programming knowledge, (2) include reflection and discussion time after each activity, (3) provide teachers with experience teaching/facilitating, grading, and learning to program, (4) facilitate discussion with experienced teachers, (5) require collaborative work to develop teacher pedagogical content knowledge and teacher peer relationships, and (6) explicitly discuss strategies for recruiting and supporting diverse students and classrooms. These design principles help keep the PD focused on teachers' professional responsibilities, giving teachers a chance to determine if the curriculum is a

good fit for their classroom and students, which could increase adoption rates. Our results suggest that these principles have promoted high teacher satisfaction and self-perceived preparedness.

The work in this BJC teacher professional development program has provided unique insight into the needs of in-service teachers first learning computer science through teacher professional development. The design changes we have implemented have helped to build a teacher community, reaching over 1000 members in our Piazza BJC teachers forum, 750 teachers attending BJC PDs from 2012-2020, and 40 Master Teachers prepared, who can independently lead BJC PD workshops. We believe that the teacher PD workshops have been refined to their shortest possible duration, while also meeting the design criteria for effective teacher professional development programs. Our continual work with and for BJC teachers has shown that teachers continue to learn from one another and from their students as they teach the BJC class. Therefore, based on the intersection of teachers' continued needs for learning, and their students' needs for formative and summative feedback on their programming assignments, the remainder of this dissertation focuses on teacher grading practices for programs, and the design and implementation of GradeSnap, an interactive tool focused on helping teachers streamline their efforts to provide students with rubric-based feedback on their programming assignments.

CHAPTER

3

LITERATURE REVIEW

This research is inspired by Penuel et al.'s design-based implementation research (DBIR) methodology [Pen11; Rus13] to continuously support the needs of teachers in classrooms over many years. In fact, I have been engaged in what Penual calls “infrastructuring” [Pen15]- continuously designing and redesigning infrastructures to support the social movement to transform education by engaging new in-service teachers to teach the creative and rigorous BJC computer science class.

In the previous chapter, I described my work in supporting teachers learning computer science and the BJC curriculum. In this chapter, I describe the context of this work, and relevant research on methods for grading programming projects, including autograders and computational analysis tools, and current teacher dashboard designs and functionalities. I will also introduce relevant user experience research methods, and the design principles used in this research.

3.1 Research Context

As described in the previous chapter, Beauty and Joy of Computing (BJC) is a College Board endorsed curriculum for the Advanced Placement Computer Science Principles (APCSP) high school course. Each summer we run professional developments (PD) across the United States by connecting with Site Coordinators, who host and organize the event, and PD Facilitators, who are experienced BJC teachers that lead the week-long workshop. During each of the PDs, we collect both teacher and facilitator feedback to learn what went well and what could be improved for their PD experience. From attendee feedback, we learned that there is a need for assessment tools and support for grading students block-based programming assignments. Using this feedback, we identified our target audience for GradeSnap as the teachers attending these PD workshops and using BJC. In

/refChapter Three, I discussed these teacher BJC PD workshops and introduced how teachers only have a 5-day workshop to prepare to teach the year-long BJC course. Based on this limited experience, teachers need more affordances within their classes to learn from and with their students, and to help improve student learning through formative and summative programming assignment feedback.

The BJC class is taught using the Snap! block-based programming language. Block-based languages (BBLs) are used in introductory computer science classes to allow novice programming students to learn computational thinking concepts without complicated syntax. The accessible syntax within BBLs even allows for courses outside of computer science to incorporate programming into their classrooms. Examples of these BBLs include Scratch, Snap!, Netsblox, and Cellular. Scratch was initially developed to allow younger ages to program and learn computational thinking concepts without traditional, complicated syntax[Res09]. Other BBLs have been created based on Scratch's original design, such as .

Snap was developed for the Advanced Placement Computer Science Principles course[Har12]. Snap incorporated many functionalities of Scratch with the additional abilities for higher order functions and other higher-level functionalities early-on, such as creating one's own block.

Over the years, Snap! has also been upgraded to meet user's needs in the Beauty and Joy of Computing curriculum. In the near future, they are intending to implement a community base similar to the one within Scratch. Currently, users can publish and share the link to their project though they are not searchable within a Snap! community[Har12].

Both Netsblox and Cellular are BBLs based on Snap!. Netsblox, originally developed at Vanderbilt[Bro17], adds extra functionality to the base Snap!, such as enabling users to send messages to each other using code blocks or users programming in the same environment simultaneously. The Netsblox environment adds new code blocks to support these functionalities. Cellular, originally developed at Monash University in Melbourne, Australia, adds other functionalities to the base Snap!. Cellular allows for users to program their sprite based on neighboring sprites, to start with multiple instances of each sprite and to track variables with a graph. Cellular is useful for programming simulations[AL12].

3.2 Grading Programming Projects

In this section, I will describe related works relevant to grading programming projects of introductory computer science courses. More specifically, I will discuss (1) how some expert instructors grade a general programming assignment and (2) current methods for assessing block-based programming projects, including autograders, a computational analysis tool, and rubrics.

3.2.1 General Program Grading Methods

Fitzgerald et al. expressed the need to share more about how they grade in order for researchers to determine the best way to help newer teachers learn how to grade text-based programming

assignments [Fit13]. Providing grades with detailed feedback of a programming assignment could benefit the student by helping them develop programming skills and heavily influence the student to determine if computer science is the right major or career choice for them. Therefore ensuring consistent grading of the open-ended programming problems from student to student is crucial. Since Fitzgerald et al. found a limited number of publications about grading programming assignments, they wanted to figure out what experienced instructors think about when grading programs. They facilitated a study to determine what he and his peer instructors thought about when they graded programs [Fit13]. Though the general goal decided prior to any grading was for the instructors to determine which students should go on to the next class and those that should not, Fitzgerald et al. found that the instructors graded with different objectives in mind by evaluating their grading methods [Fit13]. Two of the instructors used an analytic approach, one used a holistic approach (simply focusing on pass/fail), and the other used a primary trait approach (also focusing on pass/fail). Even though the instructors used a variety of methods to grade the programs, they still arrived at the same general conclusion, such as if a student was very good or very bad. Fitzgerald et al. articulated the need to share with students and peers how we grade more clearly, even if we are headed towards more automatic grading [Fit13]. Additionally, “assessment requires [faculty] to articulate... explicit and public statements of criteria of performance... [which helps]... faculty refine their own understanding of expected abilities, clarify for their colleagues the basis of their judgment, and enable students to understand what performance is required” [Loa86].

One method to disclose grading techniques for programming projects to both students and peers is by creating and sharing a rubric. Rubrics can be beneficial to give appropriate feedback to students[Eug16]. Ahoniemi found their rubric based grading tools allowed for teaching assistant’s to provide objective grades with high quality feedback in a reasonable time [Aho08; Aho09]. Becker et al. states that “learning increases...when learners have a sense of what they are setting out to learn, a statement of explicit standards they must meet, and a way of seeing what they have learned” [Bec03; Loa86], however, Eugene et al. found that differences in student demographics, slight differences in teaching, and different, although similar, assignments affects the results obtained and makes it more difficult to definitively say that rubrics can benefit the student in regards to their learning in their study [Eug16]. Eugene et al. found that for rubrics to be useful, the programming project rubric needs to be (1) simple and specific to an assignment for students and (2) detailed enough for the instructor to use [Eug16].

Rubrics can be helpful for students to use rubrics to understand what elements they are missing from their submissions. Some students are even able to improve their grade by adjusting their project when given a rubric [Bar11]. Providing the rubric to the students reduced the number of complaints from students about their grades in one study [Bar11], potentially suggesting they understood what they got incorrect and how their grade reflects it. Rust had the students assess their own assignment, which resulted in an improvement of (1) the students’ understanding of the grading criteria, (2) their self-evaluation skills, and (3) their quality of their coursework [Rus03]. However, sometimes grading by students does not align with how a teacher may grade the same assignment [Häm11].

Hamalainen found that after a student and teacher graded the same submission, it was as if they used a different scale or rubric, suggesting that the rubric needed further clarity for similar student and teacher grading [Häm11].

Stegeman et al. investigated how to design a rubric for feedback on the quality in programming courses and disseminated their methods for developing a rubric to better support novice programming students [Ste16]. In rubric development, they required that rubrics have an even number of criteria to remove the opportunity to bias towards the center for each category on the rubric [Ste16; Wal11]. Stegeman et al. facilitated the following three iterations with teachers for rubric development on the quality of programming.

1. Iteration 1:

- Teachers think aloud while grading programming submissions and suggest 2-3 changes to enhance the project. Teachers grade without a rubric.
- Teachers grade the same submissions using a rubric.
- Teachers reflect on any similarities or differences between their feedback on the previous 2 steps.

2. Iteration 2:

- Teachers give feedback on and determine 2 code quality goals from students' programming submissions. Teachers do this without using a rubric.
- Researchers reviewed a rubric for this programming assignment with the teachers.
- Teachers evaluated the same programming submissions with this rubric.
- Teachers reflect if their previously decided code quality goals were in rubric and classify other issues they experienced.

3. Iteration 3:

- Teachers give detailed code review, specify a level for each criteria within the rubric, and give comprehensive feedback, including distinguishing code references.
- Researchers interview the teacher, reviewing each level of criteria from the rubric and the detailed, corresponding feedback.

Stegeman et al.'s results were grouped into 4 design principles when designing a rubric for assessing code quality in programming assignments: Abstractions, Goals, Terminology, and Phrasing. They found that "condensing all information in the code quality model for use in the rubric sometimes leads to unclear generalizations or missing concepts" (Abstractions) [Ste16]. Stegeman et al. also found that the explicit goals of Consistency, Convention, and Lack of Trying "helped teachers' understanding of the rubric" (Goals) [Ste16]. They also found that terminology needs to be more linked to computer science and needs to be clear, and therefore, "creating a general or local glossary of

common terminology...should contribute to a better choice of words” for the introductory teachers (Terminology) [Ste16]. And finally, Stegeman et al. found that “teachers appeared to need more guidance to make better decisions” (Phrasing) [Author2]. For example, teachers were “overly strict” with following the rubric guidelines however this was solved by “adding moderating words, such as ‘generally’” [Ste16].

The grading methods and rubrics in this subsection were about grading programs which use text-based languages. More research needs to be done to understand how to communicate grading methods and feedback regarding programming assignments and projects with peers and students. In the next section, I discuss the grading methods of block-based languages, commonly used in introductory classrooms as an introduction to programming concepts. There is even less research done on how to teach teachers to grade block-based programming projects which is extremely important because many teachers who use block-based languages in their classroom may not have as much experience in computing. I discussed the intended users more in Chapter 2 to introduce the intended audience of my research.

3.2.2 Block-Based Language Programming Methods

Catete saw the need to support teachers with rubrics who were preparing to teach the Beauty and Joy of Computing [Gar15; Mor14b] Advanced Placements Computer Science Principles course [Cat16]. She found that novice CS teachers can use rubrics, created for specific projects, to grade novice student block-based projects within Snap! [Boa17]. These initial rubrics were called task based rubrics as Subramaniam and Catete later created learning-oriented rubrics. These rubrics focused more on the learning objectives which should be evident from the specific lab [Sub17]. Both the task- and learning-oriented rubrics used categories, such as correctness or efficiency, with descriptions so the teachers could identify which rating a student should receive based on their project [Cat18].

3.2.2.1 Computational Analysis Tool - Dr. Scratch

Dr. Scratch is a free and public source which allows users, or students, to upload their work from the scratch environment and grades assignments based on program complexity and gives you a computational thinking score. The concepts that it grades on include abstraction, parallelism, logic, synchronization, flow control, user activity, and data representation. Dr. Scratch also detects possible issues within the scratch code such as duplicated scripts, poor sprite naming, dead code, and sprite attribute initialization [ML15b].

Duplicated scripts are an example of code that could be used within a function or a custom block because there is more than one instance of the same code. Sprite naming would suggest to the user to rename their Sprites since naming conventions are important in computer science. When a user has a dead code alert it means they have some code within their program that is never executed. If a user receives the Sprite attribute initialization alert, it means during the program the Sprites attributes, such as costume or location, are modified during execution but were not reset for initialization for the next execution. After grading the scratch project based on the seven concepts

and detecting potential issues, the autograder and then tells you what level of programmer and you are; basic, developer, or master [ML15b].

Currently Dr. Scratch is being upgraded to work with the newest scratch version. Future attributes of Dr. Scratch include a multilingual website which will support teacher accounts to enable the teachers “group...their students to keep track of their progress”[ML15a]. Developers are currently upgrading their servers to the cloud for better performance and developing browser plugins so users will be able to analyze and grade their code in the scratch window. They are also planning to add some social network functionalities to Dr. Scratch (Moreno-Leon 2015).

3.2.2.2 Block-based Autograders

In this section, I will objectively describe some already existing, popular block-based autograders. I explore the user experience and review any publications on each. In addition, for each of the autograders.

ITCH is an autograding system which can grade the Scratch programming language[Joh16]. It is free for the first 2 weeks and then there are price options for single teachers or districts (ITCH). It cannot work as a standalone student system for developing computational problem solving. ITCH has a teacher dashboard with a repository of lessons to use. Teachers can also develop their own lessons or take existing lessons and add their own content or instructions. Within the teacher dashboard, teachers are able to follow their students’ progress. ITCH also has a student dashboard which is their homepage upon logging in.

ITCH embeds the Scratch programming environment within their online software therefore the students complete their programming using ITCH. After the teachers choose the lessons for the class, students are able to choose which project they work on from that subset. Instructions for the assignment are in embedded within the programming environment, allowing the students to see the instructions and program at the same time within one screen. On the student dashboard, students are able to share code or projects to other students or within their community (ITCH). Recently, ITCH upgraded their system to work with Scratch 3.0 and to include an iPad friendly interface. They hope to make it more mobile friendly [Joh16].

Lambda is an autograder for the Snap! programming language. The lambda developers supplied ample development documentation. They focused on helping students “improve their programming skills and complete lab exercises” making tool for grading without causing “students unnecessary stress or frustration”[Bal18]. Along with grading student work, they are wanted to give constructive feedback to the students.

The developer’s main design principle, Learner-Centered-Design, adapted from User-Centered-Design, focused on four aspects of the learners: domain expertise, heterogeneity, motivation, and changing understanding. In order to determine feedback for the students, they used Knowledge Integration, “a framework for approaching how students should synthesize information”[Bal18]. Knowledge integration also has four components: Adding knowledge, eliciting ideas, distinguishing

knowledge, and reflecting but the Lambda team focused on the second and third elements because they were the easiest to integrate into non-exploratory grading tools.

The development team made sure to include a Teacher Assistant Centered Design since Teacher Assistants (TAs) would typically be the graders for this specific research. The biggest take-away from this section in their paper is that the grader or instructor has limited time so ensuring the tool is easily usable is important[Bal18].

In order to use the autograders, teachers could take the practice problems and give the students instructions with the snap link. Within the snap window, students were able to get feedback on their current code by clicking a button in the interface. Students were also able to restore their best submission, restore their last submission, reset back to the starter file, ask for help which would display hints, and show previous test results.

3.3 Rubrics

Brookhart describes a rubric as a “[coherent] set of criteria...that includes descriptions of levels of performance quality” with a main purpose to assess performances, for example to assess student’s work [Bro16]. The grading criteria can be used to assess student performance on a task, indicating if a student learned the intended learning outcomes. Brookhart explains that the structure a rubric gives to observations of student performance changes potentially biased judgements into “descriptions of performance that can be used for feedback and teaching” [Bro16]. Rubrics allow for “the possibility of objective, consistent evaluation minimizing differences in grades” from submission to submission [Rag20].

Brookhart describes different kinds of rubrics, comparing Analytic to Holistic and General to Task-Specific. Analytic rubrics break down student work into subtasks, allowing teachers to evaluate student performance of specific criteria individually, whereas, Holistic rubrics allow for a teacher to evaluate the all criteria of a student’s work at the same time [Bro16]. Analytic rubrics are best for a class to allow the teacher to give detailed feedback and are commonly used as formative assessment, however, they can also be used for summative assessment. Holistic rubrics are best used for summative assessments and when not intending to provide qualitative information along with the numerical grade.

General rubrics can be used to evaluate multiple assignments with different instructions, grading on a set of less specific criteria, whereas, Task-specific rubrics are detailed and specific to an assignment [Bro16]. General rubrics can be shared with students and applied to assignments with the same learning objectives. Task-specific rubrics may contain solutions within the rubric and therefore teachers may not be able to share them with their students. This is an unfortunate disadvantage because students could benefit from viewing the rubric ahead of time to understand the quality of code that is expected of them [Bro16].

In addition to these rubric types and descriptors, Fluckiger shared Dietz’s introduction to a single point rubric, where “the single point rubric provides a single set of criteria” [Flu10; Die00].

3.4 User Experience Research Methods

To best aid our end users, we want to ensure a good User Experience (UX), defined by “a person’s perceptions and responses resulting from the use and/or anticipated use of a product, system or service” [Mir15]. By considering a user’s experience, the designer has a better chance of “making systems that are useful, usable, and pleasurable to use” [Bur18]. There are many User Experience Research (UXR) methods which can help researchers or developers understand a participant’s experience [Roh14].

Rohrer explained many UXR methods and introduced a 3-dimensional framework to guide the researcher to determine which UXR methods to use depending on the goal and the stage of the current product [Roh14]. The 3-dimensional framework has the three axis: (1) Attitudinal vs Behavioral; (2) Qualitative vs Quantitative; and (3) Context of Product Use. During early stages of development, Attitudinal research is used to understand what people say and can easily accommodate early stages of product development since it supports research prior to a developed product [Roh14]. Some of the UXR methods used at the beginning of tool development include personas, participatory design (specifically interviews and card sorting) [Roh14] as well as a design sprint [Kna]. With these methods, a qualitative research study allows the researcher to understand why something is a problem or which solutions might exist. The open-ended nature of qualitative studies is suited for the first two stages of product development as the researcher is gathering ideas instead of refining the concept and measuring for effectiveness [Roh14]. For the Context of Product Use dimension in Rohrer’s description of the 3-dimensional framework, there are four different ways to determine participant’s interaction with the product: (1) Natural use of the product; (2) Scripted use of the product; (3) Not using the product during a study; and (4) a hybrid or combination of any of the three previous options [Roh14]. Naturally, the simplest of the Context of Product Use options to use for a product in early stages of development is *Not using the product during a study* since in the ideation phase, there may not be a product to use and the researcher is trying to understand what would be best for the product.

Rohrer also explains how the stages of a product help influence which UXR methods a researcher may use. The three unique stages of product development include (1) Strategize, where the goal is to “explore and choose new directions” for the product; (2) Execute, where researchers “inform and optimize designs...to reduce risk and improve usability”; and (3) Assess, where researchers “measure product performance” [Roh14]. At the beginning of product development, the useful UXR methods vary greatly. For example, researchers may use participatory design methods and concept testing to confirm motivations and desired features for the end product [Roh14].

3.4.1 Personas

Personas are important because they help developers/researchers/designers consider a wider variety of users [Fra19]. Hobbies, personalities or social characteristics are important for personas because it gives the designer some insight on who might use the tool, and where or when they might be grading,

which may help the designer consider more ideas which could reduce risk later in execution [Fra19]. For example, considering that one user may work from home on a tablet instead of at work on a computer, adds additional non-functional requirements to the tool (e.g., must be accessible from multiple devices, ideally online; must have a mobile version). Additionally considering two different users' hobbies: one may be a vlogger and one may still have a non-smart phone; by considering these two hobbies creates different types of users: one that may be tech-savvy and one that may struggle with technology. With these two very different users, we would have to consider creating an informative tutorial (for the non-tech-savvy), yet the users should be able to skip the tutorial if they don't currently need it (for the tech-savvy). Without considering specific details, the designer may not be cognisant of the variation of features to consider from the different users.

3.4.2 Participatory Design

In participatory design research methods, participants are asked to construct "their ideal experience" in order to provide information on what elements of the design matters most to them, what they need and want from the design, and provide explanations for these preferences/perspectives [Roh14; Abe13]. By considering the needs and perspectives of classroom teachers, through a variety of participatory design methodologies (ex. brainstorming exercises, group affinity diagramming, rapid paper prototyping, and wireframe walkthroughs), researchers can increase the likelihood of deployment and adoption of a Teacher Dashboard [Abe13].

Interviews One method to get the participants involved in the research process early can be as simple as interviews. In order to truly learn what the user needs, "a researcher [could meet] with participants one-on-one to discuss in depth what the participant thinks about the topic in question" [Roh14].

Card Sorting Card sorting is another of the many participatory design methods. Card Sorting can be "a quantitative or qualitative method that asks users to organize items into groups and assign categories to each group. This method helps create or refine the information architecture...of a site by exposing users' mental models" [Roh14].

3.4.3 Sprint

A UX Sprint is commonly used to develop a prototype quickly and in this case, to develop a solution to solve a problem. By evaluating a prototype early in the design process, less time and money are spent implementing ideas that later are evaluated and tossed. A UX Sprint normally takes a week to complete for a team of user experienced focused researchers and designers, finishing one stage per day. A description of each stage is listed below [Kna]:

- **Research Stage** - Understand current solutions which mitigate the problem
- **Generate Stage** - Develop multiple solutions to solve the problem; some of these may be ideas
- **Choose Stage** - Decide which solutions will be implemented

- **Prototype Stage** - Develop a quick paper or digital prototype to express the chosen solutions
- **Evaluate Stage** - Test the developed solution to find early changes or misunderstandings

3.4.4 Other UXR Methods

In this section, I will introduce relevant additional User Experience Research Methods I've utilized in my research.

Card Sorting is a good way to determine an Information Architecture [Roh14], which is the organization of content to allow users “to easily adjust to the functionality of the product” [web]. To use card sorting, the researcher may write words or phrases on cards and ask the user to organize them and label the groups of cards with categories [Roh14].

Expert Review is when a single expert walks through a product to look for issues with design, accessibility, and usability [Roh14]. This process varies from profession but can retrieve quick insight for the researcher for any general issues that may be specific to the domain.

Field Studies are where the researcher observes users in the wild to measure behavior in the context where a product will actually be used [rorher]. In some field studies, it is recommended to have an observation protocol which reminds the researcher what they are looking for when observing. In the case of an ethnographic study, the researcher's goal is to “listen to the talk, watch what happens, see what people do, to write it down, tape it, [and] record [data]...[such as] conversations, diagrams of places,... transcripts of meetings and so on” with “characteristically vague objectives” [web2].

Usability Testing can be done as a session or remotely. If the usability testing is occurring as a session, or a synchronous observation session, the researcher can observe the user(s) carrying out a set of or a single user task(s) with a product[Roh14]. If the usability test is remote, it is easier to have a wider range of participants and more participants in the study. Additionally, this option saves time however the researcher cannot observe or interact with the participant.

3.5 General Good Design Principles

In order to increase the likelihood of a well-designed tool, we intentionally adhered to the following list of previously praised design principles.

1. *Problem free navigation* - Make sure navigating the tool is easy for the user [Mem19]
2. *Learn what user needs* - Don't assume what the user wants, make time to understand what they need in the tool [Mem19]
3. *Content before Visuals* - Know the content before you visually organize the site [Mem19]
4. *Keep it consistent* - If part of the tool is similar to other products, don't reinvent how it is designed unless it's solving a new problem; keep the design so the tool is more intuitive for the users [Mem19]

5. *Ask the right questions* - Know the stage of the product so you can ask the right questions to move research and development further [Mem19] [Roh14]
6. *Context is key* - Understand your target audience by creating personas and interviewing future users [Mem19]
7. *Less is more* - A clean and simple interface is better than a cluttered interface [Mem19]
8. *Feedback matters* - Ask participants or other researchers for feedback frequently and throughout the process; not just at the end [Mem19]

3.6 Teacher Dashboard

When designing a tool you expect a teacher to use nearly every day during their job, it makes sense to include them in the early design and development stages of the application[Abe13]. Abel discusses how they used Participatory Design and Contextual Design to decide the early design decisions[Abe13]. Some of the contextual design principles Abel used to create a teacher dashboard application included:

1. System design must support and extend users work practice
2. People are experts at what they do but are unable to articulate their own work practice
3. Good design requires partnership and participation with users
4. Good design is systemic
5. Design depends on explicit representations

Moving forward, Abel used participatory design in 3 phases to ensure that the teacher dashboard was regularly tested with users[Abe13]. The 3 phases included:

- Phase 1 - "Generate ideas for content and design...better understanding of target audience...gather insight to create an effective application"; Identify types of data teachers would want to understand from their class on post-it notes
- Phase 2 - Card sorting of ideas or data; "Revisit ideas from phase 1 and group into self-identified categories"; Interact with wireframe prototype of dash app
- Phase 3 - Discover data interactions and gestures teachers prefer

Molenaar et al. explored the use of a teacher dashboard with analytical information about the class progress in real-time and how or if the dashboard affected the teachers pedagogical actions. They found that teachers were able to use information from the dashboard to alter their teaching behavior [Mol17].

Teachers were supportive in Abel research as they gave advice on features for the teacher dashboard, which included (1) detailed student pages, (2) line or vertical bar graphs to help them understand the classroom's success in the class, and (3) classroom management capabilities, such as adding or removing a student to or from a class, or transferring a student to a new section[Abe13]. I considered these design principles for dashboards while planning initial design for GradeSnap.

3.6.1 Dashboard Designs

Sarikaya et al. has found that “dashboard design, development, and use has the potential to change the lives of millions of people,” however current tools are not meeting the needs of their diverse users [Sar18]. The designs need to be more flexible and customizable for more uses to benefit from using the dashboard, especially since “layout and arrangements can have significant impact on the efficacy of a dashboard” [Sar18]. Abel et al. used participatory design to develop their dashboard and said that “without the [participatory design] sessions, the application dashboard would have been created by designers void of understanding the affordances and feature requests of the teachers” [Abe13].

Dashboard designs vary from tool to tool. Naturally, the majority of the teacher dashboards are submission locations for student work [DeN17], [Zha20], [Sin17] and report each student's grades, but some do much more. Some dashboards focus on students' actions within the tool and may report that log information to the teacher. For example, the dashboards OK, Coursemology, and OpenEdX show the teacher each student's progress per assignment [DeN17] [Gro16]. Some tools, such as Fairy Assessment, go so far as to predict student's grades based on their actions, or predict estimates of how many students are working or finished with a specific task [Dia17]. In order to help teachers understand their students' learning gains, some tools will send statistics or reports to instructors with common wrong answers or mistakes [DeN17], [Sin17], [Gro16].

Some current student dashboard designs include student actions such as communicating with the instructor (coursemology) [DeN17] and viewing graded assignments as soon as they are posted (Gradescope) [Sin17].

Other common design elements and functionalities found in dashboard designs for educational practices include:

- Assignment customization [DeN17]
- Assignment management [Zha20]
- Plagiarism detection [DeN17]
- Code quality feedback [Zha20]
- Feedback review [Zha20]
- Processing handwritten assignments [Sin17]
- Support for both homeworks and exams [Sin17]

- Grade from a rubric, allowing “richer form of feedback” that enables “transparency and consistency” [Sin17]
- TA Grading [Zha20] [Sin17]
- Automatic assessment [Zha20] [Sin17] [Iha10]

User experience research methods (UXR) are appropriate for use throughout the development of our grading dashboard at the different phases of the project. At the beginning of project development, attitudinal research is needed to gain understanding of our audience’s wants and needs. As we are at the beginning stages of development for this project, we focused heavily on using these types of UXR methods, particularly personas, interviews, card sorting, participatory design, and a design sprint. We use these types of qualitative research methods to gather insight into why something is a problem or which solutions might exist.

By adopting participatory design research methods (ex. interviews, card sorting, paper prototypes) we are better equipped to understand what users need and want from the design, increasing the likelihood that our dashboard will be adopted. We intentionally selected these methods so that participant feedback would be a key part of our early research. By involving participants in this process and providing them with prototypes to evaluate early through our design sprint, we are able to best serve our target audience, teachers who are teaching AP CS Principles using the BJC curriculum. This means we will be better able to understand how the user will be using the tool and adapt our work to their needs by asking appropriate questions throughout the different phases of the project.

Throughout our creation of these prototypes during the design sprint, we intentionally adhere to research- and industry- supported design methods, including specific recommendations for dashboard designs.

3.6.2 Dashboard Impacting Teacher Actions

Part of our goal in creating a dashboard is to help teachers adjust their class lessons based on individualized information from the class’ overall learning gains. Specifically, through a dashboard grounded in distributed cognition theory, which states, “instruments can support professionals when these instructions fit seamlessly into the activities of the professional” ([Hut00] as quoted in [Mol18]). To understand how a dashboard may impact a teacher’s classroom actions, Molenaar et al. investigated how teachers used a dashboard [Mol18]. Molenaar used exploratory research methods to observe teachers using a learning dashboard, which “models the probability of a student answering a question correctly by calculating a student’s ability score, which is the representation of a student’s knowledge on a particular learning objective” [Mol18]. As the students complete work within the system, the teacher can follow students’ progress using any of the three dashboards (the lesson overview, the class overview, and the progress dashboard) in the system [Mol18].

To understand the teacher’s understanding of the learning analytics from the dashboard, Molenaar used Verberts learning analytics stages, including: (1) **Awareness stage** - where the teacher

aware of the dashboard; (2) **Reflection stage** - where the teacher interprets data by asking questions; (3) **Sense making stage** - where the teacher asks to further understand the value of the data; and (4) **Impact stage** - where the teacher understands the data and how it is used to change their behavior [Mol18; Ver13]. To evaluate changes in the teachers' instructional methods, Molenaar used a Bergh et al.'s features of feedback, developed from the literature, which are as follows [Mol18; Ber15]: (1) **Task related**: helps students fulfill task; (2) **Process related**: how students handle learning; (3) **Personal**: students behavior; (4) **Metacognitive**: students control and monitor their learning; and (5) **Social**: cooperative learning skills.

Molenaar found that the teachers who referenced the dashboard more, gave more diverse task and process based feedback to their students. In addition, they also found more **Task related** and **Process related actions** after the teacher looked at the dashboard. They only found a few **Personal actions** and no **Metacognitive or Social actions** after looking at the dashboard [Mol18].

To further support teachers new to computing or new to teaching, it is important to make dashboard elements understandable and useful for teachers using the tools. Understanding how our tools assist teachers is only a small step to improving our tools.

CHAPTER

4

INITIAL GRADES NAP DESIGN

To support teachers to provide feedback for students learning to program, we investigate **RQ2**: How do we design an effective tool to support teachers in grading programming projects?

Specifically, our research design pursued the following, more detailed research questions in the following order, based on a user-experience research methodology to: collect user needs and requirements (RQ2a), prototype a tool and get feedback (RQ2b), collect information about how the users do the task without a tool (RQ2c), and finally, evaluate the designed tool with target teacher users (RQ2d):

- **RQ2a-GS design**: What features do teachers want and need in an online grading tool for programming projects?
- **RQ2b-GS prototype**: How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change with tutorial usage?
- **RQ2c-Grading**: How do teachers grade programming projects with and without rubrics, but without tool support?
- **RQ2d-GS evaluation**: How do teachers grade with and perceive GradeSnap? How does teacher-use of the grading tools differ based on their expertise/experience or goals for student feedback?

In this chapter, I investigate RQ2a to establish the initial design for GradeSnap. I describe the initial user design principles and decisions that guided the initial study with users and experts, and were used throughout the development process in subsequent development and chapters.

To prototype a design for GradeSnap, I implemented the User Experience (UX) Research method called a UX Sprint to determine early necessary features and the desired flow of the tool. A UX Sprint is commonly used to develop a prototype quickly and in this case, to develop a solution to solve a problem. By evaluating a prototype early in the design process, less time and money are spent implementing ideas that later are evaluated and tossed. Normally, a UX Sprint takes a week to complete for a team of user experienced focused researchers and designers, finishing one stage per day[Kna]:

- **Research Stage** - Understand current solutions which mitigate the problem
- **Generate Stage** - Develop multiple solutions to solve the problem; some of these may be ideas
- **Choose Stage** - Decide which solutions will be implemented
- **Prototype Stage** - Develop a quick paper or digital prototype to express the chosen solutions
- **Evaluate Stage** - Test the developed solution to find early changes or misunderstandings

4.1 Initial user design - Phase 1

GradeSnap has been developed with a design sprint, a user experience (UX) research method, to increase the usability and usefulness of the tool for the teacher, or grader [Kna]. Non-traditionally, there was not a group of user experienced focused researchers or designers to complete a UX sprint, therefore, I completed in-depth interviews with 8 experts to imitate a group of researchers completing the UX sprint.

4.1.1 High-Level Overview of Adapted UX Stages

Here is the adapted, high-level description of my UX Sprint stages. As a reminder, I am using an adapted UX Sprint to find a solution to allow teachers to grade their student's block-based programming projects. Below the brief outline is a more in-depth discussion of each deliverable from each stage.

- **Research** - Reviewed tools teachers use to grade block-based programming projects; Developed personas; Interviewed block-based graders made up of 7 expert CS graders and 1 high school CS teacher about what they'd need in a tool
- **Generate** - Generated numerous solutions in an attempt to alleviate the problem during the interviews with the 8 block-based graders to ascertain critical features and multiple design flows for an online grading web tool.
- **Choose** - Determined a couple solutions from the 8 interviews, developed from the previous stage, to find the best solutions; this was followed by another short, individual intervention

with the same 8 cs block-based graders to elect one of the final solutions from the previous step by testing out two different options developed.

- **Prototype** - Designed a quick paper prototype of my plan, and
- **Evaluate**- Assessed the design with the same group of block-based graders to work out any last previously unforeseen issues.

4.1.2 In-Depth UX Sprint Discussion

In this section, I will go into detail about decisions related to the design flow developed using an adapted UX Sprint. As a reminder, the motivation for developing this tool is to lessen the burden of grading by creating a more fluid system for the teachers to assign, collect, and grade their students' block-based projects, as well as understand their class progress. Since I was working alone, I had to adapt the traditional UX Sprint.

4.1.2.1 Research Stage

After deciding the problem I would solve in the Research Stage, my goal was to learn about other systems which tried to solve similar challenges. In order to keep my end users in mind, at the start of the research stage I created 5 different teacher personas that included specific details about their needs and their hobbies. After researching each of the following systems, I imagined each persona using the tool to determine if and what changes would be needed within the tool.

I investigated existing systems that attempt to assist teachers with grading, such as paper rubrics, CT analysis, and auto grading systems. Here are a few key points from each of the systems. Cateté's rubrics assisted teachers by giving them a rubric for a specified Beauty and Joy of Computing lab. After teachers collected completed labs from their students, they were able to grade each student's project with the rubric by choosing a point for each category based on a textual description carefully designed by Cateté and her team. The number of categories and points for each rubric varied by lab[Cat16]. I planned to use Cateté's rubric development method to create rubrics within the system. Dr. Scratch, Moreno's CT analyzer tool, gave a CT score to a Scratch project. The tool scores a scratch project based on computational thinking categories[ML15b]. Dr. Scratch's assessment could allow teachers to grade general projects, but only based on CT components[ML15a]. Therefore, this tool would not assist teachers with grading for a specific assignment with a desired end goal. However, it could help a teacher understand what elements the student incorporated into their project and how they could improve with the feedback from the tool[ML15a].

Systems with autograding support included ITCH[Joh16], for SCRATCH; Lambda[Bal18], for Snap!. Autograders assist teachers by cutting down on the time teachers would otherwise spend on grading. This also helps students by giving them automatic feedback in many of these systems. Each of these systems had different perks. For example, within ITCH the teacher could view their students' progress on their projects, while Lambda have an entire curriculum ready for teacher use[Bal18]. Although these systems work well for specific assignments, unless a teacher has a high

level of experience with programming, they cannot easily change the autograder to assess a new project that they have created.

Beyond grading systems for Block-Based Languages, one of system I investigated was Gradescope, which is an online grading tool with multiple supports to allow teachers to grade student's work. Teachers can grade using rubrics in Gradescope. The main window will reflect the students work in a pdf form and the right margin is a rubric which a teacher can modify at any time. Gradescope can even autograde some assignments at varying levels (either grade the project and assign a grade for each student or cluster-grade projects by grouping similar solutions together and ask the teacher how they want to grade the grouped-solutions which reduces the number of projects the teacher has to grade). The autograding mechanism and cluster grading work for specific kinds of assignments, such as a math assignment[Sin17].

4.1.2.2 Generate Stage

With these systems in mind, in the Generate State, I hoped to develop multiple ideas for what should be within the tool and how the flow for the tool should work. I accomplished this by hosting interviews with experienced block-based language graders and the intended users to gain insight on what they would want or expect in a system to lessen the burden of grading block-based projects. Here is a high-level overview of my interview protocol:

- Introduction (5 minutes)
- Pre Questions (5 minutes)
- Determine Screens and Flow (10 minutes)
- Arrange Paper Prototype and Complete User Tasks (10 minutes)
- Post Questions (5 minutes)

The 8 interviewees were picked due to their availability and experience with block-based languages. For the remaining of this section, I will go into detail about the interview elements specified above. These elements in the protocol are important as the interviews worked as a substitute due to the lack of researchers working on this project.

Part 1: Introduction

In the introduction, I explained the problem.

Snap! is a block-based language designed for novice learners. This programming language is commonly used in entry-level computer science courses and STEM courses which integrate computational thinking. These classes may be taught by teachers new to computer science or computational thinking topics. In Snap!, students are able to program projects using similar programming techniques, such as loops, functions and conditionals, which are used in other popular programming languages. Snap! does not

provide an interface for teachers to grade Snap! projects which is a common inconvenience, sometimes obstacle or worry, since many of the teachers facilitating these projects do not have a computer science background. Even for experienced teachers, grading programming assignments is still a hassle.

In order to ease this burden, I am creating a tool for grading Snap! projects, called GradeSnap, based off Gradescope. Gradescope is a different grading tool that enables consistent grading by allowing teachers to create rubrics, displays classroom progress in a dashboard, and allows students to submit their homework and see their corresponding grade once their teacher publishes the grade set. These are some of the features I plan to implement in GradeSnap.

Depending if the interviewee has used or seen Gradescope, I opened the tool to show the appearance and functionality of the grading screen.

Part 2: Pre Questions

I asked these 3 initial questions prior to the bulk of the interview because I wanted their initial thoughts before diving into the flow of the tool. Throughout the rest of the interview many of these topics arise as a short discussion.

- If your tool had a dashboard, what would be on it?
- If you were teaching a course, what statistics of your class would you like to know?
- If you are four assignments into your class without an exam, how do you judge how your class is doing?

Part 3: Determine Screens and Flow

To determine which screens should be in the tool, I asked the interviewee to think of which screens they would want available within the tool. They were able to use a selection of sticky notes I already made with general screens and add on with additional stickies if I was missing a screen they envisioned to be within the tool, or they could start from scratch by developing their own sticky notes or by listing them on paper. After completing their list of screens, I asked them to indicate the flow of the tool (i.e., using arrows to show which screens were linked).

Part 4: Arrange Paper Prototype and Complete User Tasks

Now that the interviewee has developed a list of screens and the flow for the tool, I handed the participants some paper prototype (hand-drawn) potential designs of the screens. Using the participants design flow, I asked them to arrange these screens on the floor indicating out loud how they were ordering them. Participants were able to omit screens that did not exist in their plan or draw out new screens which I did not have created. I asked participants to review the screens and tell me how they were different from their vision, what they would change and what they would keep. With the paper screens arranged, I asked the interviewee to complete a set of tasks a teacher would have to do with the tool. While completing these tasks, I continued to ask the interviewee to think aloud and express when the path to doing the task was not intuitive. These tasks included:

1. Try out GradeSnap without an account. Select a project to grade. Now Select a project from your computer. You have added [the project] Draw a Square to GradeSnap. You have added some rubric items in the right margin. you play the student's code in the embedded Snap! interface to the left and marked which points the student gets. The grade is automatically calculated at the top. You decide you want to use GradeSnap.
2. You are 4 assignments into your semester. You need to complete grading assignment 4 from class 1. Please login and go to the assignment page. Grade your next student's assignment.
3. Change your email address.
4. From the gradebook for Course 1, go to the stats page for Course 1. Is there another you should be able to do this?
5. Make Course 2 inactive.

Part 5: Post Questions

After finalizing their flow and completing expected user tasks, I asked the interviewees three final questions to understand their preference for where the following actions could take place:

- You need to add another class, which web page(s) can do this action?
- You need to add another assignment, which web page(s) can do this action?
- Where can I find the median score for assignment 3 for Course 1?

4.1.2.3 Choose Stage

My goal for the Choose Stage was to decide which ideas, screens, and flows would carry through to the first prototype implementation. Based on information collected from the interviews during Generate Stage, I found the two most common flow designs for the tool and a general idea of what was intuitive to the interviewees from my notes and observations from part 3 and 4 of the interview protocol, respectively.

Many of the interviewees agreed on many of the core screens within the tool, such as:

- **Login Screen** should allow teachers to create an account, view a demo, or login.
- **Dashboard Screen** allows a teacher to access everything else they need within that course such as the following screens:
 - **Gradebook Screen** which would operate as the standard gradebook, allowing a teacher to view all of their students' scores on all of the assignments
 - **Data Center Screen** which would show a teacher interesting statistics about their course so they can learn how their students are doing

- **Assignments Screen** which would allow a teacher to add a new assignment or go and view a specific assignment to edit or publish for their class
- **Grade Screen** which is where the teacher would actually grade their students' work
- **All Courses Screen** would contain all of the other courses the teacher was using the tool for and would allow the teachers to archive, copy, delete, or create a new course
- **My Account Screen** functions as a settings tab where a teacher can update information about themselves, such as their teacher name or email
- **Student Screen** would allow for a teacher to click on a students name from the gradebook and it would open their own profile where a teacher could see a specific students progress

Major differences between the interviewees design decisions included (1) what was on the dashboard screen upon logging in and (2) how the menu was found within the tool. These two differences are what called for the first evaluation since both the landing page and menu are so critical for the tool. For the landing page, some of the interviewees wanted the tool to ask which course the teacher wanted to be signed into each time they logged in whereas other wanted the tool to remember what they were signed into last and open to this class's dashboard. For the menu, some of the interviewees wanted a menu that would drop down from the top while others wanted a menu which was always open in the left margin of the screen.

4.1.2.4 Prototype Stage

During the prototype stage, I created new paper prototypes (4.1, 4.2, and 4.3) based on the interviewees ideas and suggestions. These new paper prototypes were then evaluated which I discuss in the next section.

4.1.2.5 Evaluate stage

My goal for the evaluate stage is to decide which of the two flows of the tool I should move forward with into implementation. After I redesigned the paper prototypes of the two popular flows with the changes to the screens I asked few of the interviewees to evaluate the two options of the prototype and the following questions.

- What will I, a teacher, use this tool for most?
- What do I want to see upon logging in?
- Which menu is least tedious to use?
- Are there any other tasks this tool could do for a teacher?

Only four were able to assist with evaluating the prototypes due to time constraints and availability.

For the landing screen, one of the interviewees mentioned that “if [the teacher] walked away from their computer and it went to sleep, when they logged back in, it would be additional clicks to

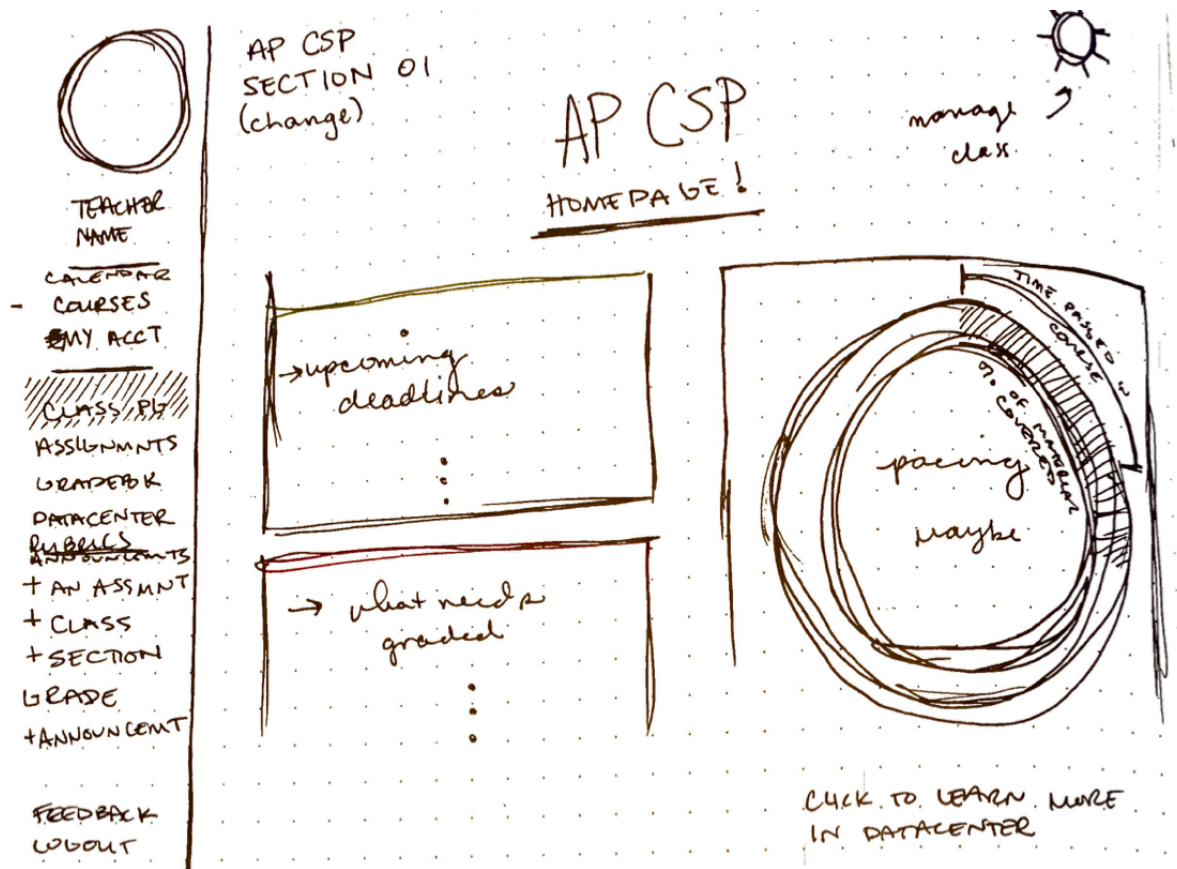


Figure 4.1 Decided dashboard (landing page) elements

choose which class" they wanted to work in. However, another interviewee stated they would not like to have to go through and choose a course each time therefore we decided we would have to come up with another way to change courses on each of the screens.

The majority of the interviewees preferred the left margin menu. In fact, one interviewee mentioned "using the left margin menu and leaving it open from screen to screen reduced the number of links they needed to click" to navigate the interface. Additionally, another interviewee pointed out using the left margin menu may reduce the confusion of using a new tool since access to many of the different screens will be accessible each of the screens.

As an off-topic thought, one of the interviewees mentioned that "if it was [the teacher's] first time logging in, they could use a tutorial which would help them set up a class". These votes and suggestions led to the final prototype design of the tool prior to the start of implementation, described in the next chapter.

4.2 Final Prototype Design, Flow and Decisions based on Sprint

One of the important decisions was whether or not students would need to have an account. Based on the user feedback here, GradeSnap is organized so both teachers and students can log in. By

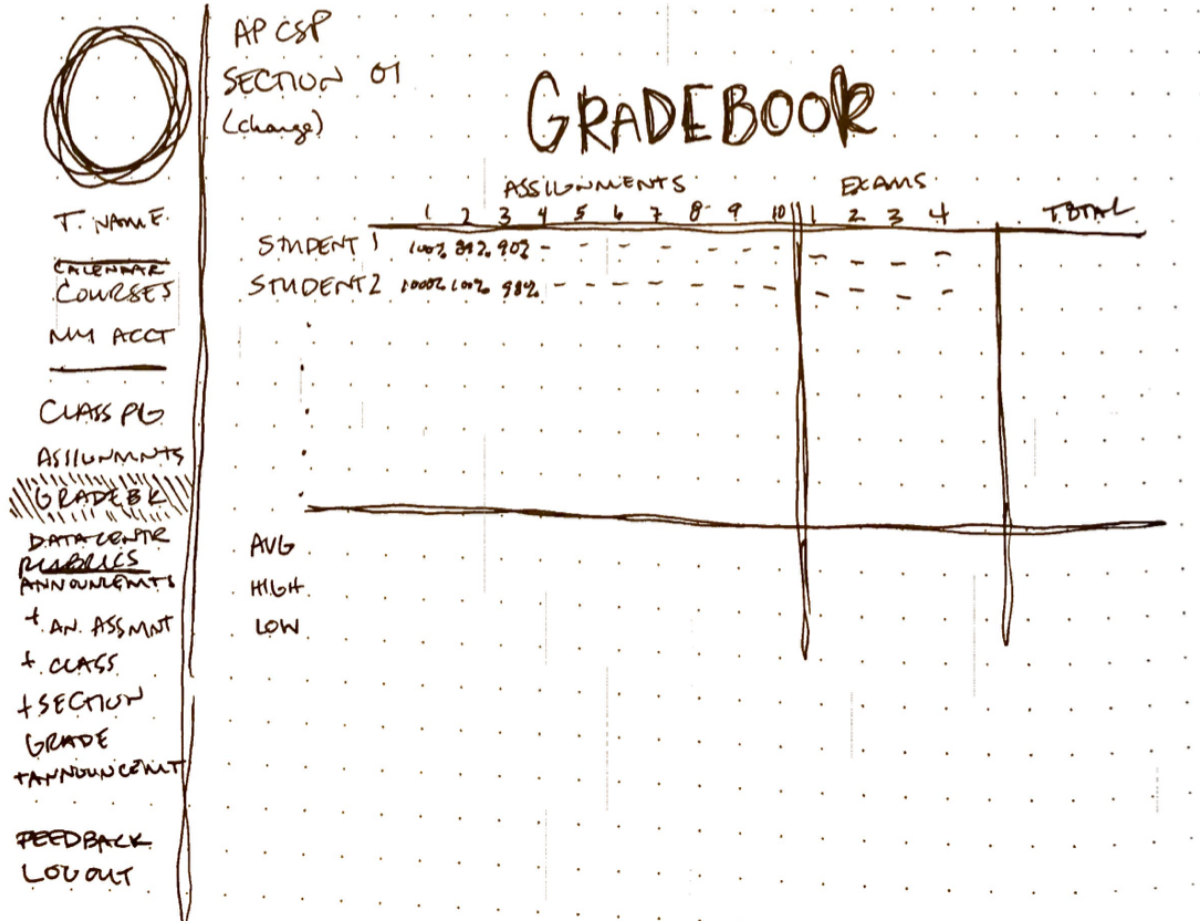


Figure 4.2 Gradebook where teachers can review students' grades

doing so, instead of sending finished projects to teachers, students only need to interact within the GradeSnap site for their block-based assignments as the tool hosts the student's instructions and has a built-in Snap! interface where students can save and submit their work instead of sharing or exporting.

Another critical decision from this sprint was the design of the grading interface (Figure 3). This design is similar to the Gradescope design where the student's work is the majority of the interface and the grading rubric lies in the right margin[Sin17]. The grade is determined by the rubric items a teacher ticks; however, the teacher is able to overwrite the grade (or edit it entirely) if they think it is not correct (too harsh or too kind).

4.3 Conclusion

In this chapter I have implemented a UX sprint to determine the initial design for GradeSnap (GS), investigating research question **RQ2a-GS design**: What features do teachers want and need in an online grading tool for programming projects? Through an iterative paper-prototyping and task-

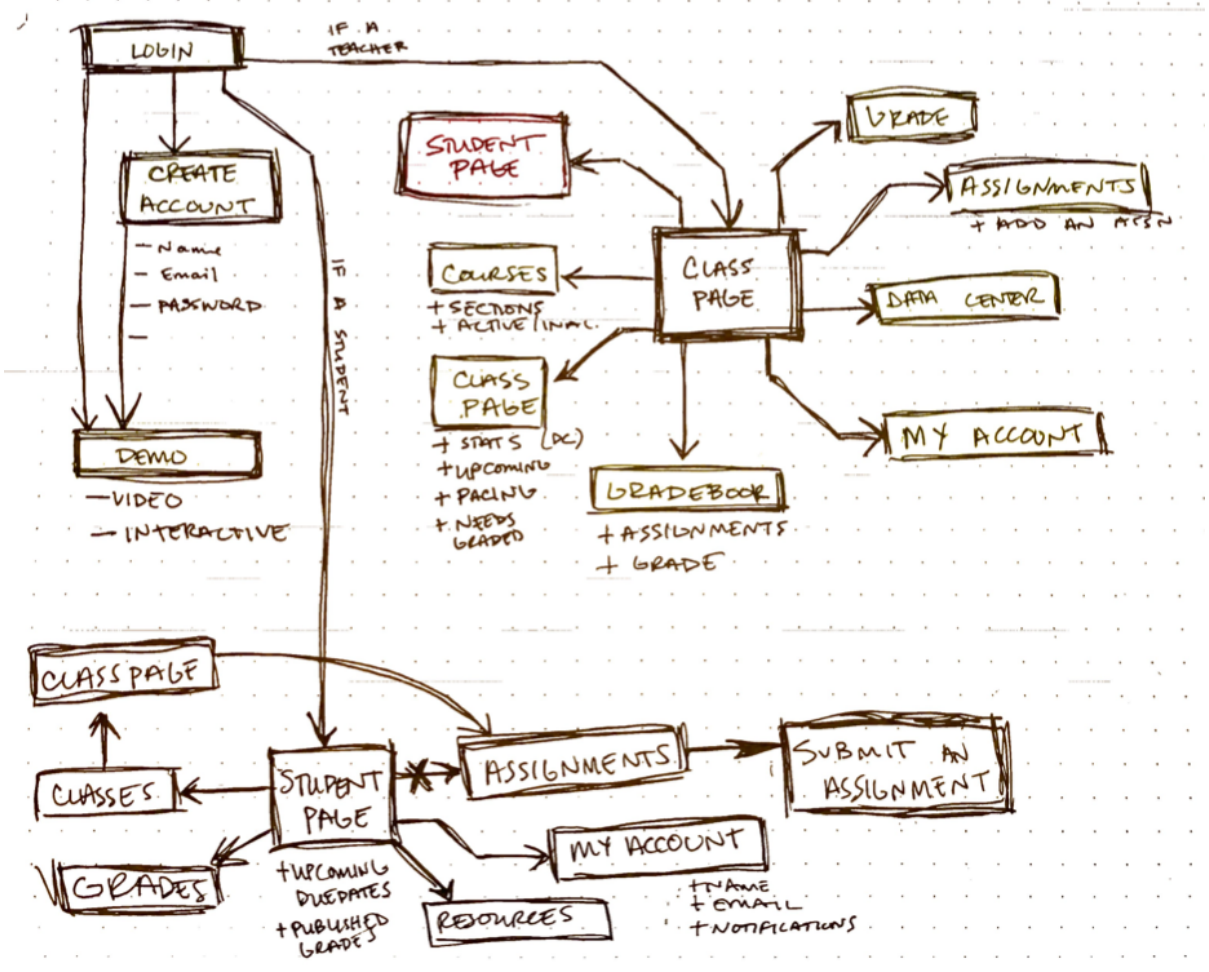


Figure 4.3 Decided flow based on interviewees and participants preferences

focused interview process, interviewees emphasized the need to have a left menu for quick access to features that would also help people know what features were available. Furthermore, interviewees sought to minimize the amount of time needed to navigate the system, and indicated that most of the time spent in the tool would be on the grading itself, though they also expressed interest in managing student scores in a gradebook. The design decisions based on this user study and suggestions by interviewees were used to create a digital prototype for GradeSnap, described in the next chapter.

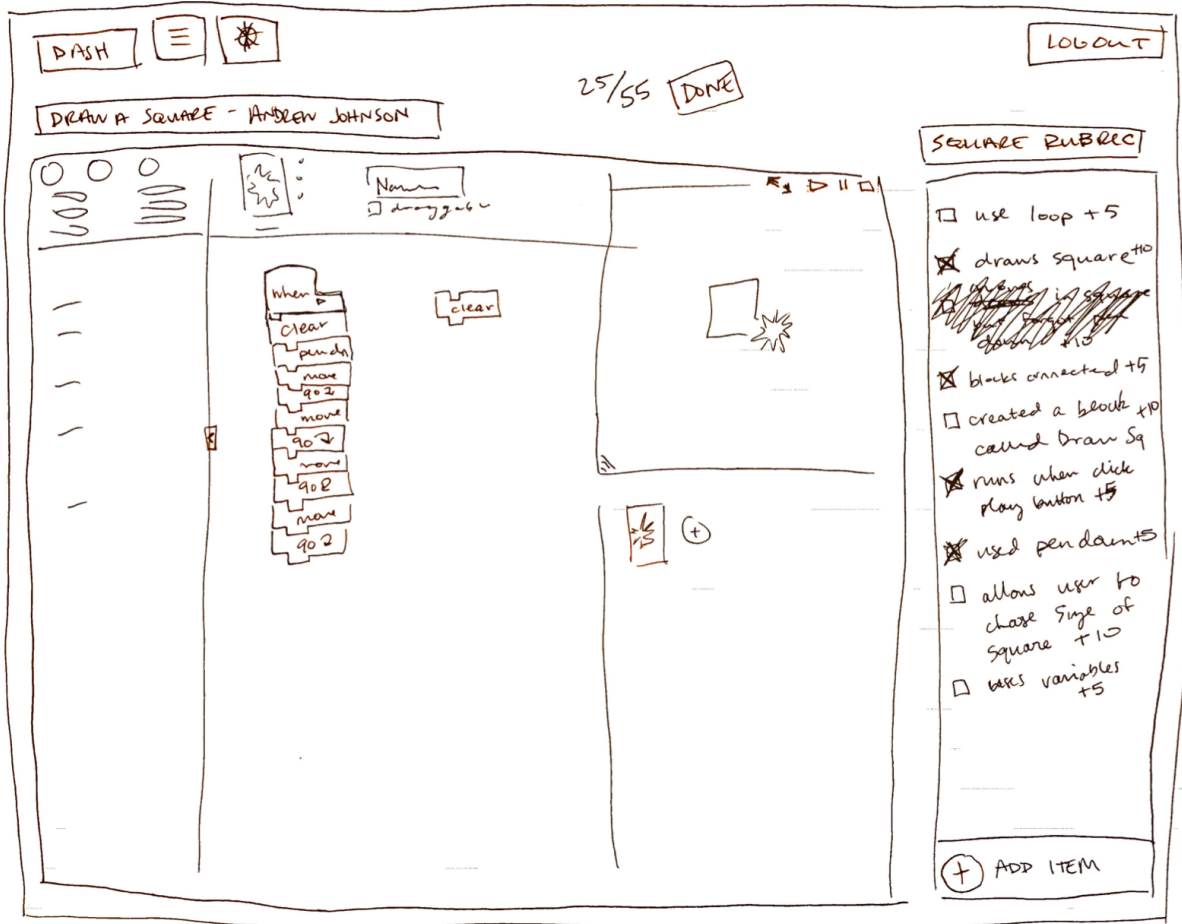


Figure 4.4 Grading interface based on Gradescope's grading interface

CHAPTER

5

INITIAL GRADES NAP PROTOTYPE

In this chapter, we investigate **RQ2b-GS prototype**: How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change with tutorial usage?

GradeSnap was made using Angular, JavaScript, HTML5, and Node.js using a model-view-controller software design pattern. In the first round of development, the team focused on developing a supportive foundation (establishing a backend to support both teacher and student accounts) and then high priority items, listed below:

- website page,
- course page,
- gradebook,
- assignment page,
- rubric page (5.2), and
- grading interface (7.4)

At the end of the first iteration of development, teachers could login, create a course, create an assignment, create or edit a rubric, and grade their student's submitted work. Students could login, find their course (if they have more than one), view an assignment, program an assignment, and save or submit their assignment to their teacher. By allowing students to save their work avoids the problem of younger students not having an email address to login to Snap! to save their work. The

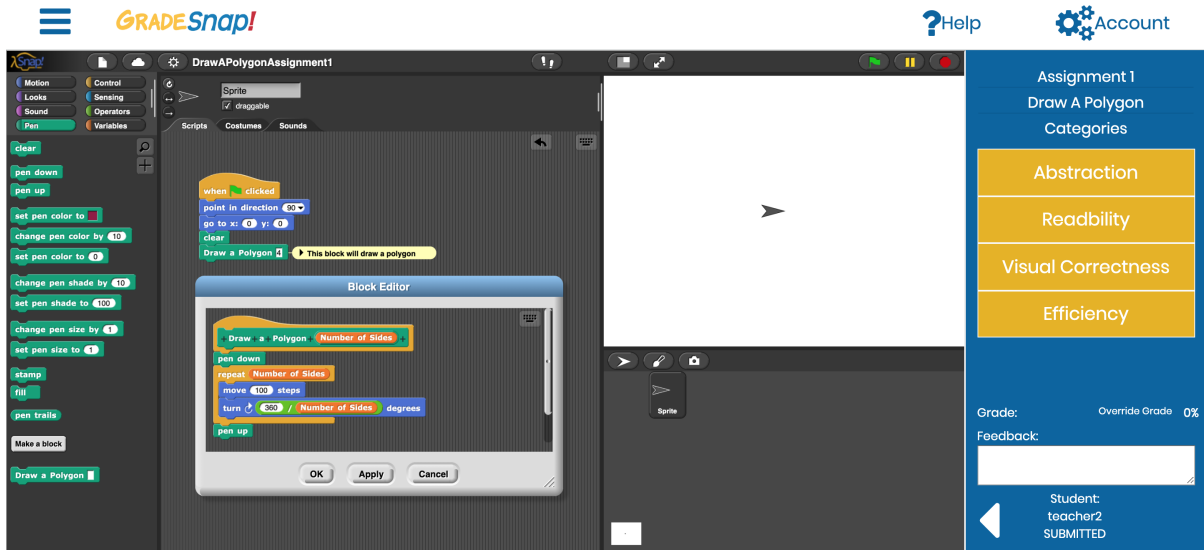


Figure 5.1 Grading interface where teachers can see their rubric and student’s code simultaneously

next two subsections discuss the next stage of implementation containing features which some of the block-based graders declared important or useful during the UX sprint described in the previous chapter.

5.1 StatSnap

In order to supply teachers with more support for their classroom, we added StatSnap, a tool within GradeSnap which synthesizes student grades submitted work based on rubric marks and reports to the teacher findings about CS skill usage.

The feature StatSnap was added in the summer of 2019 and is still in the development process. Through paper prototyping and participatory design, the idea of this tool was developed and streamlined, resulting in a final prototype which consisted of two sections: (1) grade distribution to provide a high-level overview of students progress and (2) skill usage pages to provide teachers with information about specific CS skills and how often the students are utilizing them. In a preliminary study, discussed later in section 4, teacher feedback suggested that information on specific skills should come in two forms: graphical representation and verbal, smart suggestions. Graphs of the collected data would give teachers a quantitative representation of student skill progression which would both increase teacher understanding of student needs and provide teachers with valuable statistics that could be used in reports. Verbal, smart suggestions would provide teachers with a more qualitative representation, specifying what the graphs mean in terms of student needs, and suggesting to teachers whether they should re-teach, review, or move on from specific skills, with links to further activities which could be assigned to students to promote learning in those skills. These smart suggestions would also be phrased in a way to encourage teachers with positive wording and supportive phrases.

Draw A Polygon

Pre-made rubric for Snap Draw a Polygon program

Edit Rubric Add Category

Abstraction		Edit	Delete
Readability		Edit	Delete
Assess use of comments and uses conventional names for variables and/or custom blocks.			
Point Range:	Min: 1	Max: 4	Scale: 1
Points	Reasoning		
1	The names of variables or functions do not describe what the function does/variables represent.		
2	Some names are appropriate for the functions/variables OR there is one comment or so.		
3	Some names are appropriate for the functions/variables AND there is some commenting.		
4	Each function/variable has an appropriate name. There are also comments explaining parts of the code.		

Figure 5.2 Rubric creation interface.

For StatSnap, data is automatically collected from the backend code of teacher-marked rubrics and collated into graphs and smart suggestions. “Smart suggestions” are purely based on averages. Future work on StatSnap could more closely analyze the graph and give teachers a more in-depth understanding of the trends in their students’ skill usage. To further this support, future versions of StatSnap smart suggestions could link to a database populated with Snap! exercises designed to focus on deepening student understanding of specific Snap! skills.

5.2 Interactive Tutorial

We developed an interactive tutorial to help teachers learn to use GradeSnap. Teachers can choose to complete the whole tutorial, a specific part of the tutorial or none of the tutorial. If they choose to not use the tutorial when first signing up, the system tells the teacher where to find tutorials for future help. Additionally, if a teacher does complete the tutorial, by the end they will have created their first course, adding students and an assignment. After building the tutorial, GradeSnap was evaluated by 33 teachers.

Before we can evaluate if the addition of an interactive tutorial would improve the user experience, we first had to implement it into GradeSnap in an effective manner. With teachers as the focus audience, we sought to build an interactive tutorial that will best serve their learning styles. Instruction for teachers attending an AP CS Principles Professional Development was more effective when preparatory work was small and specified and when hands-on tasks were given. In the implementation of the interactive tutorial for GradeSnap we focused on including short, step by

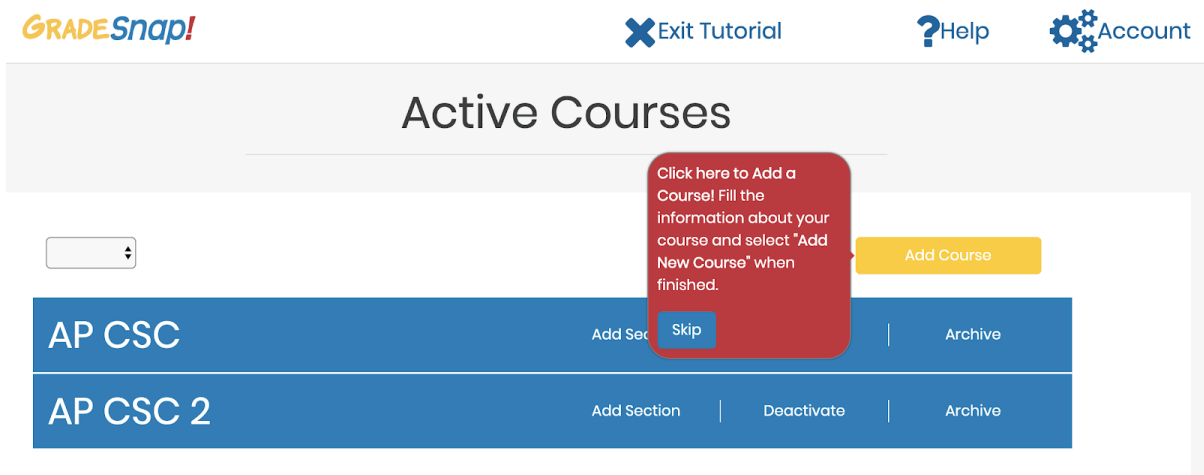


Figure 5.3 Example of the pop-up interactive tutorial

step instructions that will not just show the user how to complete a task, but will require the user to complete the steps thus setting up the user's courses and assignments.

Additionally, there are established practices that make an effective interactive tutorial such as modeling the activity, supporting the learner as they complete the task themselves, and reducing the amount of support over time. Following these ideas, we added pop-ups (see 5.3) with short messages and arrows pointing the user to the next step in setting up their classes and assignments. Furthermore, we created a help dashboard (see 5.4) that will allow the user to refresh their memory as to how to complete a given task. This dashboard serves as a long-term support system for the user. In the future, more short interactive tutorials will be added to this support system for GradeSnap.

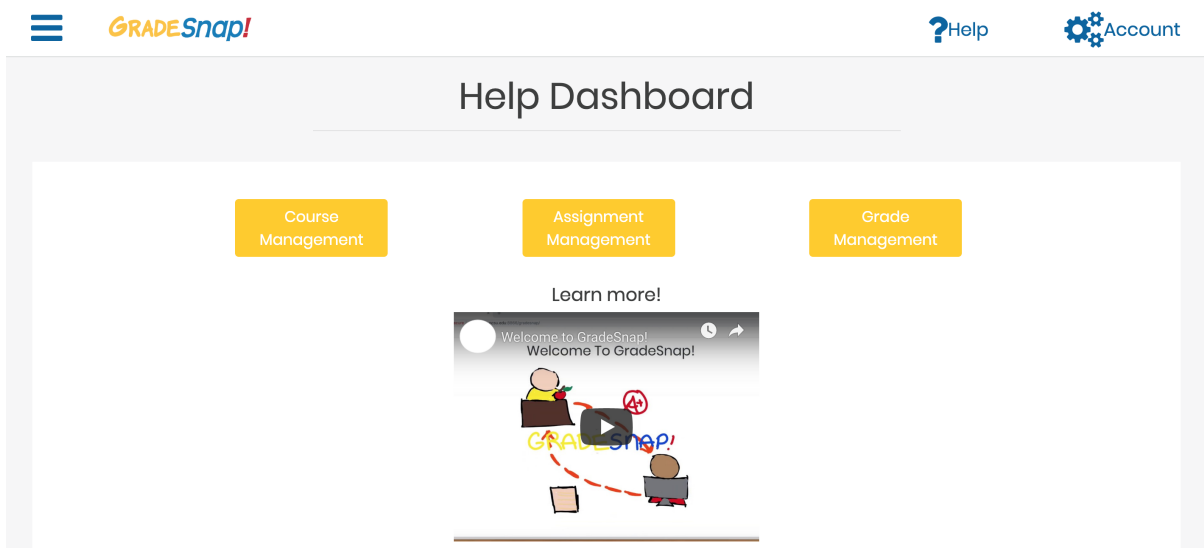


Figure 5.4 Help dashboard where teachers can find assistance with common tasks within GradeSnap

While the GradeSnap tool aims to improve the grading process of block-based programming assignments, this will only be the outcome if the tool is adopted and understood by users. Introducing a new tool to the classroom must be a careful process and adequate technical support must be provided to ease the stress of adopting new technology. Maintaining the participatory design process, we sought to determine if the addition of an interactive tutorial to GradeSnap would improve the user experience and thus increase the usage of the tool. An interactive tutorial is defined as a structured collection of navigable web pages designed to teach a specific learning outcome. In this case, that desired learning outcome is knowledge and comfort using GradeSnap. According to the Usability Engineering Approach to Acceptance, for a system to be acceptable by the user, it must meet the requirements of utility, usability, and cost. The development of an interactive tutorial for GradeSnap focuses on the usability, which requires that the tool can be used in an effective, efficient, and satisfactory manner.

5.3 First Usability Study

The 33 participants within this study include teachers from 2 different computing teacher Professional Developments and 3 were from an online forum of teachers who use block-based languages within their classrooms. Of the 33 participants, 18 participated while at a BJC AP CSP teacher professional development workshop and 12 participated while at a workshop on how to integrate computing into STEM middle and high school courses. Each of these studies used the same study design.

Our research question RQ2b is: How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change with tutorial usage?. To answer RQ2b, we collected data on (1) participant satisfaction, (2) participant's perception of the difficulty of using the tool, (3) participant likelihood of using the tool, and (4) time to complete a set of tasks. Data was collected from a pool of 30 K-12 teachers during a pilot study, who completed both the pre and post survey and a second remote study in which 3 additional teachers participated with the same surveys. The teachers were given the opportunity to use GradeSnap and give feedback while attending either a BJC Train the Trainers session, Infusing Computing professional development, or the Beauty and Joy of Computing teacher community forum in summer 2019. 18 of the participants were from the Training the Trainers event while 12 were from the Infusing Computing professional development and 3 were from the BJC online forum. In general, the participants from the BJC Train the Trainer event were more experienced in teaching computer science.

Each participant was given a pre-survey in which they answered questions about their background in teaching and computer programming. Additionally, they answered questions about their confidence with grading block-based programming assignments and using online grading tools. Additionally, all participants were given the same list of tasks starting with logging into GradeSnap, then setting up a course, adding students, creating a rubric, and grading an assignment. To compare the experience of users with and with-out the interactive tutorial, we used A/B testing. Participants

were randomly blocked into two groups: using the interactive tutorial and not using the interactive tutorial. The blocks were separated physically in the session so as to not confuse participants not doing the same thing. 18 participants completed the tasks while using the tutorial (54.54%) and 15 completed the tasks without the tutorial (45.45%). On the post-survey, participants answered different questions depending on if they used the interactive tutorial, or not. For example, participants who did not complete the tutorial were asked if a tutorial would have improved their experience using GradeSnap, while participants using the tutorial were asked if it made their experience easier. Additionally, all participants were asked a series of Likert scale questions to determine if the interactive tutorial made for a more enjoyable user experience. The participants read a statement and respond with one of the following options: Strongly Disagree, Disagree, Neutral, Agree, or Strongly Agree.

Finally, participants were given the opportunity to provide additional feedback on their experience with open ended questions. These questions allowed users to elaborate on what they liked or disliked about GradeSnap and the interactive tutorial. The results from the open-ended questions help us to continue the participatory design as well as understand some of the reasoning behind the participant's other responses.

In the second, remote study, logging was added to track how long it took each participant to complete the tasks. As this study was not completed by the research team, we created specific instructions for the instructors at the BJC PD to give to the participants. The logging was used to determine any differences in completion time between those using the interactive tutorial and those not.

To evaluate the data from both studies, we used the Mann-Whitney U test to determine if the two independent samples, the satisfaction levels of users with an interactive tutorial and those without, were selected from populations that are not equal thus indicating if the differences in the satisfaction level is statistically significant.

5.3.1 User Satisfaction

In order to gauge if the interactive tutorial alone led to a more enjoyable user experience, we compared the results of the users' self reported satisfaction with GradeSnap. Participants were asked after the completion of the tasks to rate their level of satisfaction with one of the following options: Very Dissatisfied, Dissatisfied, Neutral, Satisfied, or Very Satisfied. With a numerical score of 1 being equivalent to Very Dissatisfied and a score of 5 being equivalent to Very Satisfied, the average score of all participants was 3.66. Participants using the interactive tutorial had an average score of 3.84 while participants who did not use the interactive tutorial reported 3.43 on average, as shown in 5.1. In order to determine if the differences in sample means is significant, we completed a Mann-Whitney U test for statistical significance. The p-value associated with the test is 0.050, which indicates that with $\alpha = 0.05$, the differences in means is statistically significant (shown in 5.1).

Table 5.1 Satisfaction results comparing Tutorial Users and Non-Tutorial Users

Average Satisfaction for Tutorial Users	3.84
Average Satisfaction for Non-Tutorial Users	3.43
Overall Average Satisfaction	3.66
p-value	0.05
SD	0.645

5.3.2 Perceived Difficulty

Users who used the interactive tutorial responded “ Yes” 72.2%of the time when asked if the interactive tutorial made using GradeSnap easier. While this number is not extremely high, this could be due to the fact that the users responding to this question did not experience the tool without a tutorial. Users who did not use the tutorial respond “Yes” 76.9% of the time when asked if having access to an interactive tutorial make using GradeSnap easier. Additionally, this could just be a factor of the user’s own experience with using software tools in the classroom. As one participant commented, “I was able to figure out what I was doing by clicking around. . . some folks are less tech-savvy and really feel they need a tutorial”. Another explanation for the difference in the proportion of those who felt the tutorial made it easier and those who felt a tutorial would make it easier could be that the interactive tutorial may need to be improved.

5.3.3 Likelihood of Adoption

As shown in 5.5, those who use the interactive tutorial only had a slightly higher rating of their likelihood to use GradeSnap (average of 4.05 to 3.93). Using the Mann-Whitney U Test, the p-Value is 0.70. This shows that the differences in means is not statistically significant. Part of this could be that the demo was given while GradeSnap was in alpha stages of development so many participants may have voted that they will use the tool assuming the suggestions that they made were put into the system.

5.3.4 Limitations

The biggest limitation to the study is that more data is required to determine if the addition of the interactive tutorial significantly improves user efficiency with GradeSnap. Due to the final study being completed remotely, the participation rate was much lower. Only 3 teachers participated and while out of the 3 participants, those using the tutorial completed the set of tasks over a minute faster on average (11.15 minutes to 12.18 minutes), the results are insignificant and more data is required. This provides a potential avenue for further study.

Another limitation of this study is that early on, it was not clear to some participants if they were supposed to use the tutorial or not. To address this issue, we made the instructions more explicit, added a question to the survey asking “Did you complete the tutorial in its entirety?”, and eventually added logging to indicate whether a user started the tutorial and when they finished it. Fortunately,

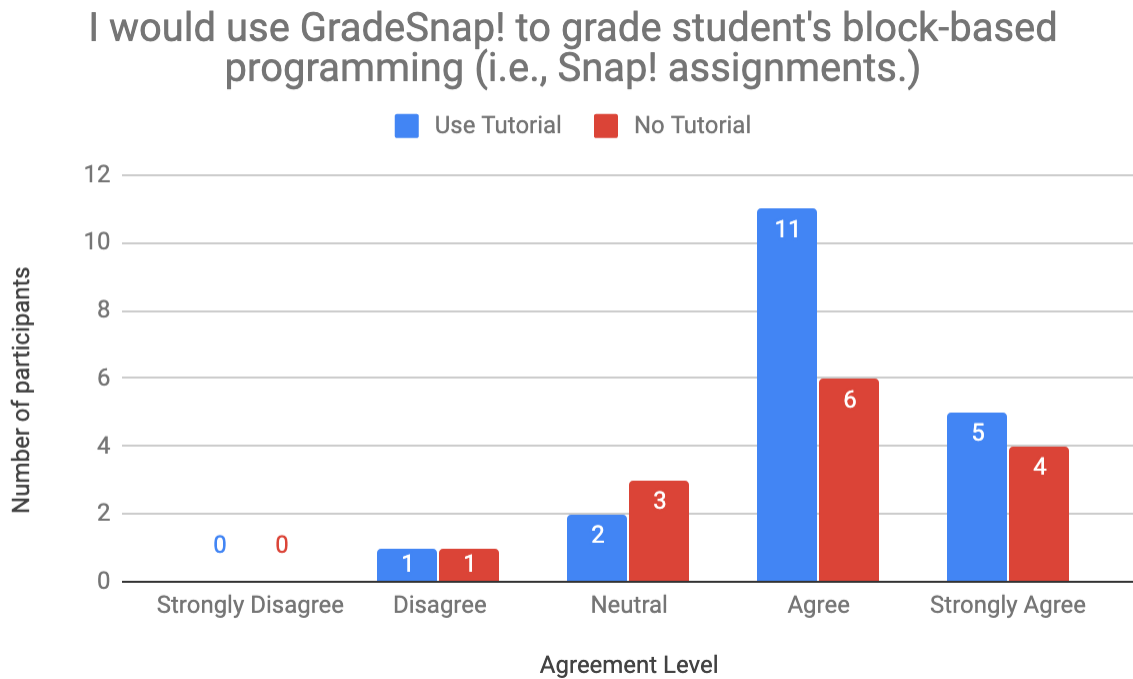


Figure 5.5 Likelihood Adoption of GradeSnap

in the cases where there was not yet logging to verify the completion of the tutorial, our team was in the room and monitoring the steps of the participants.

5.4 Conclusion

In this study, we investigated RQ2b to determine whether our initial prototype for GradeSnap would be useful for teachers who plan to teach block-based programming, and investigated the importance of including a tutorial for the tool. Based on our interview and survey results with 33 teachers, we concluded that teachers were overall satisfied with the prototype, but were significantly more satisfied with it when they experienced the tutorial. A majority of teachers using the tutorial found that the tutorial helped make GradeSnap easier to use. All teachers, regardless of tutorial use, were likely to adopt GradeSnap. These findings suggest that the design of GradeSnap generally met teacher expectations and needs for a tool for grading. In the next chapter, we investigated teacher grading practices without GradeSnap, to serve as a baseline for comparison with GradeSnap usage for grading.

CHAPTER

6

GRADING BJC PROGRAMMING LABS

In this chapter, I investigate **RQ2c-Grading**: How do teachers grade programming projects with and without rubrics, but without tool support? I conduct a think-aloud study to: (1) learn how teachers are currently grading, (2) determine what differences a rubric makes in the teachers' grading process, and (3) extract pain points and grading goals for a teacher when they are grading. In this study, I have several detailed research questions: (1) How do grading outcomes (score/grade/percentage) change when using a method to keep track of grading criteria?, (2) How does grading with a tool (such as a rubric) impact a teacher's grading goal?, and (3) What is important for teachers when grading a block based language assignment?. I hypothesize that: H1) teachers will have higher grades when not using a rubric, and H2) teachers with more block-based language grading experience will create a more informative or useful rubric.

6.1 Related Literature

The study presented in this chapter will reference multiple sections from chapter 2, mostly including the Grading Programming Projects and Rubrics sections. Fitzgerald et al. expressed the need to share more about how they grade in order for researchers to determine the best way to help newer teachers learn how to grade text-based programming assignments [Fit13]. One method to disclose grading techniques for programming projects to both students and peers is by creating and sharing a rubric [Eug16]. Brookhart describes a rubric as a “[coherent] set of criteria...that includes descriptions of levels of performance quality” with a main purpose to assess performances, for example to assess student's work [Bro16]. Brookhart describes different kinds of rubrics, comparing Analytic to Holistic and General to Task-Specific. Analytic rubrics break down student work into subtasks, allowing

teachers to evaluate student performance of specific criteria individually, whereas, Holistic rubrics allow for a teacher to evaluate the all criteria of a student’s work at the same time [Bro16]. General rubrics can be used to evaluate multiple assignments with different instructions, grading on a set of less specific criteria, whereas, Task-specific rubrics are detailed and specific to an assignment [Bro16]. In addition to these rubric types and descriptors, Fluckiger shared Dietz’s introduction to a single point rubric, where “the single point rubric provides a single set of criteria” [Flu10; Die00]. Stegeman et al. investigated how to design a rubric for feedback on the quality in programming courses and disseminated their methods for developing a rubric to better support novice programming students [Ste16]. Catete saw the need to support teachers with rubrics who were preparing to teach the Beauty and Joy of Computing [Gar15; Mor14b] Advanced Placements Computer Science Principles course [Cat16]. She found that novice CS teachers can use rubrics, created for specific projects, to grade novice student block-based projects within Snap! [Boa17].

6.2 Participants and Pre-Study Actions

I recruited 4 teachers with a varying range of experience teaching computer science, specifically the Advanced Placement Computer Science Principles (APCSP) course, and grading block-based languages. Additionally, all of the participants use the Beauty and Joy of Computing curriculum in their APCSP course. The participants also have a range of confidence levels regarding grading block-based language assignments.

Before participating in the study, teachers will complete the consent form and a pre-survey which collects demographic information such as their experience with computing and grading block-based projects. With the pre-survey information, I will ensure I can select teachers with varying levels of experience. During the study, teachers will practice thinking aloud with a computational thinking activity, grading student programs with normal grading methods (first without a rubric and then with a rubric), and a post-grading activity interview. The grading and interview activities will occur during the scheduled and recorded zoom video call. In the next section I go into more explicit detail about the study design.

6.3 Study Design

I scheduled and recorded individual zoom interviews (video calls) for each participant based on their availability. During the zoom interview, first teachers learn how to think aloud by practicing with a computational thinking activity designed to get them to think aloud while looking at block-based code. The grading portion of the study is loosely based on the first iteration of Stegeman’s process for developing a rubric to assess the quality of programming to understand how teachers think grade block-based programming projects [Ste16]. The grading portion is split into 3 parts: (1) teachers grade the 6 provided submissions using their own grading methods, omitting a rubric if they typically use a rubric, (2) teachers create a rubric for the assignment they graded, and (3)

teachers grade the same 6 submissions with the rubric they created. Teachers are thinking aloud during the grading portion of the study. During the grading portion of the interviews, teachers think aloud about what is most important to them when grading assignments. Determined grades from each student assignment, were stored in a provided google spreadsheet. At the end, teachers answer interview questions about their grading goals, pain points, and confidence. They also reflect on the 2 different grading instances, first without a rubric and then with a rubric.

Table 6.1 The study outline consisted of these six parts.

Pre Survey	Collecting Demographic information (i.e. previous grading experience)
Computational Activity	Think Aloud Practice
Grade 6 Submissions without Rubric	Thinking Aloud - Understanding Grading
Create Rubric for Assignment	Thinking Aloud - Understanding Rubric Creation
Grade 6 Submissions with Rubric	Thinking Aloud - Understanding Grading
Post-Survey - Teacher Perceptions	Collecting usability and usefulness, rubric's impact

During each zoom meeting, I will have 1-2 observers who are a part of the research team to take notes and observe actions the teacher takes which may not be found in the audio transcription.

6.3.1 Assignment Teachers are Grading

During this study, teachers will grade submissions for a provided assignment. The assignment is called Brick Wall where students should program their project to draw a brick wall. The assignment instructions give directional support, such as the blocks they may need in the assignment and the different custom blocks a student should define and use (draw brick, rowA, rowB, Draw a Brick Wall with () Rows, where the () is a parameter). The assignment also gives students visual assistance, such as a picture of the definition of draw brick, a picture of what rowA and rowB should output, and a picture of what the final project should output.

The submissions are artifacts created by CSP high school students from a previous study [Cat16]. These submissions were graded by two university researchers with K-12 teaching experience and post-baccalaureate degrees in computer science. As part of that study, each submission's score was translated to the level of achievement, specifically, each submission is also marked as High-, Middle-, or Low-Achieving. In order to observe the teachers grade a variety of submissions, 6 submissions were chosen, 2 from each of the levels of achievement previously determined.

6.4 Data Collection and Analysis

Each of the subsections below describes one of the types of data collected during these studies, the motivation for collecting this data, and how the data was analyzed.

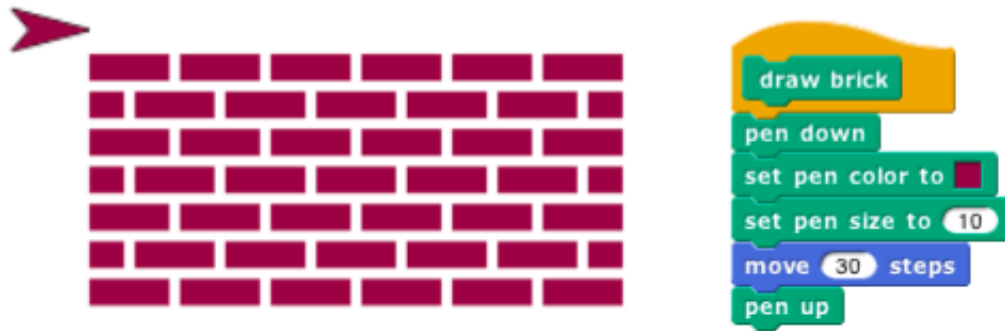


Figure 6.1 The final output of the Brick Wall assignment and an example of a defined custom block, provided in the instructions for the students.

6.4.1 Screen Recording and Audio Recording

All grading and interview parts of the study were recorded using Zoom, which provides automatic audio transcription. The research team reviewed and revised the automated audio transcripts when needed to ensure accurate audio transcription when analyzing the data. We analyzed audio transcripts using thematic analysis with four raters in multiple phases. Participant's screens were also recorded to help determine usability challenges, provide further observation into how a teacher grades, and provides another form of time collection.

6.4.2 Rubrics

During this study, participants created a rubric for grading assignments in the second grading session. Based on the participants' explanation of their rubric and the rubric design, I determined the type of rubric a teacher uses while grading these assignments. In order to compare participants' grading methods from both grading sessions, I also determined the participants' grading process during the first session by analyzing their grading methods and determining how they decided scores even though they did not create a rubric. I included information about participants' grading methods and rubric types in their respective case studies. I also compared their final rubric types with their grading goals. The 5 different types of rubric or grading processes I used are in chapter 2 and earlier in this chapter when I briefly reviewed relevant literature for this study.

6.4.3 Thematic Analysis Process

For the thematic analysis process we used, there were 5 total phases, which included 1 data reviewing phase, 2 coding phases, 1 meet and discuss codes, and the final phase where researchers group codes into themes. The data was coded by 4 researchers.

Table 6.2 Each researcher coded 2 different participant's think aloud interview. They coded their first assigned interview (indicated with a 1), then the research met as a group, then they coded their second assigned interview (indicated with a 2).

	Alice	Beth	Christina	Deborah
Researcher 1		1		2
Researcher 2			2	1
Researcher 3	2		1	
Researcher 4	1	2		

1. Phase 1 - Review Data and Initial Coding

- Researchers meet to discuss thematic analysis process.
- Researchers are assigned the two interviews they will code, indicated by Table WW.
- Researchers spend 20 minutes reviewing their first interview and creating initial codes.
- Researchers discuss code they found, add descriptions, and ask questions.

2. Phase 2 - Coding First Interview

- Researchers review all current codes before they finish coding their first interview.
- Each researcher codes their first interview.

3. Phase 3 - Meet and Discuss Codes

- Researchers meet and share all codes they've added to the codebook
- Researchers acknowledge duplicate codes and remove if applicable
- Researchers ask questions if needed

4. Phase 4 - Coding Second Interview

- Researchers review all current codes before they finish coding their second interview.
- Each researcher codes their second interview.

5. Phase 5 - Grouping

- Researchers meet and create some initial groups
- Researchers split into 2 pairs and group the codes during the meeting
- Researchers return to a full group and review the codes with which are grouped into no themes or multiple themes for discussion; Then researchers refine the groups.

6.5 Results

First, I will present and define the themes found from tagging the think aloud grading sessions and interviews. Then, I will provide case studies for each of the participants. The case studies will present many of the themes, therefore, themes will be emphasized in case studies and discussed in the discussion sections.

6.5.1 Themes

Since the participants' time during the think aloud interview was while grading, their tags and themes are mostly about grading. The four themes include: (1) Feedback is Important for Students and Teachers; (2) Other Grading Goals; (3) Pain Points; and (4) Teachers Want More Support.

6.5.2 Teacher Cases

These case studies introduce the participant's teaching and grading experience and explore details about their grading think aloud sessions. Additionally, each case study contains information about their grading methods, grading goals, and the rubric they created. The four participants' names include: Alice, Beth, Christina, and Deborah.

6.5.2.1 Alice

Alice has taught computer science for the past 3-5 years and has integrated computing into her classes for 8 or more years. She is not **confident with grading** block-based language programs, though she has been assessing block-based language projects for 8 or more years. Alice uses a **spreadsheet to grade** projects. She wants her grading to be in more context of learning objectives and wants more clarity when she is grading.

Alice was very nervous to grade assignments and for others to see her grading since she felt her grading is very different compared to how others grade or how she was supposed to grade. Alice believes she has "crazy guidelines" when she grades and shared that sometimes she goes down a rabbit hole during grading code, comparing submissions from student to student, and looking for consistency between assignments. Alice typically has students turn in supplemental course work for each assignment, such as a journal entry, because it makes her anxious to grade code. Alice does not believe grading code in order to keep her class inclusive since students enter the course with a range of experience.

During the first grading portion, Alice shared her thought process about grading these submissions and how it compared to how she normally would grade for her classroom. The first desire Alice shared was that she wants to be able to grade the student code quickly in a flow, therefore she downloaded all xmls at once and opened them in different snap tabs. She explained that she needs speed when grading because of the large quantity of students she teaches. After she prepared her grading environment, she looked through all of the student submissions, looking for specific

code elements quickly, then focused on each program's visual output. She compared the student's submission looking for consistent good or bad elements, such as if they used a loop in their code. As Alice looked through all the submissions, she shared ways that the instruction the students received could be improved if all the students missed the criteria from the assignment (referring to the rounded bricks).

When Alice started assigning grades, it seemed she was empathizing with the students to understand their thought process or to discern how much they understood of the assignment. She expressed grading without any explanation from code comments or supplemental information, such as a journal entry, makes it hard for her to understand what they comprehend. Alice would change a student's code occasionally to understand the student's mistake. When Alice approached lower achieving submissions, she would look back at the assignment instructions to essentially 'scrape for points' for the student. All 6 submissions achieved grades between (85-96) [90, 90, 96, 86-89, 90, 96]. Alice did not provide comments, feedback, or written reasoning for why she gave the grades to each student, however, she did verbally compliment students' work when they did something she liked.

The following are explanations for typical grades she gives in her school:

- 96 unless perfect to signify there is always room to grow
- 90 means it was a good effort
- 85 if you really need some help
- 70 ..really? You turned in that?

When asked to create a rubric, Alice created a Single-Point rubric, similar to a checklist, to keep track of student success, that she updated while grading. She shared that she typically makes a checklist before she grades an assignment with the goal of checking for students' comprehension. Alice shared that the journal plays a large part of the grade for programming assignments to understand what the student does or does not grasp. Alice shared that she is looking for efficiency, possibly because she believes this is an expectation from us. Her final single-point rubric focuses on the visual output of the program and the 4 custom blocks students had to make from the assignment instructions.

Rubric Elements:

- One Brick Abstraction [0,1; no explanations]
- Row A/B Abstraction [0,1; no explanations]
- Procedure with Input [0,1; no explanations]
- Rows line up? [0,1; no explanations]

During the second grading portion, Alice added her single-point rubric to her grading spreadsheet. She graded much faster than the first time and thought a lot less about how a student was trying to approach the problem, or program their solution. She didn't reference the assignment instruction and instead referenced her single-point rubric when grading. Alice was pickier than her rubric would allow her to be. For example, a few times Alice mentioned how the visual space between elements in the program did not line up with her expectations, but since her rubric did not take this into consideration, she could not take this into account when grading. When using her single-point rubric, Alice didn't total anything up to calculate the grade. Instead, she used the rubric to view what the students did correctly and then gave a percentage equivalent to the grade she would give in her class. All 6 submissions achieved grades between (89-98) [89, 96, 90, 89, 89, 98]. Again, Alice did not provide comments, feedback, or written reasoning for why she gave the grades to each student, though she verbally communicated grade reasonings to the research team as she thought aloud.

Table 6.3 Alice's grades for the submissions only had a change of over 2 percentage points for 3 submissions; Student 2, 3, and 4 between the two grading sessions. The submission level refers to the achievement of the submission.

	Grading (no rubric)	Grading (with rubric)	Submission Level
Student 1	90?	89	Low
Student 2	90?	96	Middle
Student 3	96	90	High
Student 4	86-89	96	Low
Student 5	90?	89	Middle
Student 6	96	98	High

At the end of the grading portions, she viewed the two sets of grades she gave for the student submissions side by side to see how they compared. Alice was surprised by the differences in the students scores with and without the single-point rubric. The differences between students 2, 3, and 4, caused her to go back to the students' projects, where she found some mistakes she made while using her rubric. Alice believes she lost the big picture of grading with her rubric. Sometimes Alice would get distracted when checking off the rubric about how she doesn't like grading and would make mistakes when grading. Alice concluded that the first round of grading was more important to her than the second round of grading sessions. Her **grading goal for grading session one** was more similar to her normal grading goals, which is to find something her students are passionate about and are confident in. The **grading goal for grading session two** was to determine what programming elements students were missing from their assignments. She no longer felt like she was trying to understand the student point of view, but only checking to see if they implemented something correctly. Alice's type of grading changed from the first session to the second session. In session one, Alice executed students' code and determined the grade mostly from the output that

happened, therefore her grading method was **holistic** (grading the project as one) and **task-specific** (specific to the assignment). In session two, Alice used her single-point rubric to confirm or deny if the submissions had specific elements for the assignments, therefore her grading method was **single point** and **task specific**.

Alice's other **normal grading goals** include (1) helping middle-achieving students leave the class better than they arrived and (2) understanding how to adjust her teaching due to common misconceptions among her class. Alice compared her **normal grading process** to her grading process during grading session two and reflected on what would happen if she gave these grades to students and what would happen at her school. Because of the students at her school, if she gives a grade below 90, it would most likely result in a conversation with the student and their parents. This is why Alice uses the students' journals as supplemental to justify their grades and show them misconceptions to communicate better with the students. When asked about **pain points**, Alice answered the large number of accurate solutions, and the complexity of those solutions, makes it difficult for her to evaluate if a student's code is considered "correct".

6.5.2.2 Beth

Beth is currently teaching her 3rd year of Advanced Placement Computer Science A (APCSA), using Java, and her 1st year of APCSP, using BJC with Snap!. She is **slightly confident** with grading block-based language programs. Beth **creates her own rubrics** for grading these projects. She likes using autograders, though there are none for Snap!. Beth wants to change her grading habits by creating a rubric before she starts grading instead of while she is grading.

Beth came to the grading research session after just having finished grading for one of the classes she teaches. Beth did not seem stressed about needing to grade, and seemed ready to participate in the research session about grading. Before sending the submission folder to Beth, the researcher thoroughly reviewed the assignment instructions clearly with her because she was the only teacher that had not taught and/or graded the assignment. Beth shared that she remembered the assignment from the curriculum and found it boring so she didn't assign it to her class. That said, she was happy grading student submissions on the assignment.

To set up her grading environment, Beth downloaded all 6 project submissions, opened 6 tabs with the Snap environment, and loaded a different project in each Snap tab. Though Beth is new to teaching BJC, she seemed very technologically savvy. After getting her grading environment set up, Beth started looking at the students' submissions. For each submission, Beth would clean up the script space, clear the student's stage and execute the code from the student's custom blocks. First, she focused on reading the code, then on the visual output that it should produce. When reviewing students' code, she would look for common errors in their code. After Beth looked at each submission, Beth created a short list in a notes document on her computer to remember what criteria she would look for in student submissions. Then, Beth gave grades for each of the submissions, ranging from 50-95 [70, 90, 95, 50, 90, 90]. Beth also wrote notes for each submission,

telling the students what their code was missing. The following is the notes she created to keep track of while she was grading:

1. Needs DrawBrick block
2. Are lines aligned on the right
3. Needs rowA block
4. Needs rowB block
5. Other custom blocks other than drawBrick
6. Final block (Draw a Brick Wall with () Rows)
7. Does it work for even and odd rows
8. Will it draw the correct number of lines
9. Aesthetic (space between rows are thinner than bricks)
10. Do they give comments
11. Is their code well organized
12. Minimum 50 if shown effort

When asked to create a rubric, Beth shared that she uses a rubric everytime she grades. If she doesn't have a premade rubric from the curriculum or from previous years, she has to create one. Beth separated her notes from the first grading portion into 3 categories; Appearance/Naming, Abstraction, and Functionality. She weighted the two latter categories with double points compared to the first category. One of the elements from her notes that did not carry over to her rubric is the last note: Minimum 50 if shown effort.

Table 6.4 Beth's rubric has three categories, each with 4 category items (or level of correctness for each category). Two of her categories (Abstraction and Functionality) had double the weight as the other category (Appearance & Naming), signified by a (x2).

	Score 4	Score 3	Score 2	Score 1
Appearance & Naming	- Bricks are bigger than space between rows - Blocks are organized nicely - Blocks are named appropriately - Comments are added	Missing 1 item	Missing 2 items	Missing 3 items
Abstraction (x2)	- drawBrick is defined as in instructions - Other blocks are defined/used - The final block is defined as described	Missing 1 item	Missing 2 items	Missing 3 items
Functionality (x2)	- Works for even and odd rows - Draws the correct number of lines - rowA & rowB blocks have the same length	Missing 1 item	Missing 2 items	Missing 3 items

During the second grading portion, Beth was able to grade much faster. She reviewed each students' code once or twice, in slightly less time than the first grading portion since she already knew the code well. She graded each student using her rubric, resulting in [30, 85, 95, 25, 85, 90]. When she was grading students 1 and 4 (both low achieving students), she wanted to adjust her rubric to give the students more points, however, she didn't find the room to do so. Beth did not give students feedback during the second implementation, however when she compared the two iterations of grading, she stated that the comments she supplied during the first grading iteration applied to the second iteration as well.

Table 6.5 Beth's grades for the submissions only had a change of over 5 percentage points for 2 submissions; Student 1 and 4 between the two grading sessions. The submission level refers to the achievement of the submission.

	Grading (no rubric)	Grading (with rubric)	Submission Level
Student 1	70	30	Low
Student 2	90	85	Middle
Student 3	95	95	High
Student 4	50	25	Low
Student 5	90	85	Middle
Student 6	90	90	High

After she completed grading, Beth compared the two iterations of grading. Her grading goal stayed the same during both grading sessions, which was to see if the students understood and completed the assignment. In order to support her grading goal in the first grading iteration, she created the visual checklist to grade. Beth mentioned that her rubric during the second grading portion was more specific and most likely caused the grades to change even though her grading goal stayed the same. Beth's **type of grading** changed from the first session to the second session. In session one, Beth created a list of elements specifically for this assignment that she tested individually and supplied feedback for the students on how to improve, therefore her grading method was **single-point** (grading individual parts of the project, checking if the student had the element within their project) and **task-specific** (specific to the assignment). In session two, Beth took her list of elements from the first session and created and used her rubric with multiple levels, testing each of the custom blocks independently, therefore her grading method was **analytic** (grading individual parts of the project) and **task-specific**.

She shared her **normal grading goal** includes determining what is best for her classroom by reflecting on the class outcomes and averages. From the class outcomes, she is able to determine the topics her students did not comprehend fully, and would plan to incorporate them into her next lesson.

When asked about **pain points**, Beth answered finding a good rubric for grading block-based language programs. She shared for her **normal grading process**, she looks at a sample of submis-

sions, typically representing the class, to determine the types of errors students make and then creates a rubric. **When she uses other rubrics**, sometimes her grading leads to unexpected results because the errors her students were making were unaccounted for.

6.5.2.3 Christina

Christina is currently teaching her 4th year of APCSP, using BJC with Snap!. She is a Master Teacher within BJC, meaning she is experienced and teaches other teachers how to use BJC. She is **extremely confident** with grading block-based language programs and has been grading them for 3.5 years. Additionally, last year Christina was an Advanced Placement reader (grader) for the Create Performance Task (CPT). Christina **creates her own, or modifies her previously used, rubrics** before grading class projects. The rubrics she creates are similar to the AP CPT rubrics. She uses a special set up within her learning management system, Canvas, and does not need or want anything changed about her current grading process.

Christina came to the grading research session just having finished her remote class. Christina did not seem nervous grading the Brick Wall assignment, however, she mentioned that for her classroom, she does not usually grade this lab for correctness, she grades it formatively. She shared she looks to see if her students did the assignment, and “if they [got the] brick wall to work, then they get full credit”. She explained that giving comments or feedback is more important for this assignment since this lab is about learning and figuring out how to program. With that said, she was happy grading student submissions on the assignment.

To set up her grading environment, Christina first downloaded all 6 XMLs and then imported them into Snap, each in their own tab. She needed to be reminded how to import XMLs into Snap since this was different from how she normally collects Snap submissions in her classroom. For her normal classroom, she has students submit a shared Snap URL so she can open the link to their submissions in her Learning Management System (LMS).

As she started to grade, Christina mentioned that she would like to see all student-created custom blocks in the script space when opening their projects. She requires this in her classroom to make it easier to navigate student submissions. Christina reviews submissions one at a time and in order, focusing on both code and visual output. When Christina is focusing on grading code, but only runs the final custom block that would create the final output of a brick wall. She did not compare submissions to each other until she found an exemplar submission and then compared each submission to that submission to try and be consistent with grading. Instead of using any supplemental materials, such as a notes document, she referred to the assignment instructions when grading each submission, almost as if it was her grading criteria. Christina gave a grade range of 70-100 [80, 90, 100, 70, 100, 100] and feedback to 4 of the submissions (all either lower or middle achieving).

When asked to create a rubric, Christina created a rubric with 4 categories, each with 3 levels. Each category represents a different custom block the students have to make. For each possible score, she explains what the score means. For example, if a submission receives a score of 2 in any

category, it means that the submission “meets/exceeds expectation”; if a submission receives a score of 1 in any category, it means that the submission is “below expectations;” if a student receives a 0 in any category, it means that the submission is missing this element of the project.

Table 6.6 Christina’s rubric has 4 categories and 3 category items (levels of correctness). All of her categories have the same distribution of weight.

	Score 2: Meets/Exceeds Expectations	Score 1: Below Expectations	Score 0
Draw Brick block	Brick is a block, flat line ends have been selected and saved in file.	Brick has rounded ends	Missing
Row A block	Draws consistent number of bricks with even spacing; Does not include any motion other than moving to draw bricks and mortar	Inconsistent spacing; Includes unnecessary motion, other than drawing bricks and mortar	Missing
Row B block	Starts and ends with half or partial brick; Spacing is same as Row A spacing; Does not include any motion other than moving to draw bricks and mortar	Inconsistent spacing; Does not include any motion other than moving to draw bricks and mortar	Missing
Draw Brick Wall with () Rows block	Uses a loop to draw number of rows; Uses even/odd to determine how many times to repeat each row; draws correct number of rows; includes code to transition from row to row	Draws too many rows; or rows don’t begin or end at same x value; or does not include code to transition from row to row	Missing

After creating her rubric, Christina regraded the 6 submissions with her rubric. Immediately she shared to the researchers that she didn’t care if the custom blocks were named differently because that is just semantics and is not related to the program running correctly. As Christina was grading, she wanted to make changes to her rubric though she didn’t. She said that she didn’t make changes to her rubric because she feels it is unfair to the students since she typically gives the rubric to the students before she starts grading. Christina spent more time grading submissions when using her rubric because she wanted to make sure she was being consistent. Instead of only reading each custom block a student programmed, she also executed each custom block as well to determine its correctness. She assigns grades for the submissions while using the rubric [37.5, 87.5, 100, 50, 100, 100], though she doesn’t provide any comments or feedback.

When she finished grading, Christina shared that she is “a big fan of the rubric...because it helps [her] to be more **consistent**” and supplies her with explanations for students when they ask why they got the grade they did. With that said, when asked about how her **grading goal** changed between grading sessions, Christina started by reiterating her **normal grading goal** when she would grade this assignment for her classroom. She would give them full credit if the program draws a brick wall with either verbal or written feedback to let them know how to improve their programming to manage complexity (i.e. creating custom blocks to abstract code complexity). She explained her **grading goal for grading session one** was the same as her normal grading goal: giving students individualized feedback for improvement. This was her reason why her grades were more generous in the first grading session compared to the second. Though this was her normal grading goal,

Table 6.7 Christina’s grades for the submissions only had a change of over 5 percentage points for 2 submissions; Student 1 and 4 between the two grading sessions. The submission level refers to the achievement of the submission.

	Grading (no rubric)	Grading (with rubric)	Submission Level
Student 1	80	37.5	Low
Student 2	90	87.5	Middle
Student 3	100	100	High
Student 4	70	50	Low
Student 5	100	100	Middle
Student 6	100	100	High

when asked about **pain points**, Christina mentioned that giving individualized feedback was also her biggest pain point because of the number of students she has. Christina’s **grading goal for grading session two** focused more on specific elements found in their code. Christina’s type of grading changed from the first session to the second session. In session one, Christina executed the program and graded based on the output and custom blocks, therefore her grading method was **holistic** (grading the project’s output and code simultaneously) and **task-specific** (specific to the assignment). In session two, Christina created and used her rubric with multiple levels, testing each of the custom blocks independently, therefore her grading method was **analytic** (grading individual parts of the project) and **task-specific**.

Though Christina prefers to only formatively grade labs for the class, she did notice that if she did her grading the same way in the first grading session, she would suspect “they’re all getting it and they’re gonna have no problems moving forward” when in reality, some of the students actually needed more support. With the rubric, she explained, she would be able to go back to her students that needed more help and “give them a remediate assignment...to help them move on.” She shared that her students would do the remediation work if they could earn points back because they don’t want to mess up their GPA.

6.5.2.4 Deborah

Deborah has been teaching computer science for over 8 years, specifically teaching APCSP using BJC with Snap/ for over 5 years. She is a Master Teacher within BJC, meaning she is experienced and teaches other teachers how to use BJC. She is **fairly confident** with grading block-based language programs and has been grading them for over 8 years. Deborah **creates her own or modifies her previously used rubrics** before she begins grading these projects. If possible, Deborah also uses other teacher- or curriculum- created rubrics. She also uses autograders to grade other computing work from other courses. Even though the autograder occasionally fails a student for incorrect reasons, she still appreciates the amount of time it saves her.

Deborah shares that she hates grading because of the amount of time it takes for her to give good individual feedback to her students. She grades some assignments formatively and some sum-

matively for her classrooms. When the researcher introduced the lab she would be grading, Deborah said that it was a very familiar lab, expressing it is an assignment she has seen and graded many times. She asked if there was any specific grading criteria she should be using and the researcher replied only the assignment instructions.

To set up her grading environment, Deborah asked how to download the XML files and what to do with the XML file to view the Snap code since she typically has students submit a shareable Snap link. After Deborah downloaded the XML files, she imported the first submission to Snap. She immediately opened a custom block in the first submission, readedthrough the code, in the block and stated, “I’d give this a C”. She continued on and referenced the assignment instructions and stated elements from the instructions that the student was missing. Deborah went back to the folder with the submissions and downloaded the remaining XML files. She continued to open the Snap projects one by one, give a grade, and explain aloud why by referencing errors in their code, typically referring to the instructions. As she gave a grade to each submission, she also provided feedback for each submission. Deborah gave letter grades instead of percentage [C, B+, A, I, C, B+], where the I is an incomplete grade and would require a resubmission. Normally when she gives grades, she allows students to resubmit projects to increase their grade. She shared that before the pandemic, she would regrade submissions and the student’s final grade would be the average score of the two submissions, however, with the remote classroom, she instead is giving the students the higher of the two grades. Deborah’s letter to percentage conversion is:

Table 6.8 Deborah’s translation of Letter Grades to Grade Percentages.

	Normal Range	Plus (+) Range	Minus (+) Range
A	93 - 96	97 - 100	90 - 92
B	83 - 86	87 - 89	80 - 82
C	73 - 76	77 - 79	70 - 72
D	63 - 66	67 - 69	60 - 62
F	59 and below		

When Deborah created a rubric, she used 5 categories (plus a bonus category). Each of the categories had different weight in the rubric and a different amount of levels within the category. Her categories included: (1) Output looks like sample?, (2) Does it have appropriate comments?, (3) Uses loops, (4) Does it work for all cases?, (5) Does it test for odd/even row numbers?, and (6) Asks for user input?. Deborah stated she wanted to add the extra credit in her rubric, even though she knew that none of the submissions contained user input, because she may add it for extra credit or make it a required element when she assigns this assignment in her class.

Deborah also shared that for large class projects, which she grades with a summative rubric, she creates the rubric with her class when she assigns the project so her class understands exactly what they are going to be tested on.

Table 6.9 Deborah's rubric has 6 categories, one of which is a Bonus/Extra Credit category. Her categories have different distributions of weight and a different number of category items (levels of correctness).

	Reasons and points awarded			
Output Looks like Sample?	Perfectly 10 points	Too much space between the rows 1-9 points (range)		Nope 0 points
Does it have appropriate comments?	Yes 5 points	Somewhat 1-4 points (range)		No 0 points
Uses Loops	Yes 5 points	Not every where they could be 1-4 points (range)		No 0 points
Asks for user input? (Extra Credit)	Yes - 5 points		No - 0 points	
Does it work for all cases	Yes 10 points	For 2 of 3 6 points	For 1 of 3 3 points	Not at all 0 points
Does it test for odd/even row numbers?	Yes 2 points		No 0 points	

When Deborah regraded the 6 submissions she no longer referenced the assignment instructions, instead she referenced her rubric she made. She stepped through the rubric with each submission, going in order of the rubric categories. Since she did not identify exact point values for some of the category elements (i.e., the range for the “Output looks like Sample” category), she would give a point value and then compare to other submissions that also had a less than perfect visual output. This was the only time she compared submissions. Deborah did not give feedback for submissions during the second session of grading.

Table 6.10 Deborah's grades for the submissions changed for each submission. The submission level refers to the achievement of the submission.

	Grading (no rubric)	Grading (with rubric)	Submission Level
Student 1	C	31.25%	Low
Student 2	B+	75%	Middle
Student 3	A	78.13%	High
Student 4	I	15.63%	Low
Student 5	C	68.75%	Middle
Student 6	B+	75%	High

When Deborah compared grades from the two different grading sessions, she noticed that Student 1 and Student 5 received the same grade in the first round, however, in the second round of grading, they did not. She said that the disparity in the grades was due to the rubric. When asked about **grading goals**, Deborah answered that her **grading goal of grading session one** was the same as her **normal grading goal**, which is giving individualized feedback. Deborah answered that her **grading goal of grading session two** was to determine which elements from the assignments students implemented, to give a grade. When asked about **pain points**, Deborah mentioned that giving

feedback for the number of students she teaches is challenging because she has so many students. Deborah's **type of grading** did not change between the first and second session. In Deborah's first session, she evaluated and ran each custom block required for the assignment and made sure the submissions passed all expected conditions. In her second session, she created and used her rubric with multiple levels, assessing the same thing as before except noting it with a rubric. For both of sessions, her grading methods were **analytic** (assessing individual elements) **and task-specific** (specific for this assignment).

She said that if she gave her students these grades, they would all resubmit their assignments to get it to 100 percent. When asked what her grading goal was for each of the grading sessions, she explained that the first round of grading was more formative and the second round of grading was more summative. She shared that grading as a formative assessment is usually much faster than grading as a summative assessment. Deborah shared that "when you grade without a rubric, unless you go back and you grade each assignment twice, there's no consistency."

6.6 Discussion

In this section, I compare the teachers grading methods and rubric creation processes from the think aloud sessions. Then I discuss the implications of themes found in the participants' case studies. When referencing a participant in this section, I will use A for Alice, B for Beth, C for Christina, and D for Deborah.

6.6.1 Comparing Teachers Grading

When teachers were setting up their grading environment, three of the participants (A,C,D) did not know what to do when presented with XML files to download. This led us to learn that most teachers have their students share a link to their project as a submission. This is different than we found previously when talking to teachers who had to download XMLs from their LMS, and then upload them into Snap!. After the teachers downloaded the XMLs to grade, two of the participants (A,D) were unsure how to view the code, or load the XML files back into Snap (either by importing or dragging the XML into Snap). After the teachers had their code loaded in snap, their only questions were towards themselves as they thought about how the students programmed their projects.

During the first grading portion, two of the participants reviewed all student assignments before giving grades (A,B). Participants did this by opening six tabs running Snap! and importing one submission per tab. Even though three of the teachers (A,C,D) were very familiar with the lab and have previously taught and graded the assignment, they still referenced the directions frequently when grading submissions during the first grading session and when making the rubric to follow the criteria listed in the assignment for the second grading session. One of the teachers (D) would predict the students' grade within 10 seconds of looking at their project code and then spend a couple of minutes confirming their prediction and giving reasons why they thought the student should receive that grade. All of the teachers announced when they thought they found the best submission

(we did not ask them to do this). Three (B,C,D) of the teachers then proceeded to compare other student submissions to the exemplar submission. Three of the teachers (B,C,D) wrote comments (as feedback) for the grades they gave. One of the teachers (B) created a small notes document to keep track of items they were looking for. While the other 3 teachers (A,C,D), kept track of grading criteria in their head and frequently referenced the instruction sheet.

During the second grading portion, teachers graded the same 6 submissions using their created rubrics. Three of the teachers (B,C,D) created rubrics with multiple levels of points for each criteria whereas one teacher (A) created a single-point rubric, purely looking to see if a student had something in their code or not. All four teachers added their rubrics to their spreadsheet to reduce the number of tabs they would be switching between. Teachers did not reference the instructions as much during the second grading part. Instead they referenced their created rubrics. None of the teachers supplied feedback in the gradebook during the second grading session.

Three of the teachers (A,B,C) shared they felt they were easy graders during the first part but did not feel as such during the second grading part. They felt with a rubric they had to be more strict. All of the teachers' grades changed, especially for students with lower achieving solutions.

Table 6.11 The participants' grades for each submission for each grading session. Each participant row is denoted with the first letter of their name (Alice - A) and the grading sessions are denoted with the accompanying number (1 - first grading session, no rubric; 2 - second grading session, with rubric). The level row denotes the the achieving level of the submission. The avg column is the calculated average of grades given during that grading session for that teacher.

	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6	Avg
<i>Level</i>	Low	Middle	High	Low	Middle	High	
A (1)	90?	90?	96	86-89	90?	96	91.83
A (2)	89	96	90	96	89	98	93.00
B (1)	70	90	95	50	90	90	80.83
B (2)	30	85	95	25	85	90	68.33
C (1)	80	90	100	70	100	100	90.00
C (2)	37.5	87.5	100	50	100	100	79.17
D (1)	C (73-76)	B+ (87-89)	A (93-96)	Incomplete	C (73-76)	B+ (87-89)	84.60
D (2)	31.25	75	78.13	15.63	68.75	75	57.29

Table 6.11 shows all the participants' final grades for each submission for both grading sessions. As a reminder, the first grading session, indicated by <first letter of participant's name> (1), is when a teacher graded without a rubric and the second grading sessions, indicated by <first letter of participant's name> (2), is when a teacher graded using a rubric.

From grading session one to grading session two, only one teacher produced grades that increased and also increased the overall grade average. Alice increased the score for Student 2 and Student 4 and the overall grade average increased by 1.17 percent. Alice was also the only teacher whose final grades in the same order did not align with the predefined students' achievement levels.

Although Alice has taught and graded blocked-based projects for years, she has not felt comfortable grading nor she does focus on grading code. She explained that grading the code without supplemental materials does not allow her to understand what her students are thinking when they are programming. For her, this removes the opportunity to truly figure out what her students know and how to help her students learn. For example, if a student doesn't know how to program an element within a project, that student still has the opportunity to earn a 90 if they explain well enough what they didn't comprehend and why they couldn't program it. This could explain why her grades are all higher than the other three teachers' grades. When grading the submissions, she perhaps was imagining the student also submitted the supplemental materials explaining where they went wrong in the programming project, for her to determine the grade the student would get. Additionally, Alice's lack of confidence in grading could contribute to her different scores when grading. Although there are multiple reasons for her different grades, the most important factor may be her current school environment. Alice mentioned that a grade lower than 90 would typically result in a conversation with the student's parents. Throughout Alice's grading sessions, she referenced these meetings with parents several times, constantly reflecting on how her current class of students would react based on such grades. This student and parent confrontation towards the teacher may hinder the teacher's ability to grade without anxiously awaiting a confrontational meeting. Teachers have already complained about the lack of time for grading, prepping, teaching, and helping students. It makes sense that Alice would decide if she wanted to have a confrontational meeting with the parent and student based on a small assignment grade when, in reality, she uses assignment to help students gain an interest in computing and develop some skill along the way. To classify her grading habits, **Alice thinks about student's reactions when she grades, letting this make an impact on the grades she provides.**

All three other participants' overall grade average decreased by over a grade letter, or over 10 percentage points. Beth's overall grade average decreased from 80.83% to 68.33%. Grades for students 3 and 6 stayed the same, and students 2 and 5 only decreased by 5 percentage points, however, students 1 and 4 grades decreased by at least half the grade from session one. This grade difference for the lower-achieving submissions could be a result of the lack of a baseline the teacher typically supplies the students with. For example, during the first grading session, Beth said she would give students a minimum of 50 if they showed an effort to complete the assignment. When Beth graded with the rubric, she did not account for this minimum grade. Beth shared that she normally has this minimum baseline for her students because if she gave the grades from session two to her students it would be discouraging. Since her main grading goal is to understand where the students are misunderstanding course concepts so she can adjust her class material, determining the actual grade and understanding of her students by summatively grading her students' work is important for Beth. To classify her grading habits, **Beth focuses on learning what more her students need from her, finding common misconceptions she needs to review in her future classes.**

Christina's overall grade average decreased from 90% to 79.17%. Grades for students 3, 5, and 6 stayed the same and grades for students 1, 2, and 4 all decreased. The grade for Student 2 only de-

creased by 2.5 percentage points, however student 1 and 4 decreased by 42.5% and 20%, respectively. This grade difference for the lower-achieving submissions is most likely a combination of (1) not typically giving summative grades for assignments and a (2) result of not fully understanding what students were missing when providing a formative assessment by just checking the submissions for completion. Christina said that grading as a formative assessment is much faster which she shared is how she normally grades student assignments, such as Brick Wall, for her class. However, she also exclaimed that in this particular instance, if she graded this assignment as a formative assessment and saw these grades as an output 70-100% with an average class grade of 90%, she would probably move onto the next lesson without a second thought. She continued to say that this would've been the wrong decision since two of her students needed way more help than she anticipated, only discovered using her summative assessment. To classify her grading habits, **Christina grades students with lower achieving levels easier when not using a rubric, causing her to miss common mistakes amongst her students who may need the most help, however, she is good at scoring students who are doing well on assignments.**

Deborah's overall grade average decreased from 84.60% to 57.29%. All but one of her grades were the lowest of any teacher in the second grading session. Each of her grades decreased, except for the Incomplete, since that was not a numerical grade. Grades for students 2, 3, and 6 decreased by about 15%. The grade for Student 5 decreased by 6.25%. And the grade for Student 1 decreased by 43.75%. When Deborah created her rubric, she added grading criteria from beyond the assignment instructions. For example, one of Deborah's grading criteria was if the student had appropriate comments, however, the assignment instructions did not request students to add comments to their code. Deborah would give 5 points to a project if it contained appropriate comments and the total possible score was 32 points. There this immediately removed the chance for a submission to receive higher than 85% for their grade. If Deborah did not take off points for not adding appropriate comments to their projects, the submission grade would be 37.04%, 88.89%, 92.59%, 18.52%, 81.48%, and 88.89%, respectively, which would have resulted in the final submissions' grades to mostly stay the same, excluding Student 5, which would've increased. To classify her grading habits, **Deborah created rubrics with expectations beyond the assignment instructions, suggesting she expects more from her students for this assignment. Additionally, she grades stricter compared to the other participants.**

Table 6.12 Deborah's grading in the first grading session D(1), the second grading session with her rubric (2a), and the second grading session removing one category that was not expected by the assignment (2b).

	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6	Avg
Level	Low	Middle	High	Low	Middle	High	
D (1)	C (73-76)	B+ (87-89)	A (93-96)	Incomplete	C (73-76)	B+ (87-89)	84.60
D (2a)	31.25	75	78.13	15.63	68.75	75	57.29
D (2b)	37.04	88.89	92.59	18.52	81.48	88.89	67.90

To address **Research Question 1**: How do grading outcomes (score/grade/percentage) change when using a method to keep track of grading criteria?, and to address Hypothesis 1: when using a rubric to grade block-based language assignments, teachers will result in lower grades compared to when teachers did not use a rubric to grade the same submissions, here is a summary of what changed between grading sessions one and two for the teachers.

1. For Alice, both first and second session were in a similar range, and therefore the scores/grades/percentages did not really change.
2. For Beth, the first session had a higher average grade percentage than the second session. The lower-achieving submissions' grades decreased by 25-40%.
3. For Christina, the first session had a higher average grade percentage than the second session. The lower-achieving submissions' grades decreased by 20%-42.5%.
4. For Deborah, all submission's grades were 7-43% lower in the second grading session. However, the only reason her grades from grading session 2 were so different from grading session 1 was because she added a grading criteria that the assignment did not require even though it would have been a requirement for her class. Therefore, grading with a rubric did lower the submissions' grades, however, it was due to a non-required grading criteria.

In general, when a teacher used a rubric to grade the assignment, they gave lower grades to the students with lower-achieving submissions. Beth and Christina (and Deborah when not considering the comment grading criteria) did better at predicting middle- and higher-achieving submissions. Maybe they do not have students who submit lower-achieving submissions. This could also be a result of not wanting to discourage students in their classroom, a concern each teacher mentioned when reflecting on the differences in grades. This finding confirms and contradicts the hypothesis. Therefore, teachers are better at estimating the score of their students with higher- and middle-achieving submissions, however, they may need to use a summative assessment to know when their students with lower-achieving submissions need more assistance in their classroom.

During the grading portions of the study and the post interview, participants shared their normal methods of grading, why they do that in this normal method, and why they weren't doing their normal methods during this research session. Two of the teachers (C,D) shared they use a mixture of formative and summative assessments in their normal classrooms. The other two teachers (A,B) did not explicitly share the type of assessments or grading they do in their classrooms. Three of the teachers (A,C,D) shared they feel like they are normally an easy grader. One teacher (D) discussed the difference in their grading both pre-pandemic and during the pandemic. One of the participants (A) requires supplemental materials, such as a journal, for students to explain what they understand. Two of the participants (C,D) review instructions with students and allow resubmissions for regrades before giving failing grades. Two of the participants (C,D) mentioned that occasionally they give extra credit on assignments when students go above and beyond the requirements of the assignment. One participant (A) said they never give extra credit.

6.6.2 Rubric Creation

When creating their grading criteria between the two grading sessions, three participants created a rubric (B,C,D) and one participant created a checklist, also known as a single-point rubric (A). Only two of the participants (B,D) who created rubrics added different weights to the category items, signifying which were more important, or worth more points. After the three participants (B,C,D) who created their rubrics in the template provided, they moved their rubrics to the grading spreadsheet to reduce the number of tabs they needed to reference while grading. The participant (A), who created a single-point rubric, made it within the grading spreadsheet.

Participants shared some of their normal rubric creating habits. Two of the participants (C,D) shared that they normally create their rubrics before they give the assignment to the students. They also share their rubrics with the students to help them understand how they are going to be graded. One participant (A) shared they typically create a rubric or checklist before they grade. And the other participant (B) shared they typically create a rubric after looking over a few assignments to understand common students' errors she should be prepared for. This participant also shared that they have tried to use rubrics created by other teachers, but her class makes different errors unaccounted for in their other rubric. That said, three of the participants (B,C,D) shared that they like to use rubrics when they grade and wish there were already good premade rubrics for their curriculum providers to support them better.

To address **Research Question 2**: How does grading with a tool (such as a rubric) impact a teacher's grading goal?, and to address Hypothesis 2: teachers who have more block-based language grading experience will create a more informative rubric, I determined the type of rubric that teachers created using Brookhart's explanation of different types of rubrics [Bro16]. Grading goals are self-reported during the interviews directly after the second grading session. The researcher asks the teacher what their grading goal was for each of the grading sessions. Rubric type and grading goals are found in Table 6.13. During the interview, the participants were asked if their grading goals changed between sessions. One of the participants (B) said their grading goal was the same between sessions. Three of the participants (A,C,D) stated that their grading goals changed. These teachers felt that as soon as they incorporated a rubric into their grading method, their grading process was no longer formative and was purely for investigating what the students got incorrect. They felt they were no longer giving feedback for students to improve, but instead just telling the students where they did something incorrectly.

This is interesting because an informative rubric does not necessarily determine the type of assessment for an assignment [Bro16]. In fact, the type of assessment, such as formative or summative, could both be determined when using an analytical rubric, which was the most common type of rubric participants (B,C,D) created. This finding was unexpected and suggests that teachers could benefit from a better understanding of differences in types of rubrics, when to use the different types of rubrics, and how to create them.

This finding led to further investigation about the similarities and differences in the rubrics the

Table 6.13 The participants’ prior grading and teaching experience with block based programming languages, their grading goals for both sessions, their rubric type and categories with the number of corresponding category items.

	Prior Experience & Grading Confidence	Grading Goal One	Grading Goal Two	Rubric Type	Categories - # of Levels
A	8+ y grading BBPL 3-5 y teaching BBPL Not Confident	Find passion for students; build their confidence	Determine missing assignment elements	Single-point Task-Specific	One Brick - 2 Row A/B - 2 Procedure with Input - 2 Rows line up - 2
B	1st y grading BBPL 1st y teaching BBPL Slightly Confident	Determine if students understood and completed the assignment; provide feedback for improvements		Analytical Task-Specific	Appearance & Naming - 4 Abstraction - 4 Functionality - 4
C	3.5 y grading BBPL 4th y teaching BBPL Extremely Confident	Give individualized feedback for improvement	Determine missing assignment elements	Analytical Task-Specific	Draw Brick - 3 RowA - 3 RowB - 3 Draw Brick with # Rows - 3
D	8+ y grading BBPL 5 y teaching BBPL Fairly Confident	Give individualized feedback for improvement	Determine missing assignment elements	Analytical Task-Specific	Visual output - 11 Comments - 6 Loops - 6 Cases - 4 Even/Odd - 2

participants made beyond the type of rubric. I explored the rubrics’ type of category, type of detail in category levels, and weight distribution.

Alice and Christina created categories with assignment-specific titles, such as Draw () Brick, and specific category level descriptions. Beth created categories with general titles, such as Abstraction and Functionality, however, the categories level descriptions are specific to the assignment. Deborah created a mixture of assignment-specific and general, category titles and category level descriptions.

Christina’s rubric categories explicitly explain what a student did correctly or incorrectly for any category and level within her rubric, however, her category descriptions are not general enough for unexpected student mistakes. Beth’s rubric categories explicitly explain only when a student receives full credit or no credit for a category because she generalizes the intermediate descriptions (i.e., missing one required item). Some of Deborah’s rubric categories are less detailed, therefore, only sometimes it may be possible to determine what the student is missing or what the student did correctly by using her rubric. Since Alice’s rubric is a single-point rubric, it is only clear when a student correctly or incorrectly completed a category, however, it is not clear what that category required to be done correctly.

Different types of rubrics are used for different purposes [Bro16]. Due to the level of detail, Christina’s would be useful to provide to students as feedback for what they did correctly or incorrectly, with the assumption that the student completed the project as she expected. Additionally, Christina’s rubric would be useful for other teachers adopting her rubric, but only if students completed the project as expected by Christina. Beth’s rubric would also be useful to provide as feedback to students with an additional comment pointing out which of the 3 grading criteria of the category the student missed. Beth’s rubric would be useful for other teachers adopting her rubric, so long that they supplied the additional comment. Although Beth’s rubric is specific to this assignment,

Table 6.14 A descriptive table comparing the rubrics of the four participants. For the Type of Rubric: S - Single Point, T - Task Specific, and A - Analytical.

	A	B	C	D
BASIC RUBRIC DETAILS				
Number of categories	4	3	4	5
Categories have levels?	No	Yes	Yes	Yes
Type of Rubric	S, T	A, T	A, T	A, T
TYPE OF CATEGORY				
General category (i.e., Readability)	-	Yes	-	-
Specific category (i.e., Draw Brick)	Yes	-	Yes	-
Mixture of general and specific categories	-	-	-	Yes
DETAIL IN CATEGORY LEVELS				
General enough for edge cases	-	Yes	No	Yes
Clear what student missing at each level	-	Somewhat	Yes	Somewhat
Clear what student missing at lowest level	-	Yes	Yes	Somewhat
Clear what student got correct	-	Somewhat	Yes	Somewhat
Clear what student got correct, if perfect	-	Yes	Yes	Somewhat
Same number of levels per category?	-	Yes	Yes	No
WEIGHT DISTRIBUTION				
Same weight for each category	-	No	Yes	No
One weight per level?	-	Yes	Yes	No

her rubric may also lend well to modifications for other assignments since the category titles were general and a teacher would only need to change the contents of the one category level per category for other assignments. Deborah's rubric could be given to students as feedback for a grade on a project, however, she would have to supply more additional and individualized comments to explain the score. Deborah's rubric could be useful for other teachers as it provides more flexibility in scoring some categories, however, modifying the rubric or determining the score for categories with flexibility may be more challenging for newer teachers. Alice's rubric could be useful as an internal check if students completed the overall elements of the assignment, however, without any additional comments, her rubric would not assist students in understanding what they did incorrectly. Alice's rubric may also prove challenging to use for newer teachers due to the lack of guidance in what to check for in the code.

Alice and Deborah both have been grading block-based language assignments for 8+ years, however, their rubrics were very different. Deborah's rubric seems more informative than Alice's since she provided multiple levels within her rubric categories. Christina has been grading block-based programming language assignments for 3.5 years, however, her rubric seems more informative (detailed) or useful than Deborah's or Alice's. This academic year is Beth's first year grading block-based programming assignments, however, her rubric seems more informative or useful than Deborah's or Alice's. Christina's rubric seems more informative or useful than Beth's under the assumption that student's mistakes align with Christina's expectations. However, Beth's rubric seems

more useful for adapting to use for other assignments. Therefore, a teacher's grading experience does not necessarily influence rubric types. This finding does not confirm or contradict Hypothesis 2: teachers who have more block-based language grading experience will create a more informative or useful rubric. This potentially future research could help teachers determine what type of rubric best works for their needs.

6.6.3 Themes

The following themes were found from tagging participants' think aloud data during the grading sessions and from the interviews. I refer to teachers' case studies and discuss any implications in the remainder of this section.

6.6.3.1 Feedback is Important for Students and Teachers

During the interviews, participants share their goals when they graded these student submissions. Two of the participants (C,D) shared their most important grading goal is giving enough feedback to their students. These were the same teachers who shared that they use a mixture of formative and summative grading methods in their classes. Both said that the first grading session during the study was similar to how they would grade students with formative assessments; grading less strict and trying to learn what their students did not understand to give feedback so they can learn more before a summative assessment. However one of the teachers (C) noticed that if they only graded with this method, she may miss important student misconceptions.

The other two participants (A,B) also mentioned that one of the important elements of grading is to give feedback to their students but, it was not the most important grading goal for them. However, both participants shared their most important grading goal is understanding what the students did not understand to essentially give feedback to themselves. Both participants shared that they grade students' code to find where the students still need help and then adjust their course to continue meeting the students' needs.

This result suggests that teachers would appreciate assistance in giving more direct individualized feedback to their students and understanding student grades to learn how to adjust their class for future lessons. An analytics tool that already associates a rubric item with appropriate formative feedback could be used to reduce the amount of time a teacher spent giving feedback to students. It can also supply teachers with an understanding of their class progress overall by supplying the teacher with information about their classroom's overall understanding of certain computational concepts.

6.6.3.2 Other Grading Goals

To address **Research Question 3**: What is important for teachers when grading a block based language assignment?, we asked teachers for their goals when grading. This is the complete list of all

the common grading goals amongst the participants and which participants said that this was one of their goals when they grade.

1. Give good feedback to give their students (A,B,C,D)
2. Determine next teacher action (i.e. adjust their classroom) (A,B)
3. Identify common mistakes amongst students by debugging student code to give feedback on how to change it (B,C)
4. Look for student understanding (A,B) (i.e. finding students who are struggling by comparing student code)
5. Consistency (A,C,D)

Most of these common grading goals are focused on how teachers can help their students. The first two in the list were discussed in the previous theme (Feedback is Important for Students and Teachers). The third and fourth goals coincide with the first 2 goals such that teachers want to supply better feedback to their students by fully understanding exactly what a student would need to do to receive full credit. With this knowledge a teacher can give more specific feedback. Additionally, if multiple students are making the same error, this could suggest to the teacher that they need to alter their next lessons to review these mistakes. This further suggests that teachers would benefit from an analytics tool to help teachers understand common misconceptions in their class. A tool could also be used to make it possible for teachers to give individualized feedback. For example, when a teacher selects an item in a rubric, there could be feedback linked to that item in the rubric that will be added to a feedback bank for the student.

The fifth grading goal is specifically for teachers. Most of the participant teachers shared they want to ensure they are grading consistently for all their students projects. One of the teachers (D) shared rubrics help to improve their consistency when grading. This would especially be the case when grading over 150 submissions due to the high volume of students teachers have. Previously research has shown that using a rubric or some predefined guide helps graders maintain consistency while grading student work [Rag20].

6.6.3.3 Pain points

When asked about pain points, two of the participants (C,D) mentioned that grading and giving individualized feedback for the number of students they teach is challenging. These were the same teachers that mentioned giving individualized feedback was also their main grading goal. This further suggests teachers could benefit from an analytics tool that already associates a rubric item with appropriate formative feedback could be used to reduce the amount of time a teacher spent giving feedback to students. One of the participants (A) said their biggest pain point was the large number of accurate solutions and complexity of those solutions for programming projects. It is difficult to explain this result, but it might be related to the teacher's comfortability of reading and

grading code. The other teacher (B) shared her biggest pain point is finding a good rubric to use for the assignments. This result may be explained by the fact that this teacher is new to teaching the APCSP course, using the BJC curriculum, and teaching and grading block-based languages. The following includes all the pain points participants shared.

1. Individualized comments (C,D)
2. Number of submissions to grade (and regrade) (C,D)
3. Unexpected errors are difficult to prepare for in rubric (B)
4. Complexity of the numerous solutions for a problem (A)
5. Needing to reference multiple tabs (tab switching is annoying) (A,C,D)
6. Organizing the script space (B,C,D)
7. Finding all custom blocks (A,C,D)
8. Feedback based on rubric (B,C,D)
9. Want to give other methods of feedback, such as audio comments (C)

I previously discussed the first four pain points earlier in this section. The fifth pain point is regarding to the number of tabs a teacher may need to have open when grading projects. One teacher (D) specifically shared they need to have open the instructions, the rubric, and student submissions. Without an extra monitor, she said it would be nearly impossible to grade in an organized and timely manner. It can therefore be assumed that a tool where all the teacher needs is in one screen would make their grading methods drastically different, and hopefully for the better.

The sixth and seventh pain points refer to the programming environment, Snap!. All of the teachers either found it difficult or frustrating to find all the students' custom blocks or understand what was happening within the script space of Snap!. This problem could be solved by requiring students to organize their projects a certain way before submitting it to the teacher, however, this would not always solve the problem. This finding, while preliminary, suggests that there is a lack of dedicated teacher tools and support when grading in Snap!. Teachers shared that they want tools specifically designed for teachers to be able to see all students' custom blocks in one location instead of spread throughout Snap!. One teacher shared (C) other things they typically do when loading a student project, including (1) cleaning the stage, (2) locating the sprite (as it may be off screen), (3) finding all the custom blocks students make, (4) organize the script space (to understand their code), and (5) understand how a student wants her to execute their code (hopefully connected to the "when green flag is pressed" block). This further suggests there are even more ways to improve the teachers grading experience by giving them additional navigational tools, integrated with Snap! to help with this teacher's steps 1-4.

The eighth and ninth pain points refer to the methods for teachers to give feedback to their students. Two teachers (B,C) shared it would make their lives a lot easier if the rubric already had feedback associated with rubric items. This strongly reinforces the aforementioned notion of the need for a grading tool. One teacher (C) has the ability to create audio comments for students projects, providing feedback in a faster way for her, however, it does not currently work as she needs it to. Currently she cannot add feedback associated to rubric items with audio.

Due to the pandemic, some of the participants also shared challenges due to being remote. One teacher (A) shared that not being able to communicate with students in person made grading and understanding their students' thought processes difficult at the beginning of the pandemic. At the same time, she has felt that her students have gotten better at communicating over email or in their supplemental journal entries for their code assignments. This combination of findings provides some support for the need to have a way for teachers and students to communicate outside of just submitting code for programming assignments. One of the teachers (D) mention that debugging students' code with them over email is challenging, further supporting the need for an alternate method of communication with the student, especially when debugging software in real time with the student. One teacher (D) shared that remote grading is harder due to all the stress everyone is dealing with with the pandemic. This teacher is reinforcing the idea that the current teaching and grading system is different from the typical remote teaching and grading, however, it has shown a light on elements within her normal teaching and grading system that require more support.

6.6.3.4 Teachers want more support

When asked what they could use further support with, teachers (A,B,C,D) shared they would like premade, good rubrics with explicit criteria and how it aligns with the Advanced Placement exam. Though some of the teachers had more experience with and were more comfortable with creating rubrics for grading assignments, they felt like other teachers have probably already created rubrics for these assignments and teachers should be able to share materials more easily with each other. Each teacher should not have to recreate the wheel when other teachers have materials willing to share. This suggests the need for easier access to share other teachers materials in a community-like environment. Along with more support with rubric development, two teachers (A,B) also would like more curriculum support, such as mini tasks for students to complete to 'master' a topic. Though this does not refer to further support in grading, this further reinforces the need for a community-like environment that allows teachers to share supplemental mini-tasks designed to assist students to further their knowledge on a topic.

When asked what they would like in a grading tool for block-based languages, all of the teachers shared they would like an autograder. Two of the teachers (C,D) suggested an autograder that would take both an exemplar and an acceptable submission to compare students' submissions to. Teachers shared other curriculums or languages that have autograders, however, in their experience they were too strict. For example, teachers were unable to override a grade or create the criteria the autograding was using. Teachers shared the autograder would need to be flexible for them to use.

Teachers explained they would review the grades the autograder gave in case the need for any necessary changes to the final grades. However, an autograder would reduce the amount of time spent on grading assignments, giving teachers more time to spend with students who need more help. In addition to an autograder, teachers would like rubrics with associated feedback and an analytics tool to help teachers know how to adjust their classrooms based on class performance on assignments.

6.7 Conclusions and Contributions

To investigate research question 2b, this study aimed to (1) learn how teachers are currently grading, (2) determine what differences a rubric makes in the teachers' grading process, and (3) extract pain points and grading goals for a teacher when they are grading. In this chapter, my detailed research questions were (1) How do grading outcomes (score/grade/percentage) change when using a method to keep track of grading criteria?, (2) How does grading with a tool (such as a rubric) impact a teacher's grading goal?, and (3) What is important for teachers when grading a block based language assignment?. Hypothesis H1 is that teachers will have higher grades when not using a rubric and H2 is that teachers with more block-based language grading experience will create a more informative or useful rubric.

To address **Research Question 1 and Hypothesis 1**, I found that when a teacher used a rubric to grade the assignment, they gave lower grades to the students with lower-achieving submissions. This finding confirms and contradicts the hypothesis. Therefore, teachers are better at estimating the score of their students with higher- and middle-achieving submissions, however, they may need to use a summative assessment to know when their students with lower-achieving submissions need more assistance in their classroom.

To address **Research Question 2**, three of the four participants stated that their grading goals changed. These teachers felt that as soon as they incorporated a rubric into their grading method, their grading process was no longer formative and was purely for investigating what the students got incorrect. This is interesting because an informative rubric does not necessarily determine the type of assessment for an assignment. Therefore this unexpected finding may suggest that teachers could benefit from a better understanding of differences in types of rubrics, when to use the different types of rubrics, and how to create them. Additionally, to address Hypothesis 2, I found that a teacher's grading experience does not necessarily influence rubric types, however, this finding does not confirm or contradict Hypothesis 2 and this potentially future research could help teachers determine what type of rubric best works for their needs.

To address **Research Question 3**, I investigated participant grading goals to understand what is important to a teacher when grading block-based language assignments. I extracted 5 common grading goals amongst the participants: (1) Give good feedback to give their students; (2) Determine next teacher action (i.e. adjust their classroom); (3) Identify common mistakes amongst students by debugging student code to give feedback on how to change it; (4) Look for student understanding;

and (5) Consistency. Unfortunately, giving good, individualized feedback to the large number of student submissions was also a pain point for some of the teachers. These results suggest that teachers would appreciate assistance in giving more direct individualized feedback to their students and understanding student grades to learn how to adjust their class for future lessons. An analytics tool that already associates a rubric item with appropriate formative feedback could be used to reduce the amount of time a teacher spent giving feedback to students. It can also supply teachers with an understanding of their class progress overall by supplying the teacher with information about their classroom's overall understanding of certain computational concepts. Overall, teachers expressed the need for further support in their classroom and to enrich their teacher community. One step in the right direction having a block-based grading tool, such as an autograder, and community to allow teachers to share teacher-created materials.

CHAPTER

7

GRADING WITH GRADESNAAP

In this chapter, I investigate **RQ2d**: How do teachers grade with and perceive GradeSnap? I investigate the experience of teachers grading block-based programming assignments using GradeSnap to determine whether or not GradeSnap supports teacher’s grading goals and addresses teacher needs by reducing their pain points. Chapter 6 determined the grading goals and pain points for a set of four teachers. In this study, with the same four teachers, I investigate the detailed research questions: (R1) How does grading with the GradeSnap tool impact teachers’ grading goals? and (R2) What is important for teachers when grading a block based language assignment?. I predict that (H1) teachers’ grading goals will not change from the grading goals in chapter 6, (H2) teachers will make changes to their rubrics within GradeSnap to make their rubric more analytical or task oriented, and (H3) teachers will give at least as much feedback as they provided when they graded with a rubric during the study presented in chapter 6.

7.1 Related Literature

The study presented in this chapter will reference multiple sections from chapter 2, predominantly the Rubrics and Dashboard sections. A future goal in creating a dashboard is to help teachers adjust their class lessons based on individualized information from the class’ overall learning gains. Specifically, through a dashboard grounded in distributed cognition theory, which states, “instruments can support professionals when these instructions fit seamlessly into the activities of the professional” ([Hut00] as quoted in [Mol18]). When designing a tool you expect a teacher to use nearly every day during their job, it makes sense to include them in the early design and development stages of the application [Abe13]. Dashboard designs vary from tool to tool. Naturally, the majority

of the teacher dashboards are submission locations for student work [DeN17], [Zha20], [Sin17] and report each student's grades, but some do much more. In order to help teachers understand their students' learning gains, some tools will send statistics or reports to instructors with common wrong answers or mistakes [DeN17], [Sin17], [Gro16].

To best aid our end users, we want to ensure a good User Experience (UX), defined by "a person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service" [Mir15]. By considering a user's experience, the designer has a better chance of "making systems that are useful, usable, and pleasurable to use" [Bur18]. There are many User Experience Research (UXR) methods which can help researchers or developers understand a participant's experience [Roh14]. The open-ended nature of qualitative studies is suited for the first two stages of product development as the researcher is gathering ideas instead of refining the concept and measuring for effectiveness [Roh14]. During *Usability Testing*, the researcher can observe the user(s) carrying out a set of or a single user task(s) with a product [Roh14]. This chapter is the first usability test of GradeSnap's grading feature.

Throughout this chapter, I reference rubrics that participants created in the previous chapter 6. As a reminder, Brookhart describes a rubric as a "[coherent] set of criteria...that includes descriptions of levels of performance quality" with a main purpose to assess performances, for example to assess student's work [Bro16]. Brookhart describes different kinds of rubrics, comparing Analytic to Holistic and General to Task-Specific. Analytic rubrics break down student work into subtasks, allowing teachers to evaluate student performance of specific criteria individually, whereas, Holistic rubrics allow for a teacher to evaluate the all criteria of a student's work at the same time [Bro16]. General rubrics can be used to evaluate multiple assignments with different instructions, grading on a set of less specific criteria, whereas, Task-specific rubrics are detailed and specific to an assignment [Bro16]. In addition to these rubric types and descriptors, Fluckiger shared Dietz's introduction to a single point rubric, where "the single point rubric provides a single set of criteria" [Flu10; Die00].

7.2 Updated GradeSnap Features

This section has updated images of GradeSnap, displaying the new Landing Page, Rubric Page, and Grading Page.

The Landing Page has quick links to potentially useful resources for BJC teachers. Additionally, the landing page has information about assignments and links to the teacher's courses. The menu in the left margin has links to all other important pages within GradeSnap, such as Courses and Section Roster, Gradebook, Assignments, and Rubrics, as well as a button to go to the Grading page.

The Rubric Page has a list of all rubrics associated with a teacher's account along the left side of the page (excluding the menu). The right side of the page has the categories of the selected rubric on the left. When a category is open, the user can see a point value and a description associated with each category item. This image was Beth's rubric for her Brick Wall, looking at the category 'Abstraction'. On the rubric page, a user may edit a rubric category item by clicking on the item

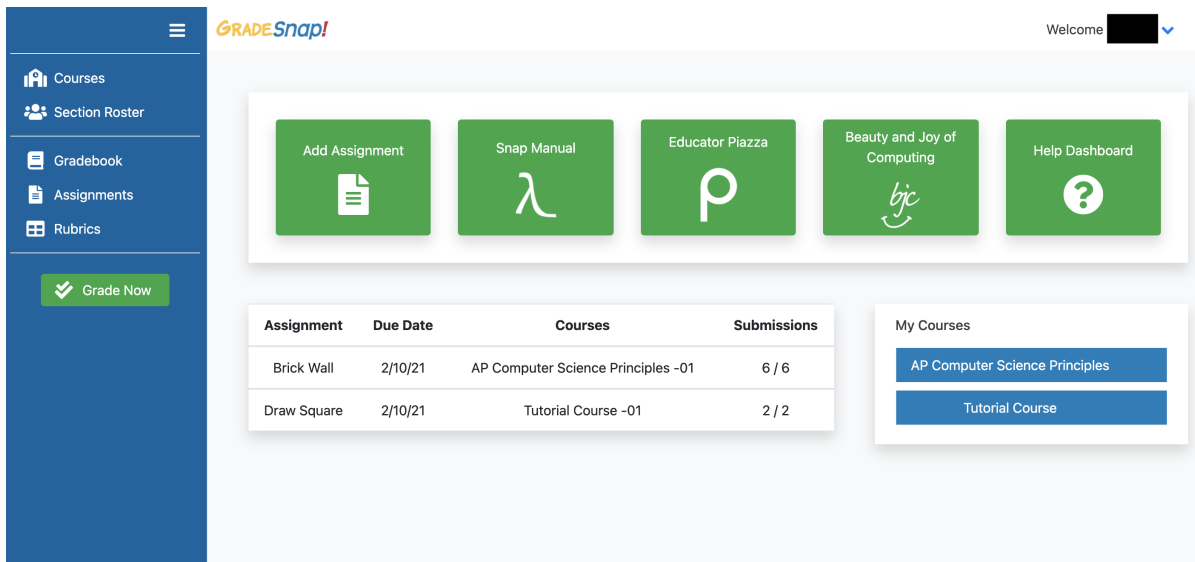


Figure 7.1 The first page a teacher arrives at upon logging into GradeSnap.

description or point, and is saved when the teacher blocks off. A user can also add a new item, category or rubric by clicking on the corresponding buttons.

The Grading Page has Snap! embedded within GradeSnap taking up most of the space on the page. On the right side of the page is the teachers rubric and feedback response box. Between the rubric and the feedback response box is a score adjustment field (for a teacher to increase or decrease the grade percentage); the score (calculated by the rubric selections); and the percentage (calculated with the score and the adjustment field). Under the feedback response box is a progress bar (showing the percentage of submissions they have left to grade); and left and right arrows (left: traveling to the previous student and right: traveling to the next student) which save the grades and feedback. A teacher grades by opening the rubric (clicking on the rubric category) and choosing a category item (one of the drop down items).

7.3 Participants and Pre-Study Actions

I recruited 4 teachers with a varying range of experience teaching computer science, who are teaching the Beauty and Joy of Computing curriculum of the Advanced Placement Computer Science Principles (APCSP) course, and using the Snap! block-based programming language. The participants have a range of confidence levels regarding grading block-based language assignments.

Before participating in this study, teachers completed the study in chapter 6 where I investigated teachers' grading habits, explored how the addition of a rubric impacted their grading methods, and extracted pain points and grading goals that teachers experience when grading block-based language assignments. During this study, teachers practice thinking aloud while they learn about and practice grading with GradeSnap. Then, they will think aloud while grading student programs using GradeSnap and a rubric they created during the previous study. At the end, they will participate

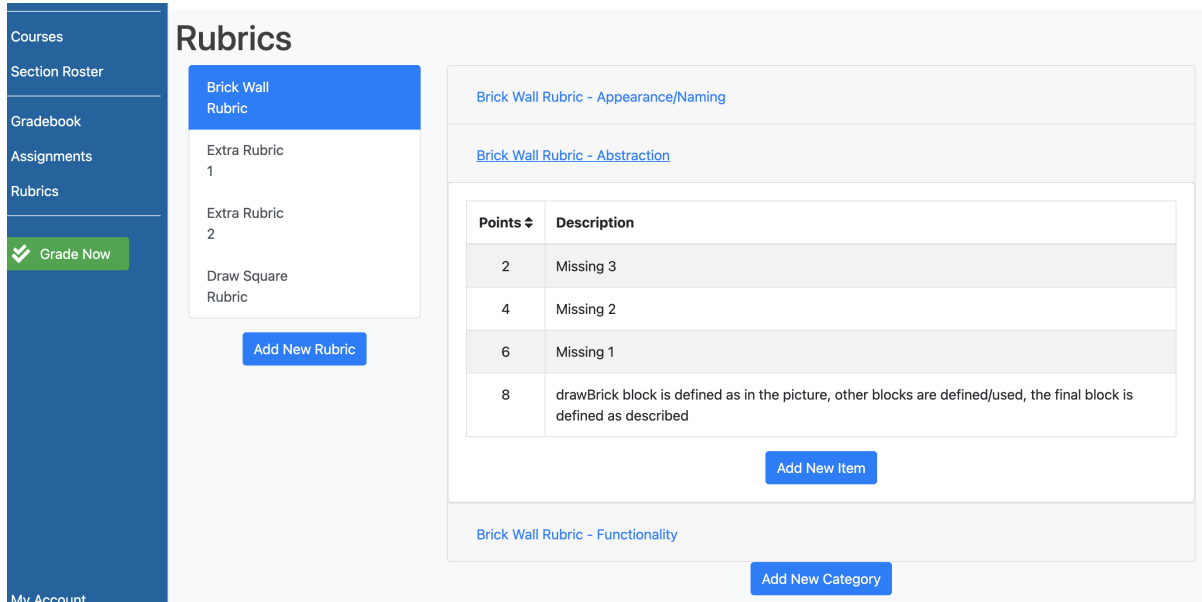


Figure 7.2 The rubric page within GradeSnap.

in a post-grading activity interview. The grading and interview activities occur during the scheduled and recorded zoom video call. In the next section I elaborate the study design.

7.4 Study Design

I scheduled and recorded individual zoom interviews (video calls) for each participant based on their availability. During the zoom interview, teachers first learn how to think aloud and how to use GradeSnap by completing a set of teacher-user tasks designed to help them learn how to use the new tool. The teacher-user tasks included:

1. Login with credentials
2. Find the tutorial course
3. Find the example assignment (familiar and called Draw Square)
4. Find the rubric you would grade that assignment with
5. View/Edit the rubric
6. Grade the six sample submissions

Before the grading portion of the study, participants can review the assignment instructions and the rubric they created, which they can also edit or re-create. After they finish reviewing their rubric, participants grade a new set of 6 submissions using the rubric they created. Participants think aloud during this study, sharing important information such as why they grade a submission a certain

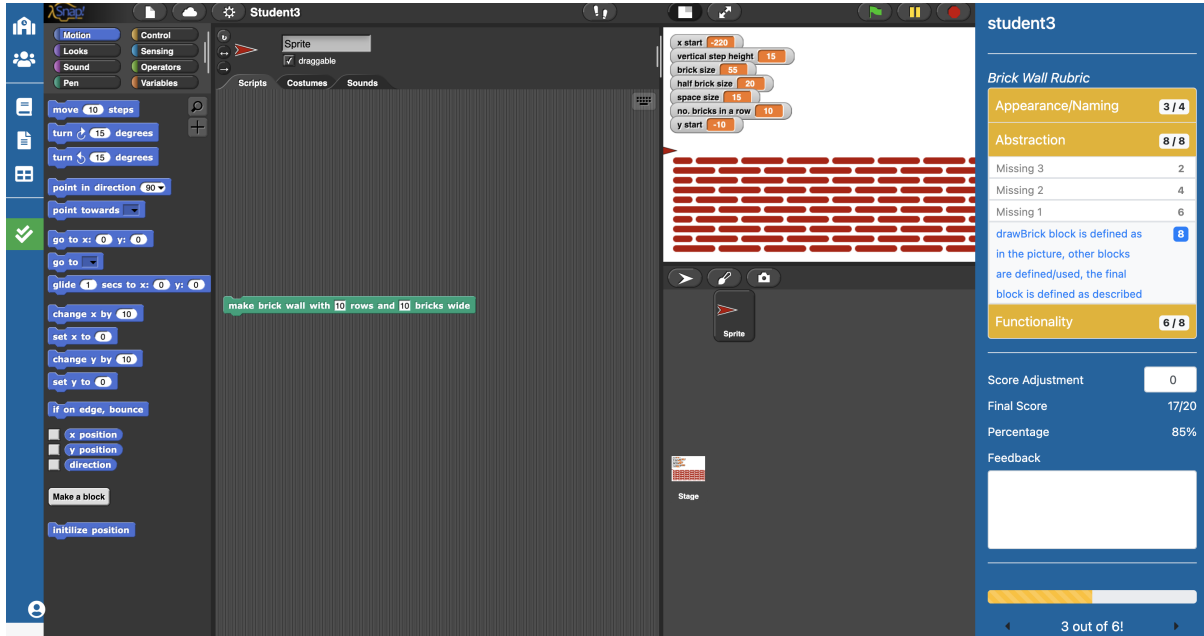


Figure 7.3 The grading page within GradeSnap.

Table 7.1 The study outline consisted of 5 parts.

Pre Survey & Grading Study	Chapter \ref{chap-six}
GradeSnap Tutorial	Practice Thinking Aloud and Grading in GradeSnap
Review Instructions and Rubric	Thinking Aloud - Reviewing their Rubric, Instructions
Grade 6 Submissions with Rubric	Thinking Aloud - Understanding Grading in GradeSnap
Post-Survey - Teacher Perceptions	Collecting usability and usefulness, tool's impact

way or their opinion after using GradeSnap. At the end, teachers answer interview questions about their grading goals, pain points, confidence, and their opinion of GradeSnap. They also reflect on this grading experience compared to their previous grading experience from the study in chapter 6.

During each zoom meeting, I will have 1-2 observers who are a part of the research team to take notes and observe actions the teacher takes, which may not be found in the audio transcription.

7.4.1 Assignment Teachers are Grading

During this study, teachers grade submissions for a provided assignment. The assignment is called Brick Wall, where students should program their project to draw a brick wall. The assignment instructions give directional support, such as the blocks they may need in the assignment and the different custom blocks a student should define and use (draw brick, rowA, rowB, Draw a Brick Wall with () Rows, where the () is a parameter). The assignment also gives students visual assistance, such as a picture of the definition of draw brick, a picture of what rowA and rowB should output, and a picture of what the final project should output.

The submissions are artifacts created by CSP high school students from a previous study [Cat16].

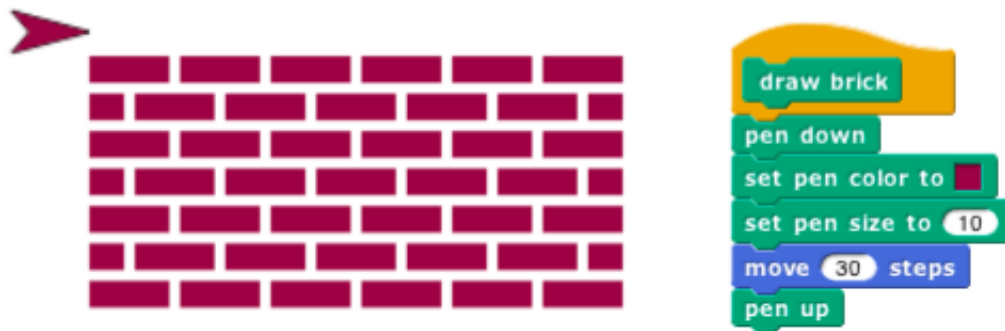


Figure 7.4 The final output of the Brick Wall assignment and an example of a defined custom block, provided in the instructions for the students.

These submissions were graded by two university researchers with K-12 teaching experience and post-baccalaureate degrees in computer science. As part of that study, each submission's score was translated to the level of achievement, specifically, each submission is also marked as High-, Middle-, or Low-Achieving. In order to observe the teachers grade a variety of submissions, 6 submissions were chosen, 2 from each of the levels of achievement previously determined.

Recall that teachers will use their own rubrics that they created in the previous study, as a starting point for their grading in GradeSnap.

7.5 Data Collection and Analysis

Each of the subsections below describes one of the types of data collected during these studies, the motivation for collecting this data, and how the data was analyzed.

7.5.1 Screen Recording and Audio Recording

As in the previous study described in chapter 6, all grading and interview parts of the study were recorded using Zoom, which provides automatic audio transcription. The research team reviewed and revised the automated audio transcripts when needed to ensure accurate audio transcription when analyzing the data. We analyzed audio transcripts using thematic analysis with four raters in multiple phases. Participant's screens were also recorded to help determine usability challenges, provide further observation into how a teacher grades, and provides another form of time collection.

7.5.2 Rubrics

During this study, participants will view and may edit their previously-created rubric for grading assignments, described in chapter 6. For participants who modify their rubric, I re-determined

the type of rubric they created. I include information about participants' grading methods and rubric types in their respective case studies described below. The 3 different types of rubrics used in this chapter include Analytic, Holistic, and Single Point rubrics. Analytic rubrics break down student work into subtasks, allowing teachers to evaluate student performance of specific criteria individually, whereas, Holistic rubrics allow for a teacher to evaluate the all criteria of a student's work at the same time [Bro16]. The Single Point rubric is binary and similar to a checklist in that a teacher can mark if a student's work contains the criteria [Flu10; Die00]. In addition to these types of rubrics, 2 important descriptors for rubrics are General and Task Specific. General rubrics can be used to evaluate multiple assignments with different instructions, grading on a set of less specific criteria, whereas, Task-specific rubrics are detailed and specific to an assignment [Bro16].

7.5.3 Thematic Analysis Process

For the thematic analysis process we used, there were 5 total phases listed below, including 1 data reviewing phase, 2 coding phases, 1 meet and discuss codes phase, and the final phase where researchers group codes into themes. The data was coded by 4 researchers.

Table 7.2 Each researcher coded 2 different participant's think aloud interview. They coded their first assigned interview (indicated with a 1), then the research met as a group, then they coded their second assigned interview (indicated with a 2).

	Alice	Beth	Christina	Deborah
Researcher 1	1		2	
Researcher 2	2	1		
Researcher 3		2		1
Researcher 4			1	2

1. Phase 1 - Review Data and Initial Coding

- Researchers meet to discuss thematic analysis process.
- Researchers are assigned the two interviews they will code, indicated by Table WW.
- Researchers spend 20 minutes reviewing their first interview and creating initial codes.
- Researchers discuss code they found, add descriptions, and ask questions.

2. Phase 2 - Coding First Interview

- Researchers review all current codes before they finish coding their first interview.
- Each researcher codes their first interview.

3. Phase 3 - Meet and Discuss Codes

- Researchers meet and share all codes they've added to the codebook

- Researchers acknowledge duplicate codes and remove if applicable
 - Researchers ask questions if needed
4. Phase 4 - Coding Second Interview
- Researchers review all current codes before they finish coding their second interview.
 - Each researcher codes their second interview.
5. Phase 5 - Grouping
- Researchers meet and create some initial groups
 - Researchers split into 2 pairs and group the codes during the meeting
 - Researchers return to a full group and review the codes with which are grouped into no themes or multiple themes for discussion; Then researchers refine the groups.

7.6 Results

First, I will present and define the themes found from tagging the think aloud grading session and interviews. Then, I will provide case studies for each of the participants. The case studies will present many of the themes, therefore, themes will be emphasized in case studies and discussed in the discussion sections.

7.6.1 Themes

Since the participants' time during the think aloud interview was while they were using GradeSnap, their tags and themes are mostly about the tool itself and what they would want in the tool. The six themes include: (1) Usable and Useful; (2) Baseline for Students; (3) Rubrics and Feedback in GradeSnap; (4) Grading in GradeSnap; (5) Gradebook and Assignments in GradeSnap; and (6) Auto-grading and Analytics.

7.6.2 Teacher Cases

These case studies (1) briefly review the participant's grading goals, pain points, grading method, and rubric from the previous study and (2) explore the participant's grading experience within GradeSnap from their think aloud session. I developed the case studies first reviewing observers' notes which included a play by play of how they completed the tasks, then referencing interview scripts for missing details, and then viewing their audio and screen recordings for details when they referenced something on their screen. Additionally, each case study contains information about their grading methods, new grading goals, and any modifications participants made to their rubric. The four participants' names include: Alice, Beth, Christina, and Deborah.

7.6.2.1 Alice

Alice created and used her **single-point, task-specific** rubric to grade 6 submissions in the previous study. When grading submissions with her rubric during the last study, **her grading goal** was to determine missing assignment elements, however, she shared that her normal grading goals include (1) helping students find their passion, (2) building their confidence in computing, (3) helping the middle-achieving students leave the class better off than they arrived, and (4) learning from the common student mistakes to understand how to modify her class work to support her students. Alice also shared that her biggest **pain point** is the large number of accurate solutions for a programming assignment, and the complexity of those solutions, making it difficult for her to evaluate if a student's code is considered "correct".

After Alice logged into GradeSnap (GS), she shared she felt that it is a **very intuitive** and inviting tool. Alice navigated through assignments and through the rubric with little issue while learning with the user steps in the tutorial. During the tutorial, Alice mentioned some simple **functionalities that would be helpful** to her and revealed insights into her grading practices. For example, when viewing the premade rubric for the tutorial in GS, Alice commented that she usually assigns more high scores than low scores, therefore, it would be more intuitive if the rubric sections were ordered highest to lowest to save Alice time. Additionally, Alice said she would appreciate a rich text format option when modifying or creating rubric categories and their category levels, such as a list.

She talked about how her rubric was more of a checklist, comparing her rubric to the example rubric that was premade for her to use during the tutorial. When Alice started grading the submissions, she continued to compliment GradeSnap's "clean" interface. As a reminder, Alice describes herself as a non-traditional grader who normally does not grade programs with a rubric or for correctness, however, **Alice was immediately drawn to GS. In fact, Alice even said "This is very nice. Wow! This might make me actually want to have a rubric"**, suggesting that the tool may influence her to grade with rubrics. **Not only does GradeSnap have the capability to support teacher practices, but GradeSnap also seems to have the power to change teacher grading practices - or at least serves as a powerful tool for reflection.**

One of the submissions for the teachers to grade was a submission that did not do anything the assignment requested. In fact, all of the participants mistook the submission as an incorrectly submitted assignment. When grading this submission, which was the fourth one she viewed, Alice realized that she could use the feedback portion of the rubric. For submissions 1-3, she did not use the feedback textbox to give feedback to the students yet. She commented that she is "so used to being anxious" and moving quickly through assignments to complete grading that she didn't even think about giving feedback. At this point, Alice outwardly said that **believed the tool removed that anxiety**, claiming "I really could've taken time to leave comments" because GradeSnap "is so wonderful." When grading with GradeSnap, Alice's grading goal was to follow the rubric. Alice shared that she felt that GradeSnap made her less anxious about the fact that each programming assignment has many possible correct solutions, addressing her **pain point**.

Alice seemed very positive and excited about GradeSnap throughout the majority of the interview.

However, once Alice realized that the students would complete their assignments through the GradeSnap portal, Alice became concerned with the pedagogical implications of GradeSnap. Alice shared that she does not believe in traditional grading and finds it limiting to the open-endedness and creativity of programming. To reconcile the differences between her pedagogical values and the tool, Alice suggested renaming GradeSnap to be less focused on the grading aspect. Because Alice's concern did not arise until the end of the interview, it is reasonable to conclude that the usability and usefulness of the tool made Alice overlook or reconsider some of her own pedagogical vision. This implies that the population of teachers who do not grade traditionally may still find value in the tool and may be open to adopting a new grading style.

7.6.2.2 Beth

Beth created and used her **analytical, task-specific** rubric to grade 6 submissions in the previous study. When grading submissions with her rubric during the last study, her **grading goal** was to determine if students understood and completed the assignment and to provide feedback for improvements which is one of her normal grading goals. Her other **normal grading goal** includes determining what is best for her classroom by reflecting on the class outcomes and averages. From the class outcomes, she can determine the topics her students did not comprehend fully, and would plan to incorporate these topics into her next lesson. Beth also shared that her biggest **pain point** is finding a good rubric for grading block-based language programs. When she uses rubrics created by others, sometimes she has found that her grading leads to unexpected results because the errors her students were making were unaccounted for.

After Beth logged into GradeSnap, she was immediately impressed with the interface of the **home screen with useful elements** such as quick links to external resources (i.e. BJC curriculum, BJC teacher's guide, BJC Piazza). Beth was very curious and unpromptedly started to **explore the application**. For example, Beth asked if the tool would support multiple courses and sections of courses, but before the researcher could answer she quickly found the answer herself on the course management page.

When Beth starts to complete the tutorial tasks, she reviews the tutorial assignment (Draw Square) and the premade rubric. She shares **expected usability functionality that was missing**, such as the ability to sort the category elements or view the rubric as a grid opposed to a list. Beth continues the tutorial steps by practicing to think aloud while grading in GradeSnap. Upon loading the grading screen, Beth (1) zooms in blocks, (2) organizes the student's code in the script space, and (3) runs the student's code. Initially when assigning a grade for an assignment, Beth tries to type in the grade but quickly realizes that when she clicks on the category name, the category reveals the category options in a drop down selection list. Beth shared other **usability suggestions** throughout the tutorial and the remainder of the think aloud session.

Beth chose not to review her rubric before she started grading the 6 submissions, instead she read through her rubric while she graded the first submission. Beth shared aloud that **since this assignment was mainly about abstraction and custom blocks, that it guided her grading**. While

grading, Beth felt constricted by her rubric. Instead of clicking on the number on the rubric to assign a score, Beth wished she could type the grade so that she could assign half points. Beth was persistent at reading the assignment code rather than relying on the visual output. When Beth grades submission 4, which was a low-achieving submission for the incorrect assignment, she gave the student the benefit of the doubt and assumed they submitted the wrong assignment. Instead of failing that student, she requested a resubmission within the feedback. For some of the submissions, Beth does try to fix the student's code in order to provide more feedback.

At the end of grading all 6 submissions, Beth shared that although she didn't give a lot of feedback for these submissions, **she felt that the tool would allow her to give more feedback to her students since the feedback textbox is on the same screen and can be updated at any time.** Beth also shared that she would like a button that would increase or add a baseline of 50% for students who got below a 50%. Beth shared her **grading goal** while using GradeSnap was to evaluate how students completed the task and give feedback so the students can do better. She explained that this is her normal grading goal for her Advanced Placement courses, but when she has a class that struggles, she awards points for attempting the assignment. Beth shared that while her **pain point is not solved**, it still enhances her grading process because this tool is very efficient. Beth shared she is pretty confident with grading block-based language assignments, however, she is more comfortable with grading Java, or a text based language, because they are easier since English is not her first language. Beth would use this tool to grade block-based programming assignments and she would recommend the tool to her peers.

When asked to compare GradeSnap to her normal grading environment, she said it is **almost exactly the same except with GradeSnap she can see everything on one screen.** She said she was happy she didn't have to upload the code or organize her screen to put her feedback right next to the code. Beth was very excited about GradeSnap and wanted to use it in her real classroom as soon as it's released. Beth stressed the importance of connecting GradeSnap to popular learning management systems to increase adoption by teachers, such as Google Classroom, Schoology, Canvas, and PowerTeacher. Beth uses Schoology and PowerTeacher for her class.

7.6.2.3 Christina

Christina created and used her **analytical, task-specific** rubric to grade 6 submissions in the previous study. When grading submissions with her rubric during the last study, her **grading goal** was to determine missing assignment elements, however, she shared that her **normal grading goal** is to give individualized feedback to her students for improvements. Christina also shared that giving individualized feedback is also her biggest **pain point** because of the number of students she teaches.

After Christina logged into GradeSnap, she said the dashboard was **attractive and clean**, and **reminded her of other dashboards** she uses. Christina **seamlessly navigated** to the tutorial assignment and reviewed the rubric. She commented that she usually creates and reads the rubric category items from highest points to lowest points, therefore, it would be more intuitive if the categories were ordered highest to lowest. Then Christina practiced thinking aloud while grading the tutorial

assignment. She shared that she liked the ability to expand and collapse the rubric categories. Assigning grades **using the rubric was intuitive** as she did not have any questions, however, she did ask for clarification of how to save her grades and feedback, which GradeSnap does automatically when the teacher advances to the next student submission. In the future, GradeSnap will have intermediate saving as well.

Christina reviewed her rubric before she started grading the 6 submissions. She **compared her rubric to the example tutorial rubric**, which was unintended, and said she “didn’t do a very good rubric” potentially because she was “kind of in a hurry”. She felt that **her rubric was not complex enough** for the Brick Wall assignment, however, she didn’t modify her rubric. For each assignment, Christina looked at the rubric to determine the next category she was scoring, found the element in the code, and scored it. For submissions 1, 2, 3, and 5, **Christina found different reasons that she needed to change her rubric**, such as requiring square bricks. When asked why she didn’t modify her rubric, she said it wouldn’t be fair to the students because they should know what it expected of them when the assignment is assigned. For submission 4, the low-achieving submission, Christina assumed they submitted the incorrect assignment and added feedback for the student to resubmit.

Christina shared her grading goal while using GradeSnap was to see how well the students were able to follow the instructions to build the brick wall. She shared that her **grading goal** didn’t change from the previous study and suggested that it was because she used the same rubric. Christina typically grades more informally, not looking for certain criteria but looking for patterns of students struggling or succeeding in order to give them meaningful feedback. When she does grade formally, she assesses what the student has learned and uses more stringent or precise rubrics.

Christina shared that GradeSnap **partially addressed her pain point** of wanting to provide individualized feedback for all her students. Her current grading environment is very similar to GradeSnap’s environment, however, it does not have a way for her to give feedback to her students in the same screen as her grading environment like GradeSnap does. However, GradeSnap doesn’t address anything to minimize the number of student submissions she needs to grade. She said **to really solve her pain point, she would need an autograder to assist with grading**.

Christina shared she is very confident with grading block-based language assignments. She would use this tool to grade block-based programming assignments, assuming that it connected to her learning management system, Canvas, and she would recommend the tool to her peers.

7.6.2.4 Deborah

Deborah created and used her **analytical, task-specific** rubric to grade 6 submissions in the previous study. When grading submissions with her rubric during the last study, her **grading goal** was to determine missing assignment elements, however, she shared that her **normal grading goal** is to give individualized feedback for improvement to her students. Deborah also shared that her biggest **pain point** is giving feedback for the large number of students she teaches.

Deborah was able to navigate and use GradeSnap superbly as a first time user. She felt the dashboard **display was pleasant** and the **application was easy to use**, experiencing little issue

while learning with the user steps in the tutorial. During the tutorial, Deborah viewed the premade rubric and mentioned she is used to viewing a rubric horizontally, however, she said she could get used to viewing the rubric categories vertically. She did not have any problems or questions using the tool, comfortably graded the tutorial assignment and **excitedly said she was ready to grade the other assignment** with the tool.

During the grading portion she was happy to see the student submissions, her created rubric, and an grading element on the same page. Having multiple tabs and switching between those tabs constantly was a pain point when grading. Deborah felt as though having the student assignment, rubric, and grading on the same page would make grading easier and may decrease the amount of time she spends grading one assignment. Deborah chose not to review the assignment instructions or her rubric prior to grading the 6 submissions. For submission 4, the low-achieving submission, Deborah assumed they submitted the incorrect assignment and added feedback for the student to resubmit.

After grading the assignment, Deborah shared that her **grading goal** when using GradeSnap was to see if the students learned and followed the instructions, which was her grading goal during the previous study when she used the rubric. Deborah also shared that GS **addressed her pain points** of tab flipping and making it easier to give individualized feedback, solely because it is in the same interface. Deborah says she is pretty confident with grading block-based language assignments, but shared that there is always more she can learn. Deborah said she would be excited to use GradeSnap and would recommend it to others. **When asked if she had any questions, she asked when she could start using GradeSnap for her class!**

7.7 Discussion

In this section, I explore differences in how participants graded within GradeSnap, and investigate how GradeSnap impacted a teacher's grading goals and/or pain points from the previous study. Then I discuss the implications of themes found in the participants' case studies. When referencing a participant in this section, I will use A for Alice, B for Beth, C for Christina, and D for Deborah.

7.7.1 Grades

In this section, I compare how the teachers graded and how their averages compared to the study in chapter 6. Table 7.3 shows all the participants' grades for each submission from the grading session. Each of the participants' comparisons below start by reviewing the participants' grading habits during the study in chapter 6 and reviewing their grading habits and grades during this study.

Previously, **Alice** thought about students' reactions when she gave grades, even for the submissions during this study. It was obvious that students' reactions make an impact on how she grades for her classroom. Her grades ranged from 89 - 98 percent during both grading sessions 1 and 2, with grade averages of 91.83 to 93 percent, respectively. During grading session three, her grades ranged from 70 - 85 percent, with an average of 77.5 percent, and her grading habit changed, which

was reflected in her stated grading goal in “respecting the rubric.” Alice said she did not consider how a student would react when she used GradeSnap to grade the similar submissions. When asked why, she said she felt less anxious and just decided to respect the rubric, almost not questioning whether it was the correct way to grade the submissions. This finding suggests that a tool formalizes the grading process and could allow teachers with grading anxiety to grade with more ease.

Previously, **Beth** focused on learning what more her students needed from her, finding common misconceptions she should revisit in a future class to reteach a concept. Her grades ranged from 70 - 95 percent in grading session 1 with an average of 80.83 percent and from 30 - 95 in grading session 2 with an average of 68.33 percent. During grading session three, her grades ranged from 50 - 85 percent, with an average of 68.0 percent, and her major grading habit stayed the same; trying to learn what students commonly struggled with. However, one similarity between grading sessions 1 and 3, not captured during grading session 2, was the ability to provide students who tried with the baseline of 50 points. Her average between grading session 2 and 3 are within a percent, which is interesting since she provided the baseline buffer for students during the third grading session but not the second grading session.

Previously, **Christina** graded students with lower achieving levels easier when not using a rubric, causing her to miss common mistakes amongst her students who may need the most help, however, she is good at scoring students who are doing well on assignments. Her grades ranged from 70 - 100 percent and 37.5 - 100 percent, with averages of 90 percent and 79.17 percent, in grading sessions 1 and 2, respectively. During grading session three, her grades ranged from 60 - 100 percent, with an average of 84.5 percent, and her grading habit did not change. Her average of 84.5 is directly between her grade averages during the previous two grading sessions. When grading student 5, Christina made a mistake while grading their work, however, it was only one mistake, which was less compared to her mistakes during grading session 1 and similar to grading session 2. Therefore, this finding suggests that she grades with a grading tool similarly to how she grades with a rubric.

Previously, **Deborah** created rubrics with expectations beyond the assignment instructions, suggesting she expects more from her students for this assignment. Additionally, she graded stricter compared to the other participants. Her grades ranged from 75 - 95 percent, with an Incomplete and an average of 84.60 percent, in grading session 1, and 15.63 - 78.13 percent in the grading session 2 with an average of 57.29 percent. For her grade range, I included the grade without modifications made when analyzing her grade in chapter 6. During grading session three, her grades ranged from 33.84 - 78.13 percent, with an average of 58.35 percent. This grade range and average is almost the same as her grading from grading session two. Her grading habit did not change since she continued to follow the same rubric, which may have impacted her grading habit in grading session 2. This suggests that for some teachers, it does not matter if the grading instrument is embedded in a digital tool or if it is a rubric they use to grade assignments.

Table 7.3 The participants’ grades for each submission for the third grading session. Each participant row is denoted with their name. The levels row denotes the achieving level of the submission. The Avg. column is the calculated average of grades given during that grading session for that teacher. The Avg row (along the bottom) is the calculated average for grades for that student.

	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6	Avg
Level	Middle	Low	High	Low	Middle	High	
A	75	70	85	0	80	85	77.50
B	70	50	85	0	50	85	68.00
C	75	60	100	0	100	87.5	84.50
D	51.03	33.84	68.75	NG	60	78.13	58.35
Avg.	67.76	53.46	84.69	0	72.50	83.91	

7.7.2 Rubrics Changes

To address **Hypothesis H2**: Teachers will make changes to their rubrics within GradeSnap to make their rubric more analytical or task oriented. None of the participants modified their rubrics therefore my prediction was not true and teachers did not feel the need to edit their rubrics within GradeSnap. Therefore, this hypothesis H2 is rejected.

Two of the participants (A,C) chose to view their rubrics before they started grading students, whereas, the other two participants (B,D) solely viewed their rubric as they graded student’s submissions in GradeSnap. Currently in GradeSnap, a teacher can only modify their rubrics from the rubric page, therefore, unless participants B or D say a problem with their rubric, they would need to go to the rubrics page. This may have reduced the chance of either of these participants would modify their rubric.

One of the participants (C) viewed their rubric said it wouldn’t be fair to modify their rubric since they have already “assigned” the assignment and have created and used the rubric. She gives her rubrics to her students to give them further guidance in what is expected of them for an assignment so she doesn’t edit her rubrics after she assigns the assignment. The other participant (A) compared her rubric to the example tutorial rubric, however, she didn’t feel the need to edit her rubric.

7.7.3 Comparing Teachers Grading Goals

To address **Research Question R1**: How does grading with GradeSnap impact a teacher’s grading goals?, and **Hypothesis H1**: Teachers’ grading goals would not change from the grading goals in chapter 6, I compared the participant’s self-reported grading goals. Table 7.4 shows how the teacher’s grading goals changed with a tool, such as a rubric or GradeSnap, or without.

Alice’s grading goals changed during each grading session. This could be because she typically does not grade these types of assignments in her classroom. Respecting the rubric for Alice meant not “fudging” the grade depending on how she felt the student would feel.

Beth’s grading goals were similar from each grading session, therefore her grading goal did not

change with GradeSnap. She was the only participant who searched for what the student missed and aimed to provide feedback for improvements.

Christina's grading goals did not change from grading session two (with the rubric) and grading session three (with GradeSnap), which was predicted. This could be because she was using the same rubric in both sessions. In fact, when asked if her third grading goal changed from the previous session, she said no because she didn't change her rubric, implying that if she did change her rubric, it would've also changed her grading goal. This could explain why her second and third grading goals were different from her first grading goal. Another reason why her grading goal could've changed from grading session one to the other two could be because she typically does not grade class assignments with rubrics but more so as completion grades.

Deborah's grading goals three encompassed both her grading goals from grading session one and two. When asked she didn't feel that her third was different from her second grading goal, however, her third goal also addressed figuring out what the students learned in order to give feedback. Possibly she did not feel that her grading goal changed since it encompassed the second grading goal. It is interesting that her grading goals changed even though her rubric did not change since during the last study, Deborah mentioned that the rubric change is what caused the change between her initial two grading goals.

Two of the participants (B,C) did not change their grading goal between the second and third grading sessions and two of the participants (A,D) did change their grading goals. Therefore, this third hypothesis is not confirmed.

Table 7.4 Comparing grading goals from the first grading session (without a rubric), the second grading session (with a rubric), and the third grading session (with GradeSnap).

	Previous Study		Current Study
	Grading Goal One	Grading Goal Two (with Rubric)	Grading Goal (GradeSnap)
A	Find passion for students; build their confidence	Determine what's missing from the assignment	Respect the rubric
B	Determine if students understood and completed the assignment; provide feedback for improvements		Evaluate how students completed the task and give feedback so the students can do better.
C	Give individualized feedback for improvement	Determine missing items from assignments	See how well the students were able to follow the instructions to build the brick wall
D	Give individualized feedback for improvement	Determine missing assignment elements	Determine if the students learned and followed the instructions to give feedback

7.7.4 Solving Teachers Pain Points

To address **Research Question R2**: How does grading with GradeSnap impact a teacher's pain points?, for each teacher, I determined whether they felt their pain points were addressed and, if not, how they think a grading tool could address them.

Alice's biggest pain point is the large number of accurate solutions for a programming assignment,

and the complexity of those solutions, making it difficult for her to evaluate if a student's code is considered "correct". Alice felt that GradeSnap made her less anxious about the fact that each programming assignment has many possible correct solutions, addressing her pain point. This was unexpected that Alice felt GradeSnap removed her pain point, however, this results may be explained by the fact that maybe her pain point is more about her anxiety with grading.

Beth's biggest pain point is finding a good rubric for grading block-based language programs which account for the mistakes her students make. Beth shared that while GradeSnap did not solve her pain point, GradeSnap still enhances her grading process because this tool is very efficient. In order to address her pain point, she shared she needs good premade rubrics for the assignments within the curriculum.

Christina shared that giving individualized feedback is also her biggest pain point because of the number of students she teaches. Christina shared that GradeSnap partially addressed her pain point of wanting to provide individualized feedback for all her students because the feedback response box was located on the same screen as the student's submission and the rubric, therefore making it quicker and easier to supply feedback. In order to really solve her pain point, she would need an autograder to assist with grading.

Deborah's biggest pain point is also giving feedback for the large number of students she teaches. Contrary to Christina, Deborah shared that GradeSnap did address her main pain point by making it easier to give individualized feedback, solely because it is in the same interface. Additionally, she shared that GradeSnap addressed a second pain point of hers by reducing the number of tabs needed to grade her student's work by having her rubric and feedback located in the same interface as the student's submission.

Therefore, half of the participants (A,D) felt that GradeSnap solved their pain points. In order to continue solving the other participant's (B,C) pain points, GradeSnap would need to have autograding capabilities and have premade or template rubrics to support teachers.

7.7.4.1 Giving Feedback

To investigate **Hypothesis H3**: Teachers will give at least as much feedback as when they graded with a rubric during the study presented in chapter 6, I compare feedback teachers gave during the previous study in chapter 6 to the feedback teachers gave while using GradeSnap. As a reminder, there were a total of 12 submissions; 6 for the first and second grading session and 6 for the third grading session.

During the study from chapter 6, three of the participants (B,C,D) wrote feedback for the submissions when they did not use a rubric. None of the participants wrote feedback for the submissions when they did use a rubric. During this study, three of the participants (B,C,D) wrote feedback for half of the submissions while using GradeSnap. Since the teachers provided at least as much feedback when using GradeSnap as when they graded with a rubric, this hypothesis H3 is accepted.

A possible explanation for the lack of feedback provided during the study where participants used rubric might be that participants saw their rubric as their feedback for their students. With

informative enough rubrics, this could be acceptable, however many of their rubrics, discussed in the chapter 6, would require further information for students to know how to improve their programming. As a reminder:

- Christina’s rubric would be useful to provide to students as feedback for what they did correctly or incorrectly, with the assumption that the student completed the project as she expected since her rubric was fairly specific.
- Beth’s rubric would be useful to provide feedback to students with an additional comment pointing out which of the 3 grading criteria of the category the student missed.
- Deborah’s rubric could be given to students as feedback for a grade on a project, however, she would have to supply more additional and individualized comments to explain the score.
- Alice’s rubric could be useful as a check for students to know if they completed the overall elements of the assignment, however, without any additional comments, her rubric would not assist students in understanding what they did incorrectly.

However, to further investigate the participants’ feedback and compare their feedback when using and not using GradeSnap, I analyzed the quality of their feedback. As mentioned earlier, during the study from chapter 6, three of the participants (B,C,D) wrote feedback for the submissions when they did not use a rubric.

Alice did not give feedback during any of her grading sessions.

Table 7.5 Beth’s feedback for the submissions from the first grading session and the third grading session.

Grading Session One (No Rubric)	Grading Session Three (GradeSnap)
1L - missing drawBrick, lines aligned on the right, other custom blocks other than drawBrick, Will it draw the correct number of lines 2M - different length rows 4L - only has only rowA and rowB custom blocks 5M - different length rows 6H - Aesthetic	5M - check your initialize pen block - the pen size 6H - Comments?

Beth gave feedback during both grading sessions, (1) when not using a rubric and (2) when using GradeSnap. While grading without a rubric, Beth supplied detailed feedback to the Low (1,4) and Middle (2,5) achieving students about what they were missing from the assignment. While grading with GradeSnap, Beth did not start leaving feedback until the last two students. At student 5, when Beth realized she hadn’t supplied any feedback, she shared that she would probably be able to give more feedback than normal for her classroom even though she did not utilize the feature much during the grading session. The following is a comparison of their feedback for the 2 sets of submissions.

Table 7.6 Christina’s feedback for the submissions from the first grading session and the third grading session.

Grading Session One (No Rubric)	Grading Session Three (GradeSnap)
1L - not building wall with variable (input) number of rows; rows don't end evenly; bricks have rounded edges; not using conditionals and getting double the number of rows 2M - rows are not exact same length 4L - missing block to draw variable number of rows, only draws 2 rows 5M - double check that rows end up even with different brick size because they go off the stage as it is	2L - Draw brick block is there, but it does not actually draw a brick. 4L - I think you submitted the wrong project. Please resubmit the Brick Wall project and I will re-grade it.

Christina gave feedback during both grading sessions, (1) when not using a rubric and (2) when using GradeSnap. While grading without a rubric, Christina provided detailed feedback to the Low (1,4) and Middle (2,5) achieving students about what they were missing from the assignment. While grading with GradeSnap, Christina gave feedback to the Low Achieving students. The following is a comparison of their feedback for the 2 sets of submissions.

Deborah gave feedback during both grading sessions, (1) when not using a rubric and (2) when using GradeSnap. While grading without a rubric, Deborah supplied detailed feedback to every student. While grading with GradeSnap, Deborah gave feedback to 3 of the students; two high achieving and one middle achieving. The following is a comparison of their feedback for the 2 sets of submissions.

Table 7.7 Deborah’s feedback for the submissions from the first grading session and the third grading session.

Grading Session One (No Rubric)	Grading Session Three (GradeSnap)
1L - Program functions, but does not have instructions on how to operate 2M - Follows instructions but could have added more - green flag when clicked 3H - Good use of custom blocks, if statements and loops 4L - Sit with the student and be sure they understand the instructions. 5M - The issue is that the brick isn't a brick. Redo for full credit. 6H - Why the spaces between the lines?	3H - Good job! 5M - Code should be connected for a more easily read program. 6H - The right end of your lines don't line up correctly.(-2)

7.7.5 Themes

The following themes were found from tagging participants’ think aloud data during the grading session and from the interviews. Since the participants’ time during the think aloud interview was

while they were using GradeSnap, their tags and themes are mostly about the tool itself and what they would want in the tool. I refer to teachers' case studies and discuss implications in the remainder of this section.

7.7.5.1 Usable and Useful

All four participants (A,B,C,D) shared that they felt GradeSnap was both useful and usable. Usable refers to how easy it is to use an application whereas useful refers to if a user is able to accomplish a task or objective.

All four participants (A,B,C,D) said that GradeSnap was usable. One participant (A) shared they felt the interface was great which would make using the tool easier to use. When asked what ways GradeSnap was usable, instead of sharing specific uses of how GradeSnap was usable, three participants (A,B,C) shared specifics on the changes they felt would make it more usable. Two participants (A,B) said changing the cursor symbol when hovering over a clickable link or items (such as the menu items in the left margin) would make it more usable. Another participant (C) shared that improved scrolling on the grading screen would make GradeSnap more usable. And the other participant (D) shared that making it possible for a teacher to view their gradebook as points or percentage would make GradeSnap more usable since some teachers primarily use points and not percentages. Towards the end of her think aloud interview, one participant (C) shared she felt that GradeSnap is "very intuitive to [her] because of the fact that it looks already like a lot of things that [she's] used to working in". She (C) continued to exclaim "it was very intuitive and very easy to navigate through."

All four participants (A,B,C,D) said that GradeSnap was useful. Each shared that they were able to complete all that they wanted to while using GradeSnap. Two of the participants (B,C) commented that it was most useful to have the student's code right next to the rubric where they can assign the grade. One teacher went so far to say that "having the assignment and being able to consistently grade and have the rubric right there was very convenient," continuing on to say that "[she] liked to be able to click on the next student" from the grading screen instead of returning to the list of student submissions to select the next submission to grade.

Two of the participants (B,D) asked when they could use GradeSnap, exclaiming they are happy to be beta testers and can ignore the small usability conflicts (previously shared). This result suggests that for some of these participants (B,D), they would value usefulness ahead of usability if that would enable them to use GradeSnap. This may suggest that these participants feel the need for a tool like GradeSnap.

One of the participants (A), who did not comment further on GradeSnap's usefulness beyond that she felt it was useful, became skeptical because she does not believe in grading learning activities. Although she did not comment on further usefulness of GradeSnap at this point in the interview, she suggested (when answering a different interview question) that GradeSnap removed her anxiety of grading, making it possible for her to grade without constantly talking about how her students

would feel about the grade or without feeling lost in grading. This may suggest that other teachers who have anxiety when grading could receive some comfort from GradeSnap.

7.7.5.2 Baseline for Students

Three of the participants (A,B,C) said they create a baseline, or a grade for students to receive if they tried even though they did not succeed in completing the assignment. Alice's baseline (or minimum grade) was 70

Although these three participants share they use a baseline when grading their classroom's assignments, none of them found a way to add a baseline when they created a rubric. In GradeSnap, the grade saved in the gradebook is calculated based on their rubric. Teachers are able to adjust their calculated grade from the rubric either in the gradebook or the grading page. These teachers adjusted their grades within GradeSnap, aligning more with their normal grading practices compared to grading from the previous study in chapter 6. Participants were excited about this feature and gave suggestions to make the feature even more useful.

7.7.5.3 Rubrics and Feedback in GradeSnap

Three participants (A,B,C) mentioned wanting rich text formatting for their rubric category items. One participant explained the formatting could save her time when grading (C). One of the participants shared they would like the ability to import rubric into the system (B). All of the participants (A,B,C,D) wanted to be able to designate feedback when making their rubric so when they select the category item when grading a student's submission, the corresponding feedback would append to the current feedback in the feedback textbox.

7.7.5.4 Grading in GradeSnap

On the grading page, teachers are able to adjust the original grade percentage in case they felt the rubric was too strict or the grade was not a good representative of the student's work. When teachers adjust the score on the grading page, they change the percentage by either (1) increasing or decreasing with an arrow or (2) by typing in the amount to add or subtract in the textbox. Two of the participants (B,D) would prefer to adjust the score instead of the percentage.

To assist with giving feedback, one of the participants (D) suggested for GradeSnap to include stock comments (or customizable comments made by the teacher). She mentioned that stock comments would specifically be helpful for students who do well and would not receive feedback from corresponding rubric category items.

7.7.5.5 Gradebook and Assignments in GradeSnap

Even though the majority of the think aloud interview session was using the grading portion within GradeSnap, all the participants also had plenty of feedback or requests on the Gradebook page and

Assignment page. This may be because the participants are accustomed to having a Gradebook and Assignment page for their class so they are more likely to know what they want in those pages.

On the assignment page, the participants can see which students submitted the assignment, which submissions the teacher has graded, and how they graded each submission according to the rubric they used. Two of the participants (B,D) shared it was really useful for them, but they each would like to know (1) when the student submitted their submission (D) and (2) an indicator for the teacher to know when they've asked a student to resubmit their submission (B).

On the assignment and gradebook page, one of the participants (B) wants to be able to see the feedback they've provided for each student submission. This could be to allow the teacher to not need to click to go to the grading page to view the student's submission. Some of the participants like percentages (B,D) in the Gradebook whereas others wanted scores (A,C). A compromise would be to allow teachers to choose which they want to use for their class.

After a teacher uses the rubric in the grading page, the grade is saved in the gradebook. In both the gradebook and grading page, teachers are able to adjust the original grade percentage. This grade adjustment is visibly noted (for the teacher) in the gradebook so they can remember that they changed the grade outside of using the rubric. One of the participants (C) shared that they liked they could see when they've 'fudged' the score so they could remember. In addition to being able to see a score adjustment, another participant (A) exclaimed it may be useful for teachers to have private notes about students, accessible from the Gradebook. They added on, suggesting the ability to 'star' students to denote students needing more help.

Two of the participants (B,C) shared they liked that they could view the average grade for an assignment and the overall grade for a student, however, one participant (B) would want to see the average grade for the assignment at the top of the gradebook and the overall grade for a student in the second column (directly to the right of the student's name). This same participant (B) expressed the need for connecting GradeSnap with common Learning Management Systems, such as Schoology, PowerTeacher, Canvas, and Google Classroom. This participant uses Schoology and PowerTeacher and says that the grades automatically sync from Schoology to PowerTeacher. She compromised and shared that an export function would be a good substitute while GradeSnap is not connected to any lms.

Other suggestions for the Gradebook page include: Assign different weights for different types of assignments (B) Would like a "set all grades" button for when something is graded on completion (B) Add more organization to the assignment view (i.e. "Module 1" or "Unit 1") (B) Ability to sort the Gradebook by assignment, grade, first name, last name..etc (B) Add a button to show the pairs or groups working on an assignment and put all their names at the top of the submission when grading (B)

7.7.5.6 Autograding and Analytics

All four participants (A,B,C,D) want an autograder added to Gradesnap. One of the participants (B) said they would want the option to release grades from the autograder automatically, but sometimes

they would want to review students' work before sending their grades. Two of the participants with a concern for giving individualized feedback to large classes (C,D) said an autograder would reduce the amount of time needed for them to grade so they could instead focus on helping those students in need of more assistance.

After using GradeSnap, one participant (A) reflected on more things that the tool could do for her. She shared that she would like to receive insights after she finishes her grading. For example, she suggested that another useful feature could indicate which students may need additional help and what they need more help on. Another participant (B) mentioned she tries to determine how to adjust her next class based on her students' submissions, therefore, would probably also benefit from this addition to GradeSnap as well.

7.8 Conclusions and Contributions

This chapter investigates **RQ2d**: How do teachers grade with and perceive GradeSnap? This research question was refined into two sub questions, R1 and R2.

To address **research question R1**: How does grading with the GradeSnap tool impact teachers' grading goals?

To address **research question R2**: What is important for teachers when grading a block based language assignment?, I asked participants if they felt that GradeSnap solved their pain points. Half of the participants (A,D) felt that GradeSnap solved their pain points. In order to continue solving the other participant's (B,C) pain points, GradeSnap would need to have autograding capabilities, and have premade or template rubrics to support teachers.

To address **hypothesis H1**: Teachers' grading goals will not change from the grading goals in chapter 6. One of the participant's (A) grading goals changed during each grading session. The other three participants (B,C,D) all had similar grading goals as from the last session. Therefore, this partially contradicts and partially confirms my prediction.

To address **hypothesis H2**: Teachers will make changes to their rubrics within GradeSnap to make their rubric more analytical or task oriented. None of the participants modified their rubrics therefore this hypothesis is rejected.

To address **hypothesis H3**: Teachers will give at least as much feedback as when they graded with a rubric during the study presented in chapter 6. Since none of the participants provided feedback for the submissions when using a rubric in the previous study, and since three of the participants (B,C,D) provided feedback for half of the submissions while using GradeSnap, this hypothesis H3 is accepted.

Overall, this study shows that teachers were able to effectively use GradeSnap to grade programs within its interface, and two teachers felt it addressed their pain points in grading. Two other teachers felt that autograding or pre-made rubrics would address their grading pain points. For three teachers, their grading goals did not change. For one teacher, the two studies combined seemed to produce a reflection and evolution of the teacher's beliefs about grading and her own goals for student feedback

and support. This was a very promising result, suggesting that even teachers who may not believe in the value of grades perceive value in a tool that can help the teachers provide feedback and support to their students. Teacher feedback seemed to improve with the use of GradeSnap when using the tool, rather than a rubric alone. However, the amount of feedback teachers provided with the rubric was strongly influenced by the teachers having graded the same assignments in the prior study and already writing student feedback. The results here suggest, however, that teachers felt confident in the use of GradeSnap to provide students with written and numeric grade feedback on their work.

CHAPTER

8

CONCLUSION

This chapter describes the contributions of this work as well as present important, more generalized findings for the research community.

8.1 Contributions

The goal of this dissertation is to investigate how to help in-service teachers, especially those with little prior computer science or programming experience, prepare to teach computer science, with two research questions:

RQ1: How do we help in-service teachers learn to teach programming with summer professional development?

RQ2: 'What is important for teachers when grading a block based language assignment?

RQ1 was investigated over a period of three years from 2016-2018, described in Chapter 2, where I refined the design of teacher professional development to reduce the amount of time teachers needed to spend preparing for teaching, while still maintaining high levels of teacher perceived readiness to teach the BJC course. Overall, my research suggests that teacher professional development for in-service teachers, should be carefully designed to limit the amount of independent work that teachers must do, especially if teachers are learning a discipline that is new to them. Furthermore, teacher professional development for computer science should continually be adapted and revised based on teacher feedback, and should build in structured time for learning, reflection, discussion with peers, learning from experienced teachers, and for explicit practice of teaching responsibilities. These design features help teachers feel prepared for teaching a new computer science course. Based on my experience with in-service teachers, I realized that much teacher learning occurs during the

academic year when teachers interact with students, and that the next challenge teachers faced after building their readiness for teaching the class, was providing students with feedback on their programs.

Based on this insight, I established **RQ2** on how to design an effective teacher support tool. I took a user-experience (UX) research focused approach to design GradeSnap. Within RQ2, I leveraged user-experience research methods to investigate the following, more specific research questions:

- **RQ2a-GS design:** What features do teachers want and need in an online grading tool for programming projects?
- **RQ2b-GS prototype:** How do teachers perceive our prototype GradeSnap tool, how likely are they to use or recommend it, and do these factors change with tutorial usage?
- **RQ2c-Grading:** How do teachers grade programming projects with and without rubrics, but without tool support?
- **RQ2d-GS evaluation:** How do teachers grade with and perceive GradeSnap grade support? How does teacher-use of the grading tools differ based on their expertise/experience or goals for student feedback?

Chapter 3 outlines the relevant research on teacher grading, grading within computer science, and teacher support tools, as well as user experience design research methods. Chapter 4 investigates RQ2a, describing the initial design of GradeSnap, implemented through a UX spring using paper prototyping and interviews with teachers, to determine what features were most important in a tool to support grading for block based programs. Chapter 5 investigates RQ2b, describing the first digital prototype of GradeSnap, the features implemented, and a user study with 33 teachers to determine the importance and impact of using a tutorial for helping teachers learn to use GradeSnap. Chapter 6 investigates RQ2c, discussing a study of 4 BJC teachers, on how they grade block-based programs using their own process, then while using a rubric of their own design. In this study, I found that teachers have diverse goals for grading, and while they had a typical process, it did not always involve an explicit rubric. The study results showed that teachers need support to achieve their diverse goals, and also that they could see some usefulness in defining a rubric. However, surprisingly, teachers did not often revise their rubrics once created, stating that, if a rubric existed, they would share it with their students and then leave it unchanged to be fair to students. Chapter 7 investigates RQ2d, presenting the final study of GradeSnap with 4 in-service BJC teachers. The goal of this study was to determine how teachers perceived the tool, how useful it could be for grading and student feedback in their classes, and to determine how teacher practices might have changed based on using the tool. Our results showed that teachers used the rubric they created in the prior study, again citing consistency and fairness for students, despite the fact that the rubrics did not result in grades that aligned with their grading goals. Teachers did find GradeSnap quite easy to use and wanted to use it in their classrooms. Most interestingly, we found that while teachers could see how the tool could be modified to meet some of their goals for grading (i.e. by suggesting a way to

provide a minimum, or baseline, score to all students), they did not make use of the ability that the tool provides to revise their rubrics. Encouragingly, however, teachers did provide more feedback for students while using GradeSnap than they did with a rubric alone.

8.2 Summary of Contributions

This dissertation contributes the following:

1. A study on how in-service teacher professional development should be designed to limit the time needed to 5 summer days while promoting teacher confidence in their preparedness for teaching computer science
2. A user research study to determine the features teachers need and want in a tool for block-based programming assignments
3. A user research study with teachers to determine the usability of the prototype GradeSnap tool and the effectiveness of a tutorial designed to help teachers use it
4. A study on how teachers grade block-based programming assignments for an introductory computer science course, both with and
5. The new GradeSnap tool for teachers to grade block-based languages
6. A study showing how teachers use GradeSnap and its implications for future grading tool design

8.3 Concluding Thoughts

Beyond the individual findings provided at the end of the prior chapters, I present 3 important takeaways for the research community. First, researchers' beliefs of what teachers know already about grading block based programming projects and rubrics does not necessarily align with what teachers actually know about grading projects and developing rubrics. Second, from the GradeSnap study, I learned that early adopters are not the same as the newer block-based programming teachers in the field. It is important for researchers to continue to survey newer populations to learn how they are currently grading or facilitating their classrooms, else previous research and assumptions may bias future research endeavors. And third, using a tool or system like GradeSnap can help teachers apply a more equitable approach for giving feedback to students. As some teachers may not want to take off points or give letter grades for incorrect code, the openness of the system allows teachers to give formative feedback, instead of only holistic or generic grades, so students can learn from their coding mistakes and improve their programming skills.

Given the findings from the last two studies (where teachers were grading without rubrics, with rubric, and grading with GradeSnap), I've identified recommendations for how I would change the

professional development (from the first study) to further support teacher grading activities during the professional development (PD).

During the existing Beauty and Joy of Computing PD, teachers program an example project and then they grade each other's work. This is a perfect opportunity to introduce teachers to GradeSnap, allowing them to understand how they could grade programming projects. Since I found in study 3 that teachers with and without a tutorial found GradeSnap intuitive and since I found in study 5 that teachers were able to make a smooth transition from grading with a rubric to grading within Gradesnap, then we would be able to allow BJC teachers to explore GradeSnap during the PD with potentially minimal problems. With this experience, teachers could use GradeSnap and determine if they want to use it for their own classroom. Additionally, due to its usable and intuitive nature, if the teachers do not get a guided introduction to GradSnap while in their different PD programs, it is likely that they could independently learn how to grade with GradSnap using the built-in tutorial. Furthermore this also implies that teachers who enter the education system laterally or are from a different discipline, could learn how to use rubrics to grade block based language projects through GradeSnap.

Additionally throughout the last two studies (where teachers graded without a rubric, with a rubric and with GradeSnap), I found that teachers would benefit from learning more information about different types of rubrics and their uses. In order to best help teachers, I would recommend that researchers and teachers should work together to create effective rubrics which align with expected learning outcomes and enable teachers to grade how they feel they need to for their classrooms.

In order for researchers and teachers to collaborate in creating rubrics, I would recommend a process similar to Stegeman's process for developing rubrics to assess programming quality [Ste16], where the researcher develops a draft or template task-specific and analytic rubric for the teacher to (1) give initial feedback to the researcher, (2) use the rubric to grade a sample of projects, (3) adjust the rubric, (4) determine if the rubric still aligns with grading goals, (5) regrade the same assignments and reflect on the changes, and (6) grade a new set of projects and again reflect on the changes. An iterative process will allow for the researcher to learn how to fine tune the rubric for the teacher and the teacher will develop their own rubric creation skills in the process. From this process the researcher may be able to develop a generalized or learning-based rubric (as utilized by Cateté [Cat16]) for teachers to use or learn how to modify for their lessons without existing rubrics. Additionally, teachers would learn how to add baselines or lower threshold, a very desired feature, to their grading practices when using rubrics because they do not discourage students who are new to programming and need further assistance. Overall, a tool like GradeSnap can help mediate the iterations between researchers and teachers as they learn to develop high quality rubrics together for both teacher and student growth.

BIBLIOGRAPHY

- [Cod] *2020 State of Computer Science Education: Illuminating Disparities*. 2020.
- [Abe13] Abel, T. D. & Evans, M. “Cross-disciplinary Participatory & Contextual Design Research: Creating a Teacher Dashboard Application.” *IxD&A* **19** (2013), pp. 63–76.
- [Aho09] Ahoniemi, T. & Karavirta, V. “Analyzing the use of a rubric-based grading tool”. *ACM SIGCSE Bulletin* **41.3** (2009), pp. 333–337.
- [Aho08] Ahoniemi, T. et al. “Improving pedagogical feedback and objective grading”. *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, pp. 72–76.
- [AL12] Aidan Lane, B. M. & Mullins, J. *Simulation with Cellular A Project Based Introduction to Programming*. first. BlockBooks Series. Online: <https://github.com/MonashAlexandria/snapapps>. Monash University, 2012.
- [Ast12] Astrachan, O. & Briggs, A. “The CS principles project”. *ACM Inroads* **3.2** (2012), pp. 38–42.
- [Bal18] Ball, M. “Lambda: An Autograder for snap”. *Masterscriptie. EECS Department, University of California, Berkeley* (2018).
- [Bar11] Barney, S. et al. “Improving students with rubric-based self-assessment and oral feedback”. *IEEE transactions on Education* **55.3** (2011), pp. 319–325.
- [Bec03] Becker, K. “Grading programming assignments using rubrics”. *Proceedings of the 8th annual conference on Innovation and technology in computer science education*. 2003, pp. 253–253.
- [Ber15] Bergh, L. van den et al. “Teacher learning in the context of a continuing professional development programme: A case study”. *Teaching and teacher education* **47** (2015), pp. 142–150.
- [Boa17] Board, T. C. *AP Computer Science Principles: Course and Exam Description*. New York, NY: College Board, 2017.
- [Bro17] Broll, B. et al. “A visual programming environment for learning distributed programming”. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM. 2017, pp. 81–86.
- [Bro16] Brookhart, S. M. *How to create and use rubrics for formative assessment and grading*. ASCD, 2016.
- [Bur18] Burd, B. et al. “Courses, content, and tools for internet of things in computer science education”. *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*. 2018, pp. 125–139.

- [Cat16] Cateté, V. et al. “Developing a rubric for a creative CS Principles lab”. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM. 2016, pp. 290–295.
- [Cat18] Cateté, V. et al. “Creation and validation of low-stakes rubrics for K-12 computer science”. *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM. 2018, pp. 63–68.
- [Cod19] Code.org. *Promote Computer Science*. 2019. URL: <https://code.org/promote>.
- [Cod18] Code.org Advocacy Coalition. *State of Computer Science Education Policy and Implementation*. Report. 2018.
- [Coo17] Cooper, S. et al. “K-12 Teachers Experiences with Computing: A Case Study”. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’17. New York, NY, USA: ACM, 2017, pp. 360–360.
- [Cun14] Cuny, J. et al. “CS principles professional development: only 9,500 to go!” *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM. New York, NY: ACM, 2014, pp. 543–544.
- [DeN17] DeNero, J. et al. “Beyond autograding: Advances in student feedback platforms”. *Proc. of the 2017 ACM SIGCSE Tech. Symp. on CS Ed*. 2017, pp. 651–652.
- [Dia17] Diana, N. et al. “An instructor dashboard for real-time analytics in interactive programming assignments”. *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*. 2017, pp. 272–279.
- [Die00] Dietz, M. *Single point rubric idea presented at INTASC Academy*. 2000.
- [Eri14] Ericson, B. J. et al. “Preparing secondary computer science teachers through an iterative development process”. *Proceedings of the 9th workshop in primary and secondary computing education*. ACM. 2014, pp. 116–119.
- [Eug16] Eugene, K. et al. “The usefulness of rubrics in computer science”. *Journal of Computing Sciences in Colleges* **31.4** (2016), pp. 5–20.
- [Fit13] Fitzgerald, S. et al. “What are we thinking when we grade programs?” *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013, pp. 471–476.
- [Flu10] Fluckiger, J. “Single point rubric: A tool for responsible student self-assessment”. *The Delta Kappa Gamma Bulletin* **76.4** (2010), p. 18.
- [Fra19] Franco, Z. *7 Great, Tried and Tested UX Research Techniques*. 2019. URL: <https://www.interaction-design.org/literature/article/7-great-tried-and-tested-ux-research-techniques>.
- [Gar15] Garcia, D. et al. “The beauty and joy of computing”. *ACM Inroads* **6.4** (2015), pp. 71–79.

- [Goo14] Goode, J. et al. "Curriculum is not enough: the educational theory and research foundation of the exploring computer science professional development model". *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM. New York, NY: ACM, 2014, pp. 493–498.
- [Gro16] Grover, S. et al. "Factors influencing computer science learning in middle school". *Proceedings of the 47th ACM technical symposium on computing science education*. 2016, pp. 552–557.
- [Häm11] Hämäläinen, H. et al. "Applying peer-review for programming assignments". *Int. J. Inf. Technol. Secur* **1** (2011), pp. 3–17.
- [Har12] Harvey, B. et al. "Snap!:(build your own blocks)". *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM. 2012, pp. 662–662.
- [Hoc15] Hochanadel, A. & Finamore, D. "Fixed and growth mindset In education and how grit helps students persist in the face of adversity." *Journal of International Education Research* **11.1** (2015), pp. 47–50.
- [Hut00] Hutchins, E. "Distributed cognition". *International Encyclopedia of the Social and Behavioral Sciences. Elsevier Science* **138** (2000).
- [Iha10] Ihantola, P. et al. "Review of recent systems for automatic assessment of programming assignments". *Proc. of the 10th Koli calling intl. conf. on comp. ed. research*. 2010, pp. 86–93.
- [Joh16] Johnson, D. E. "ITCH: Individual Testing of Computer Homework for Scratch Assignments". *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM. 2016, pp. 223–227.
- [Kna] Knapp, J. *Sprint: how to solve big problems and test new ideas in just five days*. first. Online.
- [LS18] Labor Statistics, U. B. of. *Employment by detailed occupation*. 2018.
- [Lau11] Laughran, J. et al. *Scientific Literacy Under the Microscope: A Whole School Approach to Science Teaching and Learning*. The Netherlands: Sense Publishers, 2011, pp. 42–99.
- [Loa86] Loacker, G. et al. "Assessment in higher education: To serve the learner". *Assessment in higher education: Issues and contexts* (1986), pp. 47–62.
- [Mem19] Memon, M. *16 Important UX Design Principles for Newcomers*. 2019. URL: <https://www.springboard.com/blog/ux-design-principles/>.
- [Mil19] Milliken, A. et al. "Effective Computer Science Teacher Professional Development: Beauty and Joy of Computing 2018". *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM. 2019, pp. 271–277.

- [Mir15] Mirnig, A. G. et al. “A formal analysis of the ISO 9241-210 definition of user experience”. *Proceedings of the 33rd annual ACM conference extended abstracts on human factors in computing systems*. 2015, pp. 437–450.
- [Mol17] Molenaar, I. & Campen, C. Knoop-van. “Teacher dashboards in practice: Usage and impact”. *European Conference on Technology Enhanced Learning*. Springer. 2017, pp. 125–138.
- [Mol18] Molenaar, I. & Campen, C. A. Knoop-van. “How teachers make dashboard information actionable”. *IEEE Transactions on Learning Technologies* **12.3** (2018), pp. 347–355.
- [Mor14a] Morelli, R. A. et al. “Mobile computer science principles: a professional development sampler for teachers”. *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM. New York, NY: ACM, 2014, pp. 750–750.
- [Mor14b] Morelli, R. A. et al. “Mobile Computer Science Principles: A Professional Development Sampler for Teachers (Abstract Only)”. *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. SIGCSE '14. New York, NY, USA: ACM, 2014, pp. 750–750.
- [ML15a] Moreno-León, J., Robles, G., et al. “Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills”. *Scratch conference*. 2015, pp. 12–15.
- [ML15b] Moreno-León, J., Robles, G., et al. “Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects.” *WiPSCE*. 2015, pp. 132–133.
- [Pen15] Penuel, W. R. “Infrastructuring as a practice for promoting transformation and equity in design-based implementation research”. *keynote presentation at the meeting of the International Society for Design and Development in Education Conference, Boulder, Colorado*. 2015.
- [Pen07] Penuel, W. R. et al. “What Makes Professional Development Effective? Strategies That Foster Curriculum Implementation”. *American Educational Research Journal* **44.4** (2007), pp. 921–958.
- [Pen11] Penuel, W. R. et al. “Organizing research and development at the intersection of learning, implementation, and design”. *Educational researcher* **40.7** (2011), pp. 331–337.
- [Pri16] Price, T. W. et al. “Lessons Learned from BJC CS Principles Professional Development”. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM. 2016, pp. 467–472.
- [Rag20] Ragupathi, K. & Lee, A. “Beyond fairness and consistency in grading: The role of rubrics in higher education”. *Diversity and inclusion in global higher education*. Palgrave Macmillan, Singapore, 2020, pp. 73–95.
- [Res09] Resnick, M. et al. “Scratch: Programming for all.” *Commun. Acm* **52.11** (2009), pp. 60–67.
- [Roh14] Rohrer, C. “When to use which user-experience research methods”. *Nielsen Norman Group* (2014), pp. 1–7.

- [Ros17] Rosato, J. et al. “A Comparison of Online and Hybrid Professional Development for CS Principles Teachers”. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM. New York, NY: ACM, 2017, pp. 140–145.
- [Rus13] Russell, J. L. et al. “Theories and research methodologies for design-based implementation research: Examples from four cases”. *Yearbook of the National Society for the Study of Education* **112.2** (2013), pp. 157–191.
- [Rus03] Rust, C. et al. “Improving students’ learning by developing their understanding of assessment criteria and processes”. *Assessment & Evaluation in Higher Education* **28.2** (2003), pp. 147–164.
- [Sar18] Sarikaya, A. et al. “What do we talk about when we talk about dashboards?” *IEEE transactions on visualization and computer graphics* **25.1** (2018), pp. 682–692.
- [Sin17] Singh, A. et al. “Gradescope: a fast, flexible, and fair system for scalable assessment of handwritten work”. *Proceedings of the fourth (2017) acm conference on learning@ scale*. ACM. 2017, pp. 81–88.
- [Ste16] Stegeman, M. et al. “Designing a rubric for feedback on code quality in programming courses”. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 2016, pp. 160–164.
- [Sub17] Subramaniam, M. & Cateté, V. “A Pathway to Strengthening Support for Beauty and Joy of Computing Teachers”. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM. New York, NY: ACM, 2017, pp. 779–780.
- [Tho18] Thomson, P. “Thinking about the school most of the time: studio as generative metaphor for critical reflection”. *Journal of Educational Administration and History* **51.2** (2018), pp. 1–16.
- [Ver13] Verbert, K. et al. “Learning analytics dashboard applications”. *American Behavioral Scientist* **57.10** (2013), pp. 1500–1509.
- [Wal11] Walvoord, B. E. & Anderson, V. J. *Effective grading: A tool for learning and assessment in college*. John Wiley & Sons, 2011.
- [Zha20] Zhang, J. et al. “GitGrade: A Scalable Platform Improving Grading Experiences”. *Proc. of the 51st ACM Tech. Symp. on CS Ed.* 2020, pp. 1284–1284.