An Exploration of Learning Tool Log Data in CS1: How to Better Understand Student Behaviour and Learning

by

Anthony Estey
B.Sc., University of Victoria, 2008
M.Sc., University of Victoria, 2010

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

An Exploration of Learning Tool Log Data in CS1:
How to Better Understand Student Behaviour and Learning

by

Anthony Estey
B.Sc., University of Victoria, 2008
M.Sc., University of Victoria, 2010

Supervisory Committee

Dr. Yvonne Coady, Supervisor
(Department of Computer Science)

Dr. Alona Fyshe, Departmental Member
(Department of Computer Science)

Dr. Marc Klimstra, Outside Member
(Department of Exercise Science, Physical and Health Education)

ABSTRACT

The overall goal of this work is to support student success in computer science. First, I introduce BitFit, an ungraded practice programming tool built to provide students with a pressure-free environment to practice and build confidence working through weekly course material. BitFit was used in an introductory programming course (CSC 110) at the University of Victoria for 5 semesters in 2015 and 2016.

The contributions of this work are a number of studies done analyzing the log data collected by BitFit over those years. First, I explore whether patterns can be identified in log data to differentiate successful from unsuccessful students, with a specific focus on identifying students at-risk of failure within the first few weeks of the semester. Next, I separate out only those students who struggle early in the semester, and examine their changes in programming behaviour over time. The goal behind the second study is to differentiate between transient and sustained struggling, in an attempt better understand the reasons successful students are able to overcome early struggles. Finally, I combine survey data with log data to explore whether students understand whether their study habits are likely to lead to success.

Overall, this work provides insight into the factors contributing to behavioural change in an introductory programming course. I hope this information can aid educators in providing supportive intervention aimed at guiding struggling students towards more productive learning strategies.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Yvonne Coady for leading by example and showing me just how much of an impact an instructor can have on students inside and outside of the classroom.

Anna Russo Kennedy for kick-starting this work, and helping me find a research area I am so passionate about.

Celina Berg, Mike Zastre, Bette Bultena and Veronika Irvine for the years of mentoring, advice, and even peer support.

Marc Klimstra, Alona Fyshe, and Steve Wolfman for your insightful comments and edits, and for your participation at my defense.

Members of the Learning and Teaching Centre for tirelessly working to improve the learning experience for our students.

The Computer Science Department Faculty and Staff for being my home and family for well over a decade.

My amazing friends and family for your love, support, and patience.

DEDICATION

In loving memory of my late grandfather, Ralph Estey.

This is one for the students.
The ah-hah moment is so pure and magical every time.

Chapter 1

Introduction

1.1 What is the problem?

For decades, improving the success rates of students in introductory programming courses has been one of the focuses of the computer science education community [13, 39, 44, 73, 99, 127]. In recent years, there has been a spike in enrollment in computer science programs [132], but failure rates are still high, estimated at nearly 33% worldwide [11, 124]. Beyond simply improving pass rates, it is important that undergraduate programs worldwide produce a diverse set of eager and empowered computer scientists. Improving on the fail and drop rates in computer science is a complex and difficult problem to solve, as it consists of a variety of different sociological, psychological, and even economical issues, and often one solution exacerbates an entirely different set of problems [110].

This work focuses on student success in introductory computer science programming (CS1) courses. In CS1 courses, lecture content typically builds upon material covered in previous weeks, making it difficult for students who fall behind early to catch up on their own [2, 31, 40, 94]. "At-risk" students, or those in danger of dropping or failing a course, must be identified early so that they can be supported in a way that allows them to succeed.

As class sizes increase, it is extremely difficult for instructors to identify and support struggling students. Large course sizes can act as barriers to meaningful relationships between students and instructors. In large classes, it is extremely difficult for instructors to gauge invidividual student progress and comfort; instructors simply do not have enough information to be able to adequately support students. Many

instructors do try and provide extra support for their students, but studies show that instructors do not necessarily know where exactly students are struggling [16]. On the other hand, students feel that instructors do not care about them, are hesitant to ask questions, and are not motivated to learn [126]. A number of studies have shown that the motivation, confidence, and comfort levels of students have a remarkable effect on learning [5, 12, 117], and that faculty engagement, especially in first year courses, is critical to supporting students in these areas [110].

Automated assessment tools have been used successfully as a means of providing students with plenty of feedback without overburdening instructors. Although emerging technologies can be utilized to present material in a way that may be more enticing to students, simply replacing the medium in which material is presented, without changing the underlying practices of instruction, will have little impact on learning [116]. Many assessment tools focus only on scoring solutions, rather than assessing whether students correctly understand the material [10]. Instructors need a reliable and scalable mechanism for identifying at-risk students as quickly as possible, before they disengage, drop out, or fail [40]. The current generation of tools provides instructors with the ability to move from simple, canned quizzing systems to a new model where automated, data-driven analysis can be used to continually assess and refine the quality of teaching [19]. Some institutions are already using tools with data collection features, and have begun analyzing workflow data to help improve student success [56, 81, 83], but questions about how these tools can be used to improve the learning process still remain [42, 84].

This work presents a learning tool deployed in an introductory programming course designed to provide students with a practice environment to build confidence. An analysis of log data collected over 5 semesters explores how learning tool data can be analysed to inform instructors about student learning and performance. Based on the problems described above, the findings of this work may provide valuable insight to computer science educators interested in providing supportive intervention to students at risk of failure before it is too late.

1.2 Background and Related Work

There are many contributing factors that help define the research questions explored in this work; Figure 1.1 illustrates the three core areas, which include predictors of success in computer science, learning and assessment tools, and educational psy-

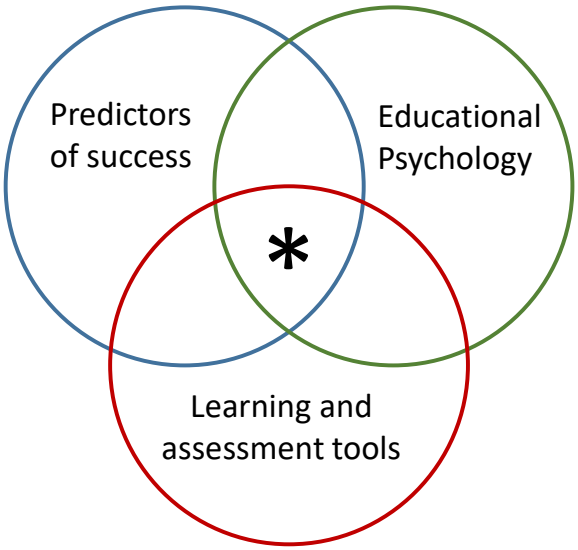


Figure 1.1: This work explores the intersection of research on predictors of success in computer science, learning and assessment tools, and educational psychology.

chology. Research has been done in each of these areas for decades, and this work only overlaps with a very small subset of each area. That being said, each area was equally important throughout the course of this research, and this work best fits at the intersection of these three areas. Within the realm of predicting success in computer science, this work focuses specifically on an introductory programming course, using data collected from a practice programming tool to investigate what patterns are associated with learning and success. Previous work in Educational Psychology provided insight on how to differentiate between learning and success, which helped in both the creation of tool features and the analysis of data. Many of the features present in BitFit, the practice programming tool used in this study, were influenced by the previous research done on assessment and learning tools.

The following subsections overview some of the influential research done in each of the areas that contributed to this work.

1.2.1 Detection of "at-risk" students

In order to improve the learning environment and experience for students in a course, the first goal of this work was to develop a system that allowed the identification of "at-risk" students as early as possible. In this work, at-risk students are defined as those in danger of dropping or failing a course. The results of a study done in first-year mathematics courses show that for courses where each week's material builds on the concepts learned in previous weeks, if students are unable to complete the first 2 graded assignments, they will very likely fail [31]. Early detection is important in computer science (CS) courses as well, as a number of studies have established that performance on early coursework correlates with final exam scores [2, 89]. Similarly, Falkner and Falkner found that the timing of a student's first assignment submission in a CS course, particularly late submissions, was a strong predictor of which students were likely to under-perform in their classes [40]. Recent studies suggest that students can be identified as early as the first two weeks of the course [1]. Grades have also been shown to correlate positively with the amount of help students receive beyond office hours and email [22], as well as through peer instruction [130].

Other research that provided helpful insights include Bergin and Reilly's research on the fifteen factors that might influence success [13], Ventura's study on predictors of success [117], and Carbone et al.'s work that explored reasons students may lose motivation [20]. Alvarado et al. found that the traditional factors of prior experience

and confidence still predict success in courses with modern curricula and improved pedagogy, but only for some students [5].

There have been a number of studies on ways to classify student behaviour in order to predict drop-out. Qu et al. found that low confidence, high confusion, and low effort were strong indicators of students giving up [91]. Challenging assignment material can keep some students engaged, while being frustrating to others [47]. Kinnunen et al. note that efficient intervention requires a combination of many different actions that take into consideration the many factors that contribute to drop-out [63]. Arroya et al. state that a student's goals and attitudes while interacting with a tutoring system are typically unseen and unknowable [8]. They note that problem-solving time, mistakes, and help requests are easily recorded, and report on their success in using a Bayesian Network to infer a student's hidden attitude toward learning, amount learned, and perception of the system. A study on student behaviour during lab periods found that confusion and boredom are two affective states associated with lower achievement [96].

Cocea and Weibelzahl have completed a number of very insightful studies on disengagement prediction [24]. They explain that engagement is an important aspect of effective learning, because time spent using an e-Learning system is not quality time if the learner is not engaged. In some of their earlier work, they showed the possibility to predict engagement from log files using a web-based e-Learning system [26]. Through follow-up analysis with different web-based learning systems, they were able to demonstrate that their solution is system-independent and that engagement can be elicited from basic information logged by most e-Learning systems: number of pages read, time spent reading pages, number of tests/quizzes, and time spent on test/quizzes [25]. Later, they were able to classify different types of disengaged users from different sets of data, and further highlight the importance of monitoring not only problem solving activities, but also time spent reading pages [27]. In more recent years, there have been a number of studies analyzing log data to better understand student programming behaviour.

1.2.2 Programming behaviour analysis

Both code generation and debugging activities within an IDE have been shown to correlate with final grades [34, 80]. Large-scale studies of compilation errors from hundreds of thousands of students hold promise to identify at-risk patterns in types

and frequencies of errors [4, 57]. Jadud used BlueJ, a beginner's programming environment, to report at compile-time the complete source code along with other relevant meta-data, and observed that a minority of different syntax errors account for the majority of errors [56]. The fact that certain kinds of errors occurred frequently directed future work efforts towards helping students break out of repetitive error cycles [109]. As an extension to BlueJ, Norris et al. used the ClockIt BlueJ Data Logger/Visualizer to monitor student software development practices [83]. Students and instructors alike were then able to view the collected data, the goal being to discover "patterns" of student practices shared by successful students that are not present in struggling or unsuccessful students. Similarly, Retina collects information about students' programming activities, and then provides useful and informative reports to both students and instructors based on the aggregation of that data [81].

Edwards et al. compared effective and ineffective programming behaviours of student programmers, using assignment submissions gathered from an automated grading system [36]. Submissions were classified by grade received, and the significant finding of the study is that when students started and finished their assignments earlier, they received higher grades than when they started an assignment the day before it was due, or later. Spacco et al. also reported that students who started their work earlier tended to earn better course grades in their study on programming habits in a CS2 course [107].

Watson et al. claim that there are still major improvements that need to be made with respect to the identification of at-risk students [125]. They state that one of the problems may be that prior research methods are based upon using static tests, which fail to reflect changes in a student's learning progress over time. They claim that if data is dynamically analyzed to identify changes in progress over time, students can be provided with appropriate interventions when required. Unfortunately, this requires a learning tool with a number of data collection and analysis features.

1.2.3 Learning tool features

An important requirement during the design of the tool in this study was that it allowed users to work with the material from anywhere. It was also important that it allowed an analysis of usage data, which is why an online tool made sense. Online tools are not a new idea, as back in 1999, Boroni et al. discussed the huge potential of self-contained, animated, interactive, web-based resources for computer science in

their creation, "hypertextbook", which paved the way for many future endeavors [15]. Learning management systems (LMSs), such as Moodle and Blackboard, have become popular in recent years, and include features that allow communication and collaboration between learners and teachers through things such as wikis and discussion forums [18]. Rößling et al. state that LMSs do not adequately support the needs of computer science students, and are particularly lacking in providing students with effective learning activities and tools [97]. They recommend integrating assessment features, algorithm visualizations, and practice problem generation into these systems. They also warn that students may become discouraged due to lack of feedback in practice problem activities if they are struggling with any of the material. They recommend building systems with the ability to provide layered feedback to students as a way of raising student motivation.

In a recent survey, Ihantola et al. identify a wide range of recent features in automated program assessment systems, including test case construction assistance, submission management, automated scoring, and security features [52]. Unfortunately, features about problem understanding, rather than scoring solutions, are noticeably absent. Gikhandi el. al provide a review of the online and blended learning formative assessment research [43]. The review provides evidence on the potential to engage both teachers and learners in meaningful educational experiences. They suggest that educators need to recognize and emphasize the value of embedding assessment within the learning process, by assessing the process of learning instead of just finished products. One set of tools that focus on supporting students throughout the learning process are Intelligent Tutoring Systems (ITSs).

VanLehn's report provides encouraging results with respect to the effectiveness of tutoring systems [114]. Different granularities of electronic tutoring were compared with one-on-one human tutoring, and both were compared to instruction scenarios where no tutoring is given at all. A significant finding is that modern ITSs are just as effective as one-on-one human tutoring for increasing learning gains in STEM topics, and it is argued that these systems replace homework, seatwork, and other classroom activities. It is important to note that none of these systems attempt to replace classroom teachers, nor does the report recommend these systems do so.

Although there have been positive results with respect to ITSs, creating an effective ITS is a challenging process, and one that needs to satisfy multiple stakeholders in an educational setting. Because tutoring systems require 200-300 hours of development for one hour of instruction, authoring tools have been created to speed up

content development [3]. Rau et al. present a methodology for designing interactive learning environments, and provide a methodology to resolve conflicts introduced by conflicting stakeholder recommendations [92].

Koedinger et al. looked into improving online learning and course offerings through data-driven learner modeling [69]. They recommend designing online learning environments that collect fine-grained, complex data about a learner. This enables researchers to use well established principles on learning and cognition, and avoid re-inventing the wheel when it comes to evaluating and improving the learning experience. They stress the importance of learning from the decades of research that have been carried out in the Intelligent Tutoring Systems (ITS) and Artificial Intelligence (AI) in Education fields, and recommend including cognitive psychology expertise to guide the design of online learning activities. They also strongly recommend adopting a data-driven approach to learner modeling with the goal of improving the learner's experience, and shifting course design away from solely an expert-driven paradigm to one that is self reflective and learns from past interactions of learners with the system.

The aforementioned works highlight how successful ITSs have been in supporting students through different learning processes. With the prevalence of tools that log student progress data, there has been some interesting research within the domain of learning analytics on how the data can be used to improve the learning experience. Triantafillour et al. provide a report on how cognitive profiling approaches have informed and influenced the research and development of ITSs for decades [112].

In his work back in 1985, Soloway stated that when solving a problem, it is often not the syntax and semantics that pose major stumbling blocks for novice programmers. He notes the real problems lie in "putting the pieces together," which involves composing and coordinating the different components of a program [105]. Based on these ideas, tools like MENO II, PROUSE, and later CHIRON were developed, and used successfully to aid novice programmers in learning Pascal [101]. In Johnson and Soloways explanation of Prouse, they note the two important components of their tutoring system: a *programming expert* which can analyze and understand buggy programs, and a *pedagogical expert* that knows how to effectively interact with and instruct students [60].

Anderson's production rule theory also influenced a number of tutoring systems [6]. The basic premise is that a cognitive skill is made up of a number of production rules, and that tasks are achieved by stringing together a series of these production rules.

Anderson et al. designed a tutor for LISP that was successful in improving the effectiveness of notive programmers [7]. Based on such models, a specific type of ITS was developed, called a "Cognitive Tutor". The idea behind Cognitive Tutors is to put students into a hands-on problem-solving situation, with the tutoring system providing instruction based on individual student needs [67]. Jin shows that students who used intelligent cognitive tutors outperform students who did not in standard classrooms by more than 50% in solving targeted programming questions [59]. Because Cognitive Tutors have been shown to lead to significant learning gains, they are now incorporated into over 3,000 schools across the United States.

1.2.4 Focus on Learning

Although assessment tools have come a long way, there are still an abundance of questions about how emerging technologies can be used to improve the quality and effectiveness of teaching. Vandewaetere et al. provide an overview of research done on the value of learner models in the development of adaptive learning environments [113]. Their results show that, although there are a number of high-quality studies, there is sparse data related to the empirical effectiveness of including specific cognitive, affective, or behavioral individual characteristics in learner models with respect to enhancing the learning process or increasing the learning outcomes. Veletsianos reminds instructors that is important to remember that what impacts learning are changes in instructional design and pedagogical practices supported by the introduction of new technologies, not the technology itself [116]. He also mentions that fields closely related with educational technology, such as instructional design and cognitive psychology, can provide us with evidence-based insights in how and under what conditions people learn.

Instructional design

Within the realm of education, curriculum development is primarily concerned with what to teach, whereas instruction is primarily concerned with how to teach it [104]. Instructional design is concerned with understanding, improving, and applying methods of instruction, and the intention is that learning outcomes produced are effective, efficient, and appealing [93]. Effectiveness is measured by the level of student understanding, efficiency is measured by the effectiveness divided by student time and/or cost of instruction, and the appeal is measured by tendency of students to want

to continue to learn. This simple framework has proven resilient and valuable for theorists and practitioners. Wilson et al. expand the framework by focusing more on social impact, engagement. and the learner's experience [128]. They note that tools, technologies, and all other forms of intervention have both positive and negative impacts. They suggest adding a fourth descriptor, "good instruction", where the aim is to lead learners toward valued ends, while minimizing any negative impacts. Instead of combating instructional problems and inefficiencies, Veletsianos proposes that technologies can be used as means to provide personally relevant and meaningful transformations for students [115].

Educational Psychology

The Attention, Relevance, Confidence, and Satisfaction (ARCS) model comes from educational psychology, and defines four major conditions that need to be met for people to become and remain motivated [61]. Within the ARCS model, gaining and retaining the learner's attention is necessary for efficient learning, relevance (of the learning content) is a condition for attention and motivation, confidence determines the level of effort invested in learning, and satisfaction refers to the reward gained from the learning experience. The purpose of the ARCS model is to employ strategies that are used to improve the motivational appeal of instruction. This, in turn, should translate into improvements in learner motivation. Huett et al. report on the potential of using ARCS-based e-mail messages designed to improve the motivation and retention of students enrolled in an online computer applications course [51]. Mihaela Cocea reported on a number of previous works that used the ARCS model within CS education, and proposed a new approach based in Social Cognitive Learning Theory, and especially related to self-efficacy and self-regulation [27]. This led to her later studies on which factors in the learning behaviour can predict drop-out.

When designing the programming practice tool, thought was put into what types of data need to be collected in order to provide a more effective, efficient, and appealing learning environment for students, and the information gained from such a process could be applied to improve the in-class experience for students, specifically by supporting students in a way that increased student confidence, self-efficacy, and satisfaction. Law et al. report on a preliminary study that investigates the key motivating factors among students taking programming courses, and adopted a research model linking various motivating factors, self-efficacy, and other effects of using their

e-learning system [72]. The results suggest that a well facilitated e-learning setting can enhance learning motivation and self-efficacy. This inspired me to look into the research on how other instructors had taken an active role in enhancing the learning environment for their students.

1.2.5 Supporting all students

A study assessing the programming skills of first year CS students illustrated how difficult learning to program can be [77]. Often students come into introductory programming courses with a variety of previous experiences. Robins et al. speculate that the distinction between an effective and ineffective novice is more important than the one between a novice and expert programmer [95]. Effective novices are those that learn to program without excessive effort or assistance. Ineffective novices are those that do not learn, or do so only after inordinate effort and personal attention. They suggest it may be a worthwhile effort to explicitly focus on trying to create and foster effective novices. Kirschner et al. note that due to the nature of human cognitive architecture, a minimally guided approach, similar to the learning environment found in traditional lecture-based courses, is not optimal for novices learning a cognitively challenging task, such as programming [64].

These results have motivated some of research into improving course programming exercises. The Cognitive Apprenticeship (CA) model emphasizes guiding students through the learning process, instead of evaluating end products [29, 30]. A number of institutions report on positive results when using CA to teach programming. In addition to fostering programming skills, improving a course's programming exercises can also impact student motivation and comfort, and numerous studies have shown that both the motivation and the comfort level of students have a remarkable effect on learning [12]. A key component to improving programming exercises is feedback. Lumsden notes that talking with students about their solutions and problem solving strategies, while giving them hints on how to improve them, also has a positive impact on student motivation [74].

Vihavainen et al. build upon many of these ideas in their Extreme Apprenticeship (XA) method, which is a variation of cognitive apprenticeship [121]. They used their XA approach to produce some encouraging results with respect to student drop-out rates, pass rates and grade distribution. The core values in XA are first, that in order to master a craft, students need meaningful activities, so that they can practice

as long as is necessary. Second, there needs to be continuous feedback between the learner and advisor, and the learner needs confirmation that he or she is progressing, and to a desired direction. The XA method has been used in a number of course offerings, including online learning environments [120].

These works all highlight the importance of, and active role required of instructors maximize the effectiveness of learning environments. In the CA model, the instructor's roll is not limited to introducing and explaining concepts. Instead, instructors are also mentors and facilitators of student learning and skill development as students progress through meaningful activities.

Instructor impact

In a review of the state of student retention research, Tinto notes that involvement, or what is increasingly being referred to as engagement, matters and it matters most during the critical first year of college [110]. He also notes that it is a widely accepted notion that the actions of the faculty, especially in the classroom, are key to institutional efforts to enhance student retention. In more recent work, McCartney et al. discuss the importance of interpersonal interactions with respect to the learning experience, especially among first-year students [76]. Student interviews show that in some learning environments students are not comfortable asking questions, and do not feel like instructors know what material "trips people up".

Janet Carter et al. provide a comprehensive review of many different approaches instructors can use to increase student motivation [21]. The work provides a repository of tips and techniques for educators, discusses challenges and concerns, and makes some suggestions to promote further progress in the field.

With the effectiveness of the current generation of learning tools, Dodero et al. studied the learning experiences of blended styles of learning compared to a purely virtual e-learning approach [33]. The study showed that technology can act as an incentive to improve students' participation during traditional classroom-teaching, but does not help increase student participation when the learning is completely virtual and not complemented by regular classes.

Jason Carter et al. found that most students wanted help that went beyond office hours and email, and that for the vast majority of them, their grades correlated positively with the amount of help they received for insurmountable difficulties [22]. In order to reduce instructor overload, their study also describes how their approach

in using log data to distinguish between surmountable and insurmountable difficulties.

1.2.6 Tying it all together

Previous work highlights the fact that many students drop out of or fail introductory computer science courses due to low confidence or motivation because they are unable to keep up with the material. Learning tools have been used to provide assistance to students, and scale to very large or online classrooms, but these tools can also act as barriers that prevent meaningful relationships between students and instructors. Recent studies suggest that a solution to both problems is a blended learning approach; learning tools can provide some assistance to students, and the data collected by the tool can inform instructors about student struggles, enabling them to support students accordingly. I tried to emulate this type of environment at the University of Victoria for this study.

A similar set of studies is being done concurrently by Spacco et al [106]. Over the past few years, they have analyzed data collected from two web-based programming exercise systems. Their work demonstrates the potential of collecting and analyzing data from short programming exercises, and provides positive, although weak, correlation between student's effort and success on CloudCoder exercises and the student's final exam score. They also discuss that an intriquing use of such systems would be to detect "flailing" [41] students early enough in the semester that the instructor might intervene, and define metrics for detecting difference between flailing and learning.

1.3 Tool Features

BitFit was developed by Anna Russo Kennedy as part of her Masters work, and the technical components of BitFit's design are covered in her thesis [100]. I added all of the content to BitFit. For sample questions, hints, and other information about the content included in BitFit, please refer to Appendix A.

All of the aforementioned works were influential in the design of BitFit, and inspired the six original goals set forth for BitFit:

1. **Give students a pressure-free place to try and fail with the material.** To build student confidence, provide them with an enviroment where they are not afraid to make mistakes.

2. **Find a way to answer all of their questions.** In large classrooms it is difficult for an instructor to answer all student questions. Support features need to be added to the learning tool to support students when working through practice material.

3. **Aim for early intervention when warning signals arise.** Collect and analyse log data to identify students who need additional support.

4. **Regularly take a pulse of how *they* think they are doing.** Use surveys to collect additional information about student comfort, confidence and progress.

5. **Show students we care about their success.** React to the information supplied by BitFit and surveys to show that we are using this technology to support learning.

6. **Show them they do belong in CS1.** Provide feedback to show students how much progress they have made.

There are many educational tools available, and many of them share common features [52]. CloudCoder is an open-source and online tool used for creating, assigning, and sharing short programming exercises [84]. CloudCoder also collects detailed data, so many of the core features in BitFit were modeled around it. In addition to code writing exercises found in a tool like CloudCoder, code-reading exercises are also included. There are also support features included in BitFit similar to those present in many other tutoring systems. To accomplish the specific set of goals for BitFit, a number of more specific teacher- and student-focused design requirements were created.

1.3.1 Instructor-focused requirements

There were a number of features we wanted to include in the tool based on what we required as part of the teaching team.

Design Requirement 1: *Build an easy to deploy, scalable and extensible open source tool using current, industry-standard open source technologies.* BitFit, shown in Figure 1.2, was built in JavaScript via Node.js, using the MEAN framework; the tool is web-based, and Node.js was built to specifically handle the operations of a

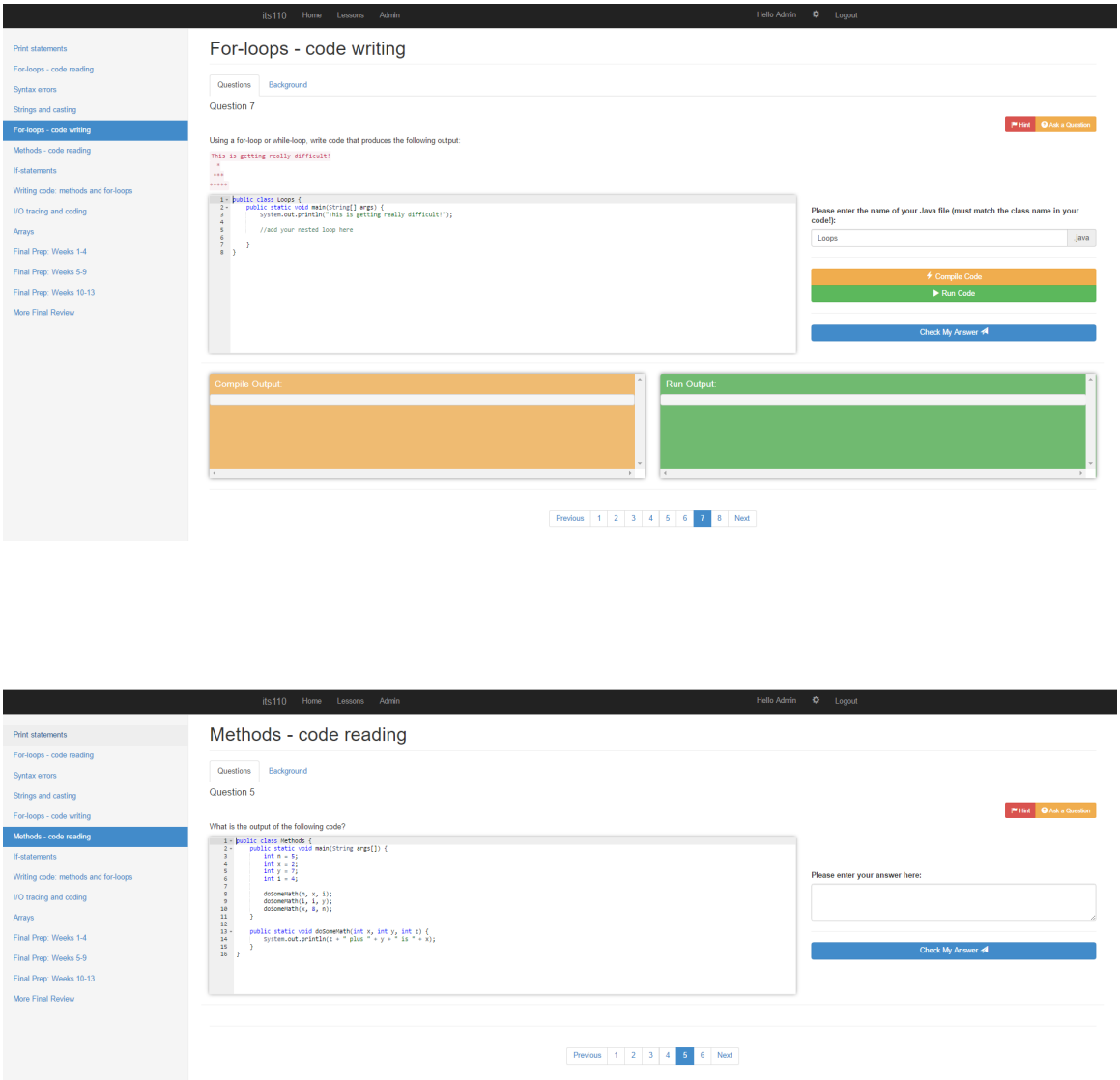


Figure 1.2: Screenshots of BitFit code writing (top) and code reading (bottom) exercises.

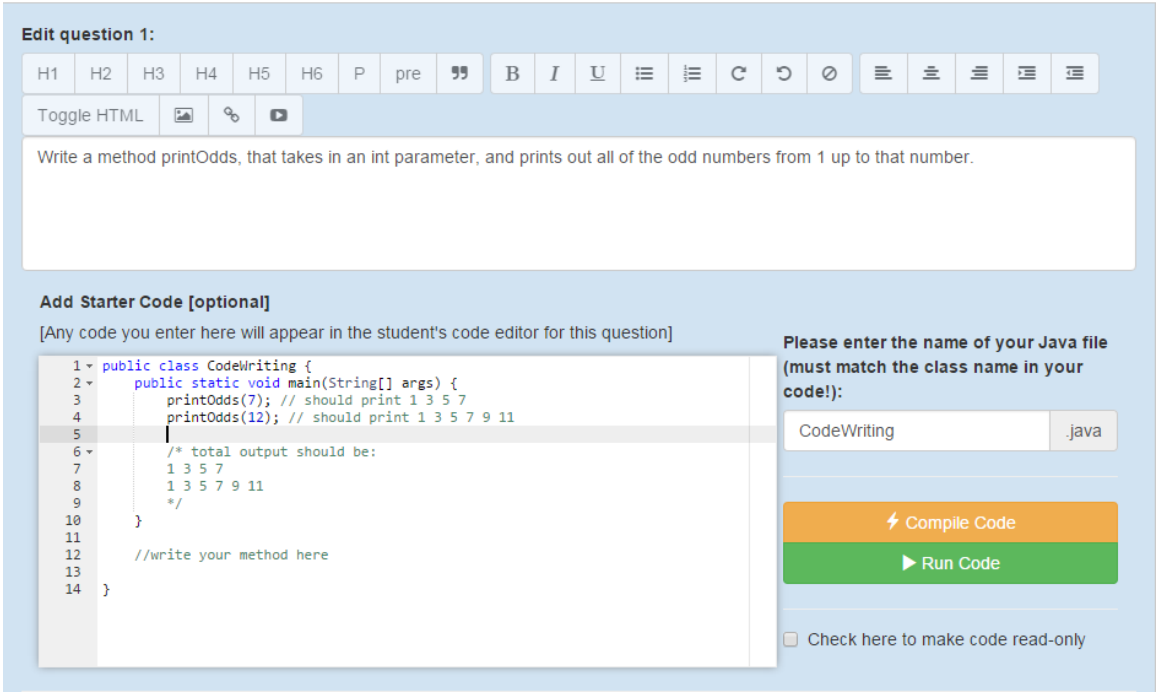


Figure 1.3: Instructor view editing a question in BitFit.

web application. The tool was also designed to be easily extensible, so it is open-source, and can currently be found on-line[1]. It is straightforward for an instructor to add functionality to the tool to allow practice exercises in a different programming language. For instance, to add support for C++, a function would need to be added to the Command Line Interface module that calls upon the C++ compiler instead of the Java compiler. The structure of the persistent data store was also chosen with flexibility in mind. MongoDB is noSQL datastore, with one major bonus of its design being that changes can be made to existing schemas on the fly, without having to restart an application for the changes to take effect. This means that students could be working through exercises at the same time as a developer is adding improved functionality. Although not a core requirement, we hope these decisions may attract future graduate students and/or community members interested in building upon their skills in these areas to extend the tool further.

Design Requirement 2: *Allow support and other instructional materials to be easily created and deployed.* As shown in Figure 1.2, the workspace where students can answer questions has another tab where instructional content can be added. Similar to other web-forms, instructors can write in information directly, or link to

[1]The tool's source code is available on GitHub at https://github.com/TheModSquad

other instructional materials used in the course. As the question database continues to grow, the initial overhead to integrate BitFit into a course will decrease. Course administrators can easily add, edit, or remove current questions. The interface to edit a question is shown in Figure 1.3. Starter code can be easily pasted into the embedded editor[2]. The "read-only" button toggles whether or not students can edit code (for non-programming questions), and hints can be added similar to how question text can be added, with support for html tags.

Design Requirement 3: *Allow for the generation of multiple question types, and include assessment features for each type of question.* In addition to program writing questions, we wanted to include code reading questions, and the possibility to ask higher-level concept questions. Although most of the tools that motivated the development of BitFit focus on improving code writing [84, 85], code reading tools have also been shown to have positive effects on program understanding [50].

Design Requirement 4: *Collect fine-grained student interaction data.* Throughout a student's use of the tool, interaction data is collected and stored on a secure server. This data includes numbers of compiles and runs for code writing questions, numbers of hints requested, time spent on each exercise, and total correct versus incorrect attempts to answer questions.

1.3.2 Student-focused requirements

There are also a number of features we wanted to include focused on providing support for students using the tool.

Design Requirement 1: *Create a programming practice environment with very low overhead to use.* Students access the tool via a webpage in a browser. All that is required is an internet connection and login info. The tool is system and platform independent, and abstracts away all the environment details, allowing students to dive straight into excerises from any machine or device they choose to do so.

Design Requirement 2: *Build a space for students to actively engage with course material: allow plenty of opportunity to practice both code writing and active code reading to increase understanding and abilities.* The tool encourages students to train the routine act of programming: carry out a high number of interactive exercises, that are somewhat repetitive, in order to master the skill needed to tackle bigger problems elsewhere [121]. The two types of questions currently incorporated

[2]https://ace.c9.io

into BitFit are code writing, where students are asked to write a method or other piece of code to produce a desired outcome, and code reading exercises, where students are asked what the output of a Java program is. The system supports other types of questions, but only these two types of questions have been added to the question repository so far.

Design Requirement 3: *Foster an environment of high interactivity in the tool, to build a continual feedback loop between student and instructor.* Students interact with and receive feedback from the tool in multiple ways. Students are provided with hints when requested, and test-case results are displayed when they submit a solution. Lectures provide an opportunity for instructors to answer student questions, but it can be difficult to formulate questions on a recently introduced topics (especially before working through an exercise), and some students may not be comfortable asking certain questions in large classrooms. In BitFit, instructors can view usage data on questions from each week's topic to better analyze student progress. This allows instructors to provide initial feedback based on student progress, which can be the first step in establishing meaningful and effective interactions between instructors and students.

Design Requirement 4: *Design a tool where questions are broken down by topic area.* The tool offers as much flexibility as possible to instructors in how they structure practice exercises. Instructors create a new topic, provide some background information on the topic area, and create a set of questions to be associated with that topic. Currently, the question repository includes questions for a number of individual topic areas, and also a variety of different combined topic sets. For example, there is a set of question on for-loops, another on parameter passing and return statements, and then a set including questions that require knowledge of both topics. Within each of these sections there a multiple questions of increasing difficulty.

One goal was for BitFit to allow for an environment where students can practice both individual, building block programming skills, as well as work on topic areas that build upon multiple concepts from previous material combined. Because the order that course topics are presented in may differ among courses or between semesters, the question repository continues to grow with different combinations of CS1 topics.

Design Requirement 5: *Provide support for all student questions.* Each question set contains a range of question difficulties. When hints are requested, as shown in Figure 1.4, students are progressively lead through the exercise, and eventually provided with a full solution to the problem. We hoped that the hint systems would

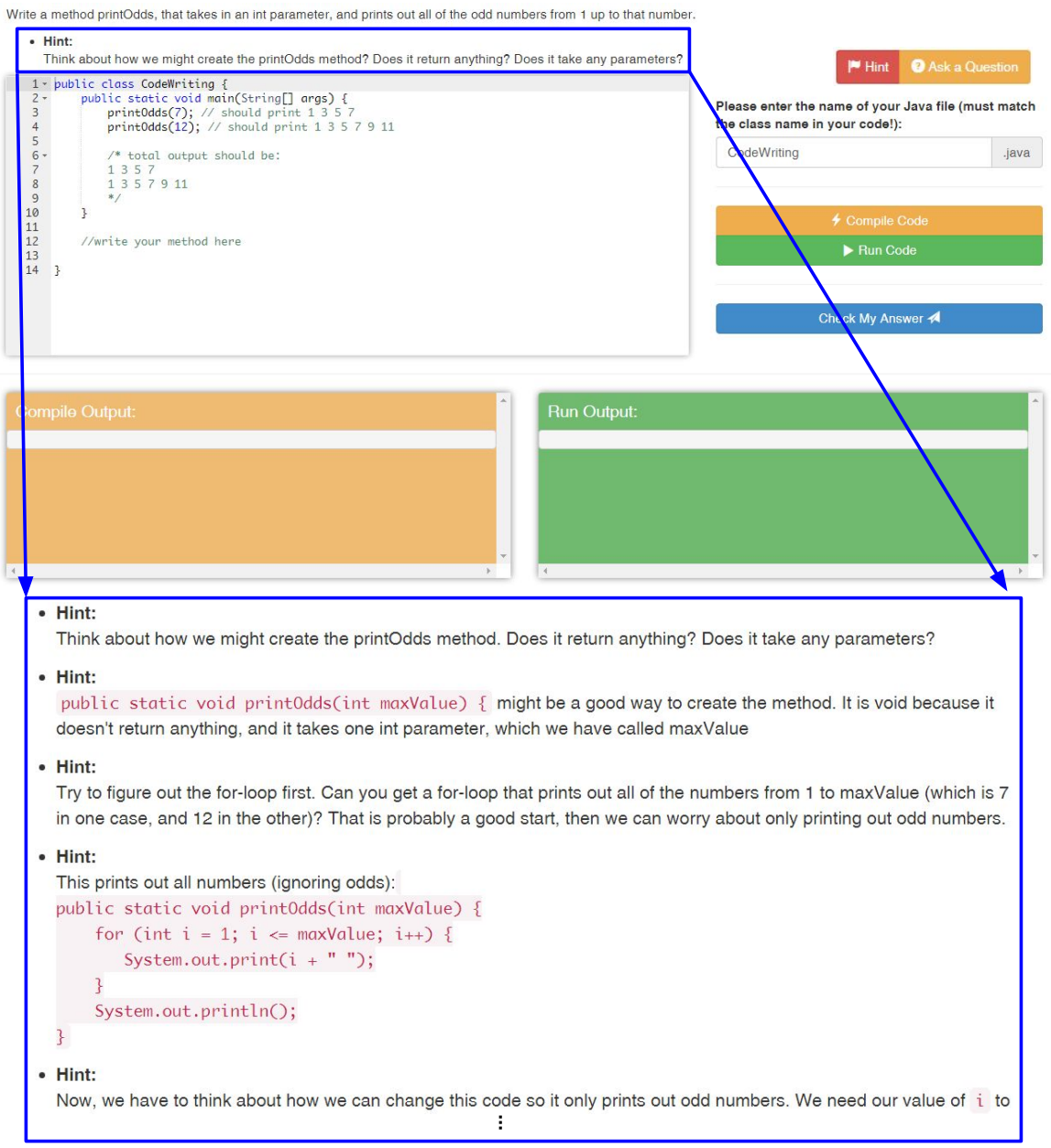


Figure 1.4: The first few hints for a question in BitFit.

provide enough support for a large number of instances where students requested assistance. Additionally, there is an "Ask a Question" button thatstudents can use to get further assitance. The button allows the user to enter a description of their problem, and provides instructors with a snapshot of their current solution. We also hypothesized that by using log data to revisit areas where students struggle may also allow us to support a number of other unasked questions, for the potentially large number of students not comfortable asking questions in traditional lectures [126].

Design Requirement 6: *Offer a safe place for students to try and fail with the material.* BitFit allows students to work through as few or as many questions as they wish, with no limit to the number of times they may attempt any of the exercises. Although within each topic area questions were generally ordered by increasing difficulty, students are able to work on or skip to any question within the system. An important aspect of this requirement is the fact that the exercises were not graded. Graded online exercies have their place in computer science education, but we believe such quizzes belong at a later stage in the learning process. Graded exercises are often used to verify that learning on a topic is complete, whereas the focus of this design requirement is associated with providing an environment for students to practice, explore, and learn.

Design Requirement 7: *Provide a way for students to accurately and rapidly gauge what they do and do not know, letting them become capable of seeing themselves succeeding with the material.* Instantaneous feedback is provided in multiple ways: on solution submissions in the form of test-case results; through a Compile Output dialogue box if student code contains errors; through a Run Output dialogue box if infinite loops or other runtime errors are detected in their code. This rapid feedback helps students gauge what areas they have mastered, and where they need to do some more work.

1.3.3 Course format

The tool was used in a first year programming course, *CSC 110: Fundamentals of Programming I.* CSC 110 is a 13-week course taught in Java[3] offered each semester at University of Victoria. CSC 110 consists of three 50 minute lectures a week, and one hour-and-fifty minute lab. Lectures take place in a lecture hall built to accommodate 250 students, and lab rooms provide computers for up to 28 students

[3]https://www.oracle.com/java/

at a time. Concepts were often introduced in lectures, and then during the weekly labs, students were provided with problem-based active learning activities, and TAs faciliated the learning by aiding students as they worked through a set of problems.

Activity on BitFit was completely voluntary each semester, and did not affect student grades in any way. BitFit was introduced as a supplemental practice resource offering exercises similar to those presented during weekly labs. BitFit is an online, open-source[4], practice programming tool where students write code in the browser. Buttons to compile code, run code, submit a solution (labeled "Check My Answer"), get a hint, and ask a question are all instrumented to collect student interaction patterns.

Compilation and execution results are displayed to the user as they work through the current problem. Submitting a solution runs the current code through a number of test cases, visible to the user, and displays the results. A solution is considered correct if all of the test cases pass. There is no restriction on the number of times a student is allowed to compile, run, or submit a question.

Each semester BitFit was used, there were over 80 questions distributed over the course's topic areas. Within each topic there are six to ten questions, ordered by difficulty. Students are not required to correctly solve "easy" questions before visiting more difficult questions within a given topic area, and are able to start on any topic area they choose.

Similar to Khan Academy[5], hints progressively lead students to a correct solution of the problem. BitFit was first designed with hints that provided high level guidance, but when students were consulted about the design of additional features, they requested sample code solutions.

The data coolected by BitFit includes the number of questions attempted, hints, compiles, runs, submissions, and correct solutions. Number of revisited questions and repeated hints were also recorded. The ratio of correct versus overall submissions, compiles versus executions, and error-free compiles versus overall compiles were also computed. All of this collected data was measured against final exam grades.

[4]https://github.com/ModSquad-AVA/BitFit
[5]https://www.khanacademy.org/

1.4 Thesis Questions

The first study in this work explores whether students will use an ungraded practice tool introduced as a supplemental resource. Additionally, the study looks into perceived impact of BitFit, specifically regarding how it may support student confidence and comfort. This lead to the following initial research questions (RQs) explored in Chapter 2:

RQ1a: Will students use a supplemental, ungraded, practice resource?

RQ1b: What are the reasons students choose not to use a practice tool?

RQ1c: What impact does BitFit have on student confidence and metacognition?

RQ1d: How well does BitFit support students in areas they feel they are struggling?

Based on the feedback from students during early BitFit deployment, BitFit was extended to allow students to optionally reveal a progressive series of hints about programming problems, eventually providing a full solution. The first study suggested that the large majority of students in a given semester will choose to use BitFit, as well as students across all grade ranges. The next step in my work was to investigate *how* students use BitFit. The goal of the second study was to uncover subtle indicators of productive learning behavior through BitFit log data, the key questions being: *Can interaction patterns with BitFit predict a student's outcome in the course?* To answer this key question, Chapter 3 considers BitFit log data over the first three semesters BitFit was used, and explores the following specific research questions:

RQ2a: How well do students' levels of engagement with the learning tool predict success in the course?

RQ2b: Can a practice tool identify differences in workflow behavior between successful and unsuccessful students?

RQ2c: What are the metrics associated with success and failure?

RQ2d: How well do metrics identified by the tool predict success?

RQ2e: How early in the semester can at-risk students be identified?

After identifying a number of metrics to identify students at-risk of failure early in the semester, I next further investigated whether the inaccuracies found in these predictors of student performance can be reduced through an analysis of log data that measures changes in programming behaviour over time. A close look at the log data in the previous study lead me to believe there was subtle difference between learning and success in CS1. If patterns associated with effective *learning* can be automatically classified in log data early in the semester, it may be possible to better guide students at-risk of failure towards more productive learning behaviour before it is too late. This motivated the key research question behind the third study: *Can analysis of patterns in interaction data help us understand how to detect and measure learning in CS1?* To answer the key question, Chapter 4 considered the following specific questions:

RQ3a: How accurately do early predictors of performance identify students who are unsuccessful in the course?

RQ3b: How well can early predictors be improved by a trajectory metric that classifies behavioural change over time?

RQ3c: How well can the trajectory metric be extended to evaluate differences in proficiency across topics?

The analysis of log data considering changes in behaviour over time was able to differentiate between transient and sustained struggling, allowing for more accurate identification of students who really need additional support. I next wanted to better understand *why* unsuccessful students so commonly exhibited ineffective study behaviour, which lead to the next key question: *Do students understand whether or not their study habits are likely to lead to success on the final exam?* My assumption is that students who are actively working through ungraded practice exercises are highly invested in learning. I wanted to know whether these students understood that their study strategies are ineffective. To consider this key question, I combined qualitative survey data with BitFit log data, and explore the following specific research questions in Chapter 5:

RQ4a: How well does time-on-task differentiate between successful and unsuccessful students?

RQ4b: How well does intended question difficulty differentiate between successful and unsuccessful students?

RQ4c: Is there a difference in a student's reflection of self-efficacy between successful and unsuccessful students?

Chapter 6 then summarizes the findings of the four individual studies, and discusses the overall implications and contributions of this work. In the Appendix, more information about BitFit, the exercises and hints contained in BitFit, sample surveys, and log data can be found.

Chapter 2

Will Students Use BitFit?

Many of the studies mentioned in the previous chapter collected student usage data during weekly labs or in a similar environment where students were graded on the code produced during the session. During the time BitFit was used at the University of Victoria, it was introduced to students as a supplemental practice resource instead of an assessment tool. The hope was that by providing students with a pressure-free environment to practice weekly material on their own, students would build confidence and be encouraged to further explore, and therefore better understand, weekly material. Because work in BitFit is completely ungraded, the idea was that students would not feel any pressure to come up with the "right" answer in only a certain number of attempts or within a specific time period.

Since BitFit usage was not a requirement of the course in any way, it was unclear who, if anyone, would use it. This study looks into overall BitFit usage statistics, as well as student feedback over the first two semesters it was used. The results of the surveys focus specifically on student perception of BitFit. Overall, this study explores the following four questions:

- **RQ1a:** Will students use a supplemental, ungraded, practice resource?
- **RQ1b:** What are the reasons students choose not to use a practice tool?
- **RQ1c:** Does using BitFit increase student confidence and metacognition?
- **RQ1d:** Can Bitfit improve a student's sense of self-efficacy?

2.1 Methodology

The data for this study was collected over three 13-week semesters. BitFit was introduced to students during the second week of the semester, and usage data was collected from then until the end of the semester.

The total population for this study is based on the total number of students registered in the course, meaning that students who may have dropped the course during the first lecture, and therefore would never have been introduced to BitFit, would be included in the statistics measuring what percentage of students chose to use BitFit. This choice was made due to the fact that it was difficult to determine when students who dropped the course stopped participating in class activities, so these students could not be filtered out of the overall population. Another option would have been to filter out all of the students who did not complete the course. As one of the overall aims of this research is to better support student success in CS1, it was important to include students who used BitFit and later dropped the course, as these students are precisely one of the groups this research aims to better support.

Qualitative data was collected through surveys given out every 4 weeks throughout the first two semesters. Participation in the study was completely opt-in; BitFit usage and survey participation was voluntary and did not count for any course credit.

2.2 Results

First, overall BitFit usage numbers are shown, followed by overall survey results. A number survey responses are included and discussed, to provide more perspective on the survey results.

2.2.1 Student opt-in

Overall, 528 out of the 652 enrolled students completed at least one question in BitFit, roughly 81%. Of the 124 students who did not use BitFit, 46 dropped from the course very early in the semester, before submitting any graded work. There were also 36 students who did not use BitFit who participated in monthly surveys. The results on why these students chose not to use BitFit are shown in Figure 2.1. The three most common reasons students did not use BitFit were that they did not know about it (39%), did not have time (31%), and did not feel like they needed extra help in the

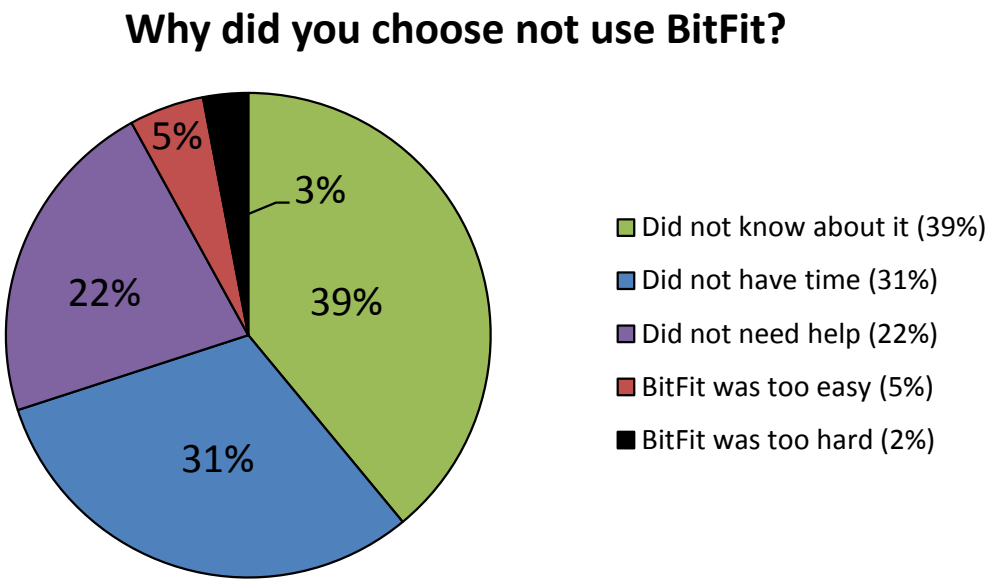


Figure 2.1: Reported reasons students did not to use BitFit.

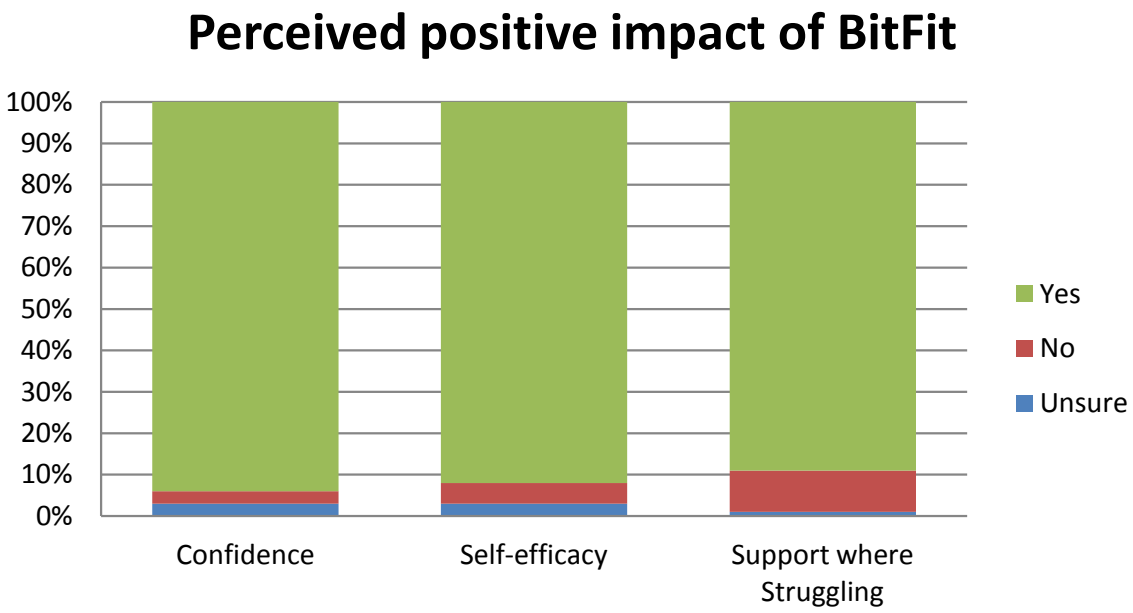


Figure 2.2: BitFit's perceived impact on student confidence, self-efficacy, and support in areas they were struggling.

course (22%).

The surveys also queried students who used BitFit about their experiences with it. Figure 2.2 shows that 94% of students reported that BitFit improved their confidence with course material, and 92% of students reported that BitFit helped them get a better idea of how they stood with respect to the course content. 89% of students reported that BitFit was effective in helping them understand areas they struggled in. Figure 2.3 shows student responses to a question about their interest levels of using BitFit in future courses.

2.2.2 Confidence

For the question *"Did [BitFit] have any affect on how you felt before writing either midterm? Why or why not?"*, many students reported that successes in BitFit made them feel better about their ability with course-related matter:

> "It made me feel better about my syntax. I often get error messages but with the tool it really helped me with it."

> "It definitely lowered my anxiety about the exams, and was a huge confidence booster to see myself answering correctly."

"I think I become more confident than before. It made me feel more able."

"Improved my confidence in my programming ability."

"I think [BitFit] gave me more confidence, because it felt good when I could do a problem correctly."

Students also often commented that the variety of exercises in BitFit helped to build confidence:

"I felt more confident because I had been able to use all of the material in a variety of code tracing and problem solving exercises"

"I found that the programming practice tool decreased my anxiety heading into both midterms, as I felt that it was a source of higher-level questions than those found in the textbook."

"It boosted my confidence, as I felt I had adequate practice and understanding."

Some of the reasons reported that BitFit did not help them feel more confident were the following:

"No, because I did not use it very much"

"It contained some difficult questions that made me worried, however, it helped me to learn a lot."

"I mostly went into the midterms worried about making logic mistakes in my programs when under stress and reading the code wrong/losing track when tracing."

2.2.3 Self-efficacy

For the question *"Did [BitFit] help you get an idea of where you stand with respect to the course content? Why or why not?"*, many students commented on the benefits of being able to use BitFit to complement the learning from lectures and assignments:

"Yes because it progressed with the course so it was easy to check your own progress and understanding."

"Yes. It makes some of the material clear and gives me more experience to work it out and think it through."

"Yes, it helped with understanding what I needed to practice more of and what I was comfortable with."

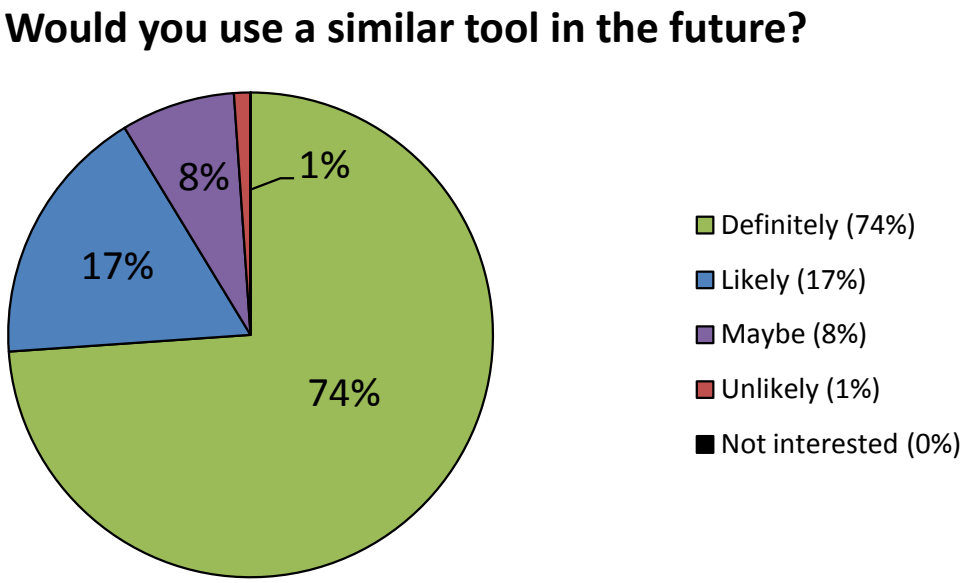


Figure 2.3: Student interest in using a similar tool in future courses, even if usage does not contribute to course credit.

> "Yes. It was good to tie it all together in questions that weren't as big as the assignment."
>
> "It did because of the sectioned layout; I think having a section for each chapter, and then combining them all together with tracing and writing code is very effective to help sort out where I was uncomfortable."
>
> "It was helpful in catching some small problem areas and made me think a little more. It was also good practice in solving novel small problems rapidly rather than the large scale types of problems presented in assignments."
>
> "The questions asked were a different style than those in lecture which made it a bit more of a challenge and made sure you understood the content."

Students also commonly reported that BitFit allowed them to better identify what areas they needed extra practice in:

> "Not all questions were same in difficulty so it showed me where I'm strong and weak at."
>
> "Yes, it helped me realize that I still had more to learn in some cases, and helped me practice that."
>
> "It did, because at the beginning I knew nothing and it really showed and now I know a little bit. I don't get the same error messages I use to!"

> "Yes, provided immediate feedback on what concepts I had trouble with."
>
> "It made me realize how little I knew, which then made me study more. It was a helpful way for me to evaluate how well I knew the topic."

On the other hand, some students reported that BitFit was not helpful in the more difficult problem-solving concepts found near the end of the semester, especially during the first semester BitFit was used, when there was not a full solution available to students:

> "I think a "solutions manual" is needed in order to compare why our codes were falling short."
>
> "In the first half of the course this statement holds true, but as the tone of the course shifted from displaying that you knew how to implement certain features into your code, and transitioned into problem solving with the material, this tool was less capable of giving me a clear idea of how good my grasp on the material was."

For the question *"Did [BitFit] help with your understanding of the topics you struggled with?"*, many students reported that BitFit did support them:

> "I often have trouble tracing code and felt that [BitFit] really helped me learn how do to it. I also learned a lot of tricks for dealing with arrays."
>
> "I used to struggled with I/O coding, because there are many java grammars I need to understand and remember. After I practiced it [in BitFit] I can understand why I need to use it and how can I use it."
>
> "Yes, helped me figure out how to make sense of nested for loops and some complex array problems."
>
> "Yes, the practice allowed me to identify areas that I need more practice in and demonstrated where I was going wrong in a useful manner."
>
> "It did, and I think I would've done far worse if not for the tool."

2.3 Discussion

Over the past three semesters, over 80% of enrolled students used BitFit. This shows that students will use a supplemental tool, even if they are not required to as a course requirement. Focusing only on students who chose not to use the tool, those that reported they did not need the help stated that if the course had been more difficult

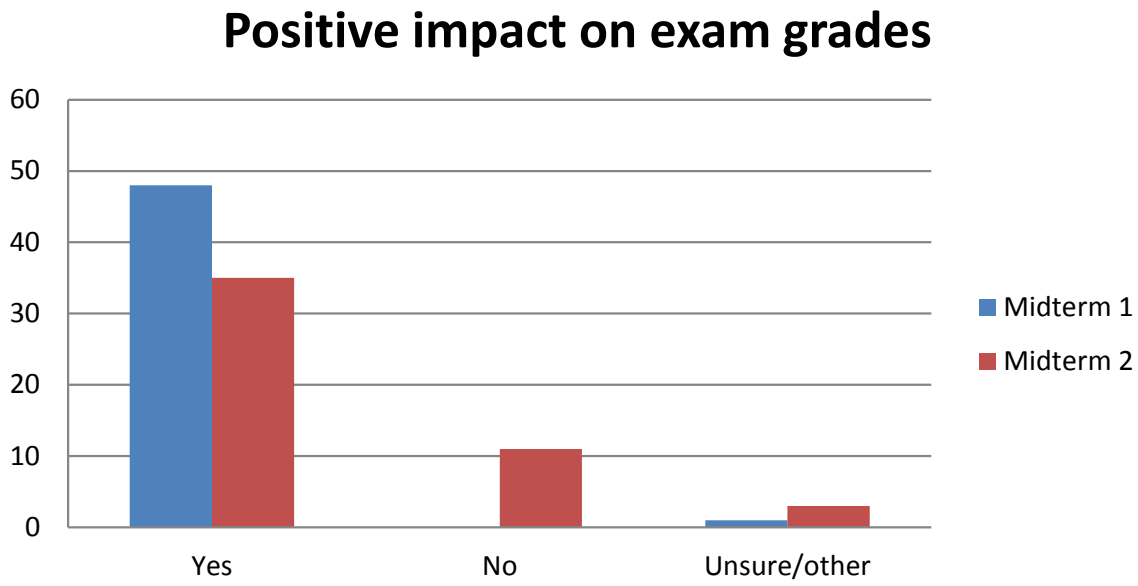


Figure 2.4: Student feedback on whether they thought using the tool improved their exam grades

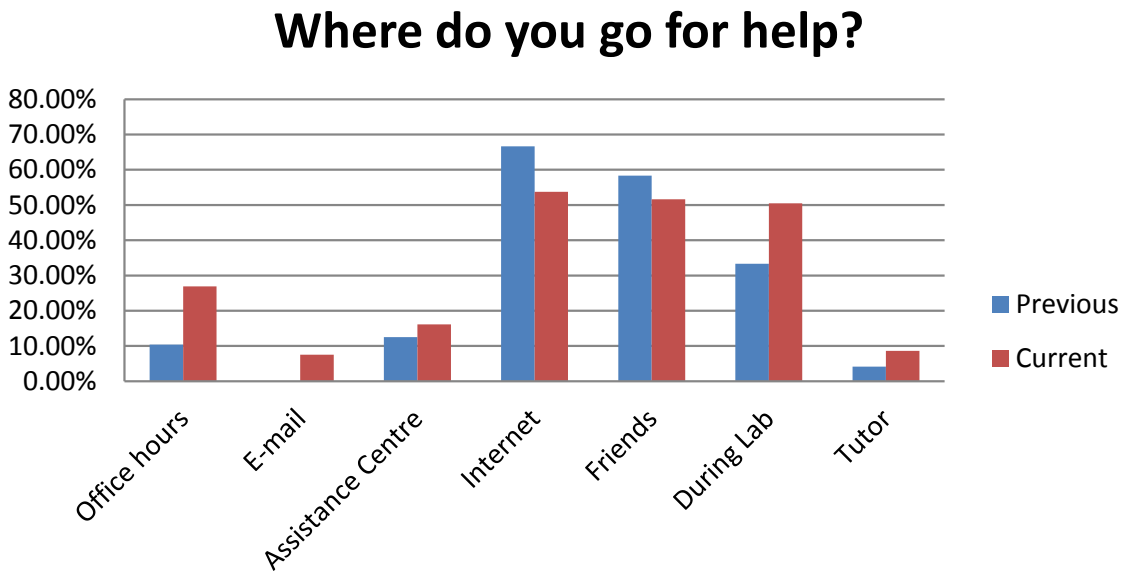


Figure 2.5: Student responses about where they go for help when struggling

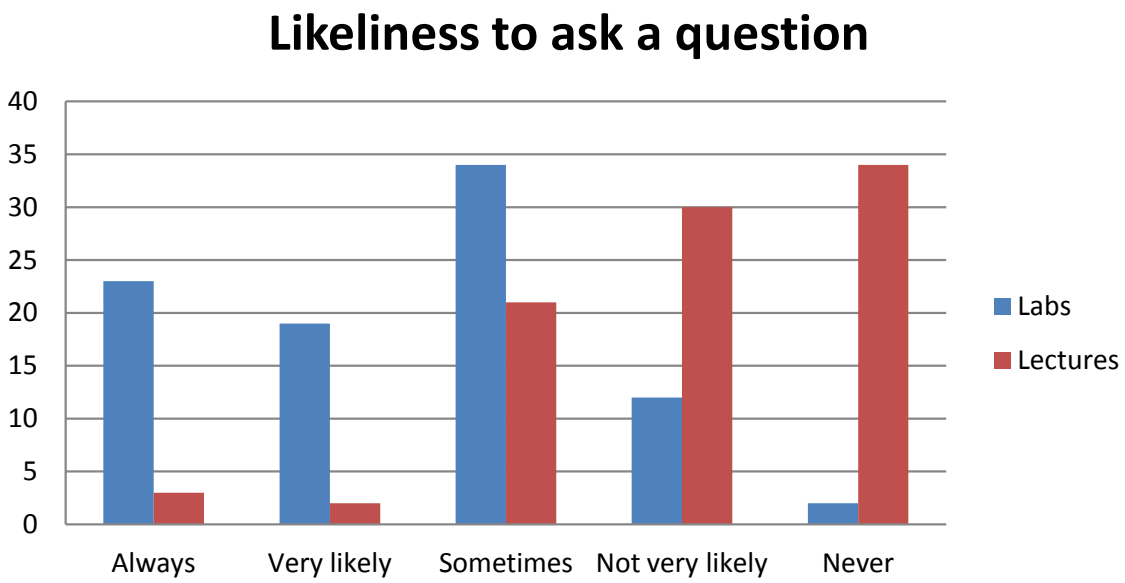


Figure 2.6: Student responses about how likely they were to ask a question in a lab or lecture when they need clarification

they likely would have used the tool for some of the more challenging material. It is unfortunate that 39% of the students who did not use BitFit reported they did not know about it at the time of the survey. BitFit was mentioned multiple times in lectures, labs, and through the course announcement system (which displays the announcement on the course website and also e-mails it to all enrolled students). The combination of students who did not know about the tool and those that did not feel like they had time to use it make up 70% of the group who did not use the tool. This is an issue what warrants future investigations, as is it unclear if this problem is associated with how well students receive information and instructions, or if certain students do not believe working through exercises is a valuable way to spend study time.

The survey results highlight how well BitFit was received by students who did use it, both as a practice resource to build confidence, and as a way students could measure their own progress with respect to course content. In addition to supporting student confidence in programming, many students commented about how BitFit helped them with code tracing. Code tracing exercises are introduced in lectures, and require keeping track of memory and generated output. This feedback highlights the importance of including both code reading and writing exercises.

Without adding any additional material to BitFit, students also report that they felt that it was an adequate resource for exam preparation. The semester following the one students used BitFit in, students were given a survey to compare BitFit to other exam preparation resources. BitFit was the most preferred resource for most students (61%), the second most preferred being access to old exams (31%). Most of the negative feedback on BitFit came when the hint system provided only high-level guidance for each question, during the first semester BitFit was used. Based on student feedback, code solutions were added into the hint system, providing students with a full solution to each problem.

2.4 Summary

This results of this study suggest that BitFit, a voluntary online learning tool, will be used by the majority of students, even if no course credit is awarded for using BitFit. Survey results also illustrate that students feel using BitFit increased their confidence and self-efficacy. Students were also interested in using a similar tool for courses in the future.

There is now a question repository that has over 100 questions throughout multiple topic areas. Very little overhead is required for other instructors interested in integrating BitFit into their own course; I hope that the tool and question repository can continue to evolve if educators from other institutions are interested in using either the tool or questions in their own courses.

This study only reports on student perceptions of the impact BitFit had on their levels of confidence and progress in course material. In the following study, interaction patterns collected by BitFit are analyzed to try and identify patterns associated with success in the course. The impact that using BitFit may have on student success rates is also an important question, but difficult to measure. Dividing students into two groups and splitting access to BitFit may be the easiest way to achieve a controlled study, but that was not an option in this course, so I wan unable to do such a study. It is my hope that as a community we can develop a different methodology that allows us to explore the impact and effectiveness of tools and other resources built to support learning and success in computer science education.

Chapter 3

How Do Students Use BitFit?

The findings of the previous chapter reveal that over 80% of the students who initially enrolled in the course over three semesters used BitFit. It was important that data from a large proportion of the student population was accessible, as I wanted to compare the workflow data of students from a number of different grade outcomes. The previous study reported on overall usage and student perception of BitFit, but did not look into BitFit usage data or connect it with student grades in any way. This study begins to explore the collected log data, and aims to identify trends associated with overall course performance. The hope is that if potentially unsuccessful students can be identified early in the semester through log data analysis, instructors may be able to support these students before it is too late. This motivated the key research question behind this chapter: *Can interaction patterns with BitFit predict a student's outcome in the course?* To answer the key question, in this chapter I consider the following four intermediate research questions:

- **RQ2a:** How well do students' levels of engagement with the learning tool predict success in the course?
- **RQ2b:** Can a practice tool identify differences in workflow behavior between successful and unsuccessful students?
- **RQ2c:** What are the metrics associated with success and failure?
- **RQ2d:** How well do metrics identified by the tool predict success?
- **RQ2e:** How early in the semester can at-risk students be identified?

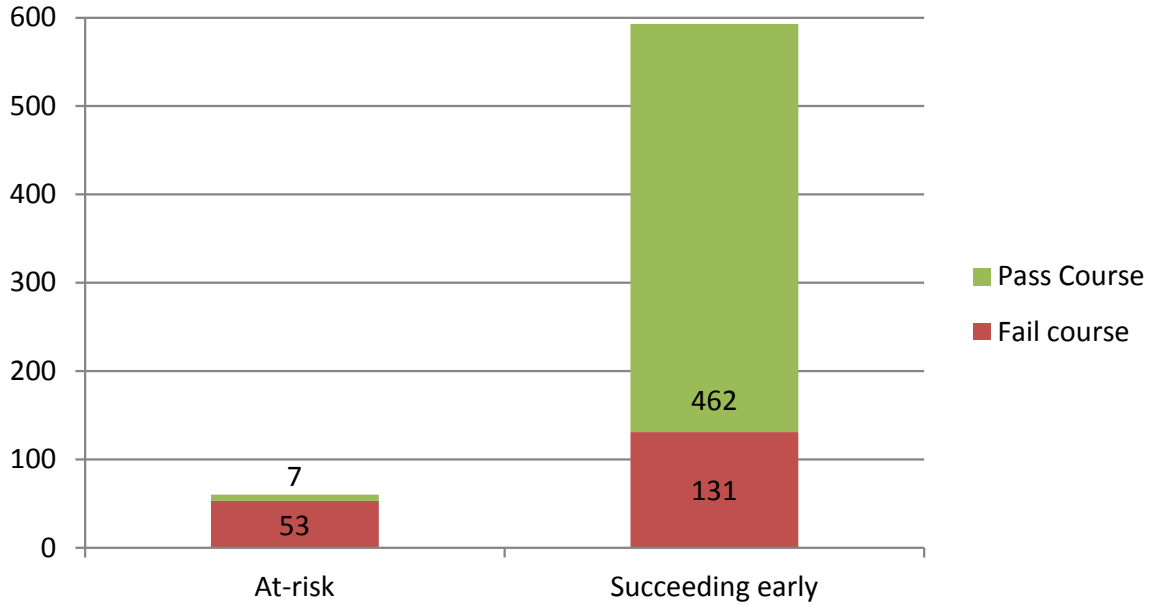


Figure 3.1: Course pass rates grouped by the "at-risk" identification metric two weeks into the course.

3.1 Methodology

For the purposes of this study, a grade of 50% or higher is a pass, and less than 50% or unsubmitted is a fail. Three semesters are considered, involving four different instructors, as Semester 3 was broken into two sections, each taught by a different instructor. An effort was made to cover the same concepts, and the assignments, exams, and grade boundaries were designed to be consistent between offerings. BitFit was introduced during the second week of all three semesters of this study.

A mixed-method approach is used, involving both quantitative and qualitative results. Similar to the previous study, BitFit was introduced during the second week of the semester, and data is collected from then until the end of the semester for log analysis. Surveys distributed every four weeks throughout each semester collected feedback on BitFit features and student progress. This feedback was used to refine features in BitFit, and compare student perception of success with usage data and course performance.

3.1.1 Objectives

The staged objectives of this study were to (1) determine how opting out of the tool was correlated to at-risk behavior; and (2) investigate whether interaction patterns

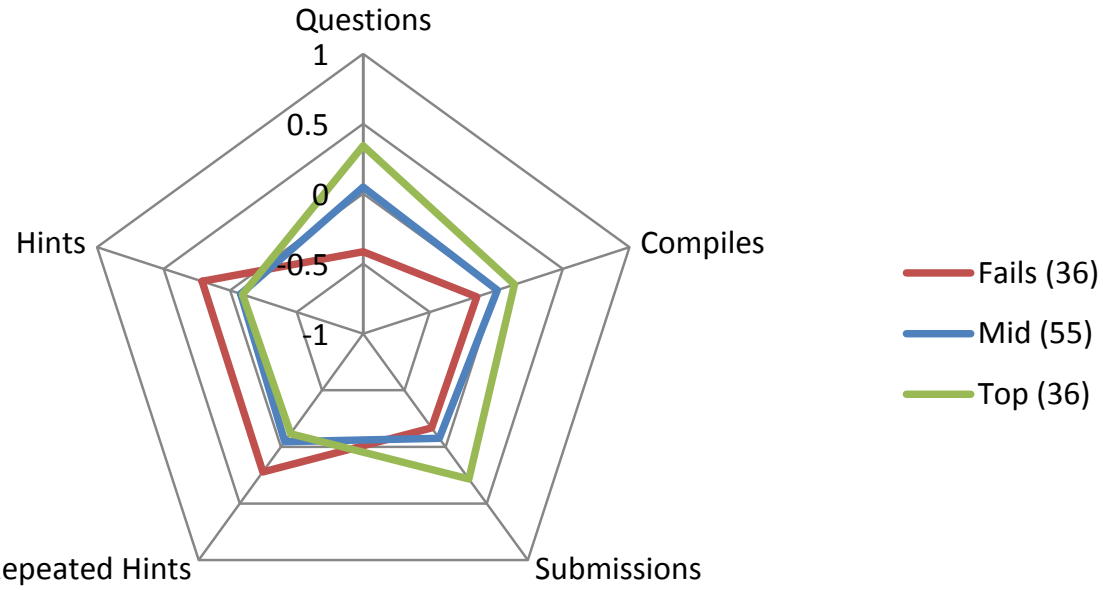


Figure 3.2: Standardized usage trends for the 127 students who used BitFit during Semester 1. Units for each axis are denoted by standard deviations from the mean (0).

with different features of the tool can be used to differentiate between successful and unsuccessful efforts to learn the material.

Previous studies show that performance on early coursework correlates with final exam scores [2, 31, 89]. I first investigated whether similar predictors could be used in this study to account for students who chose not to use BitFit.

The remaining objective was to investigate whether unsuccessful students can be identified through BitFit interaction behavior. To reach this objective, I combined quantitative BitFit data with qualitative feedback collected from monthly surveys. Features in BitFit were refined each semester. The refinements were largely driven according to students perceptions of success.

3.1.2 Threats to Validity

In terms of external validity, the limited number of semesters may not be representative of the general population. Those conducting the qualitative surveys identified themselves to the students throughout the semester, which may have introduced bias into survey responses and BitFit usage. Internal threats include behavior patterns such as students initially exploring features of the tool to familiarize themselves with it, skewing early usage patterns. Students sometimes studied in groups, affecting each individual's usage data for the questions solved together. Some questions were refined between semesters, and the hint system was restructured between Semesters 1 and 2. Students' previous computer and programming experience were not taken into account for this study. Students who had previously failed the course were able to retake the course, and would have some familiarity with the questions on BitFit.

3.2 Results

First, I wanted a metric to classify students as at-risk as early as possible in the semester that accounted for students who did not use BitFit. I investigated the 106 students who failed one of the first two assignments, and found that 84 of them eventually failed the course (79%). From this group, there were 60 students who also did not use BitFit, with an 88% fail rate. I define this group "At-risk", satisfying the first objective to accurately identify students who are unsuccessful early, potentially due to a late start.

Next, I wanted to better classify the 593 students in the group labeled "Succeeding

Early" in Figure 3.1 based on their BitFit behavior. Although only 131 out of these students failed the course (22%), these 131 students make up 71% of the overall number of course fails. The experiment is designed to investigate whether it is possible to differentiate successful students from unsuccessful students in this group based on interaction patterns found in BitFit usage data.

Radar plots are used to show the differences in BitFit usage between student groups. The axes represent the standardized average number of unique questions attempted, compile attempts, solution submissions, hints, and the number of times previously viewed hints were requested when revisiting a question. Each axes ranges from -1 to +1 standard deviations from the mean (0).

Unsuccessful students are compared with the same number of students performing at the top of the class, the third group is composed of the remaining students. Using this breakdown, Figures 3.2 through 3.6 compare three groups:

- **Fails:** the n students who failed the final exam
- **Top:** the top n students based on final exam scores
- **Mid:** the remaining students *(total BitFit users - 2n)*

3.2.1 Semester 1

The first semester BitFit was used, 156 out of 199 students attempted at least one question on BitFit, and 126 students participated in the study. A multiple regression analysis revealed that the number of questions attempted ($p < 0.01$) and questions answered correctly ($p < 0.05$) positively and significantly correlated with final exam grades. As shown in Figure 3.2, top students attempted more questions, compiled more code, and submitted more solutions. Hints were not used very often by any student group.

Qualitative Feedback

During Semester 1, hints provided high-level guidance on how to solve problems, but did not include code. I learned the following from surveys distributed throughout the semester:

- The most common *Suggestion for improvement* was to provide a full solution to each problem (47%).

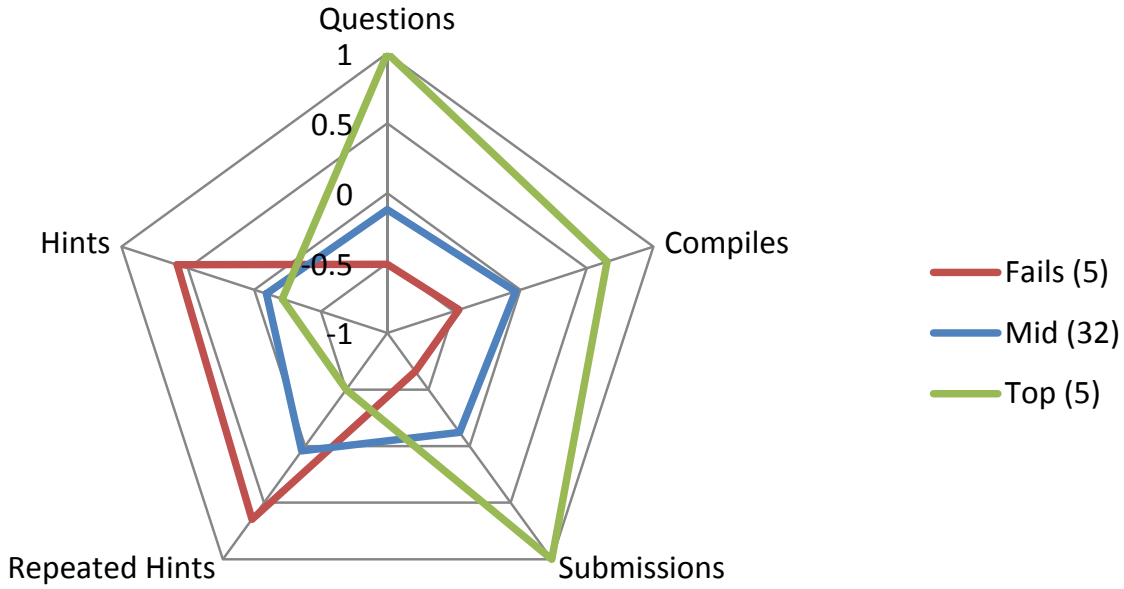


Figure 3.3: Usage trends for the 42 students who used BitFit during Semester 2.

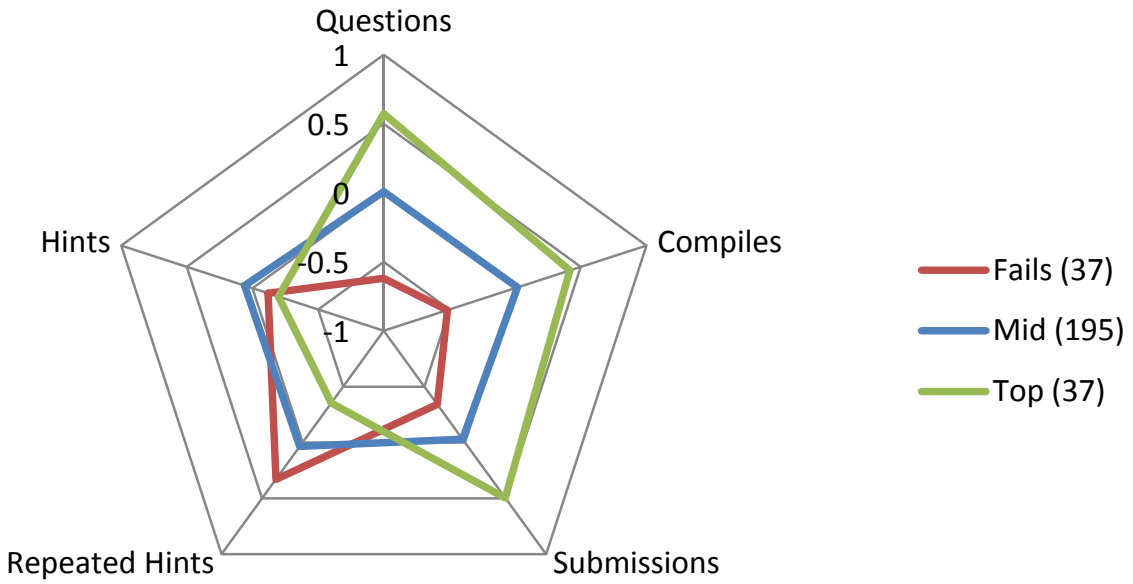


Figure 3.4: Usage trends for the 269 students who used BitFit during Semester 3.

- Students also commonly requested more questions (18%), and a larger variety of difficulty (11%).
- When asked where students went for help when stuck, the most common responses were: the Internet (66%), friends (58%), the Assistance Centre (12%), and instructor office hours (10%).

BitFit's hint features were restructured between semesters to allow access to a full solution to each problem. This change was made to accommodate the factors students perceived to be necessary for success. The question repository was also updated, to provide a more comprehensive coverage of all course topics and a wider range of exercise difficulties.

3.2.2 Semesters 2 and 3

In Semester 2, 53 out of 64 students attempted at least one question on BitFit, and 42 students participated in the study. Figure 3.3 shows BitFit usage trends for the full semester. Although many students did utilize the hint features during Semester 2, 25% of the total hint requests were made by the five students who failed. This group of students also frequently requested the same hints previously seen when revisiting a question, as denoted by the *Repeated Hints* axis. Programming trends were very similar to those found in Semester 1, with top students attempting more questions,

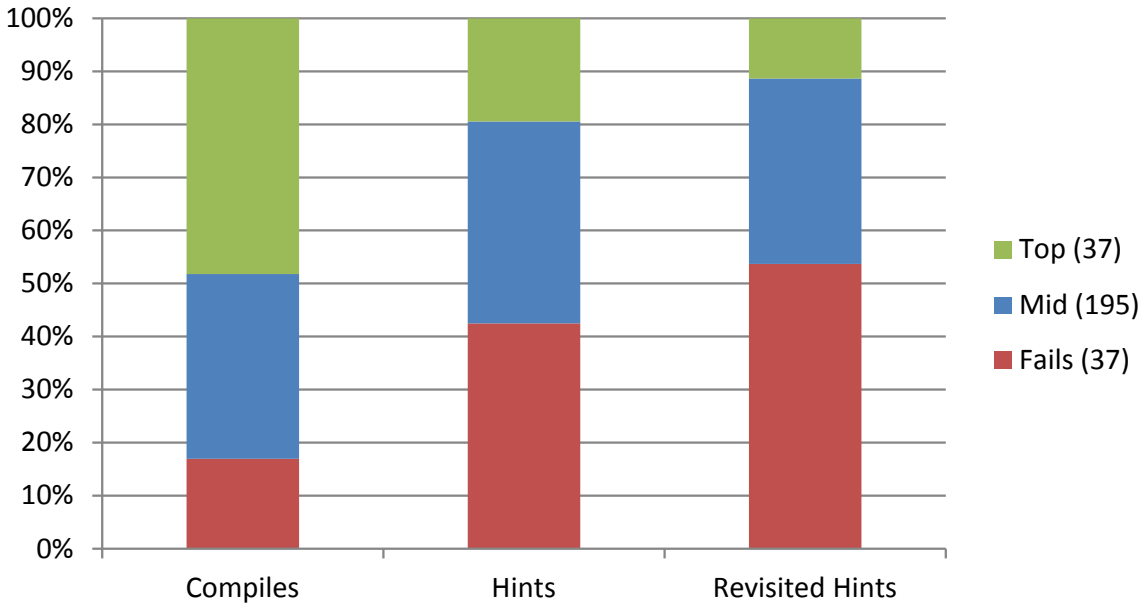


Figure 3.5: Average values for number of compiles and hint usage per question during Semester 3.

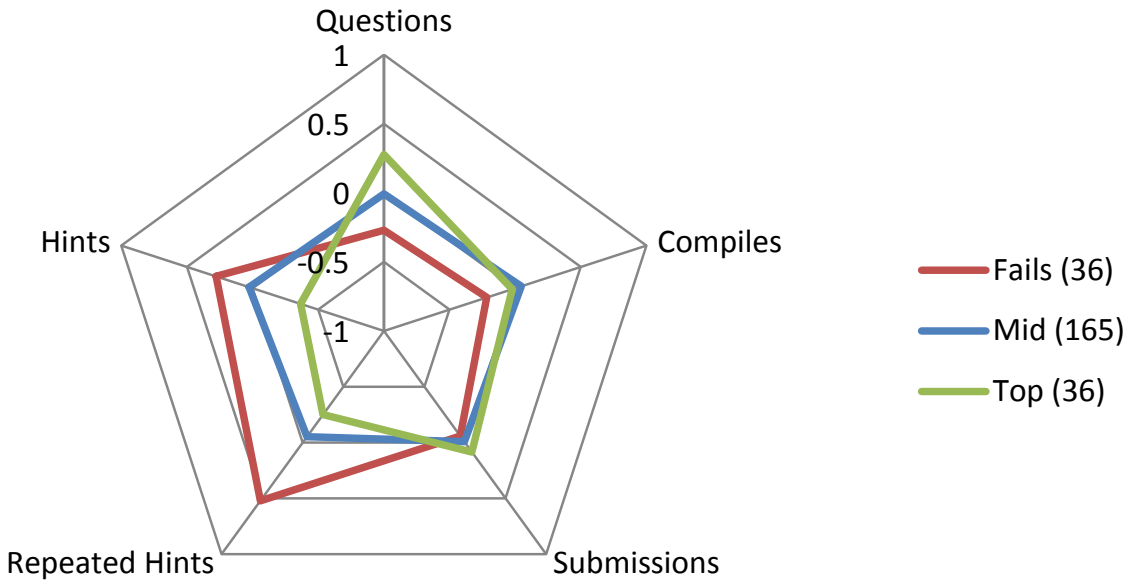


Figure 3.6: First two weeks of Semester 3 BitFit data.

and compiling and submitting more code. No changes were made to the hint features between semesters 2 and 3.

In Semester 3, 319 out of 389 students attempted at least one question on BitFit, and 269 students participated in the study. Questions attempted ($p < 0.01$), solutions submitted ($p < 0.01$), and correct solutions ($p < 0.05$) were positively and significantly correlated with final exam grades, while the number of repeated hints requested ($p < 0.05$) was negatively and significantly correlated with final exam grades.

Figure 3.4 shows student usage trends for Semester 3. Similar to Semester 2, some well-established trends can be seen with respect to compilation numbers and hint usage. Looking at BitFit usage on a per-question basis, these trends are even more pronounced, as shown in Figure 3.5. Failing students compiled 3 times less per question, but requested 4 times as many previously viewed hints as top students.

Looking specifically at the BitFit usage patterns of students with the biggest rise or drop in performance on the midterm versus final, interaction patterns with tool features align with the established successful/unsuccessful trends. Figure 3.7 shows that the 12 students with the biggest drop in exam scores exhibit behavior associated with unsuccessful students, whereas students whose grades increased the most exhibit patterns similar to top performing students.

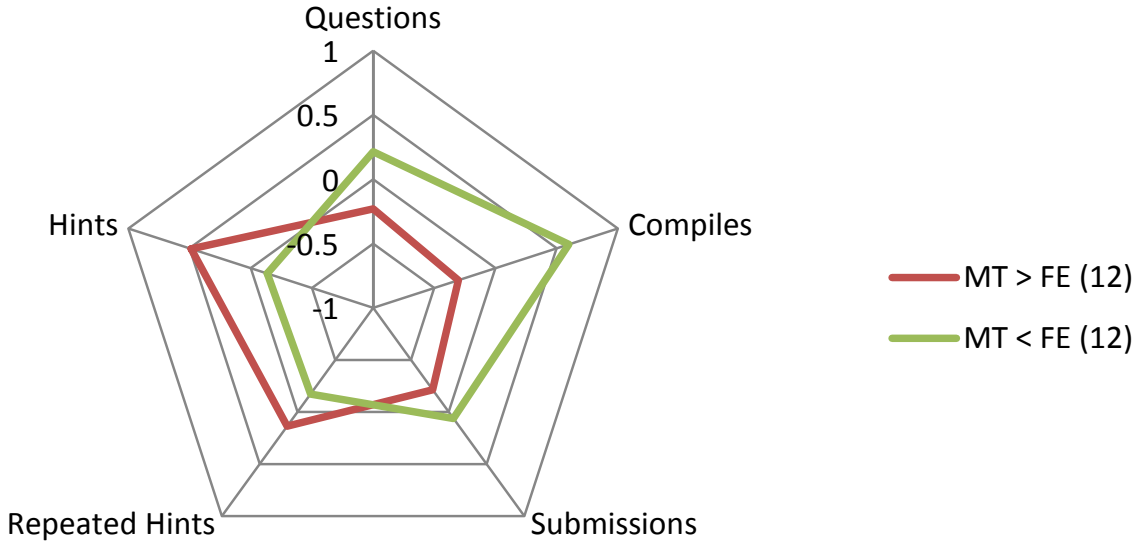


Figure 3.7: Semester 3 BitFit data for the 12 students with the biggest score differences between the first midterm (MT), and final exam (FE).

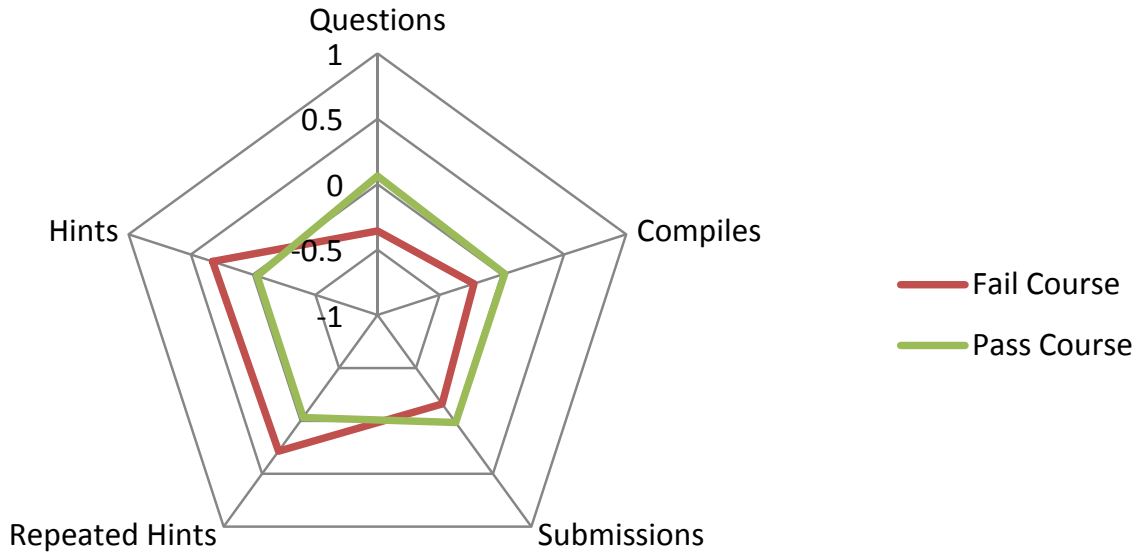


Figure 3.8: BitFit data from the first two weeks for students who pass the first two assignments.

Qualitative Feedback

Student survey feedback about hint features was positive in Semesters 2 and 3. All students selected either "Helpful" or "Extremely Helpful" on a 5-point Likert scale. Students in the failing group most commonly reported that they used hints to initially learn the material as well as when they were stuck. Students in the top group most commonly reported that they used hints to compare their solution with "an instructor's solution". At five weeks into the semester, 36 of the participants had not yet attempted a question on BitFit. Of these 36, 14 students reported they did not know the tool existed, 11 that they did not have time to use it, 8 did not feel like they needed any help, 2 found the material on BitFit too easy, and 1 student found it too difficult.

3.3 Analysis

Semester 3 had the highest number of participants, so I took a closer look at some of the aforementioned results to further analyze BitFit usage trends. First, I highlight how the findings of this study support the notion that at-risk students can be identified as early as two weeks into the semester, and how to potentially decrease false positives by sharpening a metric for engagement. Then semester-long trends are discussed, with particular focus on the effectiveness of using interaction patterns collected in BitFit

to identify unsuccessful students who were not identified early. Given that these students demonstrate engagement, the disparity between their perception of successful behavior and actual learning can be identified in their patterns of interaction with BitFit.

3.3.1 Early Identification

Figure 3.6 shows standardized BitFit usage over the first two weeks of Semester 3. Compared to the semester-long data (Figure 3.4), differences between failing and passing students are not as pronounced, but trends are already beginning to develop with respect to the number of question attempts, compilation rates, and hint usage. Only 22 of the 77 students (29%) that failed one of the first assignments used BitFit during the first two weeks of the semester, compared to 211 of the 313 students (67%) that passed both assignments. Figure 3.9 shows the usage trends between these two groups two weeks into the semester. The at-risk group attempted 13% fewer questions, compiled 33% less often, and requested 71% more previously viewed hints than passing students. Even early in the semester, the at-risk group engaged in activity that the qualitative results suggest align with what students *incorrectly* perceive as successful behavior.

Focusing only on students who passed the first two assignments, Figure 3.8 shows BitFit usage trends over the first two weeks of the course. Although students who ended up failing the course did request more hints and compiled slightly less than students who passed, usage trends are similar enough that it would be very difficult to identify students individually. At this point in the semester, these students had passed all graded work, so it is possible that they did not yet exhibit study habits associated with students at-risk of failure.

Figure 3.10 shows semester-long data for the same sets of students. Throughout the semester, the differences between students who pass and those who fail become increasingly pronounced. This suggests that although it may be difficult to identify potentially unsuccessful students after two weeks, certain trends develop as the semester progresses. Students who succeed with early material but end up failing increasingly exhibit trends associated with at-risk behavior.

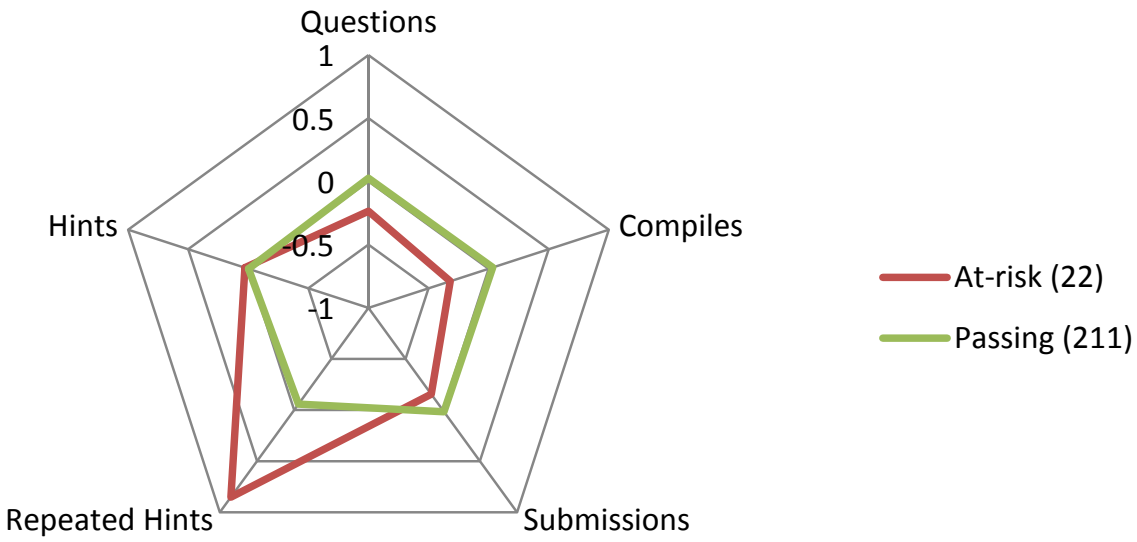


Figure 3.9: At-risk behavior patterns in Semester 3.

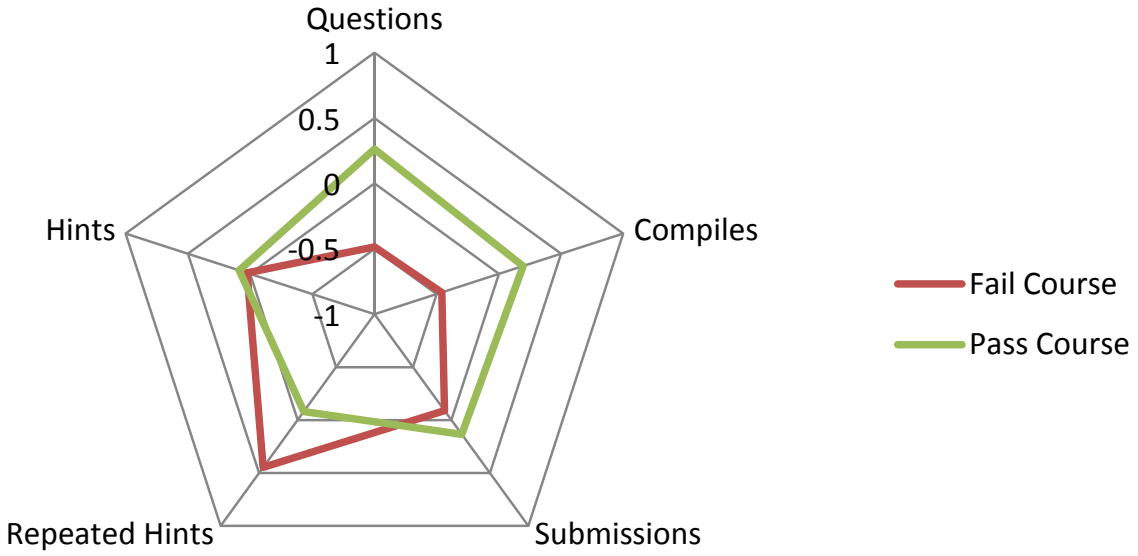


Figure 3.10: Semester-long BitFit data for students who pass the first two assignments.

3.3.2 Overall Trends

Based on the analysis of data collected over three semesters of the course, the overall trends are as follows:

- Many at-risk students can be identified with high certainty as early as two weeks into the semester.
- Low compile rates combined with high, repeated hint usage are behaviors associated with at-risk students.
- At-risk students exhibit this behavior early, and continue to exhibit such behavior throughout the semester.
- Successful students attempt more questions, and their behavior is characterized by a high number of code compilations and submissions.
- Workflow patterns of students who succeed with early material but struggle later become increasingly similar to at-risk behavior.

Over the three semesters BitFit was used, 184 out of 652 (28%) students failed the course. Figure 3.11 shows the distribution of at-risk students identified throughout this study. Overall, the metrics used in this study identify 149 out of the 184 (81%) students who failed over the three semesters.

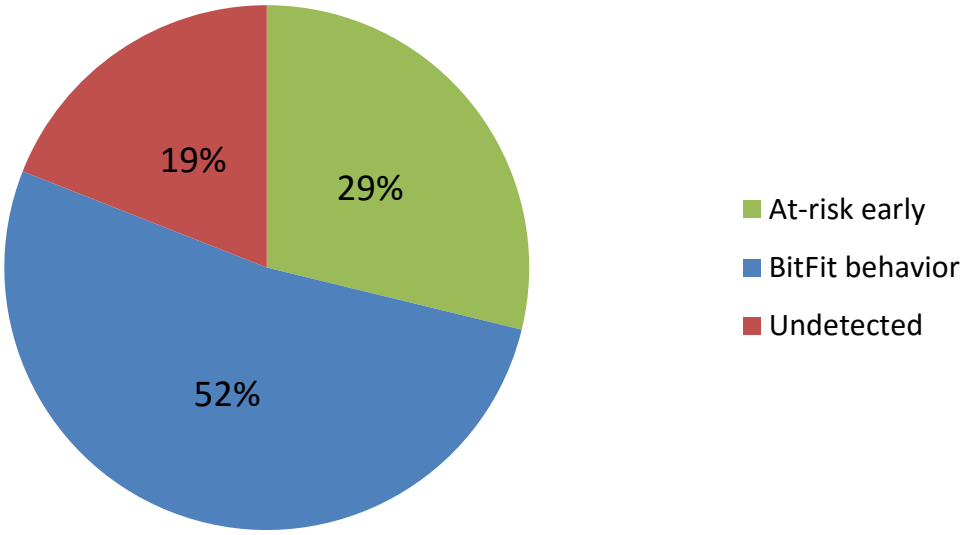


Figure 3.11: Proportion of failing students identified.

3.4 Discussion

This study shows that interaction patterns students may have perceived to be associated with success (accessing a solution) did not always result in effective strategies for learning. A number of students frequently requested all of the hints on a question, but never compiled any code. This behavior was exhibited most commonly by students who ended up failing the final exam. With the way data is collected in BitFit, it is not possible to see if there was any code written, only that the student did not choose to compile any code before leaving the question.

In response to student surveys, full solutions were added to BitFit. Although many at-risk students repeatedly failed graded coursework throughout Semesters 2 and 3, 100% of survey participants reported that the hints were helpful. Based on qualitative results from survey data, students believe that solutions assist their success. This might be true for students who use solutions to check their answers, but does not appear to be true for students who keep revisiting solutions without compiling any code.

This mismatch between perception of success and course performance warrants further investigation. Perhaps some students believe that they can learn more efficiently by memorizing a solution to a problem instead of working through a question to understand the problem-solving process. Students who were successful early but who went on to fail the course increasingly exhibited these ineffective study habits.

For upcoming semesters, one possibility is to to refine the hint system to restrict users from accessing a full solution without first writing and compiling code. Another option would be to allow students to see an instructor solution only after they pass a certain number of test cases.

From a pedagogical perspective, it is not helpful to provide a full solution that accounts for edge cases to students who do not yet understand how to begin to solve a problem. BitFit must allow struggling students to be supported, but also must convince all students that writing and exploring their own code enhances learning more than viewing a solution to a problem does.

3.5 Summary

This analysis of 652 students over three semesters of the CS1 course at the University of Victoria provides evidence that almost 30% of at-risk students can be identified

within the first two weeks of the semester. Perhaps more importantly, the results reveal that students who succeed on early topics but end up failing the course exhibit identifiable patterns of interaction with BitFit based on their compilation to hints-viewed ratios. Overall, the metrics introduced in this study identify 81% of the students who end up failing.

Looking into the BitFit data, at first glance it may seem that failing students simply work less than successful students. To a certain extent this may be true, but many failing students use BitFit heavily—an *ungraded* practice resource. The fact that these students are using BitFit suggests they want to succeed with the material. One possibility is that some students think that their study habits and methods of learning are correct, and are unaware that memorizing a solution is not an effective way to learn CS1 material.

In future research, I plan to do an analysis of questions split by topic and difficulty, as all questions were treated equally for this study. In addition, I plan to collect feedback from students about their own progress, and link this with BitFit data collected throughout the semester.

It is critical to understand precisely why students engage in ineffective learning behavior to be able to determine how to best guide at-risk students towards habits that better enable success. I have shown that tools like BitFit are able to identify at-risk students early in the semester, but many questions still remain about how to best provide intervention to support all students. Beyond this, it is difficult to evaluate the effectiveness interventions aimed at improving study habits have on overall student success rates.

I plan to continue research focused on better understanding where, when, and why students exhibit different workflow patterns. As we as educators continue to learn more about why certain students do not engage with learning opportunities effectively, we can continue to improve support initiatives to reach an increasingly broad population of students. I hope that such efforts will result in a more diverse set of strong, empowered programmers, and better success rates in our computer science programs.

Chapter 4

What Does Learning Look Like?

The results in the previous chapter suggest that it is possible to identify low-performing students, who tend not to recover, within the first few weeks of the semester. The analysis of BitFit log data was also used identify different patterns commonly associated with high and low-performing students. If low-performing students can be identified, and patterns associated with top perfroming students are known, it may seem like it should be easy to guide struggling students towards success. Unfortunately, guiding a struggling student towards programming behaviour associated with a top performing student may not be an effective support strategy. First of all, because many students enter CS1 courses with a wide variety of previous experience, a top performing student may not be the student who learned the most throughout the semester. It is unrealistic to expect an inexperienced and struggling student to be able to smoothly shift and begin to exhibit programming behaviour similar to a student with previous programming experience. Instead, I wanted to identify which students were *learning* effectively throughout the semester, and model support strategies around these students. This required that I identify what learning looks like in BitFit log data.

Learning can be defined as an enduring change in behaviour, which results from practice or other forms of experience [102]. This study builds on the previous results by investigating whether the metrics used in the early predictors of student performance can be improved through an analysis of log data that measures changes in programming behaviour over time. If patterns associated with effective learning can be automatically classified in log data early in the semester, it may be possible to better guide students at-risk of failure towards more productive learning behaviour before it is too late. This motivates the key research question behind this study: *Can*

analysis of patterns in interaction data help us understand how to detect and measure learning in CS1? To answer the key question, this analysis considers the following intermediate research questions:

RQ3a: How accurately do early predictors of performance identify students who are unsuccessful in the course?

RQ3b: How well can early predictors be improved by a trajectory metric that classifies behavioural change over time?

RQ3c: How well can the trajectory metric be extended to evaluate differences in proficiency across topics?

4.1 Methodology

In a recent ITiCSE Working Group, Ihantola et al. [53] identify the critical need for validation and replication studies, to better understand contributing factors and reasons for results. In the taxonomy for replicating studies they propose, I begin with an 'extended analysis (A)' type of verification study, as it extends the study done in the previous chapter by looking at a previously analyzed data set, but adds new analysis methods. To enable other researchers to do other types of reproduction studies, I provide a repository[1] containing the open-source learning tool used for this study, the interaction data collected over the past 2 years, and a number of interactive visualizations based on the analysis.

4.1.1 Data collection

The data for this study was collected over four semesters from the same 13-week CS1 course taught in Java. Topics covered include variables, control flow, methods, conditionals, loops, I/O, arrays, searching and sorting algorithms, and objects.

Interaction data is collected from BitFit, a programming practice tool. BitFit use was completely voluntary, and does not affect student grades in any way. Students use an embedded editor to work through exercises in a web-browser. The tool is introduced as a supplemental practice resource offering exercises similar to those presented during weekly labs. The tool provides automatic feedback to students on submission correctness. Buttons to compile code, run code, submit a solution, get a

[1] www.github.com/aestey/BitFit/

hint, and ask a question are all instrumented to collect student interaction patterns. As an optional feature, students can use a series of hints that progressively lead them through each question, and the final hint provides a full sample solution.

Data was collected from the 514 students who opted to take part in this study, out of the 718 enrolled in the course over the past four semesters. Log data was linked to final exam scores, which were standardized to account for potential changes in difficulty between semesters.

4.1.2 Early predictors

In the previous chapter, I found that, on average, high-performing students had higher compile rates and lower hint usage than students who failed. In this study, I first examine how effective these metrics are as early predictors for identifying individual students who are at-risk of failing. This first analysis considers data collected three weeks into the semester.

4.1.3 Trajectory metrics

I then extend the early analysis to further explore whether early predictors can be refined by comparing changes between subsequent programming sessions. To detect students who struggle early in the semester, and continue to struggle over time, I introduce two metrics:

Baseline: the metric calculated to provide an initial measure of proficiency on each question attempted.

Trajectory: the metric calculated to quantify changes in programming behavior over subsequent attempts.

As students work through practice exercises, they are given a score between 0.0 and 1.0 based on their compile and hint usage. A score of 0.0 on a question denotes that only hints were used, and no code was compiled. A 0.25 means hints were used before any code compilations. A 0.5 value represents a comparable split between unassisted progress and hint usage. If progress was made with minimal support a 0.75 is awarded, and a score of 1.0 signifies that a student worked through the question correctly without using any hints. The value of the baseline metric ranges between 0.0 and 1.0, and is calculated by averaging the individual question scores for each student's first attempt on each question. Table 4.1 shows sample data for two

students, and their associated baseline metric values.

	Q#	Behaviour	Score	Baseline
Student A	1	CCC	1.0	0.88
	2	CCCCC	1.0	
	3	CCCCCCCCC	1.0	
	4	CCCHHHCC	0.5	
Student B	1	HHHHH	0.0	0.25
	2	CCHC	0.5	
	3	HHHHCHH	0.25	

Table 4.1: Baseline metric showing the number and order of hints and compiles for each question (Q#)

The trajectory score measures changes in programming behaviour across programming sessions. Sessions are determined by time stamps, and are delineated by periods of time when a student does not interact with the learning tool for more than two hours. Whereas the baseline metric represents the average question score during each student's first attempt, the trajectory score represents the changes in scores between attempts. The trajectory score considers only data from exercises revisited when a correct solution was not submitted in previous attempts. When comparing individual scores between subsequent attempts, a value above 0.5 represents a change in behaviour towards unassisted progress, whereas below 0.5 signifies that hints were requested earlier or more often.

Unsuccessful students may not show signs of struggle until later in the semester, or may only struggle to progress through certain course material. The trajectory metric is also applied to interaction data separated by topic, to identify at-risk behaviour related to a specific topic.

The metric calculations are not yet a feature included in the tool, and were completed as a post-process using interaction data collected by the tool.

4.2 Results

First, I explore how well early predictors can be used to identify students in need of assistance. I then explore whether the accuracy of early predictors can be improved by measuring changes in learning behaviour over time with the trajectory metrics.

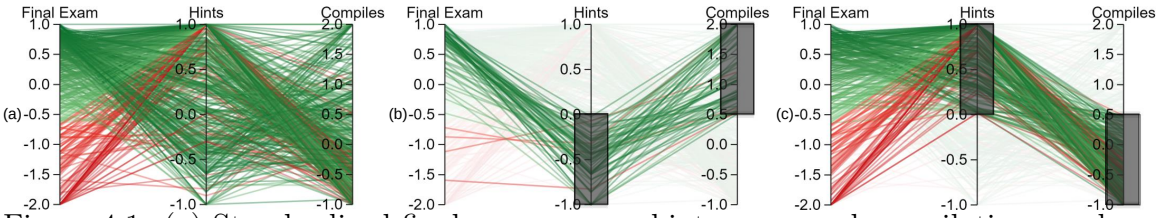


Figure 4.1: (a) Standardized final exam scores, hint usage, and compilation numbers for 514 students over four semesters. Students who passed the final exam are represented with green lines, red lines represent failure. (b) Students with below average hint usage and above average compile rates are selected, as highlighted by the rectangles on the axes. Among this group, only 4 students, represented by red lines, were unsuccessful on the final exam. (c) Selects students with above average hint usage and below average compile rates.

4.2.1 Early predictors (RQ1)

Figures 4.1 and 4.2 show standardized compilation attempts, hint usage, and final exam scores for student population in this study. The graphs are colored based on exam performance, green representing passing grades and red failure. Figure 4.1a illustrates the variation across student data sets collected for the study. The goal was to examine how well compile rates and hint usage can be used to identify students struggling early in the semester. Figure 4.1b selects only students with below average hint usage and above average compilation rates, as highlighted by the shaded rectangles on the axes. Only 4 of the 94 students who were unsuccessful on the final exam exhibited this behaviour. These 4 students represent the relatively small number of false negatives. Conversely, Figure 4.1c shows students with below average compilation rates and high hint usage, where 76 of the 94 (81%) students who received a failing final exam grade are shown.

Figures 4.1b and 4.1c suggest that, in general, high compile rates and low hint usage correlate with success. Hint and compile rate scores for each student positively and significantly correlate with midterm exam grades ($p < 0.01$), and also final exam grades ($p < 0.01$). This does not mean, however, that these early predictors can be used to accurately identify *only* students who are genuinely at-risk of failure. At this point in the semester, it is very difficult to distinguish between the 257 students identified as at-risk; Figure 4.1c illustrates the wide range of final exam grades for these students. Table 4.2 shows that 181 of the 257 students appeared to be struggling early, but managed to succeed on the final exam without intervention. These 181 students make up 43% of all students who succeed in the course, a relatively high false positive rate.

Classification:	At-risk	Not at-risk	Total
Fail exam	76	18	94
Pass exam	181	239	420

Table 4.2: Early at-risk identification

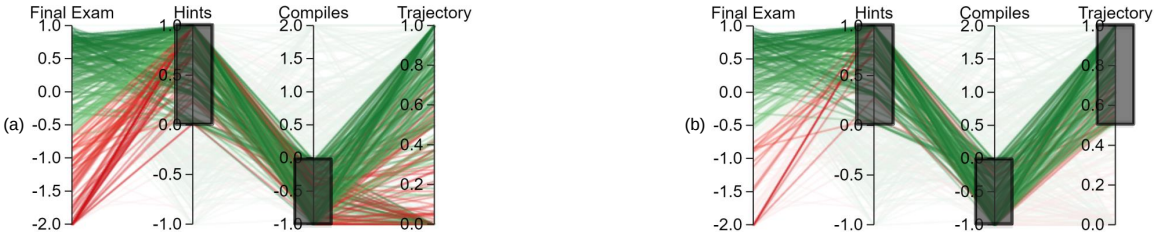


Figure 4.2: (a) students identified as at-risk by the baseline metric, shown by the rectantangles selecting above average hint usage and below average compile rates. (b) students who would be filtered from (a) by the trajectory metric, dropping the false positive rate from 43% to 11%, and the true positive from 81% to 70%.

4.2.2 Trajectory over time (RQ2)

The trajectory metric considers data from exercises revisited when a correct solution was not submitted in previous attempts. Negative trajectory scores represent instances where students are actively struggling to complete an exercise over multiple subsequent sessions.

Figure 4.2a differs from Figure 4.1c in that an additional axis is added representing trajectory scores. Figure 4.2a shows the wide range of final exam outcomes achieved by the 257 students considered in the trajectory metric analysis. Applying the trajectory metric filters out 134 false-positives, reducing the number from 181 (43%) to 47 (11%), while true positives dropped from 76 (81%) to 63 (70%). The students who would be filtered out through trajectory analysis are shown in Figure 4.2b. Overall, using the trajectory identifies 70% of the students who failed, while 11% are mis-classified as at-risk. Table 4.3 summarizes overall trajectory results.

Applying the trajectory metrics demonstrated precisely how the accuracy of the early results can be improved by considering changes in behaviour across programming sessions. The trajectory metric analysed the population of students identified by early analysis as potentially at-risk, and was used to more accurately identify students who were genuinely at-risk. Among this group, the trajectory score positively and significantly correlates with midterm exam grades ($p = 0.02$), and final exam grades ($p < 0.01$).

The trajectory metric results reveal that many students exhibit quantifiable changes in behaviour over time. However, it is entirely possible that students I claim to be

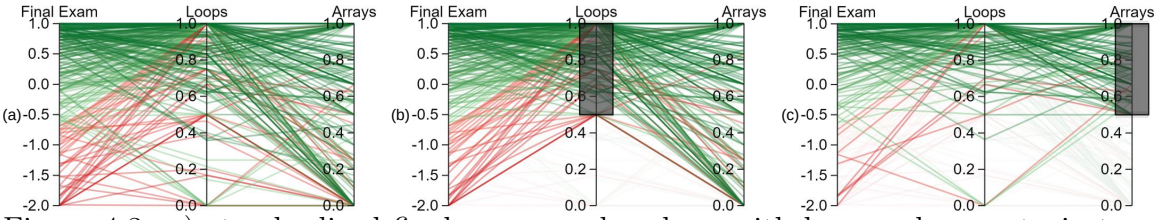


Figure 4.3: a) standardized final exam grades along with loop and array trajectory scores for data collected 6 weeks into the course. b) selects only students with positive trajectory scores on loop exercises. c) selects only students with positive trajectory scores on arrays.

mis-classified were identified accurately based on their behaviour at the time of the analysis. As there were a limited number of topics available in the programming tool at this point in the semester, the next step in the study explores if unsuccessful students who were still mis-classified could be properly identified based on their interaction data collected from later topics.

Classification:	At-risk	Not at-risk	Total
Fail exam	63	31	94
Pass exam	47	373	420

Table 4.3: Identification after trajectory filter

4.2.3 Trajectory on a topic basis (RQ3)

To measure changes in programming behaviour across topics, an analysis was performed on data collected at the end of the sixth week of the course. At this point, loops and arrays were two of the available topics in the tool. For the 223 students who completed and revisited exercises on both topics, Figures 4.3 and 4.4 show final exam grades along with scores generated by the trajectory metric applied within each topic.

Figure 4.3a shows the entire population of participants in this part of the study, whereas Figure 4.3b selects only students with positive trajectory scores on loop exercises. Of the students who ended up passing the course, 172 (95%) of them have positive trajectory scores on loops exercises compared to 31 (78%) of students who failed the course. Figure 4.3c selects students with positive trajectory scores on array exercises. At this point in the course, 125 (69%) students who passed the course exhibit this behaviour, compared to only 9 (23%) students who failed the course. Table 4.4 summarizes these results.

Figure 4.4b and Figure 4.4c illustrate how productivity levels may be different

between topics. Figure 4.4b selects students with positive trajectory scores on both loops and arrays. Among students who pass, 122 (67%) exhibit this behaviour, compared to only 8 (20%) failing students. Figure 4.4c selects students with positive trajectory scores on loop exercises, and negative scores on arrays. Figure 4.4c illustrates how 25 (63%) of the students who ended up failing the course transition from a positive trajectory score on loop exercises to a negative productivity score on array exercises.

The results illustrate distinguishable changes in behaviour across topics, especially among the studentswho eventually fail the course. Among this group, the percentage of students identified as at-risk grows by 57% when comparing interaction data collected on loop exercises to data collected on array exercises. This illustrates that, in addition to applying trajectory metrics to measure changes in behaviour over time within a single topic, measuring behavioural changes across multiple topics may also reveal students struggling in very specific areas of the course.

	Loops		Arrays	
Classification:	At-risk	Not at-risk	At-risk	Not at-risk
Fail exam (40)	9	31	31	9
Pass exam (182)	10	172	57	125

Table 4.4: Trajectory metric across topic areas

4.3 Analysis and Discussion

In this work, I explored whether behaviour patterns can be found in interaction data that allow for dynamic classification of learning in CS1. The results support the findings of previous studies that suggest that a large number of at-risk students can be identified early in the semester. On the other hand, the new approach using the

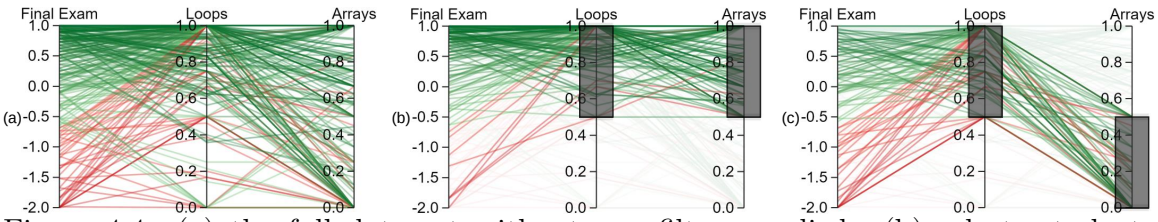


Figure 4.4: (a) the full data set without any filters applied. (b) selects students with positive trajectory scores on both loop and array exercises. c) selects students with positive loop scores and negative array scores. Of the 40 students who failed the course in this data set, 63% students exhibit these changes in behaviour between topic areas introduced two weeks apart.

trajectory metric highlights the fact that many students struggle at different times, and in different topics, throughout the semester.

It is important that we continue to improve the efficacy of early predictors in CS1; struggling students need to be identified and supported as early as possible in the semester. It is also important to acknowledge that there may not be a single metric that can accurately predict the outcome of every student at one particular time, simply because learning takes place *over* time. Although the early predictor used in this study correlates with final exam performance, the correlation strength was increased by a relatively large group of students who were able to consistently work through practice exercises without assistance and also achieve very high exam grades. Previous work suggests that many of these students may have entered the course with previous programming experience, which may have contributed to the fact that I did not measure significant behavioural changes for them. This motivates the key research question of this study: *can analysis of patterns in interaction data help us understand how to detect and measure learning in CS1?*

There was significantly more noise, variation, and change over time found in the data collected from the remaining students included in the study. The variation made it very difficult to distinguish between the large number of students initially identified as at-risk. This illustrates why a static predictor that correlates with course performance does not necessarily mean it can accurately detect at-risk students. The trajectory metric was incorporated into the analysis because the students I want to identify and support are not necessarily the students who enter the course without strong programming ability, but those who remain unable to demonstrate progress without assistance.

The results of this study illustrate how the trajectory metric was used to increase the accuracy of early predictors in identifying at-risk students, but I made no mention about how the metrics could be used to identify learning. In this study, early predictors classified 257 students as at-risk based on their programming behaviour over the first three weeks of the semester. Among this group, the trajectory metric identified 134 students who showed productive changes in programming behaviour over time. If learning can be defined as an enduring change in behaviour, it can be argued that this analysis reveals that these 134 students demonstrated learning. These 134 students make up 52% of the 257 students identified by the early predictor analysis as at-risk. If we can better understand the reasons why these students transition from at-risk behaviour to behaviour associated with success, it may serve as a model from which

to base intervention efforts aimed at supporting struggling students.

This study differs from previous works that analysed early predictors in CS1 because I incorporate metrics that measure behavioural change over time. In this study, these metrics were used to improve the accuracy of early predictors, but measuring learning is not limited to these specific metrics. Although in this study I created metrics to detect high-level changes in programming behaviour, the same general metrics can be applied to similar studies focusing on other factors that contribute to success in CS1, assuming the data collected allows for the following:

- Data points to generate a quantifiable measure.
- Capability to identify each student individually.
- Time stamps to evaluate change over time.
- Data recorded on the same or very similar exercises to compare between students and analyze progress.

These provide a guideline for what I found was necessary to measure changes in programming behaviour in CS1, but the metrics can be extended and improved in future work. It is important to recognize the limitations and threats to validity of this study. Students enter the course with a wide range of previous experience, and the factors that may contribute to student success outside of programming behaviour are not considered in this study. Also, the analysis includes only the data from students who both chose to use the practice tool and also consented to take part in the study.

The analysis metrics penalize hint usage, but students may not view hints only when struggling to solve an exercise. A student may exhibit different behaviour when learning a new concept or preparing for an exam, but I did not incorporate any session-specific behaviours into the metrics. The analysis metrics also do not take question difficulty into account. For the baseline metric, if a student was to skip through easy questions in a topic, and was then unable to progress through a difficult exercise without hints, this behaviour would result in a lower score than successfully completing trivial exercises without assistance. Similarly, for the trajectory metric, it may be important to know if a student is revisiting and struggling to complete an introductory question or a question designed to be challenging.

Despite these limitations, the results of this preliminary study using trajectory metrics are encouraging, and it is my hope that these findings measuring changes in behaviour might inspire other researchers to incorporate metrics that consider behavioural change into their own analysis.

4.4 Summary

This study presents an analysis of interaction data collected from 514 students working on exercises in BitFit, to help understand how to detect and measure learning in CS1. I first explored early predictors of at-risk behaviour, after which I proposed two metrics, baseline and trajectory, that I used to quantify changes in student behaviour over time and in different course topics. I then showed that these metrics are better at differentiating between sustained versus transient struggling.

I identify and describe several issues that arise when trying to identify struggling students. In future work, I plan to investigate whether interaction data can help me apply a difficulty rating to each exercise. Then, I can evaluate whether the analysis metrics can be improved by factoring question difficulty into the analysis. I also suggest collecting information on the objectives of the students during each session, as this information may provide additional context allowing educators to better understand the factors contributing to measured outcomes in their analysis.

I am also interested in further analysing the interaction data of students who did not show productive changes in behaviour. My assumption is that students who are actively working through ungraded practice exercises are highly invested in learning. Unfortunately, 63 students in this study exhibited patterns associated with sustained struggle. Further analysis may reveal patterns among these students that can be generalized as impediments to success in CS1.

In my following work, I plan to further explore the interaction data of the students initially identified as at risk in this study. Further analysis using different metrics that compares the interaction data of students who show changes in behaviour with those who do not may reveal interesting patterns related to the learning process. As a community, once we better understand the factors contributing to behavioural change, we can begin providing supportive intervention aimed at guiding struggling students towards productive behaviour.

Chapter 5

Do Students Know How to Prepare for Exams?

In the previous chapter, the analysis focused primarily on the group of students identified by early predictors as at-risk. By identifying changes in programming behaviour over time, I was able to differentiate between transient and sustained struggling. In this chapter I try to begin to understand why students exhibit certain programming behaviours. Surveys asked students to report on their perceived progress through different course topics, and responses are connected with BitFit log data. I hoped that the survey responses of struggling students might provide insight on how to better support them.

The key question for this study is: *Do students understand whether or not their study habits are likely to lead to success on the final exam?* The following specific questions were explored to answer this question:

RQ4a: How well does time-on-task differentiate between successful and unsuccessful students?

RQ4b: How well does intended question difficulty differentiate between successful and unsuccessful students?

RQ4c: Is there a difference in a student's reflection of self-efficacy between successful and unsuccessful students?

5.1 Methodology

In this chapter I also consider BitFit data collected during a fourth semester in which BitFit was used in the same course. For the quantitative BitFit data, final exam scores were standardized each semester, and students were split into three groups based on their standardized final exam score: students who were unable to pass the final exam (<-0.8 standard deviations below the mean), identified as the *Low* cohort; students with an A letter grade on the final exam (>0.8 standard deviations above the mean), called the *High* cohort; and the remaining students were labeled as the *Mid* cohort. Overall for this study, 101 students were placed in the Low cohort, 274 in the Mid cohort, and 90 in the High cohort.

Of the 183 students in the course, 155 opted to use BitFit. Only the fourth semester is considered for the qualitative survey data discussed in this chapter, because surveys were created based on the results of earlier studies, so were not present during earlier semesters. Students who used BitFit were asked a number of questions about their progress with course material, and the impact of the learning tool. The survey responses were set up using a 5-point Likert scale, with a score of 1 representing a strong negative response, whereas 5 represents a strong positive response. In total, 55 students filled out both surveys, 9 students from the Low cohort, 26 from the Mid cohort, and 20 from the High cohort. Similar to the learning tool, survey participation was completely voluntary, and participation and responses did not affect student grades. Survey questions were created to gather feedback based on the patterns in the learning tool log data found in the first two semesters of the study.

5.2 Results

First, I explore common trends found across student groups throughout the three most recent semesters the learning tool was used. I then explore whether survey responses about study habits align with collected log data during the most recent semester the tool was used.

Table 5.1: Aggregate usage trends recorded over all semesters

	Questions			Compiles			Hints		
Cohort	S1	S2	S3	S1	S2	S3	S1	S2	S3
Low	35.2	36.8	46.5	54	54	58	303	122	246
Mid	47.2	54.6	58.9	113	104	122	142	139	166
High	61.2	65.4	67.5	168	122	145	72	89	96

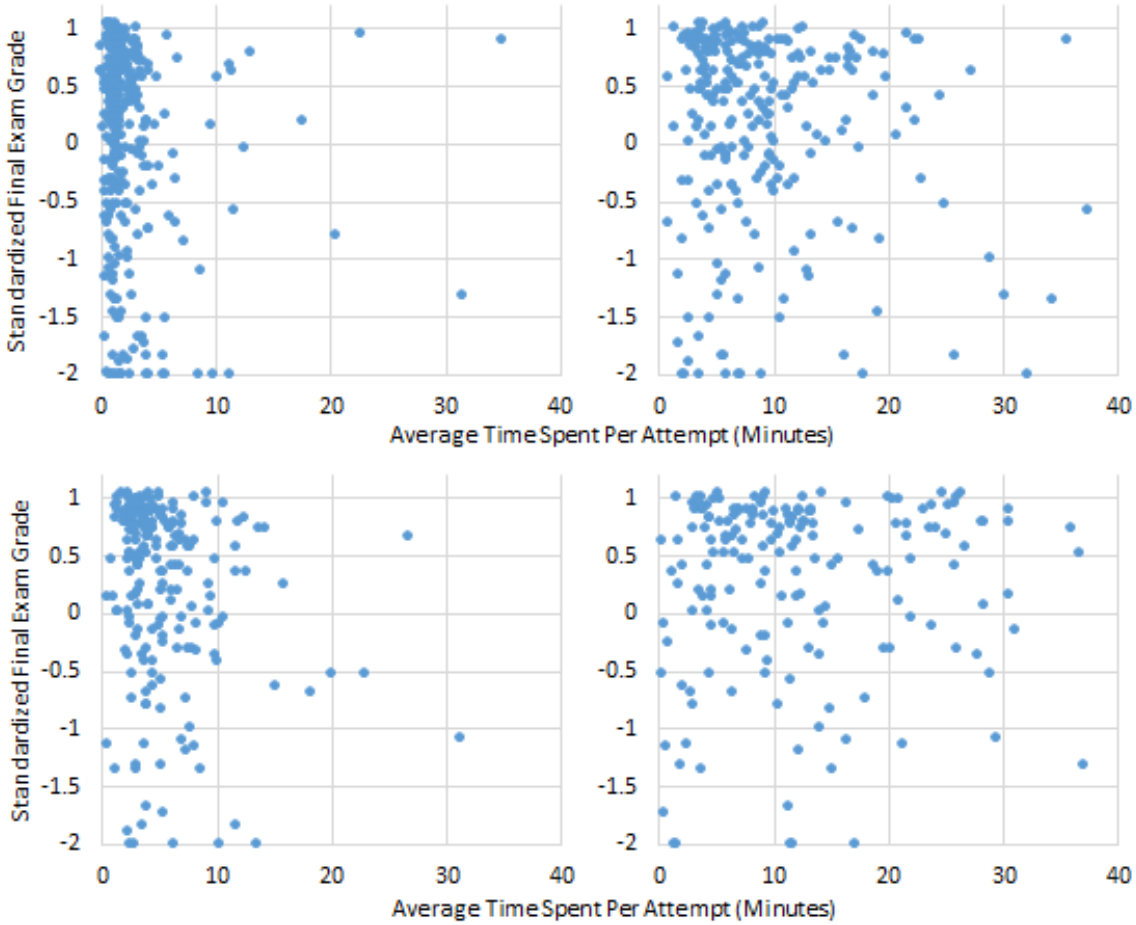


Figure 5.1: Time spent on task for students working on questions in the for-loop module, introduced in week 2. The graphs represent data collected from questions 1, 3, 5, and 7 from within this module. Final exam grades are standardized to account for possible changes in exam difficulty across semesters.

Table 5.1 shows the average number of unique questions attempted, compilations, and hints requested for the three semesters, labeled S1, S2, and S3. The data is split by cohort for each semester. Throughout all three semesters, higher performing students attempted more unique questions, compiled more often, but requested less hints. Table 5.1 shows aggregate data collected throughout the whole semester, but does not show question-specific data. For instance, Table 5.1 shows the number of unique questions attempted, but does not show how many times a student revisited a question, or which question the hints were requested on. The following sub-sections expand on this analysis by delving deeper into the data, by looking at different modules and questions independently.

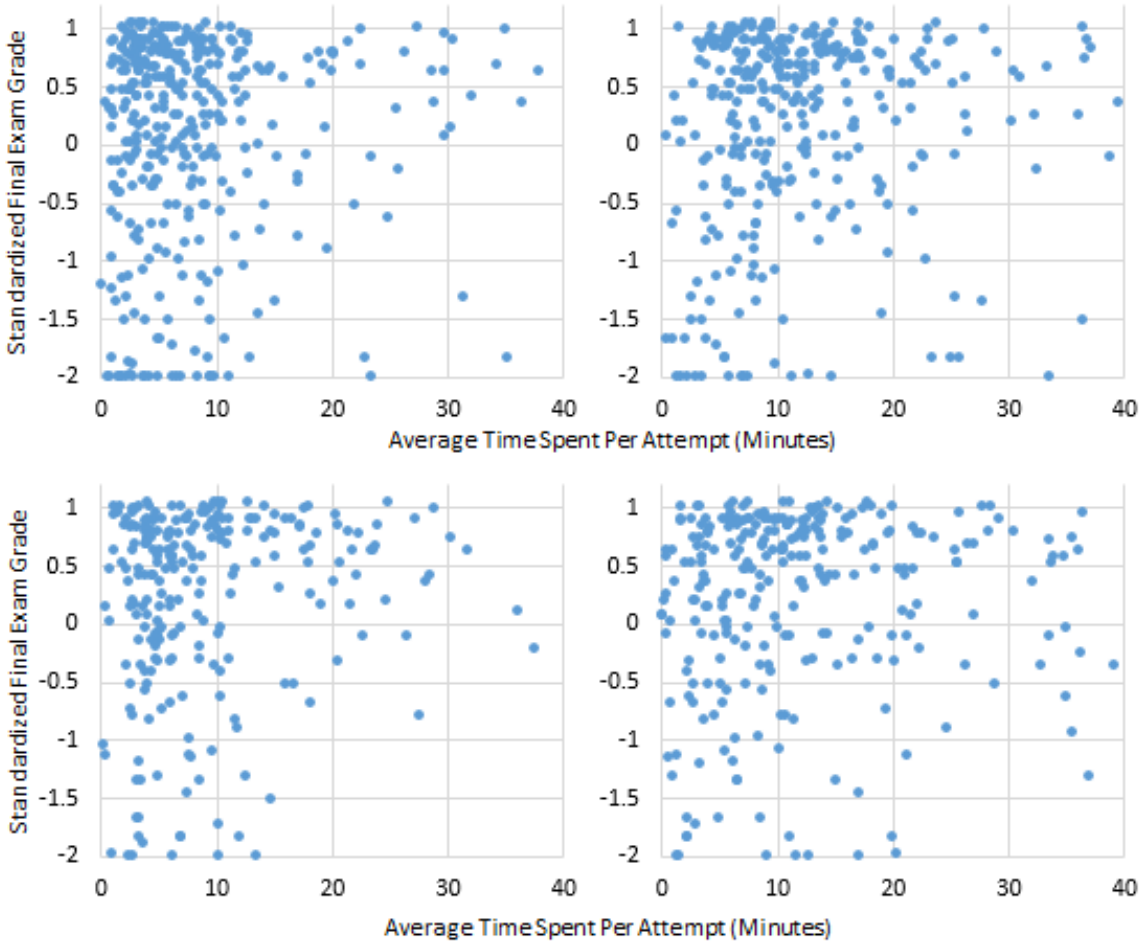


Figure 5.2: Time spent on task for questions 1, 3, 5, and 7 across all modules covered throughout each semester.

5.2.1 Time on Task (RQ4a)

To answer the first research question, I first explored whether patterns existed between student groups based on time spent on task working through exercises in the for-loop module (Figure 5.4), introduced during the second week of each semester. Figure 5.1 shows that on question 1 in the module, students across all semesters and grade ranges generally spent between 0 and 5 minutes working through the question. Although Figure 5.1 shows that time on task increased for all students when working through questions later in the module, time on task alone would not allow us to differentiate between successful and unsuccessful students in our study.

Figure 5.2 shows time on task for exercises covered in all modules throughout the semester. In comparison to Figure 5.1, the semester-long data trends show an increase

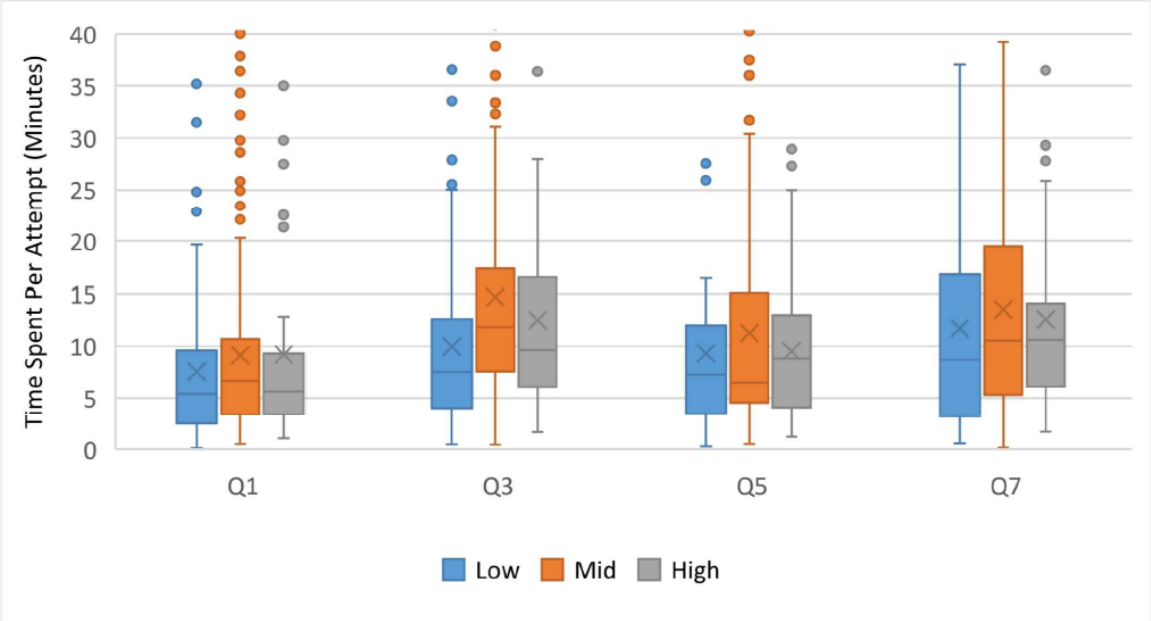


Figure 5.3: Time on task on Questions 1, 3, 5, and 7 across all modules, showing the range of the 25-75th percentile (box), min (vertical line, bottom), max (vertical line, top), average (X), median (horizontal line), and outliers for all cohorts.

the average time spent per question number, but it remains difficult to differentiate between unsuccessful and successful students using this metric. Figure 5.3 further illustrates that students cannot be identified by time on task across in this study.

It may be difficult to differentiate students based on time on task due to how the learning tool was used throughout our study. The learning tool was introduced as a supplemental ungraded *practice* tool, and students were under no pressure to complete an exercise within a certain amount of time. Students were also able to request hints, which provided step-by-step instructions and code snippets leading up to a complete solution of the problem. Figures 5.1 and 5.2 show how long students spent on each question, but fail to show *how* students spent their time working through each question. The following sub-section explores the study behaviour of students across the three cohorts, with a specific focus on hint usage in an attempt to better understand perceived question difficulty.

5.2.2 Question Difficulty (RQ4b)

The impact of question difficulty is considered from two perspectives. First, *intended* difficulty, designed to challenge the students as the questions within a module progres-

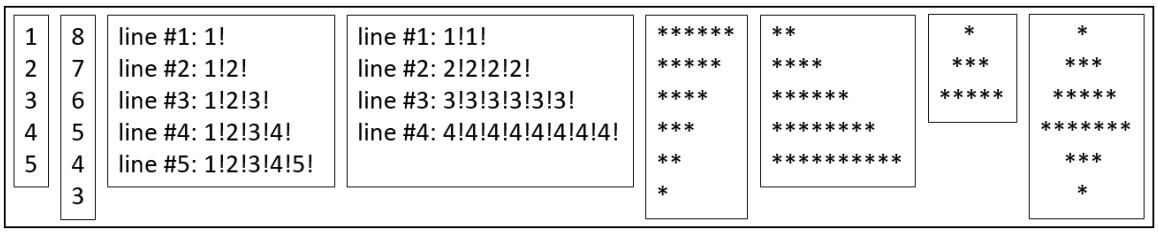


Figure 5.4: Each box outlines the output students were asked to generate in the for-loop module for Questions 1 to 8.

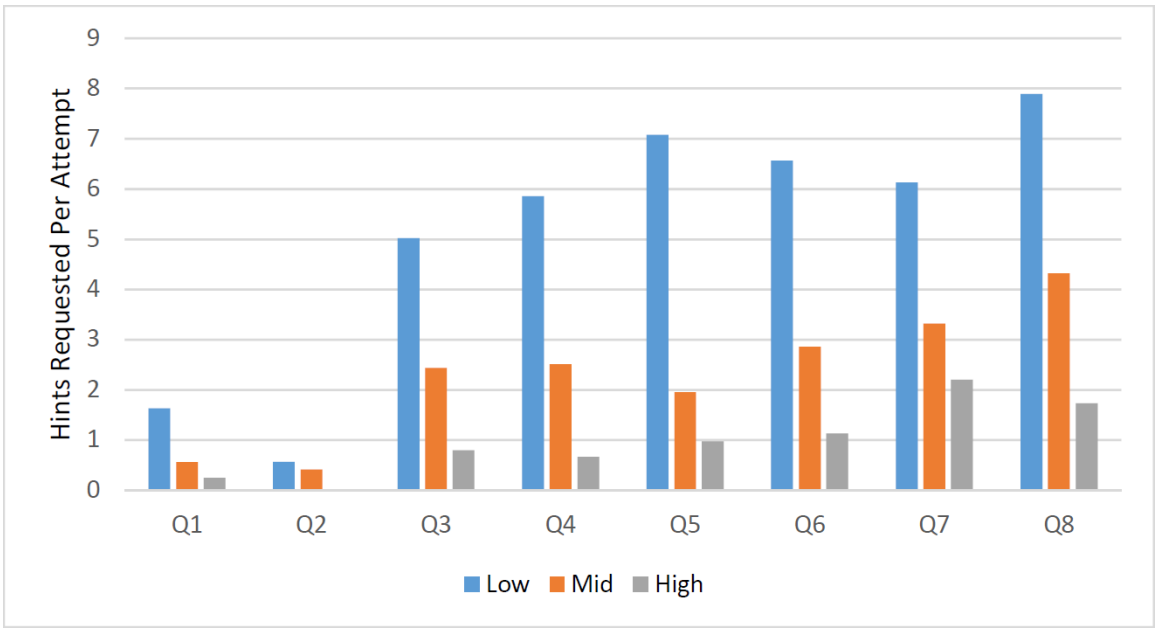


Figure 5.5: Hints requested per question number within the for-loop module.

sively build on newly acquired knowledge. Second, *perceived* difficulty, as established by student interaction with each problem within a topic module.

Within a given module, questions are designed to progressively become more challenging. Figure 5.4 shows questions 1 (Q1) through 8 (Q8) in the for-loop module. Figures 5.5 and 5.6 show hint data collected across the for-loop module on questions Q1 to Q8, which are ordered within the module by intended difficulty. These results show consistency with the trends established in Table 5.1 on the number of hints viewed among cohorts across all questions, independent of intended difficulty. Figure 5.5 shows that across all of the questions in the module, lower-performing students requested more hints on average when compared to higher performing students. In addition to this, Figure 5.6 shows that students in the Low cohort more often requested hints before attempting to write and compile any code on their own than the other cohorts.

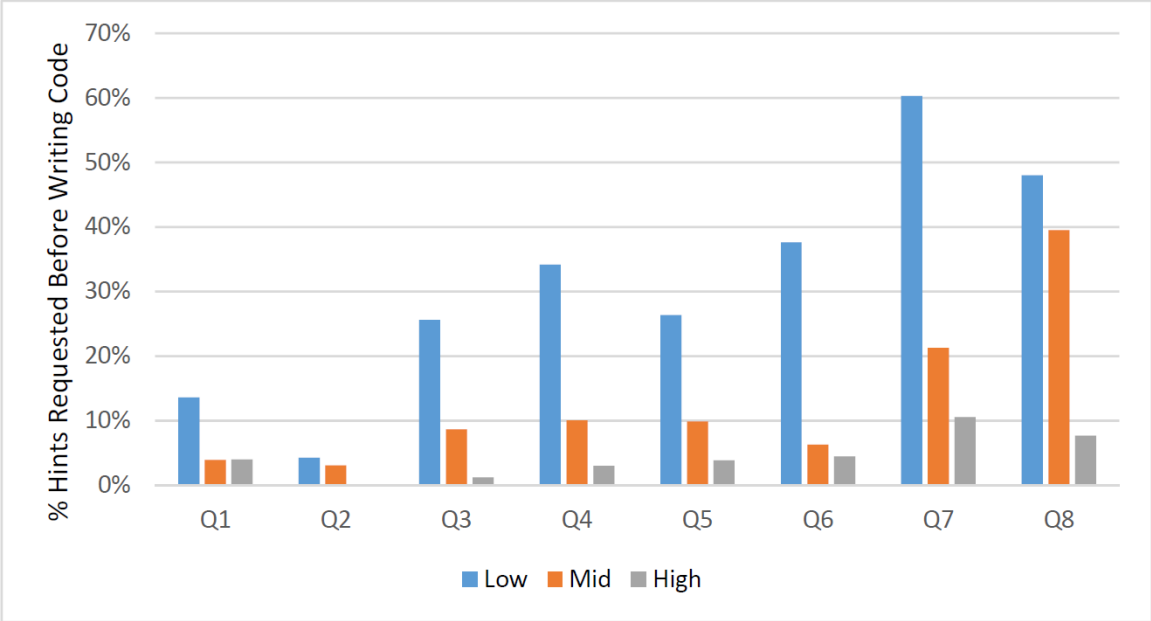


Figure 5.6: The percentage of times hints were requested before writing code within the for-loop module.

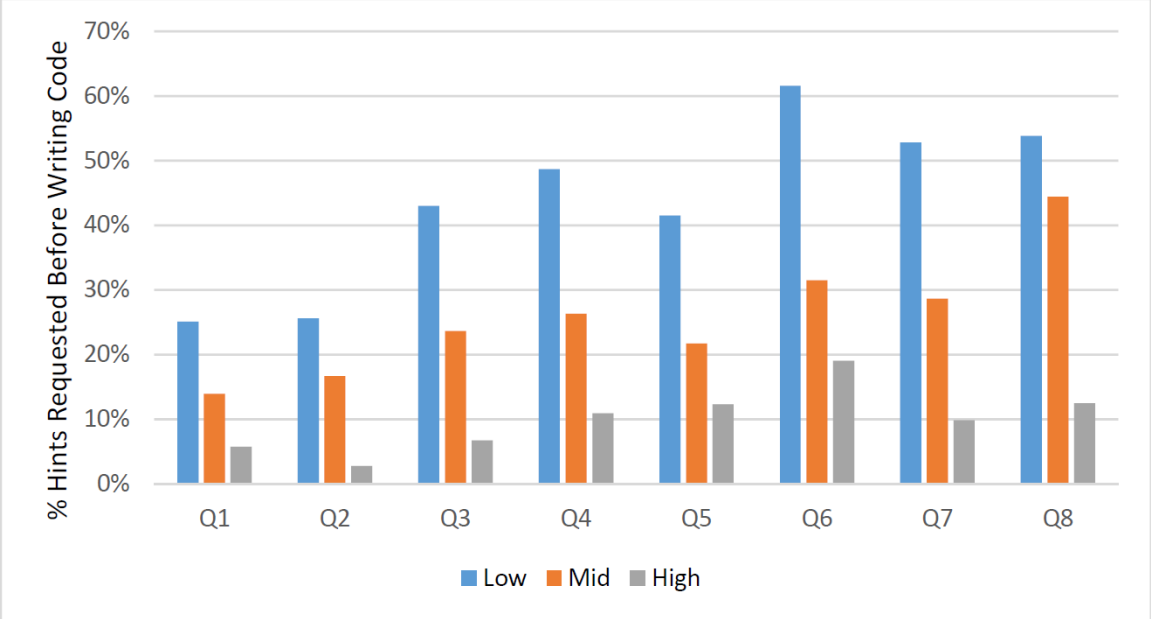


Figure 5.7: The percentage of times hints were requested before writing code across all modules.

Different questions in the module had a different number of total hints that could be requested, so Figure 5.5 alone may not effectively illustrate perceived difficulty. Figure 5.6 illustrates which questions students requested hints on before compiling

any code on their own. Taken together, Figures 5.5 and 5.6 indicate that *perceived* difficultly does not necessarily increase linearly from question to question.

Expanding this analysis to all of the modules, similar trends were found. Figure 5.7 shows the frequency hints were requested before attempting to write and compile code across all modules. Throughout each semester, students in the Low cohort consistently requested hints before attempting to solve problems on their own more often than the Mid and High cohorts.

5.2.3 Self-Efficacy (RQ4c)

To further explore hint usage behaviour, in the most recent semester (S3), students were asked a number of questions about their hint usage. Surveys distributed during the fifth and ninth week of the course. In total, 55 students filled out both surveys, 9 students from the Low cohort, 26 from the Mid cohort, and 20 from the High cohort. The surveys asked students the following 5-point Likert scale questions:

- How difficult did you find the material over the [first/second] month of the semester?
- How often did you use the hints when working through the practice tool?
- In questions you did use hints, do you think in the future you could complete a similar question without hints?

Figure 5.8 shows the results of the survey distributed during the 5th week of the semester. The Low cohort found the course material the most difficult, and reported using hints most often. The survey also shows that students in the Low cohort were also slightly less confident in their ability to complete questions they needed hints on without assistance in the future. The results from this survey align with the quantitative analysis results on tool log data.

Figure 5.9 shows results from the survey distributed during the 9th week of the semester. Figure 5.10 shows the average changes in survey responses between the two surveys. Students from all 3 cohorts reported an increased change in difficulty, but it was students in the Mid cohort who had the highest jump in perceived difficulty between surveys. In terms of how often students felt like they used hints, the Mid cohort also had the biggest increase between surveys, while the High cohort's responses barely changed at all. The Mid cohort had the biggest decrease in confidence in being

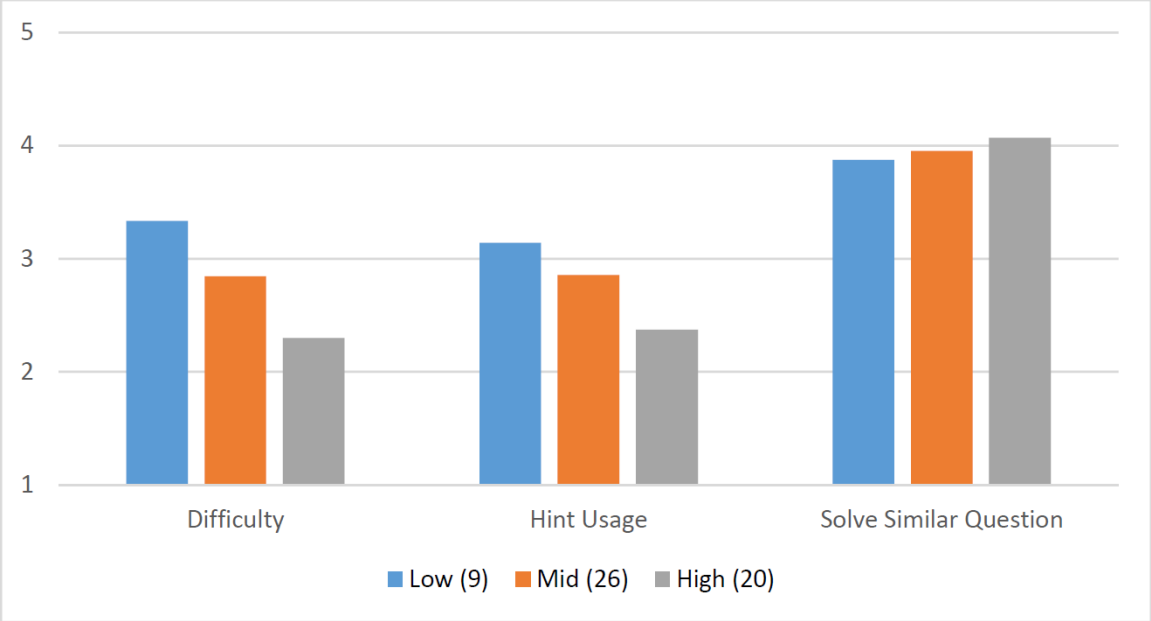


Figure 5.8: Results from the survey distributed during the 5th week of the semester. Questions used a 5-point Likert scale.

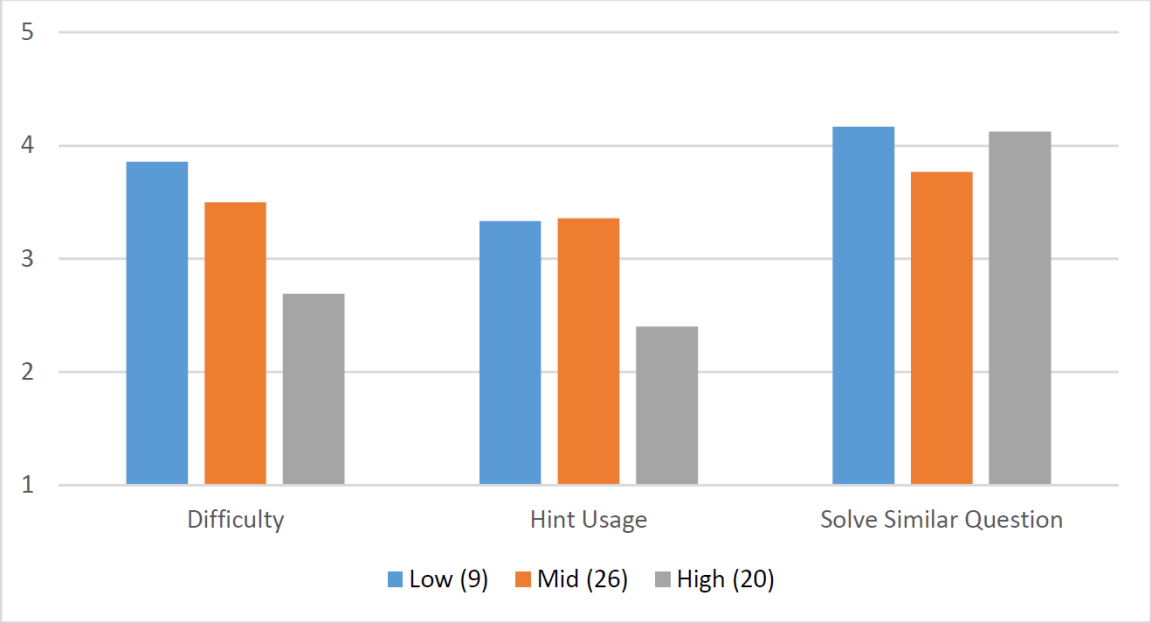


Figure 5.9: Results from the survey distributed during the 9th week of the semester. Questions used a 5-point Likert scale.

able to solve questions they used hints on without assistance in the future, whereas the Low cohort had the largest increase in confidence.

5.2.4 Connecting Survey Results with Log Data

When comparing Figures 5.6 and 5.7, it is apparent that as each semester progressed, students from all 3 cohorts increasingly resorted to viewing hints before attempting to solve the problem on their own. This trend aligns with the survey daya, as students from all three cohorts perceived that the course material increased in difficulty as the semester progressed. Similarly, students from all 3 cohorts felt like they needed to use hints more often as the semester progressed.

The changes in survey responses are different among cohorts for the third question shown in Figure 5.10, *"In questions you did use hints, do you think in the future you could complete a similar question without hints?"* Students in the Low cohort reported an increased confidence in being able to solve questions in the future, whereas students in the Mid cohort decreased in confidence. Figure 5.11 shows how often students were able to answer questions without hints when revisiting questions they previously needed assistance on. Students in the Low cohort only answered questions without hints in follow-up attempts 21.8% of the time, compared to 34.9% for the Mid cohort,

and 50.6% for the High cohort. Taken together, Figures 5.10 and 5.11 illustrate that survey responses from students in the Low cohort seem to most strongly contradict the study behaviour exhibited when working through exercises in the learning tool.

5.3 Analysis and Discussion

In this chapter, I explored whether the combination of interaction data and survey results can be used to differentiate successful from unsuccessful students. The analysis of interaction data collected from the for-loop module, introduced during the second week of each semester, supports the findings of the previous chapters, suggesting that potentially unsuccessful students can be identified within the first few weeks of the semester. This chapter highlights the fact that although these students find the material difficult, students in the Low cohort, who went on to be unsuccessful on the final exam, actually reported an *increase* in confidence with respect to answering questions without hints as the semester progressed. This disconnect between student perception and performance requires further analysis, especially when comparing the change in survey responses between students in the Low cohort and those in the Mid cohort, who were able to pass the final exam. Figure 5.10 suggests that by week 9 students in the Mid cohort may have been able to identify that viewing hints was not an effective study practice, whereas students in the Low cohort were not.

It appears that students in the Low cohort might not understand that their study behaviour is not effective in learning the content in a CS1 course. In some disciplines, flash cards are used when students employ rote learning techniques to study for exams. Students who have had positive experiences in other courses practicing this form of studying may feel like viewing solutions to programming exercises will also enable them to succeed in a computer science exam. It is unfortunate that many students in the Low cohort continued to study using the practice tool, and the survey results suggest believed they were learning effectively, but their efforts did not pay off. In future semesters, I plan to explore these results further; it may be interesting to compare these results with the results found by Robins et al. who reported that the difference between effective and ineffective novices relate to learning strategies rather than knowledge [95]. It would be quite a problem if study strategies turn out to be a significant factor in whether or not novices succeed in a CS1 course, if a large population of novices who enter the course are unaware which strategies are not likely to lead to success.

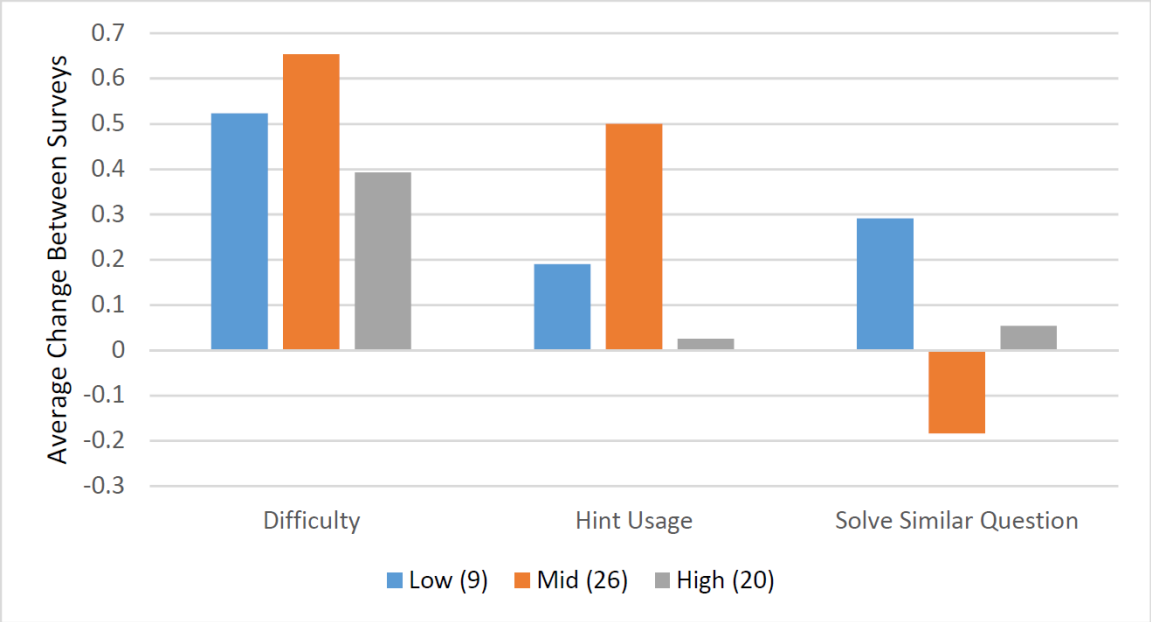


Figure 5.10: Average change in responses for each question between the surveys distributed in week 5 and 9.

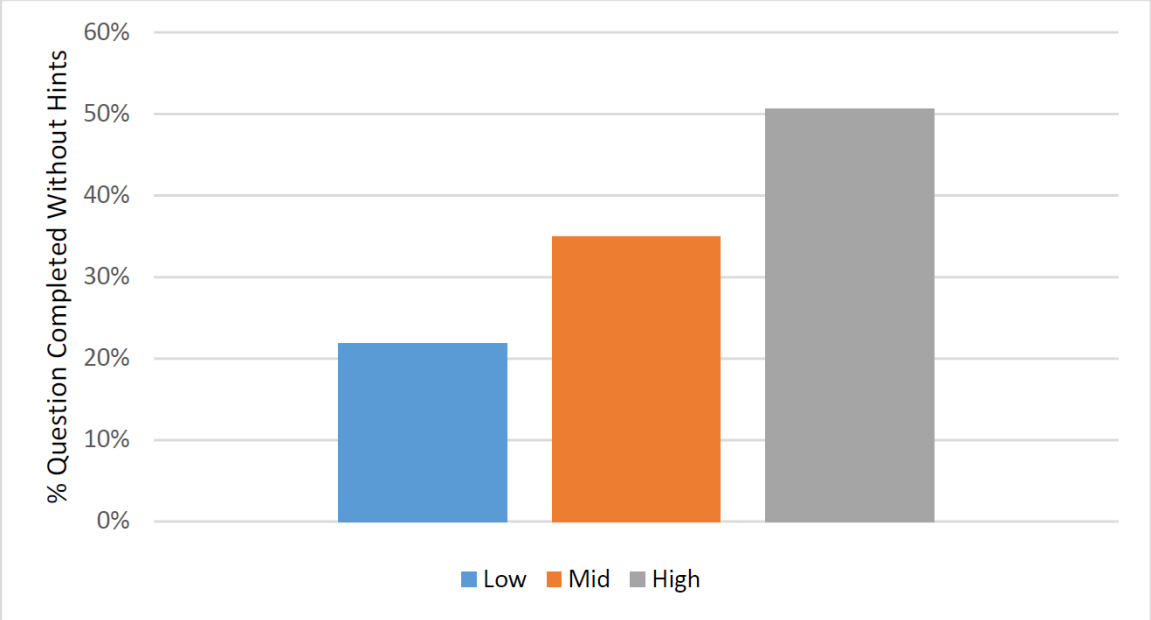


Figure 5.11: The percentage of times students were able to complete a question without hints when revisiting a question they were unable to complete previously.

The results reported in this chapter also highlighted the difference between the intended difficulty and perceived difficulty of exercises. Within a given module, questions were designed to increase in difficulty, but the log data suggests that questions were not necessarily ordered in increasing difficulty. The for-loop module exercises are shown in Figure 5.4. Within this module, Figure 5.6 shows spikes on questions 3 and 7 with respect to how often students requested hints before attempting to solve the problem on their own. Question 3 was the first exercises a nested for-loop was required to solve the problem, whereas question 7 was the first exercises a series of for-loops were required nested inside an outer for-loop. Questions 4, 5, and 6 were intended to increase in difficulty, but each question required the same general design as question 3, but with different variable values. The disconnect between intended and perceived difficulty also requires further analysis, as it may affect the generation of lab work, assignments and exams.

5.3.1 Threats to Validity

It is important to recognize the limitations and threats to validity of the study in this chapter. Students were grouped into cohorts after the semester had ended based on final exam grades, and exam grades may not accurately represent overall student success in the course. Interaction data would be affected for students who studied in groups. Also, the analysis includes only the data from students who both chose to use the practice tool and also consented to take part in the study. Our analysis metrics penalize hint usage, but students may not view hints only when struggling to solve an exercise.

Time on task was evaluated as the difference between when a student loaded a question to when they last interacted with the question, but as the tool is browser-based, the calculated time might not accurately represent the actual time a user spent working on the exercise. Students may exhibit different behaviour when learning a new concept or when using the tool to study the night before an exam, but these types of things were not considered for this study.

5.4 Summary

This study presents a quantitative analysis of practice tool log data and a qualitative analysis of survey responses collected from 465 students over 3 semesters. Students

were split into 3 cohorts, Low, Mid, and High, based on final exam grades.

First, I found that unsuccessful students could not be differentiated from successful students based on time on task data. There was some differences found across cohorts when looking into hint usage; students in the Low cohort requested more hints on average, and were much more likely to request hints before attempting to solve exercises on their own. Students in the Low cohort also reported the biggest positive change between surveys distributed in weeks 5 and 9 regarding their confidence in solving similar questions in the future without hints.

I identified and described several issues that arise when trying to use practice tool log data for analysis. Despite these limitations, the results of this study combining practice tool log data with survey results revealed some interesting findings across the student cohorts, especially with the Low cohort. I plan to further investigate the disconnect between confidence and effectiveness of study behaviour for this cohort who were unsuccessful on the final exam.

Chapter 6

Conclusions and Future Work

This work was motivated by need to support student success in CS1. Previous work suggested that increasing student confidence and motivation could have a profound effect on learning. Other prior work reported that the apathetic environment found in many STEM courses was a reason students leave. As class sizes continue to grow, it may only become more difficult for instructors to form meaningful relationships with students. Automated learning tools have recently been used to provide instructors with a scalable way to gauge student progress, and also provide timely and personalized feedback. The recent studies done on learning tools in CS1 were influential in the design of the learning tool presented in the this work, BitFit.

BitFit is an online, open-source, practice programming tool. It was introduced as a supplemental learning resource for students, and work on BitFit was completely voluntary and ungraded. For students, BitFit was created to provide an environment to practice weekly material to build confidence. For instructors, it included data logging features; I have attempted to better understand student learning and success through an analysis of this data. The end goal is to be able to also provide students with personalized and effective support as early as possible in the semester.

In addition to workflow data being collected, each study also considers qualitative data collected from monthly surveys. It is important to remember that using BitFit was completely voluntary, and survey participation was opt-in, so the population for this work is completely self-selected.

The first study looked into whether students would even choose to use a supplemental practice resource. Survey responses provided information about student perception of BitFit, and the chapter explored the following specific research questions:

RQ1a: Will students use a supplemental, ungraded, practice resource?

RQ1b: What are the reasons students choose not to use a practice tool?

RQ1c: What impact does BitFit have on student confidence and metacognition?

RQ1d: How well does BitFit support students in areas they feel they are struggling?

Throughout the study, over 80% of all *registered* students used BitFit. Students who dropped the course within the first week of the course were included in the group of students who did not use BitFit, showing that in general, students will choose to use a practice programming tool, even if it is not required or graded. The main reasons students did not use BitFit were that they did not know about it (39%), and did not have time (31%). Future work could investigate whether those who did not have time simply do not believe working through exercises is a valuable way to spend study time compared to other study methods. Overall, survey results were very positive with respect to the impact BitFit had on supporting their confidence, meta-cognition, and aiding them in areas they were struggling.

The main suggestion for improvement during early prototypes of BitFit was to improve the hint features to include full code solutions, instead of just high-level guidance. This change ended up being a significant factor in future semesters, as hint usage ended up being a major factor in the results of BitFit log data analysis.

The second study began to analyze BitFit log data, with the key question: *Can interaction patterns with BitFit predict a student's outcome in the course?*. More specifically, the analysis explored the following research questions:

RQ2a: How well do students' levels of engagement with the learning tool predict success in the course?

RQ2b: Can a practice tool identify differences in workflow behavior between successful and unsuccessful students?

RQ2c: What are the metrics associated with success and failure?

RQ2d: How well do metrics identified by the tool predict success?

RQ2e: How early in the semester can at-risk students be identified?

Overall, the number of questions attemped and submitted correctly did correlate with success in the course. In terms of behaviour and metrics for success, low compilation numbers combined with high, repeated hint usage differentiated top students from those who struggled. These metrics can be used to begin identifying students within just the first few weeks of the semester. The results also suggest that as a semester progresses, these metrics can be used to much more accurately differentiate between students, as the differences between high and low-performing students become increasingly pronounced.

When separating out only those students identified as at-risk early in the semester, there was a lot of the noise in the data, which made it difficult to differentiate between those truly at risk and those who went on to succeed in the course. The third study focused on understanding what learning looks like, by improving the early at-risk metric to include a metric that accounts for changes in behaviour over time. The goal was to more accurately identify and differentiate between students who are truly struggling and those who are learning. The key question in the third study was: *Can analysis of patterns in interaction data help us understand how to detect and measure learning in CS1?*. In order to answer this question, the following specific research questions were investigated:

RQ3a: How accurately do early predictors of performance identify students who are unsuccessful in the course?

RQ3b: How well can early predictors be improved by a trajectory metric that classifies behavioural change over time?

RQ3c: How well can the trajectory metric be extended to evaluate differences in proficiency across topics?

The early predictor metric found in the second study was the combination of low compilation numbers and high, repeated hint usage. This metric was able to identify a high percentage of students who went on to fail the course, and did correlate with overall success in the course, but this metric also produced a high number of false-positives. The trajectory metric introduced in the third study was able to filter out most of the false-positives from this group by looking at how interaction patterns with BitFit change over time. The trajectory metric also revealed that students who exhibit productive programming behaviour in one topic are not necessarily productive in later topics. The results from this study highlight the importance of combining

metrics that analyze changes in behaviour or productivity over time with baseline early predictors.

To better understand why students exhibit certain study behaviours, and students perception of their own study habits, I next explored whether there were any interesting connections between BitFit log data and student feedback received in the monthly surveys. The key question for the next study was: *Do students understand whether or not their study habits are likely to lead to success on the final exam?* The following specific questions were explored to answer this question:

RQ4a: How well does time-on-task differentiate between successful and unsuccessful students?

RQ4b: How well does intended question difficulty differentiate between successful and unsuccessful students?

RQ4c: Is there a difference in a student's reflection of self-efficacy between successful and unsuccessful students?

In terms of time on task, I found that unsuccessful students could not be differentiated from successful students. For question difficulty, there were some hint usage differences found across cohorts, specifically with respect to how often students requested hints before attempting to solve the problem on their own. Survey responses provided some insight into this issue, as their were some interesting student responses to the question about how confident students were in being able to solve a similar question on their own in the future after requesting hints. Students in the Low cohort reported the biggest positive change between surveys distributed in weeks 5 and 9 on this question. These students progressively relied on hints as the semester progressed, and became *more* confident that the hints were helping them learn effectively. Unfortunately, their efforts did not pay off on the final exam. This disconnect between student perception of progress and exam performance is something that requires further investigation in the future.

There are also a number of other important open questions related to the core research question introduced in this work. In addition to identifying interaction patterns associated with success and failure in CS1, further work should aim to determine why certain students exhibit unsuccessful patterns. Based on student feedback from surveys, it appears that some students may not realize their methods of study are in-

effective. This raises the first future research question: *What are the reasons students resort to ineffective study behaviour?*

Beyond this question, an important question would be how to best guide struggling students toward habits that better enable success. If we can better understand why our students are struggling, and how to support them, tools like BitFit can be used to identify students at-risk very early in the semester. This would allow instructors to effectively provide supportive intervention to students exhibiting subtle patterns in workflow behavior that have previously been established as having a very low probability of success. Thus, the second future research question is: *What is the best method to guide struggling students towards more effective study behaviour?*

The group of students the trajectory metric filtered out, those who early in the semester exhibited behaviour associated with students who typically struggle and fail with the material, are an interesting group to further investigate. These students were able to transition to more productive behaviour without any external support provided by the teaching team. If learning can be defined as an enduring change in behaviour, then it can be argued that these students were able to learn effectively, and the collected BitFit log data "recorded" the process. A deeper investigation of data from this specific student group may provide valuable insight into the patterns associated with learning, and this information could be very important when designing supportive intervention targeted at those who are unable to progress without assistance. The third future research question is: *Can a student support model be formed based on the data collected on learning throughout these studies?*

Further data analysis and modeling is necessary to answer each of the future research questions. Now that data has been collected over a large number of semesters, it may be possible to begin investigating whether a number of prediction models can be formed. Using predictive modeling, a number of patterns may be found allowing for dynamic classification of students in the future. This would allow a number of intervention techniques to be deployed that are customized to different types of learners. Thus, the major key research question for the future is: *How well can predictive models be used to better understand and classify students?*

Overall, what I have learned from the studies presented in this work is that learning tools can provide students with an environment to build confidence, and also provide instructors with vital information on student progress. Based on the results of these studies, I found that early identification metrics are able to identify many at-risk students, but that continually measuring changes in behaviour over time is equally

important in order to differentiate between transient and sustained struggling. Finally, a mixed-method approach combining qualitative and quantitative data might provide the insight necessary to answer questions that cannot be answered by log data alone. The combination of early and trajectory metrics based on log data with student surveys provides the information necessary to begin understanding where, when, and how are students are struggling. I hope as the computer science education community continues to strive to better understand the student learning process, our efforts to develop effective intervention strategies to support students will drastically improve. I hope that such efforts will result in a more diverse group of confident, empowered programmers, and better success rates in computer science programs overall.

Appendix A

Additional Information

A.1 Topics

BitFit included questions in the following topic areas:

1. Print statements
2. For-loops - code reading
3. Syntax errors
4. Strings and casting
5. For-loops - code writing
6. Methods - code reading
7. if-statements
8. Writing code - methods and for-loops
9. IO code reading and writing
10. Arrays
11. Classes and Objects
12. Combined: Weeks 1 - 4
13. Combined: Weeks 5 - 9

14. Combined: Weeks 10 - 13

15. Final Review

Questions in "Combined: Weeks 1 - 4" included questions on print statements, syntax errors, casting, for-loops, and methods. Questions in "Combined: Weeks 5 - 9" included questions on arrays (including multi-dimensional arrays), and IO (reading from and writing to files). Questions in "Combined: Weeks 10 - 13" included questions on classes and objects, search algorithms, and recursion. The Final Review section provided students with additional questions found on previous final exams.

A.1.1 Sample Questions

This section provides a number of sample questions from each of the included topics. Each question in the print statement topic asked the user what the output was of a pre-written Java program. Figure A.1 shows a sample question with BitFit's embedded editor and a few of the buttons.

The remaining Figures show a number of sample questions from each of the topic areas found in BitFit. For code reading questions, only the starter code is shown, as the question text just asks the user to enter the code output into the output box, shown to the right of the embedded editor in Figure A.1.

For sample code writing questions, only the question text is shown, as in most cases, there is very little to no code in the embedded editor, as the user must *write* code in the editor to solve the problem.

A few sample questions from each topic area are shown, usually in the order they are presented in. Questions were designed to increase in difficulty.

A.2 Sample Background Info

Each topic area has a list of associated questions, as shown in the previous subsection. There is also a tab students can click to access background information about the topic area. A sample Background page, the one in the "If-statements" section, is shown in Figure A.16.

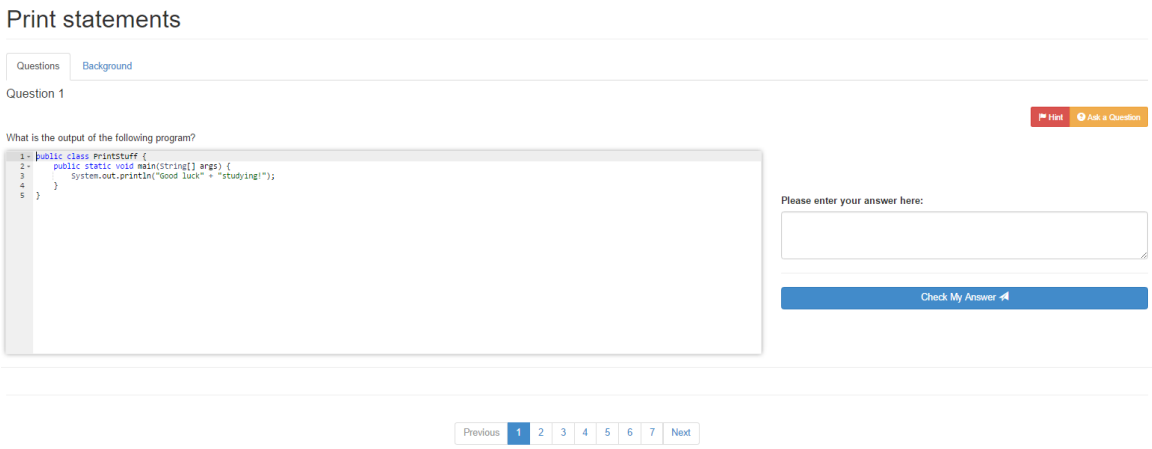


Figure A.1: Sample questions from the Print Statements topic area.

Starter Code:

```
public class PrintStuff {
    public static void main(String[] args) {
        String name = "Turing";
        System.out.println("Hello, your " + name + " is: name");
    }
}
```

Starter Code:

```
public class PrintStuff {
    public static void main(String args[]) {
        System.out.println("Hello\nCSC110! Hope\nyou\\like\\this\\tool");
    }
}
```

Starter Code:

```
public class PrintStuff {
    public static void main(String args[]) {
        System.out.println("/---\\\n|xXx|\n\\---/");
    }
}
```

Starter Code:

```
public class PrintStuff {
    public static void main(String args[]) {
        System.out.println("1" + 2 + 3 + "4" + 5 * 6 + "7" + (8+9));
    }
}
```

Starter Code:

```
public class PrintStuff {
    public static void main(String args[]) {
        System.out.println(1 + 2 + 3 + "4" + 5 * 6 + "7" + (8+9));
    }
}
```

Starter Code:

```
public class PrintStuff {
    public static void main(String args[]) {
        System.out.println("48 / 1" + 2 +  " * 3" + 15 / 3 + " is " + 1 + 4 + 8 * 0);
    }
}
```

Figure A.2: Sample questions from the Print Statements topic area.

Starter Code:

```
public class ForLoops {
    public static void main(String[] args) {
        for (int i = 1; i < 1000; i *= 2) {
            System.out.println(i);
        }
    }
}
```

Starter Code:

```
public class ForLoops {
    public static void main(String[] args) {
        int whatIsThis = 7;
        for (int i = 25; i%whatIsThis > 0; i--) {
            System.out.println(i);
        }
    }
}
```

Starter Code:

```
public class ForLoops {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i += 3) {
            for (int j = i; j <= 10; j += 5) {
                System.out.print(j + ", ");
            }
            System.out.println(i);
        }
    }
}
```

Starter Code:

```
public class ForLoops {
    public static void main(String[] args) {
        for (int i = 8; i > 0; i -= 2) {
            for (int j = 0; j <= i%3; j++) {
                System.out.print("duck, ");
            }
            System.out.println("goose!");
        }
    }
}
```

Figure A.3: Sample questions from the For-loops - code reading topic area.

Starter Code:

```
public class ErrorCode {
    public static void main(String[] args) {
        System.out.println("Hello!")
    }
}
```

Starter Code:

```
public class ErrorCode {
    public static void main(String[] args) {
        int num = 9.87;
        double num2 = 3;
        System.out.println("Why doesn't my number work?");
    }
}
```

Starter Code:

```
public class ErrorCode {
    public static void main(String[] args) {
        int num = 5;
        System.out.println("My number is "+ num);
        int num = 2;
        System.out.println("My number is now "+ num);
    }
}
```

Fix the syntax errors in the code, output should be:

```
Number started at 5
Number doubled to 10
Number doubled to 10
```

Starter Code:

```
public class ErrorCode {
    public static void main(String[] args) {
        int num = 5;
        System.out.println("Number started at "+num);
        grow();
        int biggest = num * 2;
        System.out.println("Number doubled to "+biggest);
    }

    public static void grow() {
        int bigger = num * 2;
        System.out.println("Number doubled to "+bigger);
    }
}
```

Figure A.4: Sample questions from the Syntax errors topic area.

Starter Code:

```
public class StringPractice {

    public static void main(String[] args) {
        String course = "Fundamentals of Programming";
        String students = "us!";
        String location = "Victoria";
        String vowels = "aeiou";

        String secret = "QZXI " + course.substring(4,19);
        System.out.println(secret);

        String mystery = secret.substring(3, 7) + " " + location.substring(0,6);
        String solution = mystery + vowels.substring(2,4) + students;

        System.out.println(solution);

    }

}
```

Starter Code:

```
public class StringPractice {

    public static void main(String[] args) {

        System.out.println(3*1.5);
        System.out.println((double)3*1.5);
        System.out.println((int)3*1.5);
        System.out.println((int)(3*1.5));
        System.out.println(3*(int)1.5);

    }

}
```

Figure A.5: Sample questions from the Strings and casting topic area.

Figure A.6: Sample questions from the For-loops - code writing topic area.

Starter Code:

```
public class Methods {
    public static void main(String args[]) {
        System.out.println("Start!");
        pause();
        go();
        stop();
        pause();
        System.out.println("Done!");
    }

    public static void pause() {
        System.out.println("Pause");
        stop();
        System.out.println("UnPause");
        go();
    }

    public static void stop() {
        System.out.println("Stop!");
    }

    public static void go() {
        System.out.println("Go!");
    }
}
```

Starter Code:

```
public class Methods {
    public static void main(String args[]) {
        int n = 5;
        int x = 2;
        int y = 7;
        int i = 4;

        doSomeMath(x, x, y);
        doSomeMath(7*3, n, i-y);
    }

    public static void doSomeMath(int z, int x, int y) {
        System.out.println(z + " plus " + y + " is " + z);
    }
}
```

Figure A.7: Sample questions from the Methods - code reading topic area.

Starter Code:

```
public class IfStatements {
    public static void main(String[] args) {
        int twoDiceRoll = 9;
        if (twoDiceRoll > 9) {
            System.out.println("Wow, that's high!");
        } else if (twoDiceRoll > 4) {
            System.out.println("Pretty average roll");
        } else {
            System.out.println("I hope you were trying to roll low");
        }
    }
}
```

Starter Code:

```
public class IfStatements {
    public static void main(String[] args) {
        int diceRoll = 4;
        if (diceRoll == 6) {
            System.out.println("Perfect!");
        } else if (diceRoll >= 4) {
            System.out.println("Pretty good");
        } else if (diceRoll >= 2) {
            System.out.println("Below average");
        }
        if (diceRoll == 1) {
            System.out.println("Ouch, a 1");
        } else {
            System.out.println("Are you sure you rolled?");
        }
    }
}
```

Starter Code:

```
public class IfStatements {
    public static void main(String[] args) {
        int diceRoll = 1;
        if (diceRoll == 6 || diceRoll == 1) {
            System.out.println("Extreme roll!");
        } else {
            System.out.println("Meh");
        }
    }
}
```

Figure A.8: Sample questions from the if-statements topic area.

Write a method printOdds, that takes in an int parameter, and prints out all of the odd numbers from 1 up to that number.

```
public class CodeWriting {
    public static void main(String[] args) {
        printOdds(7); // should print 1 3 5 7
        printOdds(12); // should print 1 3 5 7 9 11

        /* total output should be:
        1 3 5 7
        1 3 5 7 9 11
        */
    }

    //write your method here

}
```

Write a method drawRect, that takes in 2 int parameters, and prints out a rectangle of *'s based on the parameters.

So drawRect(4, 3) would draw a rectangle with a width of 4 and a height of 3, like so:

```
****
****
****
```

Write a method printPowersOf2 that takes in an int parameter, and prints out that many of the first powers of 2. So printPowersOf2(5) would print 1 2 4 8 16

You are allowed to use the Math library, methods can be found here: http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

(Note: The expected output is printing out ints (so no decimals), so if you use the Math.pow method from the Math class, you will need to do some casting).

Write a method called countQuarters that takes in an int parameter, and prints out how many quarters would be given out as change.

So, countQuarters(64) would print out "quarters: 2" (and you don't have to worry about the last 14 cents).

countQuarters(1278) should print out "quarters: 3". (the 12 dollars would be handled by bigger forms of currency, leaving just 78 cents, would would be 3 **quarters**, and 3 pennies)

Write a method called **printBackwards** that accepts a String as a parameter. The method will print out each of the letters in the String out backwards.

For example, printBackwards("Anthony") should result in the following output::

```
The word passed in is: Anthony
The word backwards is: ynohtnA
```

Write a method called printMultiples that accepts two integers as parameters. The method will print out multiples of the first parameter. The second parameter represents the number of multiples to print out.

For example, printMultiples(4, 5) prints out the first 5 multiples of 4 (4, 8, 12, 16, and 20). The output should be:

```
4 8 12 16 20
```

Figure A.9: Sample questions from the Writing code - methods and for-loops topic area.

Assume you have an input file containing the following data:

```
4 6 3 11
 2   9       16
8          12
7   1    5
```

Write a method called printAverage, that takes in a Scanner as a parameter, and prints out the average value of all of the numbers in the file.

So for the above file, the output would be:

Average value: 7

Assume you have an input file containing the following data:

```
the cat and the dog shared the same house but the
house was not big enough to accommodate both the cat
and the dog so the dog moved to a new house
```

Write a method called `findBiggest`, that takes in a `Scanner` as a parameter, and prints out the longest word found in the input file.

So for the above file, the output would be:

Longest word is: accommodate

Assume you have an input file containing the following data:

```
H T H H h
T t    t t h H
   h H t   h
```

Write a method called `printResults`, that takes in a `Scanner` as a parameter, and prints the number of heads flipped (which can be h or H), the fraction of heads flipped divided by total flips, and then prints "You win!" if there were at least 50% heads flipped, and "You Lose!" otherwise.

So for the above file, the output would be:

Number of heads: 9
Wins: 9/15
You win!

Assume you have an input file containing the following data:

```
H T H H h
T t    t t h H
   h H t   h
```

Write a method called `printLineResults`, that takes in a `Scanner` as a parameter, and **for each line in the file**, prints out the line number and the the fraction of heads flipped divided by total flips on that line of the input file.

So for the above file, the output would be:

Fraction of heads on line 1: 4/5
Fraction of heads on line 2: 2/6
Fraction of heads on line 3: 3/4

Figure A.10: Sample questions from the IO code reading and writing topic area.

What a method **findMin**, which takes an `int` array as a parameter, and returns the smallest value in the array.

```
public class Arrays {
    public static void main(String[] args) {
        int[] nums1 = {8, 14, 6, 9};
        int[] nums2 = {2, -5, 0, 4};

        int smallestFound = findMin(nums1);
        System.out.println("Smallest value found in nums1: "+smallestFound);
        smallestFound = findMin(nums2);
        System.out.println("Smallest value found in nums2: "+smallestFound);

    }

    //write your method here

}
```

Write a method **mostCommon** that takes in an `int` array, and returns the most common number in the array.

Write a method **contains**, that takes two int arrays as parameters. The first parameter should be a large array, and the second parameter should be a smaller array. The contains method should return what index the values in the smaller array can be found in the bigger array.

For example, if we have arrays:

a -> [1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1]

b -> [3 | 4]

c -> [4 | 5 | 4]

d -> [3 | 4 | 5 | 6]

contains(a, b) should return 2, as index 2 is where the values in array b can be found in array a.

contains(a, c) should return 3, as index 3 is where the values in array c can be found in array a.

contains(a, d) should return -1, as the pattern 3, 4, 5, 6 cannot be found in array a.

contains(d, b) should return 0, as index 0 is where the values in array b can be found in array d.

Write a method addToValues, that takes a 2d-array of integers as a parameter. The addToValues method should add 1 to all of the values found in the 2d-array.

For example, if we have the array:

[7 | 2 | 1 | 8]
[0 | 4 | 3 | 4]
[6 | 1 | 4 | 3]

the resulting array should be:

[8 | 3 | 2 | 9]
[1 | 5 | 4 | 5]
[7 | 2 | 5 | 4]

Figure A.11: Sample questions from the Arrays topic area.

This question uses the Date class as specified at the bottom of the code.

Write a static method named addDays, which accepts an integer and a date as parameters and shifts the date represented by the Date object forward by that many days. You may assume the integer value passed is non-negative. For example, consider we made the following Date d:

Date d = new Date(9, 19);

The following calls to the addDays method would modify the Date object's state as indicated in the comments. Remember that Date objects do not store the year; the date after December 31st is January 1st.

addDays(1, d); // d now represents Sep. 20

addDays(1, d); // d now represents Sep. 21

addDays(5, d); // d now represents Sep. 26

addDays(10, d); // d now represents Oct. 6

addDays(80, d); // d now represents Dec. 25

addDays(10, d); // d now represents Jan. 4 (of the next year)

addDays(1000, d); // d now represents Oct. 1 over 2 years later.

This question uses the Date class as specified at the bottom of the code.

Write a method named daysTillXmas that will be written **inside** the Date class to become part of each Date object's behaviour. The daysTillXmas method returns how many days the current Date is from Christmas, which is Month: 12, Day: 25 (December 25th) of **the same year**. For example:

Month: 12, Day: 12 (December 12) is 13 days away from Christmas.

Month: 11, Day: 22 (November 22) is 33 days away (8 remaining days left in month 11 (November) and then 25 days in month 12 (December)

Month: 9, Day: 3 (September 3) is 113 days away (27 days left in month 9 (September), 61 days in month 10 and 11 (October and November), and 25 days in month 12 (December).

Month: 12, Day: 25 is 0 days away

Month: 12, Day: 28 is -3 days away

Month: 1, Day: 1 is 358 days away.

Figure A.12: Sample questions from the Classes and Objects topic area.

What is the output of the following code?

```
public class FinalPrep {
    public static void main(String[] args) {
        int magicNumber = 33;
        int base = 3;
        while (magicNumber%15 != 0) {
            System.out.println(base*magicNumber);
            magicNumber--;
            base += 2;
        }
    }
}
```

Write a method printSpaces, that takes as a parameter one int, and prints out the digits in that integer with a space in between them. For example printSpaces(7243) should output 7 2 4 3

You are **not** allowed to convert the integer to a String.

Write a method hasMidPoint, that takes in 3 integers, and returns true if one of those integers is the midpoint between the other 2 (but the mid-point can be any of the 3 values). For example:

hasMidPoint(4, 6, 8), hasMidPoint(2, 10, 6), hasMidPoint(8,8,8), and hasMidPoint(25, 10, -5) would all return true.

hasMidPoint(3, 1, 3), hasMidPoint(1, 3, 1), hasMidPoint(21, 9, 58), and hasMidPoint(2, 8, 16) would all return false.

Write a method called **printFactors** that takes in a single int as a parameter, and prints out all factors of that number.

For example, **printFactors(24)** should output "**1 2 3 4 6 8 12 24** "

On the other hand, **printFactors(8)** would output "**1 2 4 8** "

Figure A.13: Sample questions from the Weeks 1 - 4 review topic area.

Write a method called insert that inserts a String into an array of Strings at a specified position. The method takes 3 parameters: an array of strings, an int for where to insert the new String, and the String to insert. The method should return a new String array containing the original values with the new String inserted into the correct location (and all Strings pushed back one spot after that index, so the new array will always have a length 1 longer than the original array). For example,

String[] depts = {"CSC", "PSYCH", "MATH", "SOCI"};

insert(depts, 1, "HIST"), should return an array containing ["CSC", "HIST", "PSYCH", "MATH", "SOCI"]

Write a static method named mode that takes an array of integers as a parameter and that returns an integer representing the value that occurs most frequently in the array. Assume that the integers in the array appear in sorted order. For example, if a variable called list stores the following values:

int[] list = {-3, 1, 4, 4, 4, 6, 7, 8, 8, 8, 8, 9, 11, 11, 11, 12, 14, 14};

Then the call of mode(list) should return 8 because 8 is the most frequently occurring value in the array, appearing four times. If two or more values tie for the most occurrences, return the one with the lowest value. For example, if the array stores the following values, the call of mode(list) should return 2 despite the fact that there are also three 9s:

int[] list = {1, 2, 2, 2, 5, 7, 9, 9, 9};

If the array's elements are unique, every value occurs exactly once, so the first element value should be returned. You may assume that the array's length is at least 1. If the array contains only one element, that element's value is considered the mode.

Write a full Java program called FinalPrep. Your program should use a Scanner to read a file called numbers.txt, and print out how many numbers are contained in the file, the number with the smallest value, the number with the largest value, and the average value of all numbers.

Example:

If we had a file with the following numbers in it: `4 6 3 11 2 9 16 8 12 7 1 5`

The output should be:

```
Total numbers: 12
Minimum value: 1
Maximum value: 16
Average value: 7
```

You are given a 2D array of ints that give the height of the terrain at different points in a square property. Write a method floodMap, that takes as parameters the 2D array of ints, and an int representing the current water height. floodMap should print out a `*` for each flooded point (so where the value in the array in the array is less than or equal to the water level value), and a space for each point that is not flooded. For example,

```
int height = 3;
```

```
int[][] arr =
[1 5 2 4 3]
[6 4 2 3 3]
[1 1 1 4 1]
[5 6 2 1 4]
```

would output:

```
* * *
  ***
*** *
  **
```

Figure A.14: Sample questions from the Weeks 5 - 9 review topic area.

Within the Car class, write a method **drive** that takes in an int as a parameter and increases the kms instance variable by the amount driven. Your method must also reset the kms value when the amount gets to 1000. For example, if a car has a kms value of 998 and it drives 5km, then its new value would be 3.

Write a method findMinIndex, that takes as a parameter an array of Car objects, and returns the index containing the car with the lowest kms instance variable.

Write a recursive method ***sum*** that takes two integer parameters and returns the sum of all values between the two parameters. The sum method **must not contain** any **for-loops** or **while-loops**. You can assume the value of the second parameter is always equal to or greater than the value of the parameter. For example:

sum(4,4); returns 4;

sum(2,5); returns 2 + 3 + 4 + 5 = 14

sum(11,13); returns 11 + 12 + 13 = 36

sum(103, 114); returns 103 + 104 + 105 + 106 + 107 + 108 + 109 + 110 + 111 + 112 + 113 + 114 = 1302

What is the output of the following code? The original array of integers is printed, and is then printed again after each pass of the bubble-sort algorithm.

Given the following Time class, add an instance method **minutesUntil** that accepts another Time objects as a parameter, and returns the number of minutes that must pass until reaching exact time of the passed in Time object parameter.

Figure A.15: Sample questions from the Weeks 10 - 13 review topic area.

If-statements

Questions | Background

If statements are kind of like the middle part of a for-loop, or a while-loop that only executes once. There is no initialization or increment, just the condition check, and if the condition passes (is true), then the code in the if-statement is executed.

The rule of thumb for if-statements is that you can have 1 if, followed by 0 to any number of else ifs, followed by 0 or 1 else.

Structure:

```
if ( condition ) {
   //statements here
} else if ( check a different condition) {
   //other statements here
} else {
   //even more statements here
}
```

Example:

```
int num = 6;
if ( num > 10 ) {
   System.out.println("The number is greater than 10");
} else if (num < 0) {
   System.out.println("The number is less than 0");
} else {
   System.out.println("The number is between 0 and 10.");
}
```

Output:

The number is between 0 and 10.

If any parts of the if-statement are entered (because the condition is true), **no other conditions will even be checked**.

For example:

1)

```
int num = 10;
if ( num > 8 ) {
   System.out.println("The number is greater than 8");
} else if (num > 6) {
   System.out.println("The number is greater than 6");
} else {
   System.out.println("The number is 5 or less");
}
```

Figure A.16: A sample background information page (if-statements in this case).

A.3 Sample Hints

When the hint button is clicked, a new hint is shown above the embedded editor. Hints provided contain both high-level guidance and code snippets. The final hint provides a full code solution to the problem for code writing questions, and the full output for code reading questions. Figure A.17 shows sample hints for a code reading question, and Figure A.18 shows the first few hints for a code writing question.

A.4 Sample survey

CSC110 mid-semester check-in

A survey to collect information from 110 students to be used in research by Anthony Estey to improve the learning experience for the current and future semesters of CSC110 at UVic.

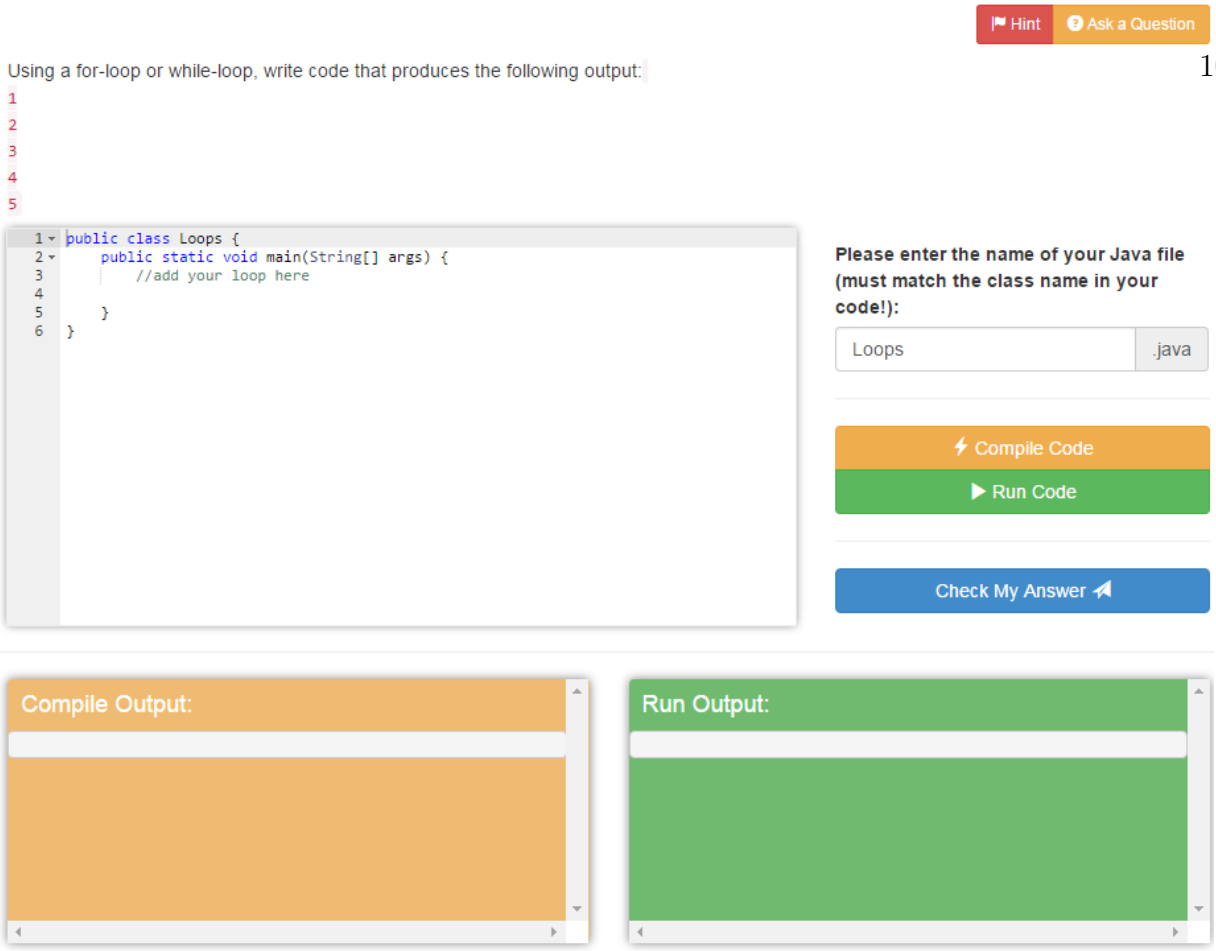
This will not affect your course grades in ANY way. If you have any questions please email Anthony at aestey@uvic.ca

*Required

1. **How difficult did you find the material over the first month of the semester (for-loops, methods, etc)?**
   Mark only one oval.
   - ◯ Very difficult
   - ◯ Difficult
   - ◯ Medium
   - ◯ Easy
   - ◯ Very Easy

2. **Do you think your midterm mark accurately represents how well you know the material?**
   Mark only one oval.
   - ◯ My midterm grade was much lower than how much I know
   - ◯ My midterm grade was a little lower than how much I know
   - ◯ My midterm grade was accurate
   - ◯ My midterm grade was a little higher than how much I know
   - ◯ My midterm grade was much higher than how much I know

3. **Did you use the practice tool (shown below) before the first midterm?** *
   Mark only one oval.
   - ◯ No *Skip to question 10.*
   - ◯ Yes *Skip to question 4.*



Learning tool questions

4. **Did the practice tool help you identify how well you understand the material for each topic?**
   Mark only one oval.

   - ◯ Not at all
   - ◯ Not really
   - ◯ Neutral
   - ◯ A bit
   - ◯ A lot

5. **Did the practice tool affect your confidence in the material?**
   Mark only one oval.

   - ◯ Way less confident
   - ◯ Less confident
   - ◯ No change
   - ◯ More confident
   - ◯ Way more confident

6. **Do you think using the practice tool affected your midterm grade at all?**
   Mark only one oval.

   - Not at all
   - Not really
   - Neutral
   - A bit
   - A lot

7. **How often did you use the hints in BitFit?**
   Mark only one oval.

   - Did not know they existed
   - Never
   - Rarely
   - Sometimes
   - Often
   - Always

8. **In questions you did use hints, do you think after viewing the hints, you could complete a similar question without hints?**
   Mark only one oval.

   - N/A
   - No chance
   - Unlikely
   - Maybe
   - Likely
   - Definitely

9. **How could the learning tool be improved to better help you succeed in this course?**

Skip to question 12.

10. **Why did you choose not to use the learning tool?**
Tick all that apply.

- [ ] I did not know about it
- [ ] I did not have time
- [ ] I did not need any extra help in this course
- [ ] The material on the learning tool was too easy
- [ ] The material on the learning tool was too hard
- [ ] Other: .....

11. **Would you be open to using a learning tool to study for future exams?**
Mark only one oval.

- ◯ No
- ◯ Yes

12. **Please list any other suggestions you have about how the teaching team could improve your learning experience in this course?**

.....

.....

.....

.....

13. **What is your connex id?**

.....

14. **May we contact you by email or over connex for further questions (if we need more information about your suggestions or another answer from this survey)**
Mark only one oval.

- ◯ No
- ◯ Yes

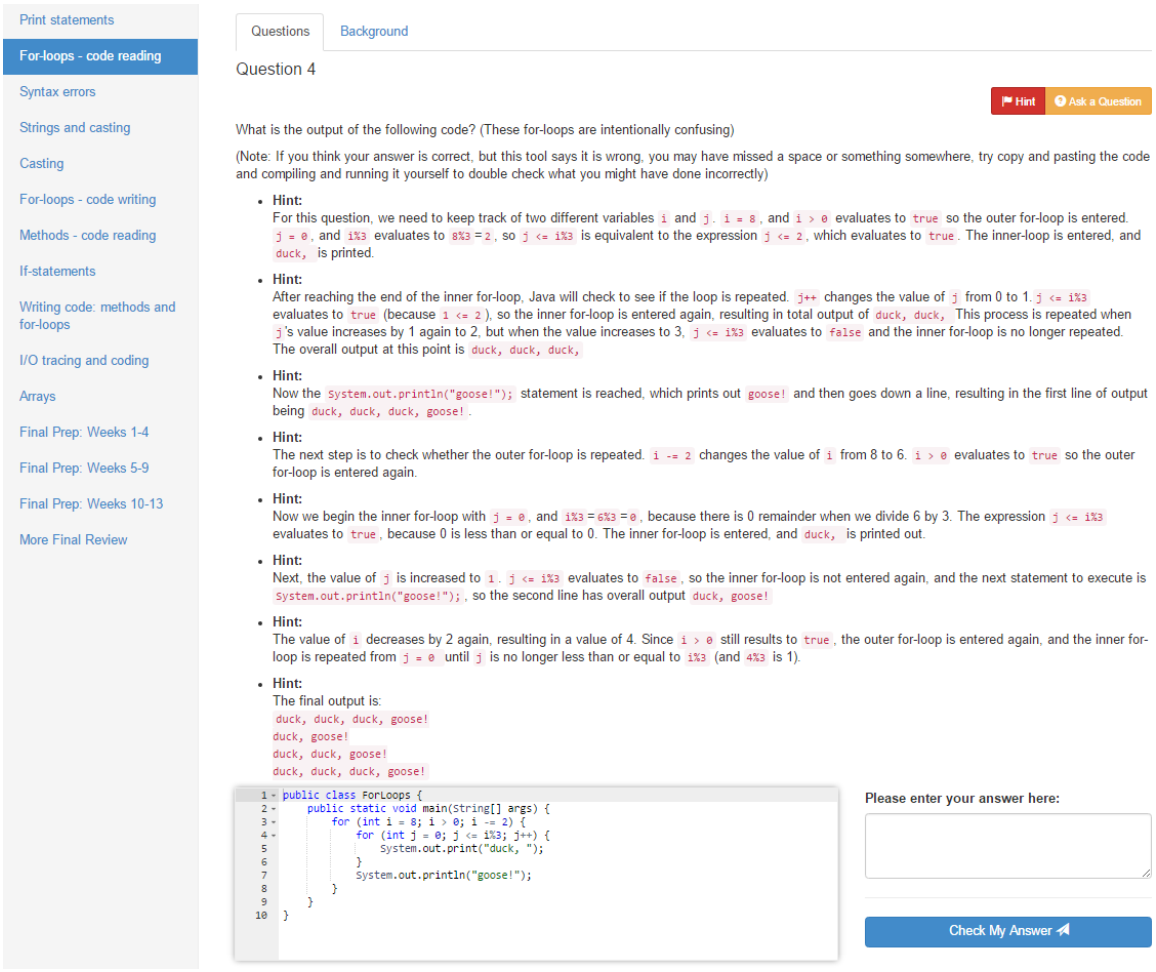


Figure A.17: A sample hint for a code reading question.

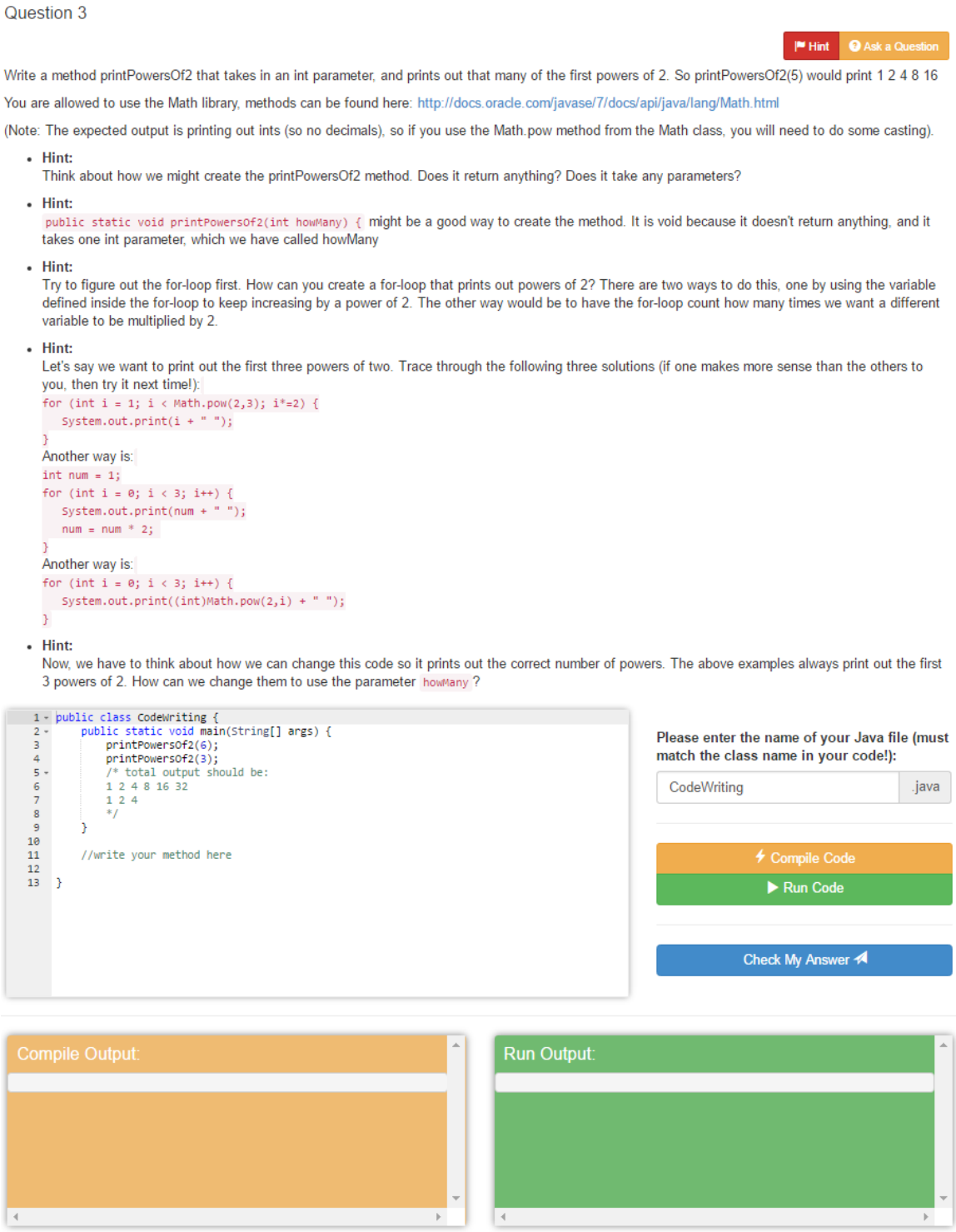


Figure A.18: A sample hint for a code writing question.

A.5 Sample log data

[{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f02
7541070e9415a3f","startTime":"2015-09-
07T18:54:47.213Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd79bc7ced0863fe5539","__v":0,"endTime":"2015-09-
07T18:54:50.258Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dcdfc704e78631e67f72e7","startTime":"2015-09-
07T18:54:50.260Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd7fbc7ced0863fe553a","__v":0,"endTime":"2015-09-
07T18:54:56.975Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dce00404e78631e67f72e8","startTime":"2015-09-
07T18:54:56.975Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd82bc7ced0863fe553b","__v":0,"endTime":"2015-09-
07T18:54:59.807Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd0fde4427a32e0b5e699f","startTime":"2015-09-
07T18:54:59.807Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd84bc7ced0863fe553c","__v":0,"endTime":"2015-09-
07T18:55:01.479Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-
07T18:55:01.480Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd86bc7ced0863fe553d","__v":0,"endTime":"2015-09-
07T18:55:03.071Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd11784427a32e0b5e69a1","startTime":"2015-09-
07T18:55:03.072Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55eddd87bc7ced0863fe553e","__v":0,"endTime":"2015-09-
07T18:55:04.527Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"54fa30dd9f61e74368f3e2f2","questi
on":"54fa320a9f61e74368f3e2f3","startTime":"2015-09-
20T22:03:37.017Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55ff2d46ed37fe98ed05438c","__v":0,"endTime":"2015-09-
20T22:03:50.447Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"5527f9d98bc4e827149df4a7","ques
tion":"5527faf28bc4e827149df4a8","startTime":"2015-09-
20T22:03:51.092Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55ff2d4bed37fe98ed05438d","__v":0,"endTime":"2015-09-
20T22:03:54.895Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"5527f9d98bc4e827149df4a7","ques
tion":"5528047c8bc4e827149df4ba","startTime":"2015-09-
20T22:03:54.896Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55ff2d4ced37fe98ed05438e","__v":0,"endTime":"2015-09-
20T22:03:56.482Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"5527f9d98bc4e827149df4a7","ques
tion":"552809c48bc4e827149df4c2","startTime":"2015-09-
20T22:03:56.482Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"55ff2d4eed37fe98ed05438f","__v":0,"endTime":"2015-09-
20T22:03:58.146Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552816458bc4e827149df4c8","ques
tion":"5528184a8bc4e827149df4d0","startTime":"2015-09-

20T22:03:58.547Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff2d51ed37fe98ed054390","__v":0,"endTime":"2015-09-20T22:04:00.667Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552816458bc4e827149df4c8","question":"552869488bc4e827149df4f4","startTime":"2015-09-20T22:04:00.667Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff2d55ed37fe98ed054391","__v":0,"endTime":"2015-09-20T22:04:05.387Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552881158bc4e827149df506","question":"552884958bc4e827149df507","startTime":"2015-09-20T22:04:06.304Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":19,"totalAttempts":0,"correctAttempts":0,"_id":"55ff2fafed37fe98ed054392","__v":0,"endTime":"2015-09-20T22:14:07.249Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552881158bc4e827149df506","question":"552888308bc4e827149df509","startTime":"2015-09-20T22:14:07.250Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3068ed37fe98ed054393","__v":0,"endTime":"2015-09-20T22:17:11.690Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552881158bc4e827149df506","question":"55288a8c8bc4e827149df50b","startTime":"2015-09-20T22:17:11.691Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff306fed37fe98ed054394","__v":0,"endTime":"2015-09-20T22:17:19.487Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552881158bc4e827149df506","question":"5528946f8bc4e827149df50d","startTime":"2015-09-20T22:17:19.488Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3076ed37fe98ed054395","__v":0,"endTime":"2015-09-20T22:17:26.052Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3977c11935d3c4f46b6c","startTime":"2015-09-20T22:17:26.744Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff307bed37fe98ed054396","__v":0,"endTime":"2015-09-20T22:17:30.924Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce39edc11935d3c4f46b6d","startTime":"2015-09-20T22:17:30.924Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff307ded37fe98ed054397","__v":0,"endTime":"2015-09-20T22:17:33.137Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3a7fc11935d3c4f46b6e","startTime":"2015-09-20T22:17:33.137Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff307fed37fe98ed054398","__v":0,"endTime":"2015-09-20T22:17:35.429Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3a8dc11935d3c4f46b6f","startTime":"2015-09-20T22:17:35.429Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3081ed37fe98ed054399","__v":0,"endTime":"2015-09-20T22:17:37.321Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3c82c11935d3c4f46b70","startTime":"2015-09-20T22:17:37.321Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3372ed37fe98ed05439a","__v":0,"endTime":"2015-09-

20T22:30:10.035Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3dfac11935d3c4f46b71","startTime":"2015-09-20T22:30:10.036Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff337aed37fe98ed05439b","__v":0,"endTime":"2015-09-20T22:30:18.483Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce8001c11935d3c4f46c03","startTime":"2015-09-20T22:30:18.484Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff337fed37fe98ed05439c","__v":0,"endTime":"2015-09-20T22:30:22.845Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"55ce34bfc11935d3c4f46b6b","question":"55ce3977c11935d3c4f46b6c","startTime":"2015-09-20T22:30:22.846Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3381ed37fe98ed05439d","__v":0,"endTime":"2015-09-20T22:30:25.019Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"552881158bc4e827149df506","question":"552884958bc4e827149df507","startTime":"2015-09-20T22:30:25.239Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff3b1fed37fe98ed05439e","__v":0,"endTime":"2015-09-20T23:02:55.555Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"54fa30dd9f61e74368f3e2f2","question":"54fa320a9f61e74368f3e2f3","startTime":"2015-09-20T23:02:55.766Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff4739ed37fe98ed05439f","__v":0,"endTime":"2015-09-20T23:54:33.048Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"54fa30dd9f61e74368f3e2f2","question":"54fa36a39f61e74368f3e2f4","startTime":"2015-09-20T23:54:33.049Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff473fed37fe98ed0543a0","__v":0,"endTime":"2015-09-20T23:54:39.122Z"},{"user":"556cfde5502cc43ec04c69b6","topic":"54fa30dd9f61e74368f3e2f2","question":"54fa3c8b9f61e74368f3e2f6","startTime":"2015-09-20T23:54:39.123Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff4741ed37fe98ed0543a1","__v":0,"endTime":"2015-09-20T23:54:41.227Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f027541070e9415a3f","startTime":"2015-09-21T04:51:49.099Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"55ff8ce3ed37fe98ed0543a2","__v":0,"endTime":"2015-09-21T04:51:52.156Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f027541070e9415a3f","startTime":"2015-09-26T18:31:53.862Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e498ed37fe98ed0543a3","__v":0,"endTime":"2015-09-26T18:31:55.355Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd12dc4427a32e0b5e69a2","startTime":"2015-09-26T18:31:55.357Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e49aed37fe98ed0543a4","__v":0,"endTime":"2015-09-26T18:31:57.112Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-

26T18:31:57.112Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e49eed37fe98ed0543a5","__v":0,"endTime":"2015-09-26T18:32:02.024Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-26T18:32:02.025Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a0ed37fe98ed0543a6","__v":0,"endTime":"2015-09-26T18:32:03.520Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-26T18:32:03.520Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a1ed37fe98ed0543a7","__v":0,"endTime":"2015-09-26T18:32:04.551Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-26T18:32:04.551Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a2ed37fe98ed0543a8","__v":0,"endTime":"2015-09-26T18:32:05.871Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dce00404e78631e67f72e8","startTime":"2015-09-26T18:32:05.871Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a3ed37fe98ed0543a9","__v":0,"endTime":"2015-09-26T18:32:06.415Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd0fde4427a32e0b5e699f","startTime":"2015-09-26T18:32:06.415Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a3ed37fe98ed0543aa","__v":0,"endTime":"2015-09-26T18:32:07.063Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dce00404e78631e67f72e8","startTime":"2015-09-26T18:32:07.064Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a4ed37fe98ed0543ab","__v":0,"endTime":"2015-09-26T18:32:07.794Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dcdfc704e78631e67f72e7","startTime":"2015-09-26T18:32:07.794Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a5ed37fe98ed0543ac","__v":0,"endTime":"2015-09-26T18:32:08.352Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-26T18:32:08.352Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a5ed37fe98ed0543ad","__v":0,"endTime":"2015-09-26T18:32:08.768Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd12dc4427a32e0b5e69a2","startTime":"2015-09-26T18:32:08.768Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a8ed37fe98ed0543ae","__v":0,"endTime":"2015-09-26T18:32:11.192Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f027541070e9415a3f","startTime":"2015-09-26T18:32:11.192Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4a9ed37fe98ed0543af","__v":0,"endTime":"2015-09-

26T18:32:12.896Z"},{"user":"54db0c3527541070e9415967","topic":"556ca495502cc43ec04c68b4","question":"556caf4c502cc43ec04c68dc","startTime":"2015-09-26T18:32:13.106Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4aced37fe98ed0543b0","__v":0,"endTime":"2015-09-26T18:32:15.129Z"},{"user":"54db0c3527541070e9415967","topic":"556ca495502cc43ec04c68b4","question":"556cb0be502cc43ec04c68dd","startTime":"2015-09-26T18:32:15.130Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4b1ed37fe98ed0543b1","__v":0,"endTime":"2015-09-26T18:32:20.279Z"},{"user":"54db0c3527541070e9415967","topic":"556ca495502cc43ec04c68b4","question":"556cb1d0502cc43ec04c68de","startTime":"2015-09-26T18:32:20.280Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4b3ed37fe98ed0543b2","__v":0,"endTime":"2015-09-26T18:32:22.615Z"},{"user":"54db0c3527541070e9415967","topic":"556ca495502cc43ec04c68b4","question":"556cb283502cc43ec04c68df","startTime":"2015-09-26T18:32:22.615Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4b6ed37fe98ed0543b3","__v":0,"endTime":"2015-09-26T18:32:25.831Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dc1e914b5edd6a055b93a2","startTime":"2015-09-26T18:32:25.977Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4baed37fe98ed0543b4","__v":0,"endTime":"2015-09-26T18:32:29.584Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce5704427a32e0b5e6943","startTime":"2015-09-26T18:32:29.584Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4bbed37fe98ed0543b5","__v":0,"endTime":"2015-09-26T18:32:30.584Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce7cb4427a32e0b5e6946","startTime":"2015-09-26T18:32:30.584Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4bded37fe98ed0543b6","__v":0,"endTime":"2015-09-26T18:32:32.375Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce83c4427a32e0b5e6948","startTime":"2015-09-26T18:32:32.376Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4bded37fe98ed0543b7","__v":0,"endTime":"2015-09-26T18:32:33.008Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dceaca4427a32e0b5e6951","startTime":"2015-09-26T18:32:33.008Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4bfed37fe98ed0543b8","__v":0,"endTime":"2015-09-26T18:32:34.360Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dceaef4427a32e0b5e6952","startTime":"2015-09-26T18:32:34.360Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4bfed37fe98ed0543b9","__v":0,"endTime":"2015-09-26T18:32:34.904Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dced744427a32e0b5e695a","startTime":"2015-09-

26T18:32:34.904Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4c1ed37fe98ed0543ba","__v":0,"endTime":"2015-09-26T18:32:36.344Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dcef264427a32e0b5e695b","startTime":"2015-09-26T18:32:36.344Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4c4ed37fe98ed0543bb","__v":0,"endTime":"2015-09-26T18:32:39.374Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","question":"54dbb5d627541070e9415a40","startTime":"2015-09-26T18:32:39.529Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4c7ed37fe98ed0543bc","__v":0,"endTime":"2015-09-26T18:32:42.315Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","question":"54dd08f94427a32e0b5e697a","startTime":"2015-09-26T18:32:42.315Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4c7ed37fe98ed0543bd","__v":0,"endTime":"2015-09-26T18:32:43.072Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","question":"54dd093f4427a32e0b5e697b","startTime":"2015-09-26T18:32:43.073Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4c8ed37fe98ed0543be","__v":0,"endTime":"2015-09-26T18:32:43.768Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","question":"54dd09eb4427a32e0b5e697e","startTime":"2015-09-26T18:32:43.768Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4caed37fe98ed0543bf","__v":0,"endTime":"2015-09-26T18:32:45.571Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","question":"54dd09f04427a32e0b5e697f","startTime":"2015-09-26T18:32:45.572Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5606e4caed37fe98ed0543c0","__v":0,"endTime":"2015-09-26T18:32:45.979Z"},{"user":"556941e2502cc43ec04c6848","topic":"556ca538502cc43ec04c68b7","question":"5606e9efed37fe98ed0543c1","startTime":"2015-09-28T21:06:42.646Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5609abeab3d6aa670a1e1282","__v":0,"endTime":"2015-09-28T21:06:50.652Z"},{"user":"556941e2502cc43ec04c6848","topic":"556ca495502cc43ec04c68b4","question":"556caf4c502cc43ec04c68dc","startTime":"2015-09-28T21:06:50.896Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5609abebb3d6aa670a1e1283","__v":0,"endTime":"2015-09-28T21:06:52.043Z"},{"user":"556941e2502cc43ec04c6848","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f027541070e9415a3f","startTime":"2015-09-28T21:06:52.193Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5609abedb3d6aa670a1e1284","__v":0,"endTime":"2015-09-28T21:06:53.727Z"},{"user":"556941e2502cc43ec04c6848","topic":"54db0c6a27541070e9415a3e","question":"54dbb5d627541070e9415a40","startTime":"2015-09-28T21:06:53.880Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"5609ac38b3d6aa670a1e1285","__v":0,"endTime":"2015-09-

28T21:08:09.562Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dbb3f027541070e9415a3f","startTime":"2015-09-
29T19:42:13.392Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560ae998b3d6aa670a1e1286","__v":0,"endTime":"2015-09-
29T19:42:15.709Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd0fde4427a32e0b5e699f","startTime":"2015-09-
29T19:42:15.710Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560ae999b3d6aa670a1e1287","__v":0,"endTime":"2015-09-
29T19:42:16.876Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-
29T19:42:16.876Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560aea81b3d6aa670a1e1288","__v":0,"endTime":"2015-09-
29T19:46:08.581Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd11784427a32e0b5e69a1","startTime":"2015-09-
29T19:46:08.581Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560aea82b3d6aa670a1e1289","__v":0,"endTime":"2015-09-
29T19:46:09.956Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd12dc4427a32e0b5e69a2","startTime":"2015-09-
29T19:46:09.956Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560aeb12b3d6aa670a1e128a","__v":0,"endTime":"2015-09-
29T19:48:33.851Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dbb5d627541070e9415a40","startTime":"2015-09-
29T22:52:02.589Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560b1615b3d6aa670a1e128c","__v":0,"endTime":"2015-09-
29T22:52:03.807Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dd08f94427a32e0b5e697a","startTime":"2015-09-
29T22:52:03.807Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560b1615b3d6aa670a1e128d","__v":0,"endTime":"2015-09-
29T22:52:04.231Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dd093f4427a32e0b5e697b","startTime":"2015-09-
29T22:52:04.231Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560b1615b3d6aa670a1e128e","__v":0,"endTime":"2015-09-
29T22:52:04.575Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dd09eb4427a32e0b5e697e","startTime":"2015-09-
29T22:52:04.575Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560b1663b3d6aa670a1e128f","__v":0,"endTime":"2015-09-
29T22:53:22.120Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dd09f04427a32e0b5e697f","startTime":"2015-09-
29T22:53:22.120Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560b1665b3d6aa670a1e1290","__v":0,"endTime":"2015-09-
29T22:53:24.806Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3e","qu
estion":"54dd09f44427a32e0b5e6980","startTime":"2015-09-

29T22:53:24.806Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20d9b3d6aa670a1e1291","__v":0,"endTime":"2015-09-29T23:38:00.184Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dc1e914b5edd6a055b93a2","startTime":"2015-09-29T23:38:00.465Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20dcb3d6aa670a1e1292","__v":0,"endTime":"2015-09-29T23:38:03.168Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce5704427a32e0b5e6943","startTime":"2015-09-29T23:38:03.168Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20dfb3d6aa670a1e1293","__v":0,"endTime":"2015-09-29T23:38:05.952Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce7cb4427a32e0b5e6946","startTime":"2015-09-29T23:38:05.952Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e1b3d6aa670a1e1294","__v":0,"endTime":"2015-09-29T23:38:08.793Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dce83c4427a32e0b5e6948","startTime":"2015-09-29T23:38:08.793Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e3b3d6aa670a1e1295","__v":0,"endTime":"2015-09-29T23:38:10.712Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dceaef4427a32e0b5e6952","startTime":"2015-09-29T23:38:10.712Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e4b3d6aa670a1e1296","__v":0,"endTime":"2015-09-29T23:38:11.304Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dceaca4427a32e0b5e6951","startTime":"2015-09-29T23:38:11.304Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e6b3d6aa670a1e1297","__v":0,"endTime":"2015-09-29T23:38:12.937Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dceaef4427a32e0b5e6952","startTime":"2015-09-29T23:38:12.937Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e7b3d6aa670a1e1298","__v":0,"endTime":"2015-09-29T23:38:14.696Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dced744427a32e0b5e695a","startTime":"2015-09-29T23:38:14.696Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20e9b3d6aa670a1e1299","__v":0,"endTime":"2015-09-29T23:38:16.449Z"},{"user":"54db0c3527541070e9415967","topic":"54dbb8cc27541070e9415a41","question":"54dcef264427a32e0b5e695b","startTime":"2015-09-29T23:38:16.449Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560b20ebb3d6aa670a1e129a","__v":0,"endTime":"2015-09-29T23:38:18.216Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dbb3f027541070e9415a3f","startTime":"2015-09-30T13:20:06.750Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be189b3d6aa670a1e129d","__v":0,"endTime":"2015-09-

30T13:20:07.579Z"},{"user":"54db0c3527541070e9415967","topic":"556ca495502cc43ec04c68b4","que
stion":"556caf4c502cc43ec04c68dc","startTime":"2015-09-
30T13:20:07.818Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be18eb3d6aa670a1e129e","__v":0,"endTime":"2015-09-
30T13:20:13.156Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dbb3f027541070e9415a3f","startTime":"2015-09-
30T13:20:13.305Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1eeb3d6aa670a1e129f","__v":0,"endTime":"2015-09-
30T13:21:48.932Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dcdfc704e78631e67f72e7","startTime":"2015-09-
30T13:21:48.934Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f0b3d6aa670a1e12a0","__v":0,"endTime":"2015-09-
30T13:21:50.743Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dce00404e78631e67f72e8","startTime":"2015-09-
30T13:21:50.743Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f1b3d6aa670a1e12a1","__v":0,"endTime":"2015-09-
30T13:21:51.925Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd0fde4427a32e0b5e699f","startTime":"2015-09-
30T13:21:51.925Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f2b3d6aa670a1e12a2","__v":0,"endTime":"2015-09-
30T13:21:53.089Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-
30T13:21:53.089Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f3b3d6aa670a1e12a3","__v":0,"endTime":"2015-09-
30T13:21:54.207Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd11784427a32e0b5e69a1","startTime":"2015-09-
30T13:21:54.207Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f5b3d6aa670a1e12a4","__v":0,"endTime":"2015-09-
30T13:21:56.050Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-
30T13:21:56.050Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1f8b3d6aa670a1e12a5","__v":0,"endTime":"2015-09-
30T13:21:58.483Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd11784427a32e0b5e69a1","startTime":"2015-09-
30T13:21:58.484Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1fab3d6aa670a1e12a6","__v":0,"endTime":"2015-09-
30T13:22:00.646Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd12dc4427a32e0b5e69a2","startTime":"2015-09-
30T13:22:00.646Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAtte
mpts":0,"correctAttempts":0,"_id":"560be1fcb3d6aa670a1e12a7","__v":0,"endTime":"2015-09-
30T13:22:02.953Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","qu
estion":"54dd11784427a32e0b5e69a1","startTime":"2015-09-

30T13:22:02.954Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be1fdb3d6aa670a1e12a8","__v":0,"endTime":"2015-09-30T13:22:04.094Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd0fde4427a32e0b5e699f","startTime":"2015-09-30T13:22:04.094Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be1feb3d6aa670a1e12a9","__v":0,"endTime":"2015-09-30T13:22:05.282Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-30T13:22:05.283Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be1ffb3d6aa670a1e12aa","__v":0,"endTime":"2015-09-30T13:22:06.178Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-30T13:22:06.179Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be204b3d6aa670a1e12ab","__v":0,"endTime":"2015-09-30T13:22:10.356Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-30T13:22:10.356Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be205b3d6aa670a1e12ac","__v":0,"endTime":"2015-09-30T13:22:11.681Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-30T13:22:11.682Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be206b3d6aa670a1e12ad","__v":0,"endTime":"2015-09-30T13:22:13.214Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-30T13:22:13.214Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be208b3d6aa670a1e12ae","__v":0,"endTime":"2015-09-30T13:22:14.646Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd11784427a32e0b5e69a1","startTime":"2015-09-30T13:22:14.647Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be209b3d6aa670a1e12af","__v":0,"endTime":"2015-09-30T13:22:16.072Z"},{"user":"54db0c3527541070e9415967","topic":"54db0c6a27541070e9415a3d","question":"54dd108c4427a32e0b5e69a0","startTime":"2015-09-30T13:22:16.072Z","numCompiles":0,"numErrorFreeCompiles":0,"numRuns":0,"numHints":0,"totalAttempts":0,"correctAttempts":0,"_id":"560be20bb3d6aa670a1e12b0","__v":0,"endTime":"2015-09-30T13:22:17.518Z"}]

Bibliography

[1] Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 121–130, New York, NY, USA, 2015. ACM.

[2] Alireza Ahadi, Raymond Lister, and Donna Teague. Falling behind early and staying behind when learning to program. In *25th Anniversary Psychology of Programming Annual Conference (PPIG), Brighton, England, 25th-27th June*, 2014.

[3] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. The cognitive tutor authoring tools (ctat): preliminary evaluation of efficiency gains. In *Intelligent Tutoring Systems*, pages 61–70. Springer, 2006.

[4] Amjad Altadmri and Neil C.C. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 522–527, New York, NY, USA, 2015. ACM.

[5] Christine Alvarado, Cynthia Bailey Lee, and Gary Gillespie. New cs1 pedagogies and curriculum, the same success factors? In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 379–384, New York, NY, USA, 2014. ACM.

[6] John R Anderson. Rules of the mind. *Lawrence Erlbaum Associates, Hillsdale, New Jersey*, 1993.

[7] John R Anderson and Brian J Reiser. The lisp tutor. *Byte*, 10(4):159–175, 1985.

[8] Ivon Arroyo and Beverly Park Woolf. Inferring learning and attitudes from a bayesian network of log file data. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning Through Intelligent and Socially Informed Technology*, pages 33–40, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.

[9] Ryan Shaun Baker, Albert T. Corbett, Kenneth R. Koedinger, and Angela Z. Wagner. Off-task behavior in the cognitive tutor classroom: When students "game the system". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 383–390, New York, NY, USA, 2004. ACM.

[10] Soumya Basu, Albert Wu, Brian Hou, and John DeNero. Problems before solutions: Automated problem clarification at scale. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 205–213, New York, NY, USA, 2015. ACM.

[11] Jens Bennedsen and Michael E. Caspersen. Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32–36, June 2007.

[12] Susan Bergin and Ronan Reilly. The influence of motivation and comfort-level on learning to program. In *Proceedings of the PPIG*, volume 17, pages 293–304, 2005.

[13] Susan Bergin and Ronan Reilly. Programming: Factors that influence success. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '05, pages 411–415, New York, NY, USA, 2005. ACM.

[14] CJ Bonk and V Dennen. We'll leave the light on for you: Keeping learners motivated in online courses. *Web-based Learning. Englewood Cliffs, NJ: Educational Technology Publications*, 2007.

[15] Christopher M. Boroni, Frances W. Goosey, Michael T. Grinder, Jessica L. Lambert, and Rockford J. Ross. Tying it all together: Creating self-contained, animated, interactive, web-based resources for computer science education. In

The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education, SIGCSE '99, pages 7–11, New York, NY, USA, 1999. ACM.

[16] Neil C.C. Brown and Amjad Altadmri. Investigating novice programming mistakes: Educator beliefs vs. student data. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 43–50, New York, NY, USA, 2014. ACM.

[17] Russel E. Bruhn and Philip J. Burton. An approach to teaching java using computers. *SIGCSE Bull.*, 35(4):94–99, December 2003.

[18] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, Jaime Urquiza, Arto Vihavainen, and Michael Wollowski. Increasing adoption of smart learning content for computer science education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, ITiCSE-WGR '14, pages 31–57, New York, NY, USA, 2014. ACM.

[19] Kevin Buffardi and Stephen H. Edwards. Adaptive and social mechanisms for automated improvement of elearning materials. In *Proceedings of the First ACM Conference on Learning @ Scale Conference*, L@S '14, pages 165–166, New York, NY, USA, 2014. ACM.

[20] Angela Carbone, John Hurst, Ian Mitchell, and Dick Gunstone. An exploration of internal factors influencing student learning of programming. In *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*, ACE '09, pages 25–34, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc.

[21] Janet Carter, Dennis Bouvier, Rachel Cardell-Oliver, Margaret Hamilton, Stanislav Kurkovsky, Stefanie Markham, O. William McClung, Roger McDermott, Charles Riedesel, Jian Shi, and Su White. Motivating all our students? In *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education - Working Group Reports*, ITiCSE-WGR '11, pages 1–18, New York, NY, USA, 2011. ACM.

[22] Jason Carter, Prasun Dewan, and Mauro Pichiliani. Towards incremental separation of surmountable and insurmountable programming difficulties. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 241–246, New York, NY, USA, 2015. ACM.

[23] Yuliya Cherenkova, Daniel Zingaro, and Andrew Petersen. Identifying challenging cs1 concepts in a large problem dataset. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 695–700, New York, NY, USA, 2014. ACM.

[24] Mihaela Cocea. Assessment of motivation in online learning environments. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 414–418. Springer, 2006.

[25] Mihaela Cocea and Stephan Weibelzahl. Cross-system validation of engagement prediction from log files. In *Proceedings of the Second European Conference on Technology Enhanced Learning: Creating New Learning Experiences on a Global Scale*, EC-TEL'07, pages 14–25, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Mihaela Cocea and Stephan Weibelzahl. Eliciting motivation knowledge from log files towards motivation diagnosis for adaptive systems. In *User Modeling 2007*, pages 197–206. Springer, 2007.

[27] Mihaela Cocea and Stephan Weibelzahl. Log file analysis for disengagement detection in e-learning environments. *User Modeling and User-Adapted Interaction*, 19(4):341–385, October 2009.

[28] J. McGrath Cohoon. Toward improving female retention in the computer science major. *Commun. ACM*, 44(5):108–114, May 2001.

[29] Allan Collins, John Seely Brown, and Ann Holum. Cognitive apprenticeship: Making thinking visible. *American educator*, 15(3):6–11, 1991.

[30] Allan Collins, John Seely Brown, and Susan E Newman. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, 18:32–42, 1989.

[31] Lorraine Frances Dame. *Student readiness, engagement and success in entry level undergraduate mathematics courses.* PhD thesis, University of Victoria, 2012.

[32] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. All syntax errors are not equal. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 75–80, New York, NY, USA, 2012. ACM.

[33] Juan Manuel Dodero, Camino Fernández, and Daniel Sanz. An experience on students' participation in blended vs. online styles of learning. *SIGCSE Bull.*, 35(4):39–42, December 2003.

[34] Gregory Dyke. Which aspects of novice programmers' usage of an ide predict learning outcomes. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 505–510, New York, NY, USA, 2011. ACM.

[35] Lorna M Earl. *Assessment as learning: Using classroom assessment to maximize student learning*. Corwin Press, 2012.

[36] Stephen H. Edwards, Jason Snyder, Manuel A. Pérez-Quiñones, Anthony Allevato, Dongkwan Kim, and Betsy Tretola. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, ICER '09, pages 3–14, New York, NY, USA, 2009. ACM.

[37] A. F. Elgamal, H. A. Abas, and E. S. Baladoh. An interactive e-learning system for improving web programming skills. *Education and Information Technologies*, 18(1):29–46, March 2013.

[38] Anthony Estey, Amy Gooch, and Bruce Gooch. Addressing industry issues in a multi-disciplinary course on game design. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 71–78, New York, NY, USA, 2009. ACM.

[39] Gerald E. Evans and Mark G. Simkin. What best predicts computer proficiency? *Commun. ACM*, 32(11):1322–1327, November 1989.

[40] Nickolas J.G. Falkner and Katrina E. Falkner. A fast measure for identifying at-risk students in computer science. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, pages 55–62, New York, NY, USA, 2012. ACM.

[41] Sue Fitzgerald, Gary Lewandowski, Renee McCauley, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2):93–116, 2008.

[42] EricJ. Fox. Constructing a pragmatic science of learning and instruction with functional contextualism. *Educational Technology Research and Development*, 54(1):5–36, 2006.

[43] J. W. Gikandi, D. Morrow, and N. E. Davis. Online formative assessment in higher education: A review of the literature. *Comput. Educ.*, 57(4):2333–2351, December 2011.

[44] Annagret Goold and Russell Rimmer. Factors affecting performance in first-year computing. *SIGCSE Bull.*, 32(2):39–43, June 2000.

[45] Irene Govender. The learning context: Influence on learning to program. *Comput. Educ.*, 53(4):1218–1230, December 2009.

[46] Mark Guzdial. A media computation course for non-majors. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '03, pages 104–108, New York, NY, USA, 2003. ACM.

[47] Stuart Hansen and Erica Eddy. Engagement and frustration in programming projects. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 271–275, New York, NY, USA, 2007. ACM.

[48] Alireza Hassanzadeh, Fatemeh Kanaani, and Shában Elahi. A model for measuring e-learning systems success in universities. *Expert Syst. Appl.*, 39(12):10959–10966, September 2012.

[49] Kenny Heinonen, Kasper Hirvikoski, Matti Luukkainen, and Arto Vihavainen. Using codebrowser to seek differences between novice programmers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 229–234, New York, NY, USA, 2014. ACM.

[50] Daniel Malcolm Hoffman, Ming Lu, and Tim Pelton. A web-based generation and delivery system for active code reading. In *Proceedings of the 42Nd ACM*

Technical Symposium on Computer Science Education, SIGCSE '11, pages 483–488, New York, NY, USA, 2011. ACM.

[51] Jason Bond Huett, Kevin E Kalinowski, Leslie Moller, and Kimberly Cleaves Huett. Improving the motivation and retention of online students through the use of arcs-based e-mails. *The Amer. Jrnl. of Distance Education*, 22(3):159–176, 2008.

[52] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA, 2010. ACM.

[53] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports*, pages 41–63, 2015.

[54] Essi Isohanni and Hannu-Matti Järvinen. Are visualization tools used in programming education?: By whom, how, why, and why not? In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 35–40, New York, NY, USA, 2014. ACM.

[55] Ville Isomöttönen and Ville Tirronen. Teaching programming by emphasizing self-direction: How did students react to the active role required of them? *Trans. Comput. Educ.*, 13(2):6:1–6:21, July 2013.

[56] Matthew C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 73–84, New York, NY, USA, 2006. ACM.

[57] Matthew C. Jadud and Brian Dorn. Aggregate compilation behavior: Findings and implications from 27,698 users. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 131–139, New York, NY, USA, 2015. ACM.

[58] Tony Jenkins. The motivation of students of programming. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, pages 53–56, New York, NY, USA, 2001. ACM.

[59] Wei Jin. Pre-programming analysis tutors help students learn basic programming concepts. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 276–280, New York, NY, USA, 2008. ACM.

[60] W Lewis Johnson and Elliot Soloway. Proust: Knowledge-based program understanding. *Software Engineering, IEEE Transactions on*, 3(3):267–275, 1985.

[61] John M Keller. Development and use of the arcs model of instructional design. *Journal of instructional development*, 10(3):2–10, 1987.

[62] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 41–46, 2016.

[63] Päivi Kinnunen and Lauri Malmi. Why students drop out cs1 course? In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 97–108, New York, NY, USA, 2006. ACM.

[64] Paul A Kirschner, John Sweller, and Richard E Clark. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, 41(2):75–86, 2006.

[65] René F. Kizilcec and Sherif Halawa. Attrition and achievement gaps in online learning. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 57–66, New York, NY, USA, 2015. ACM.

[66] Daniel Knox and Sally Fincher. Where students go for knowledge and what they find there. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 35–40, New York, NY, USA, 2013. ACM.

[67] Kenneth R. Koedinger and Albert Corbett. Cognitive Tutors: Technology Bringing Learning Sciences to the Classroom. In Keith Sawyer, editor, *The Cambridge Handbook of the Learning Sciences*, pages 60–77. Cambridge University Press, Cambridge, 2006.

[68] Kenneth R. Koedinger, Jihee Kim, Julianna Zhuxin Jia, Elizabeth A. McLaughlin, and Norman L. Bier. Learning is not a spectator sport: Doing is better than watching for learning from a mooc. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 111–120, New York, NY, USA, 2015. ACM.

[69] Kenneth R. Koedinger, Elizabeth A. McLaughlin, and John C. Stamper. Data-driven learner modeling to understand and improve online learning: Moocs and technology to advance learning and learning research (ubiquity symposium). *Ubiquity*, 2014(May):3:1–3:13, May 2014.

[70] Vitomir Kovanović, Dragan Gašević, Shane Dawson, Srećko Joksimović, Ryan S. Baker, and Marek Hatala. Penetrating the black box of time-on-task estimation. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, LAK '15, pages 184–193, New York, NY, USA, 2015. ACM.

[71] Maria S. Lam, Eric Y. Chan, Victor C. Lee, and Y. T. Yu. Designing an automatic debugging assistant for improving the learning of computer programming. In *Proceedings of the 1st International Conference on Hybrid Learning and Education*, ICHL '08, pages 359–370, Berlin, Heidelberg, 2008. Springer-Verlag.

[72] Kris M. Y. Law, Victor C. S. Lee, and Y. T. Yu. Learning motivation in e-learning facilitated computer programming courses. *Comput. Educ.*, 55(1):218–228, August 2010.

[73] R. R. Leeper and J. L. Silver. Predicting success in a first programming course. In *Proceedings of the Thirteenth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '82, pages 147–150, New York, NY, USA, 1982. ACM.

[74] Linda Lumsden. *Student Motivation: Cultivating a Love of Learning.* ERIC, 1999.

[75] Daniel S. McCain, Christos Sakalis, and Arnold Pears. Exploring assessment practices at university. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 171–172, New York, NY, USA, 2014. ACM.

[76] Robert McCartney and Kate Sanders. First-year students' social networks: Learning computing with others. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 159–163, New York, NY, USA, 2014. ACM.

[77] Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '01, pages 125–180, New York, NY, USA, 2001. ACM.

[78] Monica M. McGill. Learning to program with personal robots: Influences on student motivation. *Trans. Comput. Educ.*, 12(1):4:1–4:32, March 2012.

[79] Leslie Moller and James D Russell. An application of the arcs model design process and confidence-building strategies. *Performance Improvement Quarterly*, 7(4):54–69, 1994.

[80] Jonathan P. Munson and Elizabeth A. Schilling. Analyzing novice programmers' response to compiler error messages. *J. Comput. Sci. Coll.*, 31(3):53–61, January 2016.

[81] Christian Murphy, Gail Kaiser, Kristin Loveland, and Sahar Hasan. Retina: Helping students and instructors based on observed programming activities. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 178–182, New York, NY, USA, 2009. ACM.

[82] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. Compiler error messages: What can help novices? In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 168–172, New York, NY, USA, 2008. ACM.

[83] Cindy Norris, Frank Barry, James B. Fenwick Jr., Kathryn Reid, and Josh Rountree. Clockit: Collecting quantitative data on how beginning software developers really work. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, pages 37–41, New York, NY, USA, 2008. ACM.

[84] Andrei Papancea, Jaime Spacco, and David Hovemeyer. An open platform for managing short programming exercises. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 47–52, New York, NY, USA, 2013. ACM.

[85] Nick Parlante. Nifty reflections. *SIGCSE Bull.*, 39(2):25–26, June 2007.

[86] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '07, pages 204–223, New York, NY, USA, 2007. ACM.

[87] David N Perkins, Chris Hancock, Renee Hobbs, Fay Martin, and Rebecca Simmons. Conditions of learning in novice programmers. *Journal of Educational Computing Research*, 2(1):37–55, 1986.

[88] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 505–513, 2015.

[89] Leo Porter and Daniel Zingaro. Importance of early performance in cs1: Two conflicting assessment stories. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 295–300, New York, NY, USA, 2014. ACM.

[90] Leo Porter, Daniel Zingaro, and Raymond Lister. Predicting student success using fine grain clicker data. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 51–58, New York, NY, USA, 2014. ACM.

[91] Lei Qu, Ning Wang, and W. Lewis Johnson. Using learner focus of attention to detect learner motivation factors. In *Proceedings of the 10th International Conference on User Modeling*, UM'05, pages 70–73, Berlin, Heidelberg, 2005. Springer-Verlag.

[92] Martina A. Rau, Vincent Aleven, Nikol Rummel, and Stacie Rohrbach. Why interactive learning environments can have it all: Resolving design conflicts between competing goals. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 109–118, New York, NY, USA, 2013. ACM.

[93] C. M. Reigeluth. Instructional design: What is it and why is it? *Instructional-design theories and models*, pages 3–36, 1983.

[94] Anthony Robins. Learning edge momentum: a new account of outcomes in cs1. *Computer Science Education*, 20(1):37–71, 2010.

[95] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003.

[96] Ma. Mercedes T. Rodrigo, Ryan S. Baker, Matthew C. Jadud, Anna Christine M. Amarra, Thomas Dy, Maria Beatriz V. Espejo-Lahoz, Sheryl Ann L. Lim, Sheila A.M.S. Pascua, Jessica O. Sugay, and Emily S. Tabanao. Affective and behavioral predictors of novice programmer achievement. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '09, pages 156–160, New York, NY, USA, 2009. ACM.

[97] Guido Rößling, Mike Joy, Andrés Moreno, Atanas Radenski, Lauri Malmi, Andreas Kerren, Thomas Naps, Rockford J. Ross, Michael Clancy, Ari Korhonen, Rainer Oechsle, and J. Ángel Velázquez Iturbide. Enhancing learning management systems to better support computer science education. *SIGCSE Bull.*, 40(4):142–166, November 2008.

[98] Hamzeh Roumani. Design guidelines for the lab component of objects-first cs1. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '02, pages 222–226, New York, NY, USA, 2002. ACM.

[99] Nathan Rountree, Janet Rountree, and Anthony Robins. Predictors of success and failure in a cs1 course. *SIGCSE Bull.*, 34(4):121–124, December 2002.

[100] Anna Russo Kennedy. Towards a data-driven analysis of programming tutorials' telemetry to improve the educational experience in introductory programming courses. Master's thesis, University of Victoria, 2015.

[101] W Sack and E Soloway. From meno to proust to chiron: Ai design as iterative engineering: Intermediate results are important. In *Proceedings of the Invited Workshop on Computer-Based Learning Environments*, 1998.

[102] Dale H Schunk. Learning theories. *Prentice Hall Inc., New Jersey*, 1996.

[103] Elaine Seymour, Nancy M Hewitt, and Cynthia M Friend. *Talking about leaving: Why undergraduates leave the sciences*, volume 12. Westview Press Boulder, CO, 1997.

[104] G. E. Snelbecker. *Raising the bar for instructional outcomes: Toward transformative learning experiences.* McGraw Hill, New York, 1974.

[105] Elliot Soloway. Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9):850–858, 1986.

[106] Jaime Spacco, Paul Denny, Brad Richards, David Babcock, David Hovemeyer, James Moscola, and Robert Duvall. Analyzing student work patterns using programming exercise data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 18–23, New York, NY, USA, 2015. ACM.

[107] Jaime Spacco, Davide Fossati, John Stamper, and Kelly Rivers. Towards improving programming habits to create better computer science course outcomes. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 243–248, New York, NY, USA, 2013. ACM.

[108] Calkin Suero Montero and Jarkko Suhonen. Emotion analysis meets learning analytics: Online learner profiling beyond numerical data. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 165–169, New York, NY, USA, 2014. ACM.

[109] Emily S. Tabanao, Ma. Mercedes T. Rodrigo, and Matthew C. Jadud. Predicting at-risk novice java programmers through the analysis of online protocols. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 85–92, New York, NY, USA, 2011. ACM.

[110] Vincent Tinto. Research and practice of student retention: what next? *Journal of College Student Retention: Research, Theory and Practice*, 8(1):1–19, 2006.

[111] Ville Tirronen and Ville Isomöttönen. Making teaching of programming learning-oriented and learner-directed. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, pages 60–65, New York, NY, USA, 2011. ACM.

[112] Evangelos Triantafillou, Andreas Pomportsis, and Stavros Demetriadis. The design and the formative evaluation of an adaptive educational system based on cognitive styles. *Comput. Educ.*, 41(1):87–103, June 2003.

[113] Mieke Vandewaetere, Piet Desmet, and Geraldine Clarebout. Review: The contribution of learner characteristics in the development of computer-based adaptive learning environments. *Comput. Hum. Behav.*, 27(1):118–130, January 2011.

[114] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

[115] George Veletsianos. Designing opportunities for transformation with emerging technologies. *Educational Technology*, 51(2):41, 2011.

[116] George Veletsianos. The significance of educational technology history and research. *eLearn*, 2014(11), November 2014.

[117] Philip R Ventura Jr. Identifying predictors of success for an objects-first cs1. *Computer Science Education*, 25(3):223–243, 2005.

[118] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the Tenth Annual Conference on International Computing Education Research*, ICER '14, pages 19–26, New York, NY, USA, 2014. ACM.

[119] Arto Vihavainen, Matti Luukkainen, and Petri Ihantola. Analysis of source code snapshot granularity levels. In *Proceedings of the 15th Annual Conference on Information Technology Education*, SIGITE '14, pages 21–26, New York, NY, USA, 2014. ACM.

[120] Arto Vihavainen, Matti Luukkainen, and Jaakko Kurhila. Multi-faceted support for mooc in programming. In *Proceedings of the 13th Annual Conference on Information Technology Education*, SIGITE '12, pages 171–176, New York, NY, USA, 2012. ACM.

[121] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 93–98, New York, NY, USA, 2011. ACM.

[122] Rebecca Vivian, Katrina Falkner, and Claudia Szabo. Can everybody learn to code?: Computer science community perceptions about learning the fundamentals of programming. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 41–50, New York, NY, USA, 2014. ACM.

[123] Christopher Watson, Frederick W. B. Li, and Jamie L. Godwin. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 319–323, 2013.

[124] Christopher Watson and Frederick W.B. Li. Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ITiCSE '14, pages 39–44, New York, NY, USA, 2014. ACM.

[125] Christopher Watson, Frederick W.B. Li, and Jamie L. Godwin. No tests required: Comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 469–474, New York, NY, USA, 2014. ACM.

[126] Naomi Rosh White. Tertiary education in the noughties: the student perspective. *Higher Education Research & Development*, 25(3):231–246, 2006.

[127] Brenda Cantwell Wilson and Sharon Shrock. Contributing to success in an introductory computer science course: A study of twelve factors. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 184–188, New York, NY, USA, 2001. ACM.

[128] Brent G Wilson, Patrick Parrish, and George Veletsianos. Raising the bar for instructional outcomes: Toward transformative learning experiences. *Educational Technology*, 48(3):39, 2008.

[129] Hanna Yakymova, Yoann Monteiro, and Daniel Zingaro. Study strategies and exam grades in cs1. In *Proceedings of the Western Canadian Conference on Computing Education (WCCCE)*, pages 24:1–24:3, 2016.

[130] Daniel Zingaro and Leo Porter. Peer instruction in computing: The value of instructor intervention. *Comput. Educ.*, 71:87–96, February 2014.

[131] Stuart Zweben and Betsy Bizot. 2013 taulbee survey. *COMPUTING*, 26(5), 2014.

[132] Stuart Zweben and Betsy Bizot. 2015 Taulbee Survey. *Computing Research News*, 28(5), 2016.