Engagement in Practice: The Development of and Lessons Learned from a Community-Focused App Development Course

Ms. Jessica N. Jones, University of Florida

Jessica N. Jones is a Ph.D. student at the University of Florida studying Human Centered Computing in the Department of Computer and Information Sciences and Engineering. She received her B.S. in Computer Science from Hampton University in 2011 and her Master's Degree in Computer Science from Clemson University in 2014. Her research interests include educational technologies, robotics and natural interaction.

Ms. Tiffanie R. Smith, University of Florida

Tiffanie R. Smith is a Ph.D. student at the University of Florida studying Human Centered Computing in the Department of Computer and Information Sciences and Engineering. She received her B.S. in Computer Engineering from North Carolina A&T State University in 2013. Her research interests include educational technologies, culturally relevant education, and broadening minority participation in STEM.

Naja A. Mack, University of Florida

Naja A. Mack is is a Ph.D. student at the University of Florida studying Human Centered Computing in the Department of Computer and Information Sciences and Engineering. She received her B.S. in Computer Engineering from Claflin University in 2010 and her Master's Degree in Computer Science from Clemson University in 2013. Her research interests include educational technologies and natural language processing.

Imani Sherman, University of Florida
Dr. Juan E. Gilbert, University of Florida

Juan E. Gilbert is the Andrew Banks Family Preeminence Endowed Professor & Chairman of the Computer & Information Science & Engineering Department in the Herbert Wertheim College of Engineering at the University of Florida where he leads the Human-Experience Research Lab.

Engagement in Practice: The Development of and Lessons Learned from a Community Focused Mobile Application Development Course

Abstract

As the number of jobs for those with development skills grows, the potential pool of applicants remains inadequate. Although many entry-level software developer positions require a Bachelor's degree, the shortage of workers has encouraged employers to consider hiring and training those with experience and knowledge gained from development bootcamps. This paper, submitted as an Engagement in Practice Paper to the Community Engagement Division describes a first attempt at one such bootcamp. This article describes the partnership, development, design, and outcome of a 12-week community based iOS app development course taught in a partnership between the Computer Science department at University of Florida (the University), a state sponsored workforce development program, and a local technology training company. It details the generalized struggles and successes of the students, the lessons learned, and a second curriculum and class structure based on those findings. Finally it presents unanswered questions and presents recommendations for future courses presented by University/community/business partnerships.

1 Introduction

According to the Bureau of Labor Statistics, between 2014 and 2024, the job market for Software Developers will grow by 17%[1] which is "much faster than average". In Florida, Application and System Software Developers will grow by approximately 31% and 24%, respectively[2]. These rates are 1.8 and 1.3 times the national projection. In order to fill these job openings, recruiters are looking beyond students with traditional educations. According to a 2015 article[3],

> in certain cases, it does not even matter whether a candidate has a bachelor's degree in a specific area: companies are looking for candidates with hands-on experience in software development through "hack-a-thons," extracurricular projects, and internships.

The purpose of this article is to describe an attempt to provide non-traditional students with this sort of hands-on knowledge and experience via a 12-week programming course provided by a University and community partnership. We describe the partnership and the resulting course including several problems that were encountered and lessons learned. We then describe a course that was planned based on these lessons. Finally, we present several lingering issues and problems that still must be addressed.

2 The Partnership

The courses described here were created in a partnership between the University, a local for-profit software development school, and a state funded job training and placement program. This group will hereafter be referred to as ”the partnership”. The University provided an experienced graduate student to teach the course. The development school provided project management. They also leveraged connections with the local chamber of commerce and technical companies in the area to secure speakers and guest lecturers for the course. Finally, the career training center recruited students, provided connections to scholarships, and housed the class sessions.

Three main facts spurred the creation of this partnership. First, the mission of the University states that it should serve the state and community in which it is located by using its knowledge bases and areas of expertise. Second, the city in which the University is located is seeing job growth in the mobile development arena; companies are coming to town and several of them are looking for individuals who have mobile development skills. Finally, when these companies reach out to the local career placement center, the center is unable to recommend any of their participants. With this in mind, representatives from each group came together to determine the best way to provide permanent members of the community, as opposed to college students, with the skills necessary to apply for these new technical positions.

3 First Course: iOS Development with Swift

3.1 Background and Course Logistics

Because many of the new job postings were seeking iOS developers specifically, the team decided that it would be best to proceed with an iOS development course whose focus was creating apps using the new Swift programming language. A graduate student with experience teaching Swift was recruited to teach the course. In order to recruit students for the course, the team advertised to the local community via flyers and emails. In order to brief the community on the course, an information session where they could interact with the instructor and receive information about financial aid was held at the local chamber of commerce.

The cost of the course was $7,000, an amount close to the costs of courses offered by the software development school. These funds went to provide funding for the graduate student instructor and for the assistance of the for-profit software development school. Students who could not afford to pay for the course out-of-pocket were encouraged to apply for a state funded scholarship whose aim is to help prepare un- and underemployed residents from a particular region for STEM careers. Before enrolling, all students were informed that they would need to earn 70% or better to pass the course and receive a certificate of completion.

The course was held in a large, open room at the career placement center in the downtown section of The City. Due to the configuration of the space and the need to be connected to a standalone projector, the instructor sat and projected from the rear of the room. Students sat at tables in front of the instructor, facing a wall on which the instructor's computer screen was projected. In front of every student was an iMac or a Mac Mini with an accompanying monitor, mouse, and keyboard.

3.2 Student Demographics

When the class began, 12 students were registered. Of the 12 students, six were of African descent, five were White/Caucasian and one was of Southeast Asian descent. The course contained five women and four people over the age of 40. Six of the students entered the course with at least one year of Computer Science experience. One had an Engineering degree, one was a web designer, two who had taken college level CS classes and one learned to program in the 1960s. All students were un- or underemployed and 11 were scholarship recipients. Only one of the students was a veteran.

3.3 Course Structure and Content

iOS Development with Swift was taught for 12 weeks from 6 to 9 PM over a 13 week period, accommodating for the instructor's Spring Break. The course was held for 4 days per week. Every other week, the course would rotate schedules. On even weeks, three of the four days would be used for iOS lectures and assignments. The fourth day was used for technical presentations, supplemental iOS and Swift lectures, and guest speakers arranged by the the software development company. On odd weeks, two days were reserved for iOS lectures and workdays while the others were used for guest lectures. The vast majority of the guest lectures were provided by employees of technical companies from the area.

As a hybrid lecture/hands-on programming course, students were introduced to concepts while writing code and building apps in class. During the first week of class, students were introduced to logical and sequential thinking by writing the instructions to completing everyday tasks. They were also introduced to iteration, branching, and pseudo code by writing the rules to their favorite childhood games. In the second week, students were given an introduction to Computer Science concepts using Swift. These lessons included an introduction to Swift syntax, variables, functions, collections, branching, iteration, and debugging. Starting in week three, students were introduced to Xcode and app creation. Concepts were introduced as the class built apps. The concepts covered in class include labels, input and buttons, data manipulation, persistent storage, tables, sounds, and gestures. In the first week students were given take-home assignments every class period in order to help emphasize the concepts learned in class. These assignments focused on improving their attention to detail and order. One assignment included simulating a game of ”Red Light, Green Light” using pseudo code. In week two, in and out of class debugging or problem solving assignments were given each day.

In week 10, students were asked to propose a Final Project utilizing the material they learned in either the main iOS class or any of the guest lectures. The Final Project consisted of three parts: proposal, project and documentation submission, and presentation. No more lectures were provided and students used the remaining course time to work on their individual projects. Final project presentations were given on the final day of class.

Six students met the 70% pass/fail benchmark and were given certificates of completion. All the students were male, two were of African descent, one of Southeast Asian descent, and three were White/Caucasian. Two students were over 40 and four had prior CS knowledge.

3.4 Encountered Problems and Attempted Solutions

Wide Range of Experience with Technology Although all of the students in the course had previous experience using a computer, the range of expertise varied greatly. While some of the millennial students were expert computer users, at least one older student had difficulties saving to folders and navigating file structures. Although all the students had prior experience using a Windows operating system, only two were familiar with Mac OSX. This unfamiliarity made it difficult to progress with lessons, necessitating an overview of ”How to Use a Mac”.

Wide Range of CS Experience Half of the students entered into iOS Development with Swift with no Computer Science experience. By the second week, it was evident that more than half of the students were not grasping the material and that the pace of the course was too fast. Half of the syllabus was eliminated. Only the content listed previously was covered. Concepts like games, social network integration, online storage and maps were eliminated. Additionally, the structure of the course was changed. Instead of two or three days of lecture where several concepts were introduced followed by a homework assignment, students were introduced to one or two concepts before being tasked with implementing the concept themselves. On the first day of lecture, students were given a demonstration and code walk through. As the instructor explained the concept and typed each line of code, students took notes and copied code and comments. On the second day of lecture, students and instructor walked through each step needed to build an app similar to that which was completed on day one. As students described each step, code comments were written. After all steps were identified, the group started from the beginning and wrote the code for each step together. Finally, on the third day of lecture, students were asked to create an app similar to those created on days one and two, on their own. On weeks where there were only two lecture days, the presented concepts were simple enough to where the typical ”day two” activities could be skipped.

Lack of Hardware iOS apps can only be created on Apple machines. At the start of the course, only one student owned an Apple computer. In order for the class to continue, the job training and placement program provided each student with either an iMac, a Macbook Pro, or a Mac Mini on which they could work during the class period. In the initial course syllabus, students were expected to complete weekly assignments and to practice in-class examples at home. However, since only one student in the course owned a Mac computer, they were unable to comply. Three attempts were made to address this issue.

First, the job training and placement program allowed students to utilize the machines during the day. Several students had part-time jobs and were unable to take advantage of this extra ”lab” time. Second, the software development school provided each student with access to virtual, Apple computers. Most students owned older model computers with relatively slow processors and small hard drives. Coupled with slow Internet speeds, these computer configurations resulted in a great amount of ”lag” when using the virtual machines. Because of this lag, students reported difficulties with and frustration from using the systems at home. Finally, out of class assignments were eliminated and were converted to in class assignments.

4 Second Time Around: Android Application Development

The same partnership responsible for the initial iOS course was involved with the creation of a second course, Introduction to Android Development. It too was intended to offer mobile development instruction for members of the community, with the main difference being the alternate operating system.The course’s creators continued to offer scholarships to students enrolled in the 12 week course who were unable to afford the $7,000 cost. Unlike the iOS course, the Android course was projected to occur five nights a week for three hours each night. On Mondays and Fridays, representatives of the software development school would lecture on topics relating to prototyping, visual design and varying others that would supplement the Android lectures being taught the remaining three days. The Android lectures would cover a host of topics including utilizing the Android SDK and other development tools, understanding activities and intents, and creating applications with location based services. The lectures would allow for various in-class programming exercises that would mirror examples that would be useful in a mobile development position in industry.

Unfortunately the planned course never came into fruition. The course was originally planned to begin in early September, but was pushed back in an effort to elicit a higher enrollment. Flyers to attract more students were distributed 2-3 weeks before the second date that the class was scheduled to begin. At the time of cancellation, there were only two students enrolled in the course: one owned their own software company and the tuition of the other enrollee was being covered by their employer. Although the interest in the course was high, the members of the organizations involved in the planning of the course surmised that the low enrollment rate was most likely attributed to the prerequisite for the course. Programming in Android requires the prior knowledge of the Java programming language. Students would not have been able to develop a deep understanding of Android if they were taught Java simultaneously. In addition to low enrollment, there were also many logistic issues that were either not settled or was agreed upon at the last minute. For example, there were several location options mentioned, but the final location was never selected.

5 Recommendations for Future Courses and Unanswered Questions

There are several recommendations for future courses. First, app development courses should have strict pre-requisites that cover a specified computing skill set. The vast majority of the problems in the iOS course were due to the knowledge base of the students and the mismatch with the initial design and content of the course. The students came to the course with a wide range of prior knowledge, most of which did not prepare them to take the app development course that was prepared. We attempted to remedy this issue in the Android course by requiring students to know Java. This requirement most likely pushed students away. In order to avoid these issues (ranges of knowledge and exclusionary prerequisites), future partnerships should consider offering a tiered system of courses. Instead of offering app development course first, partnerships could offer computer literacy, introduction to computer science (in one or two courses) and end with an app development course. Using this model, the partnership can help ensure that students in the final course have the appropriate background while at the same time providing other necessary skills. With these courses in existence, the partnership will have the necessary materials to create assessments that check the prerequisite knowledge of any student who did not take any of their offered courses.

It is also recommended that students have access to any special equipment. Another major issue in the iOS course involved access to hardware. In a programming based course, students need access to computers so that they can practice. Any future partnership must be successful in ensuring that the targeted audience is well equipped to successfully complete the course. One way to meet this goal while keeping prices low, could be to solicit donated or price reduced computers from the University and other businesses. These computers could be loaned to students for a small fee with a guarantee of ownership after course completion. In order to keep students accountable for the machines, the partnership could levy a charge for lost or damaged hardware.

If students from a low-income area are being targeted, their ability to attend and afford the course and balance coursework, a job and family obligations may be impacted. Another recommendation is to lower the cost of the course. Although most students in the iOS course were recipients of a scholarship, the $7,000 price tag may be unsustainable as state and local funds may not always be available. One way to to reduce the cost is to augment the role of the graduate student instructors. Instead of teaching the course as part of funding packet, students could teach for course credit or as a means to gain experience for future academic careers. Instructor roles could also be filled by post-docs and/or junior faculty. The partnership could also solicit support from local technology companies and software development schools in the form of scholarships, lab space, hardware, or special topic instructors who teach as part of volunteer projects. These companies and businesses would in turn receive advertisement and face-to-face interactions with potential employees and current or future customers.

Despite the problems, the University believes that partnerships between Universities and the surrounding community have the potential to have positive impacts. Therefore, there will be another attempt to provide these courses. Although we provide these recommendations, several questions related to cost and course scheduling and working with non-traditional students remain. It is the hope that future attempts at community-based courses will provide flexibility to meet the needs of the communities and result in well prepared entry level mobile application developers.

References

[1] Bureau of Labor Statistics. Occupational outlook handbook, 2016-17 edition, software developers. `https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm`.

[2] Projections Central State Occupational Projections. Long term occupational projections. `http://www.projectionscentral.com/Projections/LongTerm`.

[3] Yi Xue and Richard C Carson. Stem crisis or stem surplus? yes and yes. `https://www.bls.gov/opub/mlr/2015/article/stem-crisis-or-stem-surplus-yes-and-yes.htm`.