Fall 12-20-2019

The Effects of Automated Grading on Computer Science Courses at the University of New Orleans

Jerod F A Dunbar
University of New Orleans, New Orleans, dunbar.jfa@gmail.com

The Effects of Automated Grading on Computer Science Courses at the University of New Orleans

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Jerod Dunbar

B.A. Louisiana State University, 2012

December 2019

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

This is a study of the impacts of the incorporation, into certain points of the Computer Science degree program at the University of New Orleans, of Course Management software with an Autograding component. The software in question, developed at Carnegie Mellon University, is called “Autolab.” We begin by dissecting Autolab in order to gain an understanding of its inner workings. We can then take out understanding of its functionality and apply that to an examination of fundamental changes to courses in the time since they incorporated the software. With that, we then compare Drop, Failure, Withdrawal rate data from before and after the introduction of Autolab. With this collection of data, we can conclude, to a certain extent, that Autolab has had a negligible impact on course outcomes, but a measurable impact on course structure and pedagogy as well as improved quality of life for students and professors, alike.

Keywords: autograding, computer science pedagogy, autolab

INTRODUCTION

The goal of this thesis is to ascertain what effects, if any, the adoption of the Autolab software suite from Carnegie Mellon University has had on some of the earliest programming courses in the undergraduate degree path for the Computer Science Department (referred to henceforth as the "Intro Sequence"). For the majority of students, the Intro Sequence acts as the very first exposure to software development. As such, the goal of these courses is to lay a foundation of good habits by drilling the fundamental concepts of programming into the students through increasingly rigorous and varied assignments alongside routine formal testing. This is hard work, for student and instructor alike. Students are regularly working to quickly consume and understand complex syntax, thought patterns, and theories while Instructors continue the perpetual, noble struggle to support these students while battling the drudgery of course administration. Frustrations and backlogs grow well in this type of environment, but Autolab offers to assist with this. The idea is to automate a number of core instructor responsibilities in order to free that instructor's time to further focus on directly improving student outcomes. With that in mind, it would be fair to assume that any course that incorporates Autolab into its structure should show an overall improvement in efficiency and progress semester over semester. In the Fall semester of 2018, three of the five Intro Sequence courses (CSCI 1583, CSCI 2120, and CSCI 2467) made the switch to Autolab. This investigation will seek to examine how this change has affected not only those three courses, but the entirety of the Intro Sequence in the time since then.

AUTOLAB

An Overview

Autolab is course management software created at Carnegie Mellon University (CMU) in 2010 under the leadership of Dr. David O'Hallaron. "The Autolab Project," as it is referred to by CMU, was first designed and implemented by a diverse group of CMU students with a common mission of simplifying teaching for instructors and students everywhere by addressing some of the time-sink inherent to course administration through automation of some of the more mundane aspects. Through these optimizations, an instructor's time is freed for more tailored time with students. "There's the obvious time commitment required for classes and lectures, but striving for excellence in education necessarily requires countless hours spent outside of the classroom." (Zimmerman, 2015) When you free some of that time previously allocated for course management tasks, you might find that there is more time for walk-in office hours, tutoring sessions, online Question and Answer sessions, and more.

Autolab is a full course management platform, but its primary benefit lies in tackling one particular head of the course management hydra: assignment grading.

> "Very few people sit down to grade assignments with enthusiasm and vigor, which is unfortunate; students can't learn from their mistakes until they know what they are. The time required to grade fundamentally opposes student learning and growth." (Zimmerman, 2015)

Automatic grading allows for a drastic decrease in lead-time between submission and feedback. Getting feedback to a student sooner in turn means that student has more time to digest submission comments, make changes, and ask follow-up questions in a timely manner, theoretically resulting in a higher-quality final submission. Thus, students have an easier time catching up or avoiding falling behind in the first place.

Autolab in Action

Through the Autolab webapp, instructors create profiles and link them to any number of custom course modules. These stored "courses" contain all the information relevant to course participants, Autolab's users. In the webapp, these users are split into three different categories, "Instructors," "Course Administrators," and "Students." Each type of user interacts with the webapp differently, and each has different access rights to various types and amounts of course data.

For example, an Instructor creates a course module that corresponds to a section of CSCI 2467 in the upcoming semester. She is granted total control over the structure of the module as a result of the Instructor status. Be it basic info like scheduled semester, course name, and semester start and ending dates or more specific options like the standard late penalty, number of submissions allowed, number of grace days per semester, and even how many seconds after a deadline Autolab will still accept submissions, the Instructor has complete control. This includes the assignment schedule, of course, with options to hide or reveal assignments with the push of a button. The assignments themselves are, as always, instructor-generated, but done so in a way

specific to the functionality of Autolab. Courses are modular, so each assignment exists in a vacuum within Autolab. As such, changes can always be made, and fresh assignments uploaded, with ease. What's more, the level of possible customization really comes into focus when adding assignment modules to a course. Since Autolab allows for the use of custom autograding scripts, and in fact is built around that very concept, it is no exaggeration to say that the scope and design of any single assignment hosted in Autolab is limited only by its accompanying grading script, and thereby by the imagination or creativity of the person who writes it. The only requirement is that Autolab be able to read and interpret the results of the grading script (typically through the use of a JSON string), everything else is up to the Instructor or to the assignment and script designers utilized by the Instructor. Of course, our Instructor's influence is not limited to the creation and customization of courses and assignments. She is also privy to assignment results and how they are portrayed through the webapp. Instructors may reveal as much or as little information to students as they like; Autolab even goes so far as to include a scoreboard option for every assignment where students may adopt pseudonyms and post their scores in friendly competition with their course mates. All these features and more are at the disposal of Instructors right from the start.

Carrying on with our original analogy, let us say that our Instructor has found herself managing multiple courses and in need of assistance with balancing her workload. She enlists the help of a Graduate Assistant (GA) to assist with some of her responsibilities for the duration of the semester. This GA will be a lab instructor, and thereby the first point of contact for any student issues or questions. That being the case, there is a minimum level of confidential course information to which he or she should be privy in order to best perform this function. For example, say a student has received a middling score on an assignment submission and is unsure as to what went wrong and what might be done to improve the assignment score before the submission deadline. In order to offer advice or guidance, the GA will need to see what work the student has already done. He or she could look directly at the student's working code, but more useful information might be found more quickly by viewing the feedback generated by the autograding script for each of the student's submissions. In order to view this, the submission records must be accessed. After finding the most recent submission of the student in question, our GA checks the autograder feedback against the contents of the submission. At this point, our GA might find some bit of compiler feedback or some such that points to direct problems with submitted code. From there, the student can be instructed on the mistakes revealed and make the necessary improvements. Other times, the GA might find the feedback to be erroneous or unclear, and after some subsequent digging, it may be found that the student's code is correct, but in an unexpected way. In other words, the solution produces an accurate result, but not one that our custom autograding script was created to handle, and thus the assignment has been incorrectly graded. It is now the GA's responsibility to either alert the Instructor or access the autograding script and make updates to grant the student his or her proper grade and avoid this mistake in the future. So, our GA requires some of the access rights afforded to our Instructor, but clearly not all of them. The "Course Administrator" access profile allows for just that. With proper access, the submissions and scripts are brought up to date and the student receives a proper score. As an aside, this scenario also serves as a cautionary reminder of the importance of maintaining a level of human involvement in assignment grading in Autolab-supported courses.

Grading scripts are a wonderful boon to instructors and students alike, but they are still only code, and thusly subject to the same pitfalls as any other program. It is up to the script creator, and the Instructor in the case that they are separate people, to ensure that mistakes are absolutely minimized at the outset and addressed as they arise. A high-quality, creative, thorough grading script is possibly even more important than the assignment itself.

Once classes begin, all students are tasked with creating accounts in Autolab and confirming enrollment in their respective course sections. Once enrolled, each student user is granted the "Student" access profile which grants whatever course information the Instructor has chosen to reveal at that time. With that, the course clock starts ticking and the schedule of assignments rolls onward. Months later, our Instructor examines course results and feedback, and begins to plan for the next semester.

Courses in Autolab

Instructors go to the Autolab webapp and register their course for the time period matching the target semester. Once a course is created, individual assignments can be crafted and uploaded into Autolab as part of the created course. Each assignment requires three components. The first is the "starting point" for students. This is the minimum level of information that will be revealed to Students at the start of the semester (whether a handout, a compressed directory of stubbed code, etc..). Creating this content here is no different than for standard non-Autolab courses. The second assignment component is the operating system image that will be responsible for running the grading script and recording the result. Autolab is a truly compartmentalized system, from the separation of courses right down to the parallelization of isolated grading processes. Those grading processes are hosted in Virtual Machines or Containers, and so require information regarding their specific runtime environments. That information is packaged, as a system image, together with the grading script at creation and can be changed at any time. Finally, every assignment requires the grading script itself.

These three components provide powerful avenues for assignment customization. Firstly, the starting point for any assignment frames the way that a student will view the assignment as a whole, the entire learning experience is shaped by it. Next, system images are customizable in much the same way as grading scripts. When assignments are created, the creator must reveal the structure to Autolab in the form of "problem" labels. Every component of an assignment that is separately scored is considered a "problem." So long as the resulting output of a grading script is a JSON string matching the assignment structure and problem labels known to Autolab, anything goes in terms of design.

Tango

Up to this point, the discussion has been of Autolab as a course management suite, but its most important strength lies in one particular feature, autograding. The software driving the autograding functionality of Autolab is CMU's custom back-end, known as "Tango." Tango is a Python-based, "RESTful" web service. This means that it utilizes the Representational State Transfer (REST) architectural style of programming, which imposes design constraints on how a service stores and represents its resources in order to ensure that those resources are easily and reliably interacted with whenever the Application Programming Interface (API) of the service is

utilized. In other words, being RESTful means following a common set of software design standards. This allows Tango to be somewhat customizable by Instructors while still reliably managing and facilitating every one of Autolab's myriad tandem grading jobs.

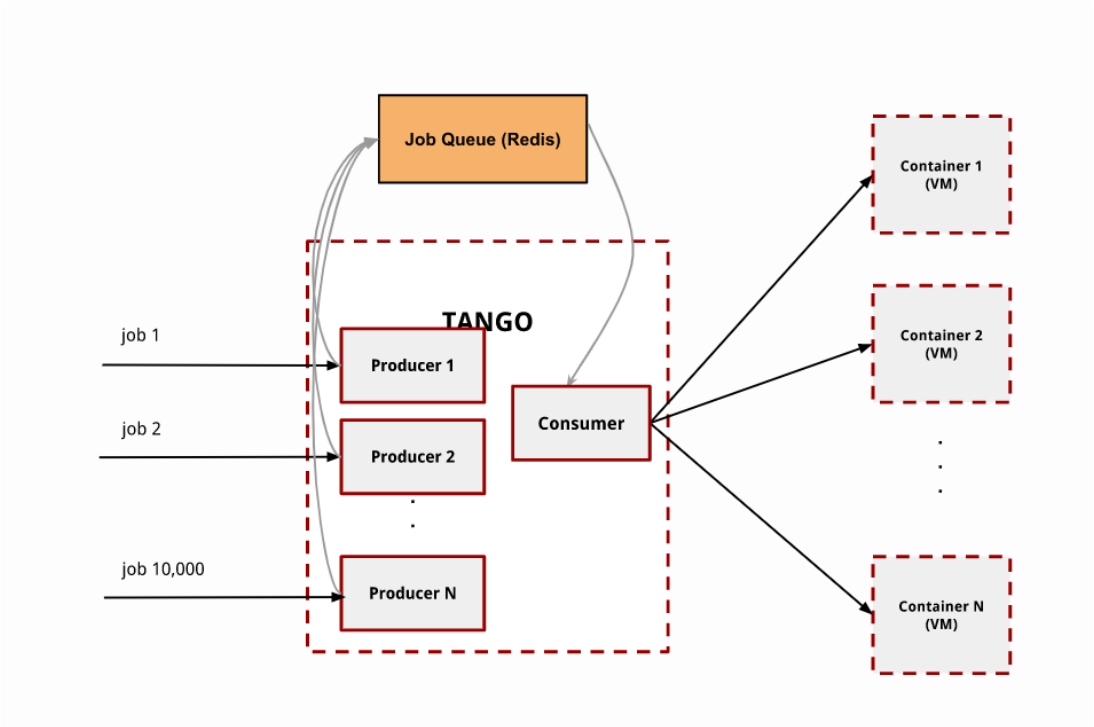

Figure 1: Tango Producer-Consumer Model (Canberk, 2015)

It works to process assignment submissions by following the Producer-Consumer Model. First, incoming submissions are assigned to one of many Producer processes which packages the submission file with its related autograder and enqueues the result (into a Redis-backed queue) to await further processing. A single separate Consumer process works to dequeue each submission and match it with a Virtual Machine or Container that meets the minimum requirements of the system image information packaged with the submission as part of the autograder. If none can be found, then the Consumer creates one and then uploads the submission and its autograder. When the Virtual Machine or Container is ready, the autograder runs its contained grading script. The basic form of any grading script consists of any number of focused, pre-defined "problems." Each assignment or lab is made up of a series of such problems (which are themselves made up of a problem name and a point value). A grading script intended for use with a particular assignment processes a given submission, determines the number of points scored by the submitter for a given problem, and appends the name of the problem and the point value to a JSON string that is output to the Autolab webapp upon completion of grading. There it is parsed into the graded results of the individual problems in the assignment and stored as a single submission. All submissions are stored, with student-identifying information, in the Autolab "Course" to which they correspond. The results of each test are output to one or more endpoints hooked into Autolab. These results are used to populate point values for the corresponding entries on the grading rubric or to otherwise generate error alerts for students (instructors have

the option to allow for corrections through a variable number of submissions). At the end of the pipeline, the submission is noted, a grade is recorded, and a student is left with high-quality feedback (along with contact points for questions/complaints) all without the teacher lifting a finger.

In the case where there is an error in the submission processing, the autograder should always be designed to return a JSON string of information regarding the error. This way, be it either student error or inherent design flaw, the problem can be quickly addressed. In a perfect scenario, the JSON returned represents an accurate scoring of the submission, which is then simultaneously logged to the gradebook and returned to the submitter along with whatever feedback the grading script is meant to return. In this way, Autolab handles any number of grading jobs simultaneously.

The Study

My Work with Autolab

In the Summer of 2017, I was awarded a Graduate Assistantship with the IT Director for the Computer Science Department at UNO, Professor Matt Toups. Professor Toups is also the instructor for CSCI 2467 as well as the first to incorporate Autolab into his course at the University. As such, my assistantship work with him has since then been centered around Autolab as well as acting as a Course Administrator and Lab Instructor for his course. Autolab ships ready to work with one of three virtualization platforms: Amazon AWS Cloud EC2 instances (running Virtual Machines), Docker containerization software, and Carnegie Mellon's own containerization software called "Tashi." It was decided that courses at UNO would use Docker to facilitate their autograding, so my job was to ensure that Docker and Autolab worked well together in our department. This work consisted primarily of configuring and testing both technologies and making adjustments where necessary until the two worked together smoothly. Once Autolab was up and running, my role shifted to that of a Teaching Assistant and Lab Instructor. All of this allowed me the opportunity to experience Autolab from both the "back end" (through my work with Tango and autograders) and the "front end" (through my time in the classroom).

Hypothesis

Based on my direct experience with Autolab in a classroom setting, it is my expectation that while the purported time-saving benefits of Autograding are real for professors and students alike, their impact on overall course outcomes will not profoundly affect the preexisting student success trends within the Intro Sequence. This is, in my opinion, due mostly to the nature of the students and their tendency to seek out extracurricular help with understanding course material (or lack of tendency, as is often the case). That said, the ability to receive instant feedback is an undeniable quality-of-life boon to all parties involved. In other words, I believe that Autolab will likely have a great impact on course pacing, but its quantitative impact on course statistics will likely be relatively muted.

Brief Overview of the Intro Sequence

The Introductory Sequence of the Computer Science curriculum of UNO is the very first progression of programming courses for students new to the program. It begins with Software Design and Development I (CSCI 1583), commonly referred to as "Java I" because the course imparts the fundamentals of programming using the ubiquitous and well-developed Java programming language. The goal here is to give prospective computer science students a rigorous introduction to Object-Oriented programming by combining a rich lecture schedule with hands-on lab assignments in the connected lab section, CSCI 1581. Successful students emerge from this course with a considerable understanding of the use of Java classes, variables, methods, and syntax, and basic algorithmic problem solving using these tools and the Object-Oriented approach.

This leads into the second course of the series, Software Design and Development II (CSCI 2120), or "Java II" for its reliance on language similarly to CSCI 1583. If Java I is intended to give students familiarity with the practical "building blocks" of Java and show them how to stack them in order to solve fairly complex problems, then Java II seeks to turn that familiarity into expertise with its introduction to more abstract concepts (like inheritance and polymorphism) along with next-level algorithmic techniques (like sorting, searching, and recursion) in order to construct larger, more complex, and more complete software solutions.

Following that is Data Structures (CSCI 2125) which continues the Java focus, but with a plan of study centered around the concepts and implementations of structured data objects (lists, stacks, trees, dictionaries, etcetera). Then comes CSCI 2450, Machine Structure and Assembly Language Programming. This course serves as a step down from higher-level programming and nearer to the bare metal beneath it all; the focus is on the basics of computer organization, operating systems, and the programming language hierarchy. On that note, it is also the first class to introduce a new language to the students, Assembly, and the role that this language plays in facilitating human-computer communication.

Finally, we come to the gateway course of the undergraduate curriculum, Systems Programming Concepts (CSCI 2467). Up to now, students have learned how to digest problems in order to design and implement compatible solutions utilizing a practiced understanding of the nuances of a rich, powerful language. They have also been taught the concepts and principles behind the technology, the layers of complex machinery and architecture that make programming more than magic. CSCI 2467, however, is the first course to bring all of these things together. It introduces the concepts that make up a computer system, and the tools used to create, manage, and manipulate that system. Using shell scripting, system calls, and the language C, students are taught to navigate the internal world of their computers in order to solve problems of an entirely new scale and complexity. This course is the capstone of the Intro Sequence, it is meant to take students beyond their meager familiarity with the walled garden of their programming environment to an embrace of the computer as a whole, predicated by the idea that programming is more than "something a computer does," it is "what a computer is."

Together, these courses lay a foundation of substantial, if far from complete, understanding of the broad field of Computer Science. The students who come out of the back end of the sequence emerge practiced, confident, and eager to foray into the myriad, more focused specializations represented in the upper level courses of the degree program and beyond.

The descriptions above are just a broad overview, however. The contents of the courses themselves are fluid, by nature, owing to the ever-evolving state of technology. In order to stay relevant, the courses must adapt. The goal of this research is to determine the impact of one such recent technologically-driven change to the department, the introduction of automated grading. As such, any investigation into the extent of the effects of this change must take into account the content history of the affected courses and their relationships to any perceived changes in student success.

The Old

The Autolab software was first utilized for CSCI 1583, CSCI 2120, and CSCI 2467 in the Fall semester of 2018, so we must first examine the course trends prior to this date. Given what we know of the nature of Autolab and its purported benefits (namely, the automation of course administration), we can safely rule out an examination of lecture changes from semester to semester. Autolab does not touch the lecture half of its courses, so it cannot affect them. What it does touch directly is graded programming assignments, namely labs (for those courses that have separate lab sections, namely CSCI 1583 having 1581 and CSCI 2120 having 2121).

The lab assignments of CSCI 1581 and CSCI 2120 prior to the Fall semester of 2018 were created by the same authors, so they share an overall similar structure. They all begin with a brief introductory paragraph, which serves as a quick abstract of the concepts being covered, followed by a lecture of sorts on the topics previously abstracted. Once all relevant concepts have been expounded upon, one or more exercises are listed in succession. These, too, share a common structure. Specifically, each exercise is built as a problem followed by a series of steps. Each step corresponds to a series of actions that bring the student closer to completion. Students follow along with the exercise, with the instructor dictating and demonstrating simultaneously, until the listed outcome is accomplished. For example, CSCI 2121's Lab 3 introduction states that the topic of choice is unit testing. Students are then shown the JUnit framework for unit testing Java code and walked through setting it up on their own machines and using JUnit commands in a terminal. An exercise then follows in the form of an instruction set, with code snippets and command examples, for creating a new class "String," creating unit tests for that class, and running the tests from the command line. The process is repeated for a custom class "Rectangle" and its unit tests. Finally, students are left with instructions to create their own unit tests for a "Fraction" class file completed in a previous lab along with a Summary detailing assignment submission. In short, the labs of the pre-Autolab era contain a few important commonalities: sections of conceptual instruction, step-by-step coding and command line walkthroughs, and a brief solo-exercise to practice the skills presented in the lab.

Each lab document contains a trove of pedagogical data. In fact, a great deal can be learned from the structure of the document alone. For instance, in the particular lab mentioned above (Lab 3 of CSCI 2121 prior to the Fall semester of 2018), of the roughly eight full pages that make up the student handout, four of them are used for introducing the topic of Unit Testing, most of the remaining four are used for step-by-step instruction, and exactly eleven lines are dedicated to the solo practice and submission sections at the very end. This structure indicates that the author intended for the labs to be used primarily as extra lecture sessions to supplement CSCI 1583 and CSCI 2120. The amount of exposition in the packet hints at the focus being on providing something of a hands-on tutorial experience for tools and concepts not covered in dedicated lecture or only partially covered (either way due, no doubt, to time constraints in lecture). By that same logic, the size of the solo practice and submission section in comparison would indicate that the unguided learning is of much less importance, possibly even an afterthought.

CSCI 2467 is structured much differently from the rest of the Intro Sequence. The final grade is split between one midterm exam, one final exam, and five large-scale lab assignments. On the surface, this is not a particularly unique curriculum in the sequence, each of the other courses has at least five major homework assignments over course of any given semester, with CSCI 1583 and CSCI 2120 adding separate weekly CSCI 1581 and CSCI 2121 lab sections on top of that. However, the difference lies not in the number of assignments, but in their sheer breadth of material.

Prior to Autolab, typical assignments for the rest of the sequence tended to be relatively straightforward single-problem projects (assignments for CSCI 1581 and CSCI 2121, in particular, were designed to be completed within the span of the class period in which they were introduced). The assignments of CSCI 2467, however, are designed with multiple stages meant to be completed in succession. The number and intensity of sections varies from one assignment to the next, but this trend of staged advancement holds true across them all. For example, the first assignment of the semester is always the "Intro Lab" where students are tasked with walking through three sections of introductory material. Students learn to navigate directory structures, create new files and directories, change permissions, move and copy existing files, and unpack compressed tarball files all from the terminal and all before Part 1 of the assignment. By the end of Part 3 of the assignment, students have learned how to edit files using in-terminal text editors like Nano or Vim, how to compile C programs using gcc, how exactly gcc compiles programs (by manually stepping through the process from pre-processing to linking), how to run C programs from the terminal, how to measure program speed with the "time" command, and how to use output redirects to save data. All of this happens in the first assignment, and the complexity only goes up with subsequent labs. This is not to say that there are no similarities between the CSCI 2467 labs and those of CSCI 1581 and CSCI 2121. For the Intro Lab specifically, there is a stark resemblance in the use of step-by-step directions for completing each part. This is intentional, as the goal here, as in the previously discussed assignments, is to instruct rather than to reinforce. However, this similarity is summarily absent in the other four labs of CSCI 2467. In the Data Lab, for instance, students are presented with a myriad of logic puzzles in the form of binary arithmetic. The number of puzzles varies a bit between semesters, but usually stands around thirty, each of which is given a difficulty rating from one to four (four being the most difficult). Students are given one "free" example walkthrough per lecture over the duration of the working period for the assignment (typically two weeks), adding up to about three or four walkthroughs. Otherwise, students are left to figure out the remaining puzzles alone or with the assistance of graduate assistant tutors. The fundamental shift in teaching style is blatantly apparent here. In the Intro Lab, students are led through to completion so that they can focus on the theories and technologies at play. With the introduction of Data Lab, that notion is gone, the lessons relegated to lecture alone, and in its place is the idea of reinforcement through repetition. There is an unspoken expectation, here, that students who have made it to this point in the degree path have crossed a threshold of sorts. These are no longer fledgling programming students stumbling about in the search for solutions; they are past the point of needing to be led by the hand through their assignments. If they can master this step, then they are prepared for the upper level courses lying in wait beyond the Intro Sequence.

The New

Having examined the labs from before the advent of Autolab, we must now make a similar investigation of the time since it was brought into the department. In the three semesters since its adoption, much of what we've seen above regarding the labs has changed.

In CSCI 1581, for instance, what was once a monolithic step-by-step instructional assignment has now become a practical gauntlet of sorts. There are nine total labs given over the course of the semester, and each of them contains an average of just over seven problems per lab packet. Each problem is self-contained, and worth a maximum of ten points (two points for successful compilation, three points for correct output using given sample inputs, and five points for utilization of additional custom inputs) and each lab has a score ceiling of forty points. For most of the labs, this means that students only have to complete roughly half of the packet in order to get full credit, the remaining problems can be completed as well, but only for bragging rights on the course scoreboard.

Examining the general lab structure, as above, and comparing them to those from before Autolab reveals a fundamental shift in the focus behind the labs themselves. Every one of them begins the exact same way, a small table indicating what the exact scope of the lab will be sits right above the eight-item submission procedure. The very next page is the very first problem. The problems themselves vary in size, seeming to grow and shrink with their complexity, but all follow a similar format: a brief preface to set the context, a list of new or uniquely important jargon, an explanation of expected inputs and outputs, and a final list of input and output samples. Overall, the problem design is concise without being vague, and the overall lab exudes a feeling of quickness. Going through the packet is like doing calisthenics, or getting a stretch in before a run, it feels like the kind of repetition that really nails down a good habit. If that feeling isn't enough of a motivator, since the score maximum eliminates the need to complete the entire packet, the addition of a scoreboard adds a certain level of friendly competition. Taking all of these things together, it feels as though the Autolab era of CSCI 1581 places extra weight on a deep practical understanding of the fundamentals of Java and of programming in general.

The labs for CSCI 2121 are another story. Though the handouts are new, and the submission guidelines have been updated for Autolab, the lab structure remains about the same as before. It is unclear as to whether this is intentional, if the material does not lend itself easily to autograding, or if there simply has not been enough time since the addition of Autolab for deep changes to be made to the labs here as well. Regardless, there is nothing new to examine here.

Likewise, the labs for CSCI 2467 have not seen much change with the arrival of Autolab. The difference here, however, is that this is most certainly intentional. When CSCI 2467 changed professorial hands in the Spring of 2016, the lab structure was changed as well. Specifically, the change came as a result of a simultaneous courses text change to the third edition of *Computer Systems: A Programmer's Perspective* by Randal E. Bryant and David R. O'Hallaron (the very same O'Hallaron behind Autolab), which was published in 2016 out of Carnegie Mellon University. As an accompaniment to the textbook, CMU made a series of labs available. These are the very labs that were adopted into CSCI 2467 in 2016, and are still in use today, two of

which were mentioned briefly above. The thing to take note of here, of course, is the connection between the Bryant and O'Hallaron text, the course labs, and Autolab: all three come from the same Systems course at Carnegie Mellon University. When the labs were adopted in 2016, they were already made to work with Autolab (which was created some years earlier in 2010). Once Autolab was incorporated into CSCI 2467, almost no changes had to be made to the labs.

These are the only three courses in the department to have adopted the use of Autolab as of the writing of this thesis, and each of them has experienced the change differently. Still, this data is not enough to determine the full scope of the impact that Autolab has had on the department. For that, more quantitative data is needed.

The Numbers

Having investigated the ways that adoptive courses have been changed to work with autograders, we are left with a general idea of the state of the courses before and after Autolab came to UNO. These specifics, though, are hardly the only things that matter to this investigation. As such, it is important to consider other metrics.

As stated before, Autolab was introduced into the undergraduate curriculum of the Computer Science department at UNO in the Fall of 2018. If we include the current semester, which is still in progress, that means that there is a total of three semesters worth of data regarding courses affected by Autolab. In contrast, we have the entirety of the history of the Computer Science department to examine what life was like before. Though, in reality, it is only useful to consider data from the last few years or so in order to ensure the preservation of variables such as curriculum content and course instructor assignment. It is important to eliminate as much noise from the data as possible, so we will restrict ourselves to examining data from as far back as the Fall semester of 2014, when the majority of the most recent labs prior to the Autolab overhaul of CSCI 1581 were last modified. This will allow a wide enough range of data to examine trends across semesters while also ensuring minimal administrative disruption to courses. The sole exception is the overhaul of CSCI 2467 in the Spring of 2016. In order to address this, no CSCI 2467 data from before that semester will be compared to similar data for CSCI 1581 and CSCI 2121 when it would interfere with said comparison. This will protect the integrity of any trends found in the data for the latter two courses.

Class Size

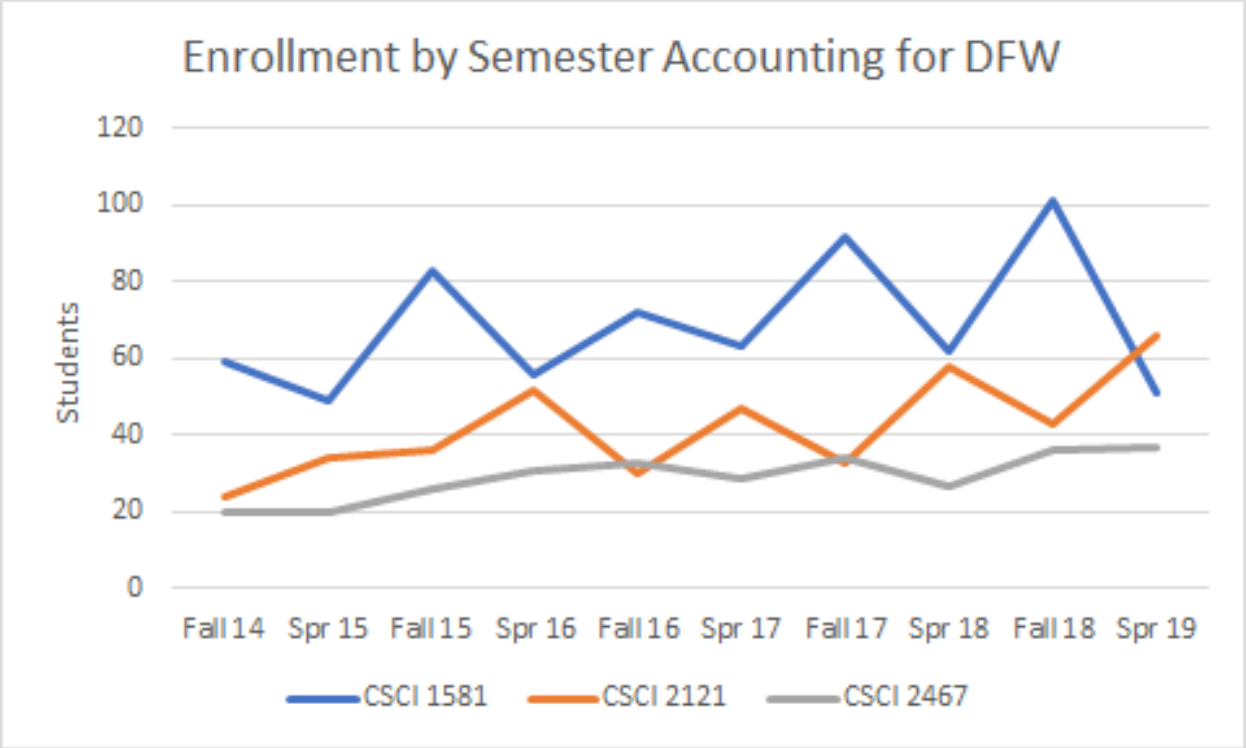

Figure 2: Enrollment by Semester Accounting for DFW

The most obvious first metric for examination is enrollment data. The chart above was created using departmental enrollment numbers for CSCI 1583, CSCI 2121, and CSCI 2467 accounting for drops and withdrawals after initial enrollment. CSCI 2467 has a fairly steady, linear progression for the length of the graph. There is little of interest to glean from this, so it is safe to ignore it. For the other courses, however, a more interesting relational pattern is immediately apparent. The complementary crests and troughs that CSCI 1583 and CSCI 2121 share from semester to semester represent the logical progression of students through the program. As students complete CSCI 1581, they enroll in CSCI 2121, so the latter class trails in the pattern by one semester. Looking closer still, though CSCI 2121 is the reactionary party in this relational pattern, it is not following the exact trends of CSCI 1581; for every semester on the graph save for one, the total census for CSCI 2121 is less than that of CSCI 1581 for the previous, complementary semester. This is due to the natural student attrition between semesters, but it is still important to note, as the difference is stable until the Fall of 2018 when it narrows significantly before reversing in the Spring of 2019. More data will be required to understand why the trend so suddenly changed, but it is worthwhile to note that the changes began the very semester that Autolab was introduced.

DFW Rates

When investigating the effects and efficacy of a new technology, like Autolab, it would be reasonable to assume for some waves to be created within those courses directly affected and

with subsequent courses in adjacent semesters. This is particularly true of those classes in the intro sequence, as they act as the "gateway" for the other courses. When one course passes a "successful" class of students, it would make sense to see the success rates of subsequent courses positively affected.

In order to investigate this supposition, attrition data was obtained from the Computer Science Department at UNO. For years, departmental officials have collected information regarding the Drop Failure Withdrawal (DFW) rates of every course within the department in every semester. Changes in this data from semester to semester are noted and are commonly used as a primary metric (along with noted trends over multiple semesters) for the success of a course over the previous semester and, by extension, whether the current form of that course is still valid or in need of updating. Thanks to this, a large amount of data for the Intro Sequence courses was gathered. This data was examined in order to test the earlier supposition regarding Autolab's potential to cause systemic, rippling change across courses. In the Intro Sequence DFW data, we find the expected ripples, but not exactly as expected.

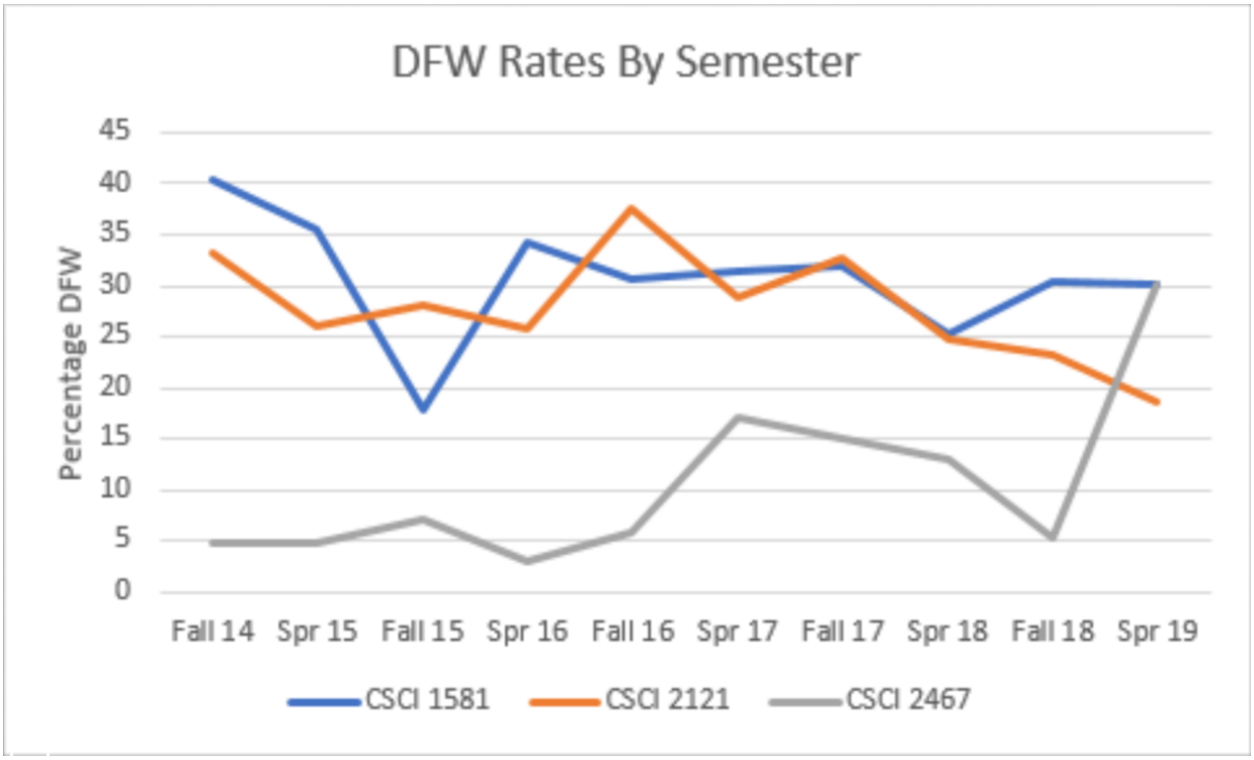

Figure 3: DFW Rates By Semester (DePano, 2019)

The most noticeable thing about this graph is the similarity of the relationship between CSCI 1583 and CSCI 2121 to the one apparent in the Enrollment By Semester graph. From the Fall of 2015, whenever DFW rates increase for CSCI 1583, the rates for CSCI 2121 in the following semester increase as well, with the same playing out for rate decreases. This trend continues until the Fall of 2017, where the courses synchronize for one semester before seemingly ending the trend. From the Spring of 2018 onward, as DFW rates have risen for CSCI 1581, they have fallen for CSCI 2121.

Looking at the graph around the time that Autolab was introduced, it is easy to see the kind of impact it had on CSCI 2467, but it is more difficult to tell what kinds of effects it may have had on the other courses. The largest changes in the relationship between CSCI 1583 and CSCI 2121 actually begin the semester before Autolab is introduced. If Autolab-driven course adjustments were implemented in the Fall semester of 2018, then we would expect to see some degree of change in the data from the previous semester. Fortunately, the data still supports that idea. Looking at the Spring of 2018, CSCI 1583 ended the semester with a DFW rate of 25.30% (meaning that roughly 25% of the enrolled students either dropped, failed, or withdrew from the course), CSCI 2120 ended with 24.36%, and CSCI 2467 ended with 12.90%. This gives us the last set of baseline percentages before Autolab was introduced to the curricula of these courses.

In the very next semester, Fall of 2018, CSCI 1583 ended with a DFW rate of 30.34%, CSCI 2120 ended with 24.53%, and 2467 ended with 5.26%. When compared to the previous semester, we see upfront that CSCI 1583 suffered a DFW rate increase of ~5%, while CSCI 2120 remained roughly unchanged, and CSCI 2467 actually decreased by ~8%. On the surface, this would seem to indicate that Autolab's initial impact was almost net neutral (with only a slightly positive impact on the three courses together). However, we must take all the numbers into consideration to get the full picture. Specifically, the changes in DFW rates must be studied alongside the changes in enrollment numbers between the semesters. Between Spring and Fall of 2018, there was a very dramatic increase in the number of enrolled students in CSCI 1583 and its accompanying lab section, CSCI 1581. In fact, the census for those courses almost doubled. This must be taken into consideration along with the fundamental changes to the lab assignment structure. Considering that the total number of solo assignments per lab increased from one-or-two per lab to four-to-seven per lab, one might say that the difficulty of CSCI 1581 was increased with the changes. Based on this, it would not have been beyond the realm of expectation for the DFW rate for Fall of 2018 to be much higher. Since actual rate change is relatively low, despite the census boom, we can assume that the lab changes had a net positive effect.

For the Spring semester of 2019, what immediately jumps out is the seasonally reduced enrollment census for CSCI 1583 (cut nearly in half). Next, we find that the DFW rates have gone through an interesting reversal. For CSCI 1583, the rate remains stable, for CSCI 2120 we see a modest decrease of ~6%, and for CSCI 2467, there is an uncharacteristically large spike of ~25%. The reason for this is unclear from the data alone. The knee-jerk explanation is that the Autolab changes affected the course adversely. Sadly, there is only data from the previous semester with which to compare, and as seen above, it shows us that the DFW rate of CSCI 2467 was roughly 8% lower than in the semester before that. Otherwise, we are left with only conjecture. The original supposition above was that it would be logical to expect that the Autolab changes may be profound enough to ripple out to neighboring courses, but this ignores the case where the reverse is true: Courses not directly affected by Autolab may have an impact as well. There is no clear evidence to support this supposition, however.

Overall, the DFW rates provide an interesting look into trends from semester to semester, but they do not show any particularly large waves made by Autolab, aside from the spike in

CSCI 2467. Furthermore, it is important to note exactly what DFW numbers represent for a course. Generally speaking, the numbers are a broad representation of student outcomes in a particular course. What this means is that they only take into consideration whether a student dropped, withdrew, or failed the course in question. Other important metrics like grades and attendance are not considered, so the resulting rates are truly an indicator of the number of students who "did not fail," instead of the number who "did pass." This is problematic for several reasons, but its breadth of scope is what really causes trouble for this investigation. These numbers must be tempered with supplementary data to eliminate the noise and extract its true value.

Number of Assignments

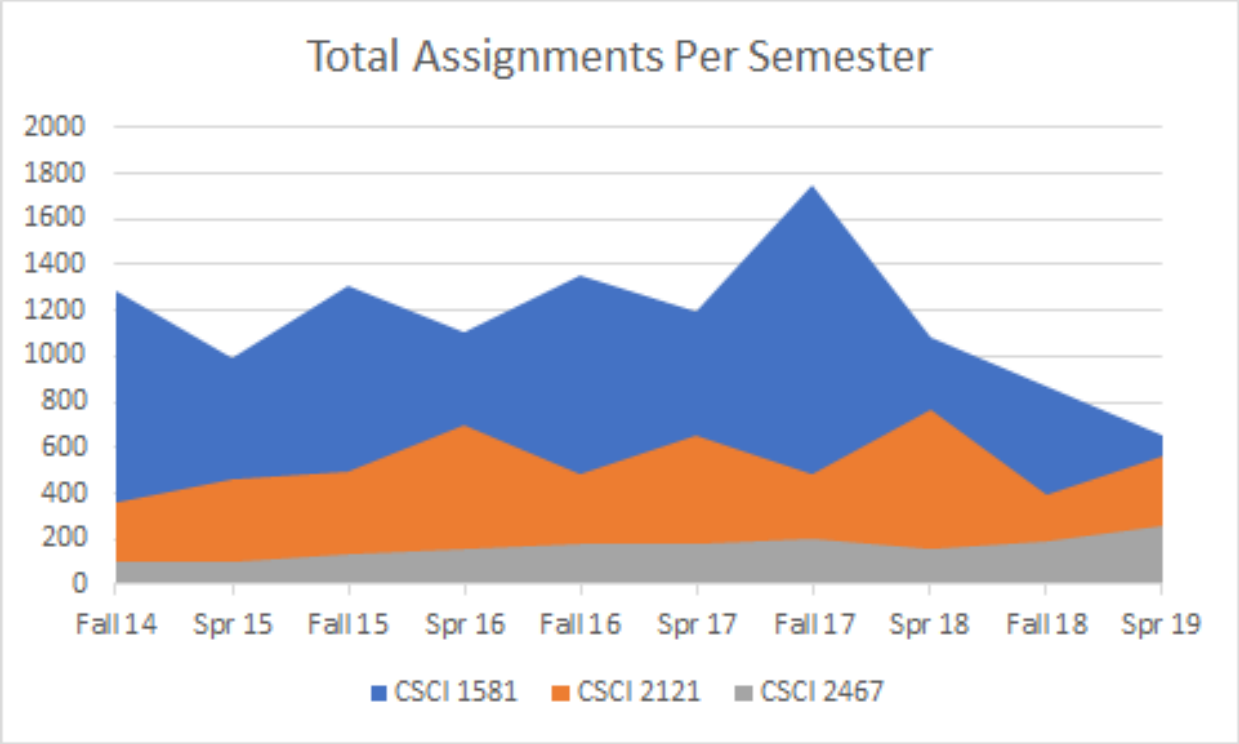

Figure 4: Total Assignments Per Semester (Holmberg, 2019)

One such supplemental metric is a count of the number of assignments submitted per course per semester. For every student enrolled in a course, a certain minimum number of assignment submissions is expected for the semester. The above chart illustrates this data. The result is a picture that greatly resembles the Enrollment By Semester chart. Once again, CSCI 2467 can be ignored, owing to its mostly-linear progression caused by a steady census between semesters and an invariable number of assignments per semester (data prior to Spring of 2016 is excluded from this chart due to the course changes enacted that semester). What is left is the same relationship between the remaining courses from before with some slight differences. First, the lines are farther apart here than in the earlier graph. This magnified separation is due to the multiplication of CSCI 1581's larger enrollment numbers by its simultaneously larger number of

assignments. This effect is muted started in the Fall of 2018, when the total number of assignments for CSCI 1581 experiences a sharp decline. Since the lab section was completely remodeled before the start of the semester, the number of assignments was changed. The difference is a noted downturn all the way through the Spring of 2019 despite an increase in the number of assignments in that semester, though the effect on the graph is somewhat exaggerated owing to the large difference in enrollment numbers between the two semesters. Still, despite the changes, this is an important graph for understanding the sheer volume of submission data that is handled over the course of a single semester. Considering that professors were responsible for grading upwards of two thousand assignments in four months before the adoption of Autolab highlights both the monumental task of grading and the size of the time saving opportunity that autograding programs like Autolab bring to the table. Unfortunately, the relationships presented here, much like the earlier graph, still tell precious little regarding and pronounced effects that can be attributed to Autolab. There is still too much noise in this graph, so a more specific filter metric must be used.

Number of Graded Programs

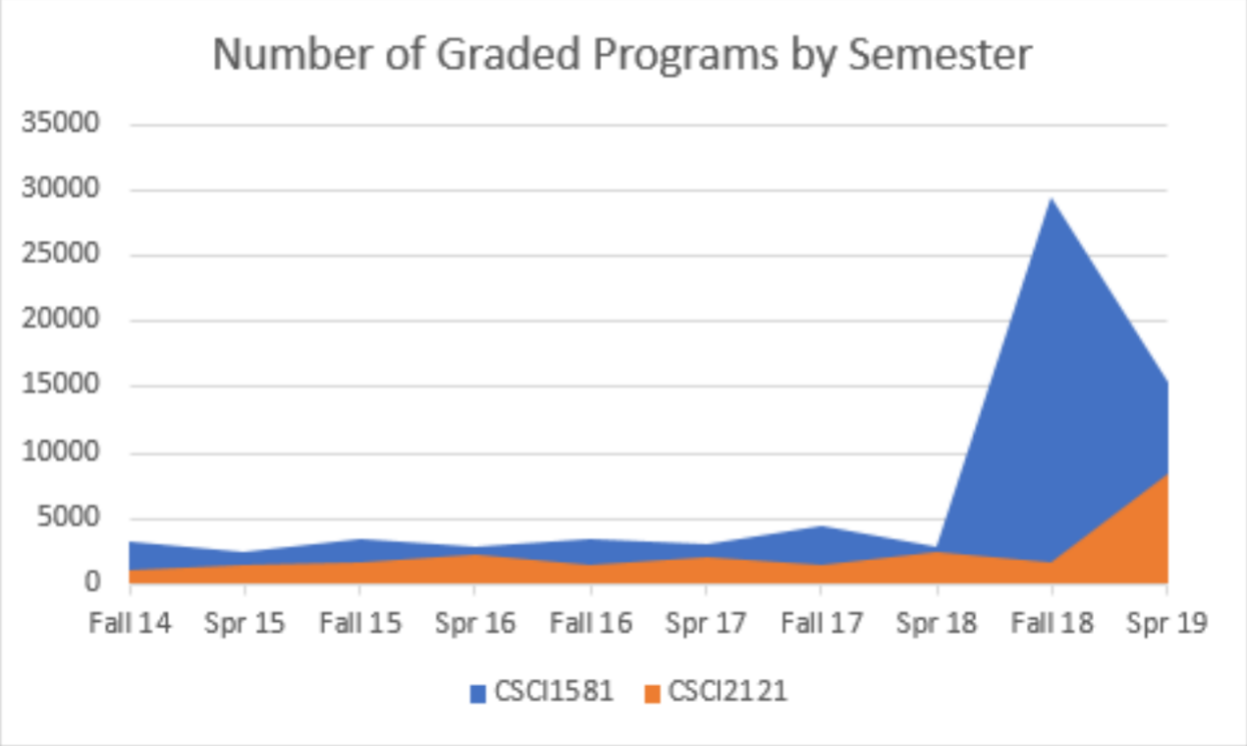

Figure 5: Number of Graded Programs By Semester (Holmberg, 2019)

The above graph represents a more focused version of the previous graph. It can be divided into two parts based on how the portrayed data was produced. The first part ranges from the beginning of the timeline to the Fall of 2018 and was obtained by dissecting one semester's worth of assignments from pre-Autolab CSCI 1583 and CSCI 2121, counting the number of individual programs that are completed with each submission, and multiplying those numbers by the enrollment census for each semester (relying on the single-submission rule of the old labs). The second part of the graph shows similar data produced instead by utilizing submission data

pulled from Autolab's logs for the Fall of 2018 and Spring of 2019. Since Autolab keeps meticulous records of every submission by default, no inferences nor extra manipulation of the data was required to fit the rest of the data set. What results is a striking new set of relational data.

The first thing of note here is that CSCI 2467 has been excluded from the graph. This is due to its unchanged structure after the adoption of Autolab. If it were included, the data would offer no new insights and would get in the way of the rest of the representation. What is shown is a display of the minimum expected lump sum of programs submitted in any given semester, the inspection of which reveals yet another representation of the now familiar relationship between CSCI 1583 and CSCI 2121 with a very noted difference beginning in the Fall of 2018. Prior to this point, the number of graded programs per semester was on average thirty-three per student for CSCI 1581 and thirty-one per student for CSCI 2121. Those numbers appear to have remained stable for a number of years, which explains the mirroring of the relationship between CSCI 1581 and CSCI 2121 witnessed in previous graphs, up to the Fall of 2018. At that point, the average number of programs per student increased to just over two hundred and three for CSCI 1581 alone. In the Spring of 2019, that number increased yet again for CSCI 1581 to just over two hundred nine programs per student (due to an increase in the overall number of labs) while CSCI 2121 shows a first-time jump to one hundred four programs per student.

This data is much more revealing than the previous sets. In the case of CSCI 1581, there is a stark illustration of the fundamental shift in pedagogical strategy that was discussed in the earlier examination of the lab write ups. The new focus on iterative learning is clearly defined, and that makes its implications clear as well. This is particularly interesting given how this segment of the graph matches up with its counterpart in the previous graph, where the number of assignments per student per semester for CSCI 1581 takes a sharp drop. If students are submitting this many completed programs over the course of a single semester, then it stands to reason that they must surely be receiving largely increased and intimate exposure to Java, making long strides in confidence and fundamental comprehension. The only problem with this assumption lies in yet another quirk of the data. This time it lies with the Autolab logs used to calculate the last two semesters of data, specifically with the way that Autolab logs every single submission with equal weight. Since the courses in question each allow for unlimited assignment submissions prior to deadlines, a significant number of the logs are likely to show neighboring submissions that are very similar, if not exactly the same. Autolab works by returning feedback to students after every submission, in this way the self-testing of code is disincentivized. If there are no limits to the number of submissions allowed, then students are far more likely to utilize Autolab's automatic feedback than they are to try to generate their own. This "throw code at the wall and see what sticks" approach pollutes the dataset, leaving it less valuable than originally believed. Still, bloated as the numbers may be, the fact remains that students are writing more distinct programs under the new lab structure than they have in previous semesters. This caveat does inspire a bit more confidence, but it does not completely alleviate the problems with this data set; even more specific data is needed.

Tracking Students

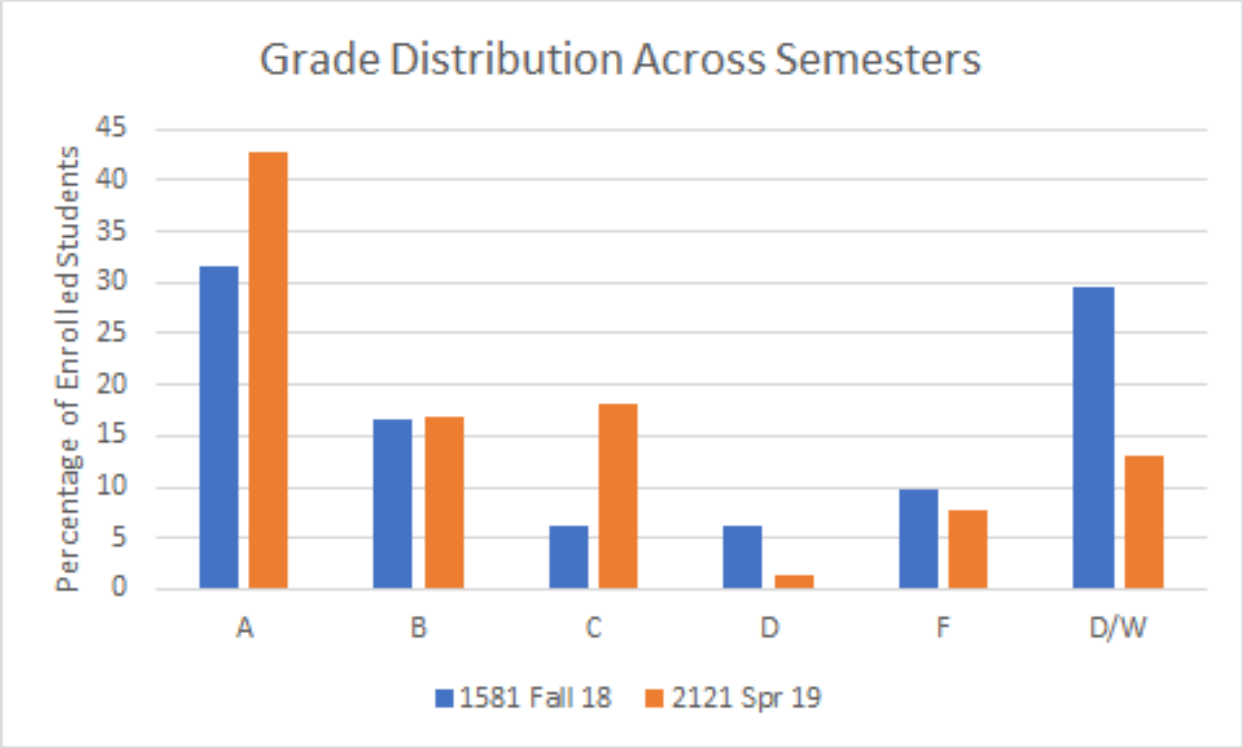

Figure 6: Grade Distribution Across Semesters

The image above represents another unique perspective of the Autolab courses. In this graph, data representing the grade distribution for all students enrolled in CSCI 1581 in the Fall of 2018 is imposed next to similar data for the Spring 2019 CSCI 2121, with a section for Drops and Withdrawals included as well. It is a common grade spread comparison on the surface, but the data set becomes truly interesting when you consider that it is also essentially a diagram showing the progress of one group of students over two subsequent semesters. Since CSCI 1583 is a direct prerequisite for CSCI 2121, students often schedule them back to back. The result is a smooth transition through a consistent learning environment for students, while researchers gain a new understanding of how these students are moving through the system. This is particularly helpful in semesters where broad changes are made to courses. Much like the changes brought with Autolab.

In other words, since it is a chart of results, the graph above tells the story of the entire semester. For example, the Fall 2018 class of CSCI 1583 seems to have finished the semester with an almost equal number of A's and Drops/Withdrawals. That latter metric comes out to a rather large washout rate of about thirty percent for the semester, but this is easily explained by the course being a the very first in the Computer Science undergrad track. No experience is expected when entering, so students are often encouraged to try it out. This low barrier for entry is simultaneously a low barrier for egress, as students have not invested much time into the Computer Science program up to this point. The result is that some students are quick to drop the course when they grow disinterested or begin to experience difficulties with the coursework. For the remainder of the spread, the students appear to have done quite well with a combined B and

C percentage of about twenty-four and a failure rate (combined D's and F's) of around sixteen percent. This would seem to suggest that many of the students who remained after that semester's drop date took quite well to the newly-designed labs.

Next, the data from the following semester, Spring of 2019, shows an interesting shift. The students who successfully passed CSCI 1581 in the previous semester seem to have gained quite a solid foundation. The graph shows a DFW rate of twenty-two percent, with failing grades accounting for a remarkably low nine percent portion of that whole. It appears that either the steep DFW rate of the previous semester is truly representative of a case of weeding out unprepared or uncommitted students, or the gauntlet of assignments presented to those students who stayed did quite a good job of preparing them for the new course. One way or the other, an overall passing rate of around seventy-six percent is quite remarkable.

Of course, there are a number of other possible explanations for these rates. It could be the case that the course content for CSCI 2121 (freshly introduced in the Spring of 2019) was of a lower difficulty than that of CSCI 1581, or that differences in section or professor lead to an overall easier grading experience. Without a more nuanced investigation of course records, it is impossible to determine if any of this is truly the case.

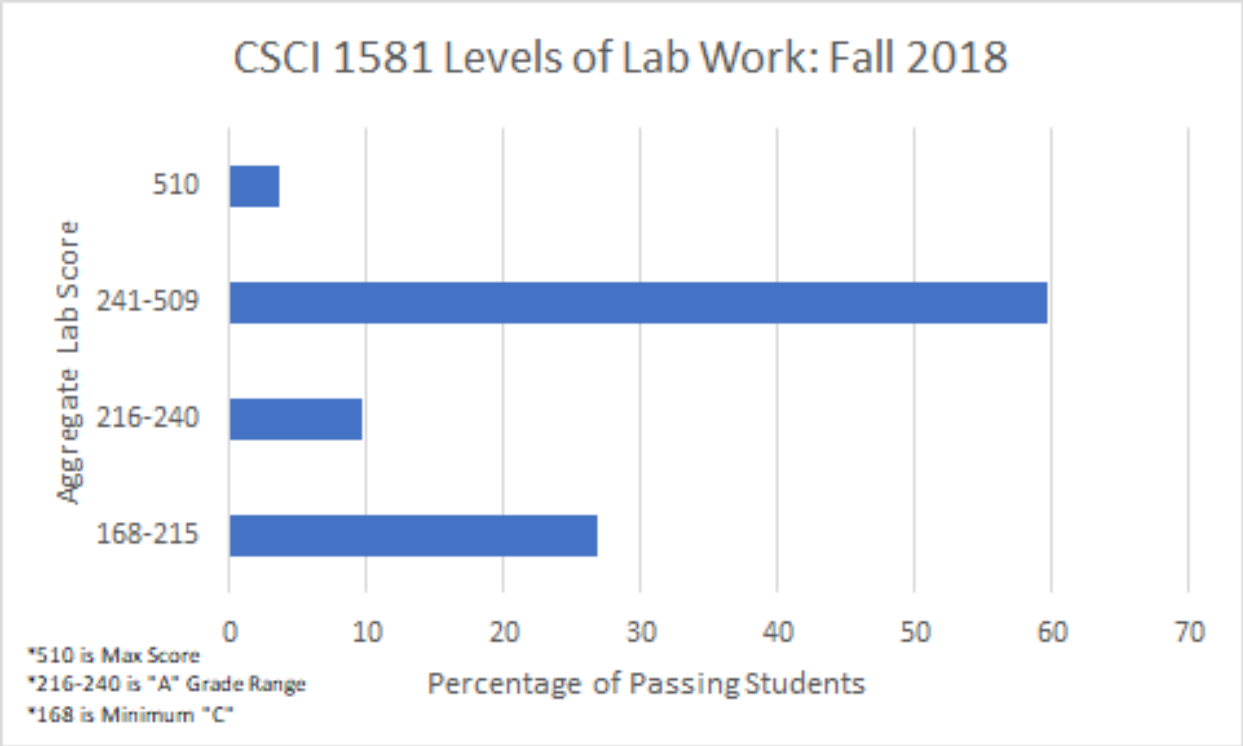

Figure 7: CSCI 1581 Levels of Lab Work: Fall 2018

Quantitative measurements of course outcomes and workload changes are very important datasets for an investigation such as thus, but it must also be mentioned that to utilize these data alone is to forgo a number of very important qualitative considerations. It is important to also consider student reactions to the changes in question. Positive indicators, such as whether the students demonstrate an increased level of involvement or interest in the course material, provide

a more nuanced glimpse of the effects of new changes. To that end, consider the graph above. It contains data representing roughly how much work was done by members of the passing cohort of CACI 1581 students for the given semester, the Fall of 2018. The graph is a bit complicated to parse, but the idea behind the data has to do with the structure of the CSCI 1581 Autolab labs. As stated before, the overhauled labs are made up of a series of four to six programming challenges, each with its own focus, nuance, and point value. The labs have varying levels of maximum assignment score, but they all share the same grading rule that restricts recorded scores to a maximum value of forty points. Since there were six total assignments in the Fall of 2018, the maximum possible grade that semester was an aggregate score of two hundred forty points. This meant that students only had to complete four assignments per lab packet, out of the six or more present, to get an A in the class. This system serves the dual purpose of providing students with a plethora of practice opportunities for those who wish to do extra work, and a variety of problem types from which students can choose their requisite four problems to obtain an A. Completing all of the problems is completely optional, however, sticking to the bare minimum means leaving a significant amount of practice material untouched, and a significant number of points unclaimed, so students are encouraged to think carefully about the benefits of each option. The extra points may not be worth anything in a gradebook, but they do have a value of their own in the form of bragging rights. Students are granted a perfect opportunity to show off their programming skill, with Autolab's built-in scoreboard capabilities providing the perfect backdrop. All participants are given the option of selecting a custom anonymous screen name, which often leads to hilarious outcomes and the occasional friendly competitive ribbing. Still, some students may not be completely comfortable in a competitive environment, even such a low-stakes example, and that may be reason enough to avoid the scoreboard altogether. One way or the other, all these considerations come into play when deciphering this data.

As previously stated, the Fall semester of 2018 version of CSCI 1581 consisted of a total of six labs with a maximum grade of two hundred forty points, though the maximum aggregate lab score was five hundred ten points. In the vertical axis of the previous graph, we see a series of value ranges representing consecutive groups of aggregate lab scores. Each of the bars represents the percentage of students, from the total number of passing students for the course, who achieved that level of completion by the end of the semester. Based on this, the graph shows that three students from this group managed to end the semester with the maximum aggregate score of five hundred ten points, while forty-nine other students completed at least some extracurricular practice, eight more students completed the number of problems required to get an "A" in the course, and another twenty-two students completed enough to finish below an "A," but at or above the minimum passing grade for the course. Looking at all of the numbers, it appears that a very strong majority (roughly sixty-three percent) of the cohort of passing students for the semester worked above and beyond the two-hundred-and-forty-point grade maximum. Another ten percent of the group stayed in the "A" range of two hundred sixteen to two hundred forty points, and the final twenty-seven percent worked enough to get a passing grade, but not an "A." These numbers tell the best story so far regarding student reception to the Autolab changes and the iterative learning strategy. With previous graphs, the information was always too broad or left too many variables unaccounted for. With this, one can see firsthand how motivated students take advantage of the opportunities and technologies that are presented to them.

The following graph takes this notion and applies it to the very next semester. In the Spring of 2019, changes were made to the course content. Specifically, the number of labs was increased from six to nine. As such, the aggregate score numbers are different, as well. Six hundred sixty is the new max score, while three hundred fifteen to three hundred fifty is the new "A" range, and two hundred forty-five is the new minimum passing grade. Taking a closer look at the graph, those don't appear to be the only changes in the data. Firstly, no students were able to meet the maximum score. Actually, where over sixty percent of the students of the previous semester appear to have taken full advantage of the "bonus" lab problems, here only thirty-six percent of the students of Spring seem interested in working beyond an "A." In fact, almost as many people in this cohort chose not to opt for any points beyond a "B" or "C."

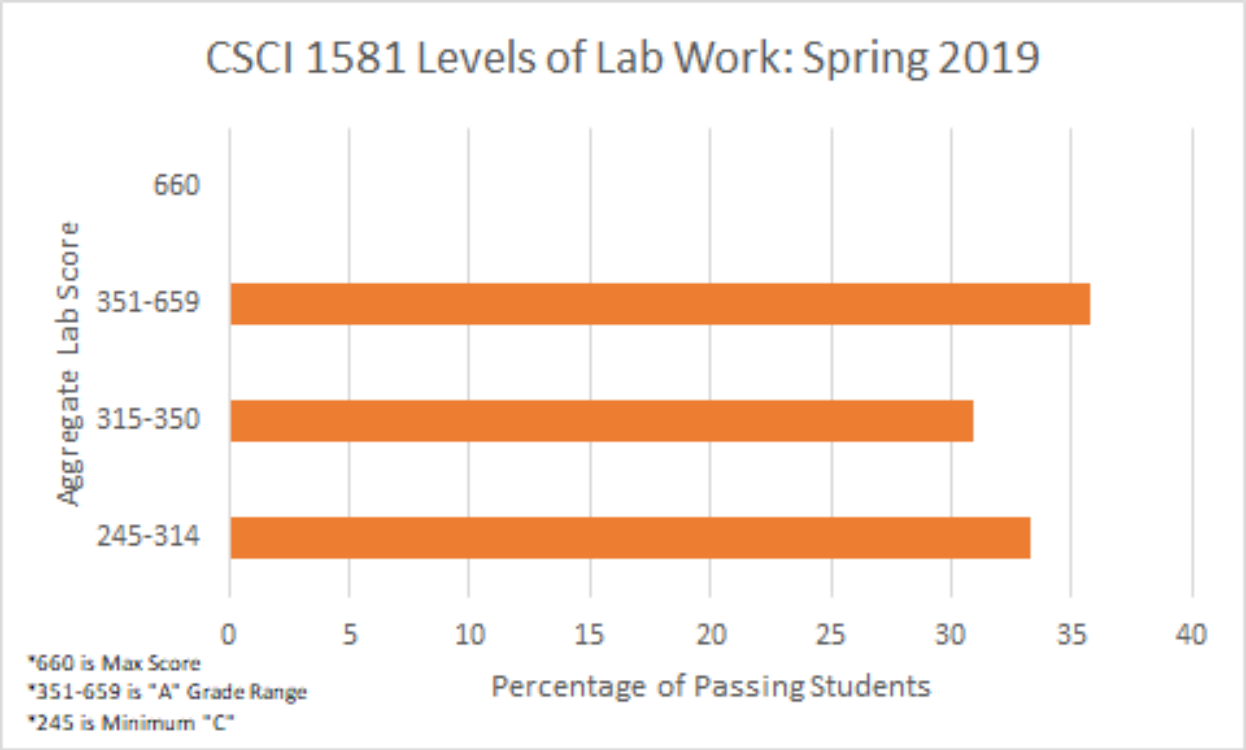

Figure 8: CSCI 1581 Levels of Lab Work: Spring 2019

Perhaps the differences between these graphs can be explained as byproducts of the continuing changes to the underlying lab materials. The increase in the overall number of labs may have created too much of a hurdle for students with full course loads to overcome, or the bonus workload may have ballooned to the point where students are too intimidated to try to keep up with it. Another important caveat to remember is the differences in enrollment numbers between semesters. In the earlier Enrollment graph, there is a clear cycle involving the ebb and flow of student enrollment between Fall and Spring semesters. Specifically, in the Fall of 2018, CSCI 1581 had an enrollment of one hundred forty-five students, which decreased to seventy-two in the Spring. Perhaps this lower student population plays a role as well. Also, it is worthwhile to consider how the structure of Autolab has affected these numbers. Has the unlimited submission policy promoted the same scattershot submission strategy that was previously discussed? Perhaps the change in the number of assignments was accompanied by a

corresponding change in the quality of feedback automatically generated by Autolab. If an autograder returns subpar or unclear feedback with each submission, then the speed benefits of autograding are severely diminished as students are left to discover coding problems on their own. Whatever the case, it is clear that there was an overwhelmingly positive reaction to Autolab assignments in the Fall semester. However, more evidence will be required to determine if this is the start of a new trend. In the end, what this data shows is the change in enthusiasm levels from semester to semester. The causes behind such a subjective state of mind are quite difficult to gauge from quantitative data alone. Still, there is quite a lot of value to be gleaned from even a surface-level reading such as this. If it can be found that there is some level of a trend forming regarding student engagement in Autolab from semester to semester, then instructors will be left with a new starting point from which to investigate possible improvements to student outcomes. It will become that much easier for the department to make more informed decisions regarding possible positive changes to this course and others. It is in this way that thoughtful decisions can be reliably made to promote a positive future for the department. This is paramount to ensuring that the program remains fresh and relevant to future generations of students, while promising an enriching and exciting environment where these students can learn and grow.

CONCLUSIONS

Autolab is a course management software suite sporting a bevy of features aimed at decreasing the workload for teachers, improving student outcomes, and creating a fulfilling experience for everyone involved. It gives course administrators the ability to rethink and redesign their courses using the very types of technology they lecture about. When it arrived at the University of New Orleans' Computer Science Department, Autolab landed in the place where it had the most potential to make an impact, the Introductory Sequence, the chain of courses that serve as every new student's first exposure to the world of computer programming at UNO. Of the three courses that incorporated Autolab, CSCI 2120 and CSCI 2467 kept to their original course designs and philosophies while CSCI 1583 made a fundamental shift to an iterative learning pedagogy in order to take advantage of the opportunity to utilize the full extent of what Autolab has to offer.

Now, two semesters since its adoption, some of its effects on those courses can begin to be examined. Using the enrollment numbers, the Drop Failure Withdrawal (DFW) rates, the total number of assignments, the number of programs graded per student, and the grade distribution data for each of the affected courses, along with the aggregate lab score data for students of CSCI 1581, a broad preliminary depiction of Autolab's effects on the department can be created. Looking at the data for one semester before the adoption of Autolab (Spring 2018), the semester it was adopted (Fall 2018), and one semester after (Spring 2019), we find that the relative DFW rate of CSCI 1583 experienced a net increase of about 5%, CSCI 2120 went through a net drop of roughly 6%, and CSCI 2467 underwent a net increase of roughly 12%. Overall, this equals to a roughly 11% increase in failure rate across all three courses over three semesters. However, these metrics alone do not reveal a great deal alone.

Looking at the changes in course structure, particularly the fluctuations in total assignments and number of graded programs per student per semester, helps to show how the addition of time saving technologies has allowed professors to embrace a new level of creativity in course design. Specifically, the multiplicative increase in number of programs per assignment brought about by the shift in priorities from lecture-based learning to iterative-learning for CSCI 1581 is something that would have been very difficult to handle if the instructor had to hand grade each of the hundreds of small programs submitted by students. Autograding allows for the instructor to focus on the minutia of instruction rather than the full extent of the drudgery of grading. Looking into the statistics following these changes reveals that this boon was taken full advantage of right from the beginning. Though the total number of labs per student per semester decreased sharply in the Fall of 2018 for CSCI 1581, the number of individual programs graded per student rose by nearly an order of ten. This type of workload would not be possible were it not for some level of autograding ability. Still, it is one thing to notice the course changes inspired by the software, and another entirely to examine how those changes affected students.

Looking into the grade data for these changed labs gives us a peek into the kinds of student trends that have emerged in CSCI 1581. Examining the aggregate lab scores of all the students who passed for the Fall of 2018 and accounting for the lab rules regarding maximum

grade scores versus maximum lab scores reveals a remarkable motivation in the students. Though they only needed to score an aggregate two hundred forty points to get the highest A in the class, a solid sixty-three percent of the group opted to complete supplementary problems with point values maxing out at five hundred ten. Whether this surge in extracurricular work ethic is a product of the course structure or a trait of this particular class of students is debatable. Without a larger pool of submission data, it is impossible to say definitively whether this is a burgeoning trend. Especially, since the proceeding semester showed a more balanced motivation level across students (with almost the same percentage settling for a sub-A passing score as striving for the maximum), but this can be attributed to yet more changes in the structure of the lab section and the seasonal drop in enrollment numbers. Still, taking this information with the rest of the data for CSCI 1581, and a moderate helping of salt, creates the outline of a rather compelling argument for a positive impact.

In conclusion, whether these numbers are indicative of the overall success or failure of the Autolab changes is hard to say. The technology is still very much in its infancy at the University of New Orleans, so there is a noticeable dearth of information upon which to base a claim. It will be some time before enough semesters have passed to generate really significant trend data, but the current outlook is positive overall. In our investigation, we have studied one course that was always intended to work with Autolab, and so has remained steady and consistent despite the new addition. Another has remained in its pre-Autolab state, choosing not to reinvent itself just yet, instead opting to stay the pedagogical course and serving as an important litmus test for the changes brought to other courses. The final subject has completely reinvented itself around Autolab's core features, and as such has become the focus of this investigation. Through the data presented, CSCI 1581 has exuded a sense of optimistic growth despite the air of obscurity that hangs over its long-term prospects. For now, it is enough to say that Autolab has not been a detriment to the department at this time, and in fact shows the potential to fundamentally change the way that the program is structured. Only time will tell if this turns out to be the case.

REFERENCES

Canberk, I. (2015, April 3). *Making Autolab's Backend Scalable.* Retrieved from Autolab Development Blog: https://autolab.github.io/2015/04/making-backend-scalable/

DePano, A. (2019, September 01). Study of Enrollment and Attrition Rates for Computer Science Students at the University of New Orleans from 2014 to Present. *Unpublished Raw Data*.

Hollingsworth, J. (1960). Automatic graders for programming classes. *Communications of the ACM*, 529-529.

Holmberg, E. A. (2019, June 06). Study of CSCI 1581/2121 Labs Before and Since the Integration of Autolab into the Computer Science Department of the University of New Orleans. *Unpublished Raw Data*.

Pandya, M. (2015, May 25). *Autolab and Docker.* Retrieved from Autolab Development Blog: https://autolab.github.io/2015/05/autolab-and-docker/

Zimmerman, J. (2015, March 26). *Autolab: Autograding for All.* Retrieved from Autolab Development Blog: https://autolab.github.io/2015/03/autolab-autograding-for-all/

VITA

The author was born in Baton Rouge, Louisiana. He obtained his Bachelor's degree in English from Louisiana State University in 2012. He joined the University of New Orleans Computer Science undergraduate program in 2017 and transferred to the graduate program to pursue a Master's degree in 2018. He became a Graduate Assistance under Professor Matthew Toups in 2017.